Blekinge Institute of Technology
Doctoral Dissertation Series No. 2024:06
ISSN 1653-2090
ISBN 978-91-7295-479-3

Mining Evolving and Heterogeneous Data

Cluster-based Analysis Techniques

Vishnu Manasa Devagiri

DOCTORAL DISSERTATION
for the degree of Doctor of Philosophy at Blekinge Institute of Technology to be publicly defended on May 22nd, 2024, at 09:00 in room J1630, Campus Gräsvik

Supervisors
Prof. Veselka Boeva, Blekinge Institute of Technology, Sweden
Prof. Niklas Lavesson, Blekinge Institute of Technology, Sweden

Faculty Opponent
Asst. Prof. Shehroz Khan, KITE, Toronto Rehabilitation Institute, Canada
Grading Committee
Prof. Ivan Koychev, Sofia University, Bulgaria
Assoc. Prof. Sindri Magnusson, Stockholm University, Sweden
Assoc. Prof. Farhana Zulkernine, Queen's University, Canada

Abstract

A large amount of data is generated from fields like IoT, smart monitoring applications, etc., raising demand for suitable data analysis and mining techniques. Data produced through such systems have many distinct characteristics, like continuous generation, evolving nature, multi-source origin, and heterogeneity, and in addition are usually not annotated. Clustering is an unsupervised learning technique used to group and analyze unlabeled data. Conventional clustering algorithms are unsuitable for dealing with data with the mentioned characteristics due to memory, computational constraints, and their inability to handle the heterogeneous and evolving nature of the data. Therefore, novel clustering approaches are needed to analyze and interpret such challenging data.

This thesis focuses on building and studying advanced clustering algorithms that can address the main challenges of today's real-world data: evolving and heterogeneous nature. An evolving clustering approach capable of continuously updating the generated clustering solution in the presence of new data is initially proposed, which is later extended to address the challenges of multi-view data applications. Multi-view or multi-source data presents the studied phenomenon or system from different perspectives (views) and can reveal interesting knowledge that is invisible when only one view is considered and analyzed. This has motivated us to continue exploring data from different perspectives in several other studies of this thesis. Domain shift is another common problem when data is obtained from various devices or locations, leading to a drop in the performance of machine learning models if they are not adapted to the current domain (device, location, etc.). The thesis explores the domain adaptation problem in a resource-constraint way using cluster integration techniques. A new hybrid clustering technique for analyzing the heterogeneous data is also proposed. It produces homogeneous groups, facilitating continuous monitoring and fault detection.

The algorithms and techniques proposed in this thesis are evaluated on various data sets, including real-world data from industrial partners in domains like smart building systems, smart logistics, and performance monitoring of industrial assets. The obtained results demonstrated the robustness of the algorithms for modeling, analyzing, and mining evolving data streams and/or heterogeneous data. They can adequately adapt single and multi-view clustering models by continuously integrating newly arriving data.

Keywords: Domain Adaptation, Evolving Clustering, Heterogeneous Data, Multi-View Clustering, Streaming Data

Blekinge Institute of Technology
Doctoral Dissertation Series No. 2024:06

Mining Evolving and Heterogeneous Data

Cluster-based Analysis Techniques

Vishnu Manasa Devagiri

Doctoral Dissertation in Computer Science

Department of Computer Science
Blekinge Institute of Technology
SWEDEN

Blekinge Institute of Technology
Department of Computer Science

Blekinge Institute of Technology Doctoral Dissertation Series No. 2024:06
ISBN 978-91-7295-479-3
ISSN 1653-2090
urn:nbn:se:bth-26098

Printed in Sweden by Media-Tryck, Lund University, Lund 2024

”If we knew what we were doing, it would not be called research, would it?”

Albert Einstein

Acknowledgements

I want to initially acknowledge my supervisors, Veselka Boeva and Niklas Lavesson. Veselka, you have been a constant support for me throughout this journey. Thank you for always having confidence in me, from motivating me to take up PhD studies until today. I would not have reached this point without you. You are always available for discussions, regardless of the time. Niklas, your valuable inputs and ideas during our discussions have helped me to look at things from a new perspective, and you have always been active at sharing new opportunities to take up. The opportunity you gave me to work with your research team during my Master's has, in a way, been a starting point for this journey; thank you.

Shahrooz, my mentor, coauthor, and friend. Thank you for always being there to help and share your knowledge and experiences. Thanks to all my colleagues and friends at BTH for the happy working environment and all the fun, discussions, and encouragement.

Next, I want to acknowledge our collaborators from the industry, Farhad Basiri (iquest AB), Andrej Petef (Sony), and Peter Exner (Sony). Thank you for sharing the data and allocating your time to collaborate with us. Elena, Pierre, Michiel D., Annelies, Amir, and others at Sirris, Belgium. Thank you, guys, for being so welcoming and making it really easy for me to integrate during my visit. It has been a good and fruitful learning experience. Those two months were some of the best days of my PhD journey.

Finally, my family, thank you for encouraging and backing me in all my decisions, for constantly believing in my potential even when I do not, and for motivating me to achieve my dreams.

April 2024
Vishnu Manasa Devagiri

List of Papers

This thesis is a compilation of the following papers. The formatting of the papers is changed to be adapted to the thesis.

Paper I

V. Boeva, M. Angelova, V. M. Devagiri, and E. Tsiporkova. "Bipartite Split-Merge Evolutionary Clustering". In: Agents and Artificial Intelligence. Ed. by J. van den Herik, A. P. Rocha, and L. Steels. Cham: Springer International Publishing, 2019, pp. 204–223. DOI: 10.1007/978-3-030-37494-5_11

Paper II

V. M. Devagiri, V. Boeva, and E. Tsiporkova. "Split-Merge Evolutionary Clustering for Multi-View Streaming Data". In: Procedia Computer Science 176 (2020). Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020, pp. 460–469. ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.08.048

Paper III

V.M. Devagiri, V. Boeva, and S. Abghari. "A Multi-view Clustering Approach for Analysis of Streaming Data". In: Artificial Intelligence Applications and Innovations. Ed. by I. Maglogiannis, J. Macintyre, and L. Iliadis. Cham: Springer International Publishing, 2021, pp. 169–183. ISBN: 978-3-030-79150-6. DOI: 10.1007/978-3-030-79150-6_14

Paper IV

V. M. Devagiri, V. Boeva, S. Abghari, F. Basiri, and N. Lavesson. "Multi-View Data Analysis Techniques for Monitoring Smart Building Systems". In: Sensors 21.20

(2021). ISSN: 1424-8220. DOI: 10.3390/s21206775

Paper V

C. Åleskog, V. M. Devagiri, and V. Boeva. "A Graph-Based Multi-view Clustering Approach for Continuous Pattern Mining". In: Recent Advancements in Multi-View Data Analytics. Ed. by W. Pedrycz and SM. Chen. Cham: Springer International Publishing, 2022, pp. 201–237. ISBN: 978-3-030-95239-6. DOI: 10.1007/978-3-030-95239-6_8

Paper VI

V. M. Devagiri, V. Boeva, and S. Abghari. "Domain Adaptation Through Cluster Integration and Correlation". In: 2022 IEEE International Conference on Data Mining Workshops (ICDMW). 2022, pp. 1–8. DOI: 10.1109/ICDMW58026.2022.00025

Paper VII

V. M. Devagiri, V. Boeva, and S. Abghari. "A Domain Adaptation Technique through Cluster Boundary Integration". Submitted for journal publication (under review).

Paper VIII

V. M. Devagiri, P. Dagnely, V. Boeva, and E. Tsiporkova. "Putting Sense into Incomplete Heterogeneous Data with Hypergraph Clustering Analysis". In: Symposium on Intelligent Data Analysis (IDA 2024), Stockholm, Sweden, April 2024 (In press).

Other research publications related to but not included in the thesis are:

Paper IX

M. Angelova, V. M. Devagiri, V. Boeva, P. Linde and N. Lavesson. "An Expertise Recommender System based on Data from Institutional Repository (DiVA)". Leslie Chan and Pierre Mounier (Eds.): Connecting the Knowledge Commons – from projects to sustainable infrastructure. OpenEdition Press, pp.135-149, 2019. DOI: 10.4000/books.oep.9078

Paper X

V. Boeva, M. Angelova, V. M. Devagiri, E. Tsiporkova, ”A Split-Merge Framework for Evolutionary Clustering”, 31th Swedish AI Society Workshop SAIS 2019, Umeå, Sweden, June 2019.

Paper XI

V. Boeva, E. Casalicchio, S. Abghari, A.A. Al-Saedi, V. M. Devagiri, A. Petef, P. Exner, A. Isberg. and M. Jasarevic. 2022. ”Distributed and Adaptive Edge-based AI Models for Sensor Networks (DAISeN)”. Position Papers of the 17th Conference on Computer Science and Intelligence Systems, Annals of Computer Science and Information Systems 31 (2022): 71-78. DOI: 10.15439/2022F267

Funding

The research work done as a part of this thesis is partially funded by the following:

- ”Scalable resource efficient systems for big data analytics”, project funded by the Swedish Knowledge Foundation (grant: 20140032).
- ”Distributed and Adaptive Edge-based AI Models for Sensor Networks”, Sony Research Award Program 2020 Project.
- ”Human-centered Intelligent Realities (HINTS)”, project funded by the Swedish Knowledge Foundation (grant: 20220068).

Author's contribution to the papers

The author is the main driver and the first author for all the papers except for papers I and V. For the studies where she was the main driver and first author, she was involved in all the phases of the research, that is, idea generation, designing and conducting experimentation, analysis of results, writing the original draft, reviewing and editing the manuscript. For Paper I, she was mainly involved in designing and conducting experiments, analyzing results, reviewing and editing the manuscript. For Paper V, the author was involved in the idea generation, experimental design, analyzing results, reviewing and editing the manuscript. The author was also the co-supervisor for the master thesis project at the foundation of this study.

Abbreviations

AMI	Adjusted Mutual Information.
ARI	Adjusted Rand Index.
ED	Euclidean Distance.
FCA	Formal Concept Analysis.
HAR	Human Activity Recognition.
IoT	Internet of Things.
MI	Multi Instance.
ML	Machine Learning.
MST	Minimum Spanning Tree.
RI	Rand Index.
SI	Silhouette Index.
SNN	Shared Nearest Neighbors.
SNNS	Shared Nearest Neighbor Similarity.
TEDA	Typicality and Eccentricity Data Analysis.

Table of Contents

Acknowledgements i

List of Papers iii

Abbreviations vii

Chapter 1 Introduction 1

1.1 Research Problem . . . 2
1.2 Contributions and Papers Included . . . 4
1.3 Thesis Structure . . . 8

Chapter 2 Background 9

2.1 Domain Adaptation . . . 9
2.2 Evolving (Stream) Clustering . . . 9
2.3 Formal Concept Analysis . . . 10
2.4 Graph-Based Clustering . . . 11
2.5 Multi-Instance Clustering . . . 12
2.6 Multi-View (Stream) Clustering . . . 12

Chapter 3 Related Work 13

3.1 Evolving Clustering . . . 13
3.2 Multi-Source Data Analysis . . . 14
3.3 Domain Adaptation . . . 15

Chapter 4 Methodology 17

4.1 Data sets . . . 17
4.2 Distance measures . . . 17
4.3 Evaluation measures . . . 19
4.3.1 Internal Measures . . . 19
4.3.2 External measures . . . 20
4.3.3 Information Theory . . . 21
4.4 Research Methodology . . . 22
4.4.1 Challenges . . . 24
4.5 Validity Threats . . . 25
4.5.1 Internal Validity Threat . . . 25
4.5.2 External Validity Threat . . . 25
4.5.3 Construct Validity Threat . . . 25
4.5.4 Conclusion Validity Threat . . . 26

Chapter 5 Results and Analysis **27**
5.1 Evolving Clustering 27
5.2 Multi-Source Data Analysis 29
5.3 Domain Adaptation 33
5.4 Summary 35

Chapter 6 Conclusions and Future Work **37**

Chapter 7 Experiences and Learning Outcomes **39**

Bibliography **41**

Paper I Bipartite Split-Merge Evolutionary Clustering **51**

Veselka Boeva, Milena Angelova, Vishnu Manasa Devagiri, Elena Tsiporkova

In: Agents and Artificial Intelligence, Ed. by J. van den Herik, A. P. Rocha, and L. Steels. Cham: Springer International Publishing, 2019, pp. 204–223, DOI: 10.1007/978-3-030-37494-5_11

1 Introduction 51
2 Related Work 54
3 Methods and the Proposed Solution 56
3.1 Problem Description 56
3.2 Pivot Bi-Clustering Algorithm 56
3.3 Dynamic Split-and-Merge Clustering Algorithm 56
3.4 Bipartite Split-Merge Evolutionary Clustering Algorithm 57
4 Experimental Setup 61
4.1 Data 62
4.2 Metrics 63
4.3 Experiments 64
4.4 Implementation and Availability 65
5 Results and Discussion 65
6 Conclusion and Future Work 69
References 70

Paper II Split-Merge Evolutionary Clustering for Multi-View Streaming Data **75**

Vishnu Manasa Devagiri, Veselka Boeva, Elena Tsiporkova

In: 24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems KES 2020, Procedia Computer Science 176, 2020, pp. 460–469, DOI: 10.1016/j.procs.2020.08.048

1 Introduction 75
2 Related Work 77
3 Methods and Background 78
3.1 Split-Merge Clustering 79
3.2 Formal Concept Analysis 79
4 Proposed Multi-View Split-Merge Clustering 80

5 Initial Evaluation and Results 83
5.1 Data and Experimental Setup 83
5.2 Results and Discussion 85
6 Conclusions and Future work 88
References 88

Paper III A Multi-View Clustering Approach for Analysis of Streaming Data 91

Vishnu Manasa Devagiri, Veselka Boeva, Shahrooz Abghari

In: Artificial Intelligence Applications and Innovations, Ed. by I. Maglogiannis, J. Macintyre; L. Iliadis. Cham: Springer International Publishing, 2021, pp.169–183, DOI: 10.1007/978- 3-030-79150-6_14

1 Introduction 92
2 Related Work 93
3 Background 94
3.1 Multi-Instance Clustering and Hausdorff Distance . . . 94
3.2 Formal Concept Analysis 94
3.3 Closed Patterns 95
4 MV Multi-Instance Clustering using Closed Patterns 95
5 Evaluation 98
5.1 Data Sets and Experimental Setup 98
5.2 Results and Discussion 100
6 Conclusion and Future Work 103
References 104

Paper IV Multi-View Data Analysis Techniques for Monitoring Smart Building Systems 107

Vishnu Manasa Devagiri, Veselka Boeva, Shahrooz Abghari, Farhad Basiri, Niklas Lavesson

In: Sensors 21, 2021, DOI: 10.3390/s21206775

1 Introduction 108
2 Background 109
2.1 Multi-View Clustering 109
2.2 Stream Clustering Algorithms 110
2.3 Multi-Instance Learning 110
2.4 Distance Measures 111
2.5 Formal Concept Analysis 111
2.6 Closed Patterns 112
3 Related Work 112
4 Materials and Methods 114
4.1 Data 114
4.2 Multi-View Data Analysis Approach 114
4.3 Data Visualization and Analysis 120
5 Experimentation and Analysis 122

5.1 Data Preparation 122
5.2 Experimental Setup and Results 124
6 Applicability and Limitations 140
7 Conclusion and Future Work 141
References 142

Paper V A Graph-based Multi-view Clustering Approach for Continuous Pattern Mining 147

Christoffer Åleskog, Vishnu Manasa Devagiri, Veselka Boeva

In: Recent Advancements in Multi-View Data Analytics, Ed. by W. Pedrycz and SM. Chen. Cham: Springer International Publishing, 2022, pp. 201–237, DOI: 10.1007/978-3- 030-95239-6_8

1 Introduction 148
2 Related Work 149
2.1 Multi-view Clustering Algorithms 149
2.2 Stream Clustering Algorithms 150
2.3 Multi-view Stream Clustering Algorithms 151
3 Background 152
3.1 Minimum Spanning Tree Clustering 152
3.2 Non-negative Matrix Factorization 153
3.3 Cluster Validation Measures 154
4 MST-MVS Clustering Algorithm 157
4.1 Multi-view data integration 159
4.2 Extraction of multi-view patterns 160
4.3 Transfer of knowledge through artificial nodes 161
4.4 CNMF-based labelling algorithm 162
4.5 Pattern-based labelling algorithm 163
4.6 Computational Complexity 164
5 Data and Experimental Settings 166
5.1 Data 166
5.2 Data Preparation 168
5.3 Experiments and Validation 168
5.4 Implementation and Availability 170
6 Results and Discussion 171
6.1 Algorithm configuration 171
6.2 Tuning of algorithm parameters 172
6.3 Evaluation of algorithm performance 173
7 Conclusion and Future Work 180
References 181

Paper VI Domain Adaptation Through Cluster Integration and Correlation 187

Vishnu Manasa Devagiri, Veselka Boeva, Shahrooz Abghari

In: 2022 IEEE International Conference on Data Mining Workshops (ICDMW). 2022, pp. 1–8, DOI: 10.1109/ICDMW58026.2022.00025

1 Introduction . . . 187
2 Related Work . . . 189
3 Problem Statement . . . 190
4 Proposed Domain Integration Clustering Algorithm . . . 191
4.1 Range-based correlation measure . . . 191
4.2 The proposed algorithm . . . 191
5 Experimentation . . . 194
5.1 Public data . . . 195
5.2 Real-world use case . . . 196
6 Results and Discussion . . . 197
6.1 Public data . . . 197
6.2 Real-world use case . . . 199
7 Conclusions and Future Work . . . 201
References . . . 202

Paper VII A Domain Adaptation Technique through Cluster Boundary Integration 205

1 Introduction . . . 206
2 Related Work . . . 208
3 Proposed Algorithm . . . 209
3.1 Model Generalization and Cluster Representation . . . 211
3.2 Range-based Distance Measure . . . 211
3.3 DIBCA++ . . . 213
3.4 Learning Algorithm . . . 215
3.5 Computational Complexity . . . 216
3.6 Algorithm Explainability . . . 217
3.7 Algorithm Applicability . . . 218
4 Data Sets . . . 220
5 Evaluation Measures . . . 220
5.1 Adjusted Rand Index . . . 220
5.2 Adjusted Mutual Information . . . 221
6 Experiments . . . 221
6.1 Data Pre-processing . . . 221
6.2 Experiments on Smart Logistics Use Case . . . 222
6.3 Experiments on HAR Use Case . . . 224
6.4 Experiments with DIBCA . . . 226
7 Result Analysis and Discussion . . . 228
7.1 Smart Logistics . . . 228
7.2 Human Activity Recognition . . . 229
7.3 Comparison with DIBCA . . . 231
7.4 Explainability . . . 231
8 Conclusion and Future Work . . . 233
References . . . 235

Paper VIII Putting Sense into Incomplete Heterogeneous Data with Hypergraph Clustering Analysis 241

1 Introduction 241
2 Related Work 242
3 Hypergraph-based Clustering Analysis Method 244
3.1 Step I: Hypergraph Construction 244
3.2 Step II: Transformation to Simple Graph 245
3.3 Step III: Cluster Integration and Analysis 246
3.4 Step IV: Deriving KPIs to analyze performance 247
4 Evaluation in Industrial Use-case 247
4.1 Step I: Hypergraph Construction 248
4.2 Step II: Transformation to Simple Graph 249
4.3 Step III: Cluster Integration and Analysis 249
4.4 Step IV: Deriving KPIs to analyze performance 251
5 Conclusion 252
References 252

1 Introduction

Thanks to the growth and technological advancements in Internet of Things (IoT), sensor networks, smart monitoring applications, etc., a lot of data is available today. These applications generate data continuously, and there are various challenges in storing, processing, and obtaining valuable information from such huge amounts of data [1]. Machine Learning (ML) and data mining fields provide methods and techniques that can be applied to analyze data for extracting useful knowledge and insights that can be used to understand or monitor the studied system. There are various ML and data mining algorithms that can be broadly categorized into four groups, namely supervised, semi-supervised, unsupervised, and reinforcement learning [2]. Different types of ML algorithms are used based on the learning capabilities, nature of data, and target outcome [3]. Both supervised and semi-supervised learning algorithms require a large amount of labeled data for training, which is either generally unavailable when dealing with real-world data and/or is a costly process to label such large volumes of data. Unsupervised learning techniques have a greater demand and use in real-world scenarios, as they do not require labeled data. In addition to this, they are capable of mining hidden patterns from data [2]. Clustering is one of the most popular unsupervised learning techniques [4]. Clustering techniques are used to group data such that data points placed in a cluster are similar to each other and different from the data points of other clusters [5]. This thesis focuses on clustering techniques that handle evolving and/or multi-source/view data.

Data is generated continuously in many application scenarios, and the ML models built become obsolete over time or space due to changes in data characteristics causing the occurrence of phenomenons like concept drift [6] or domain shift [7]. These are important aspects to be considered while developing algorithms suitable for evolving data. When the data characteristics of newly arriving data are no longer the same, it becomes difficult to accommodate them into the existing model as it becomes outdated [8]. This provides the need for evolving algorithms, which can be updated regularly to be suitable also for the new data with potentially different characteristics. When new information is generated, many computational resources are required to rebuild the model. Hence, evolving or adaptive clustering algorithms are required to discover and accommodate data with new characteristics. In addition to the streaming nature, concept drift, and domain shift, there is also a need to develop clustering algorithms that can handle data generated from multiple sources, as

most applications, like smart monitoring systems, collect information from various devices. Such data generated from multiple sources (also known as multi-view data) observing the same event can present interesting details that are otherwise invisible when studying data from a single source [9]. Data heterogeneity is an additional problem that needs to be addressed when working with multi-view data as it is collected from different sources, e.g., multiple sensors or devices [10].

Traditional clustering algorithms are unsuitable for addressing the above-stated challenges of multi-source [11], multi-domain [7], and streaming [6] data. In this thesis, novel clustering approaches are developed to address different aspects of the above-discussed challenges. The developed algorithms can monitor, analyze, and interpret the data obtained from different smart applications used for monitoring, providing personalized recommendations, etc.

1.1 Research Problem

This thesis combines several research works presenting robust clustering techniques suitable for analyzing and extracting knowledge from evolving and heterogeneous data from single and multiple sources. In this framework, three evolving clustering algorithms, four multi-view clustering analysis algorithms/approaches, and two domain adaptation algorithms are proposed. These are part of the eight different studies that have been conducted as a part of this PhD thesis.

The main aim of the thesis is to develop and study clustering techniques to mine and analyze streaming data by handling its evolving, heterogeneous, and multi-source nature. The research objectives below are defined to achieve this aim.

Obj 1. To propose and explore data analysis techniques that can adapt the clustering model based on the changes in data characteristics.

Obj 2. To analyze multi-source streaming data by developing multi-view cluster integration techniques capable of capturing knowledge across different views.

Obj 3. To study and propose clustering-based analysis methods that can handle data with missing values generated from multiple heterogeneous sources.

Based on the aim and objectives set to be accomplished, the following research questions are stated and addressed in this thesis.

RQ1 *How can a clustering solution be updated to accommodate and catch evolving characteristics of continuously arriving data?*
Motivation: Many current-day applications continuously generate a lot of data whose characteristics tend to change over time. It becomes difficult to accommodate and fit new data into the current model in such contexts. In such situations, one alternative is

to rebuild the clustering model, which is not optimal as (i) it would consume a lot of computational resources [12], and (ii) higher importance should be given to the latest trends while retaining the existing knowledge, which would not be entirely possible by rebuilding the model from scratch. Hence, there is a need for a clustering approach that can integrate new data into the existing model based on its characteristics.
Papers: This research question is dealt with in papers I, II, III, IV, and V. Paper I proposes a novel *Split-Merge Evolutionary Clustering* technique, also used in Paper II. The algorithm can split or merge existing clusters based on the clustering model of newly arriving data, thus obtaining one final updated clustering model. In Paper III, *Bi-Correlation MI-Clustering* is proposed based on Multi Instance (MI) learning and bipartite correlation clustering. This algorithm is further evaluated in a real-world use case of smart building systems in Paper IV. Another algorithm *MST-MVS* is proposed in Paper V, which can be used for mining and analyzing multi-view streaming data.

RQ2 *How can clustering models generated in a multi-view streaming context be integrated into a robust global model capturing knowledge from different views?*
Motivation: Multi-source data is common in many application scenarios, and information from these sources could complement each other. When viewed together, this data can give or produce information that cannot be obtained when only each of these views is considered [9]. Therefore, successfully integrating this knowledge from multiple views into a single model might be helpful. Even though there have been lots of works in multi-view clustering and stream clustering, not much has been done in the area of multi-view stream clustering [13, 14].
Papers: This research question is addressed in papers II, III, IV, and V, where the first three papers use Formal Concept Analysis (FCA) to integrate knowledge from different views. In addition to FCA, closed patterns have also been used in papers III and IV to extract frequent patterns and reduce the complexity of the global model. In Paper V, the local clustering models of different views are initially evaluated to avoid negative impact and those of the desired quality (Silhouette Index (SI) greater than the set threshold) are used to build the global model. The selected representatives of the cluster and their attributes from other views are extracted to build an integrated matrix, which is clustered using a Minimum Spanning Tree (MST) based clustering algorithm.

RQ3 *How can we develop a resource-efficient domain adaptation algorithm for a robust clustering model alignment to new domains?*
Motivation: There has been a lot of work in the field of domain adaptation, but the majority of the state-of-the-art works use deep learning or are developed for computer vision-based applications. There are limited works addressing this for time-series data using unsupervised learning techniques. Edge devices usually generate data in such applications with strict resource constraints. Novel domain adaptation

techniques that are resource-efficient and can support robust model adaptation to new contexts are needed.
Papers: Papers VI and VII address this research question by proposing novel unsupervised domain adaptation algorithms, namely *DIBCA* and *DIBCA++*, which are based on cluster integration. The algorithms produce an integrated clustering model that can be used across the domains and adapted domain models, one for each domain. Both the algorithms are developed to be resource efficient as all the operations are performed using only the clusters' representatives.

RQ4 *What kind of clustering analyses can be performed on heterogeneous multi-source data with missing values to produce useful homogeneous clusters?*
Motivation: In many application scenarios related to monitoring the performance of different types of assets, data is usually generated from multiple sources and is likely to be heterogeneous. Analyzing and deriving meaningful insights from such data is challenging [15]. As stated before, certain interesting aspects might be associated with a subset of features. In such cases, using multi-view or multi-layered techniques to analyze a few features at a time might be helpful [11, 16, 17]. Another common concern with data from multiple sources is having many missing values due to lack of standardization, equipment malfunctioning, registration errors, communication issues, etc. Common practices to deal with missing values include imputation techniques or removing the features with a high degree of missing values, but such practices negatively affect the data quality [18]. This raises the need for studying and proposing clustering techniques suitable to analyze multi-source heterogeneous data with missing values. The obtained homogeneous groups are expected to have similar behavior and are useful for tasks like performance monitoring.
Papers: Paper VIII mainly focuses on this research question. A multi-layered clustering approach followed by hypergraph clustering based on k-medoids and nearest-neighbor similarity is proposed to achieve this. The other multi-view clustering algorithms (papers II, III, IV, and V) proposed in this thesis are also capable of handling heterogeneous data. However, *MST-MVS* is not capable of handling missing values, and it can be noted that even if the others (papers II, III, and IV) are capable of working in scenarios with missing values, they are not evaluated concerning this aspect in those studies.

A visualization of the connections between the aim, objectives, and research questions is presented in Figure 1.1.

1.2 Contributions and Papers Included

The main contribution of this thesis is the development and empirical analysis of novel clustering techniques suitable for the mining and analysis of evolving and heterogeneous (multi-source/view) data. This thesis includes eight papers, five of which,

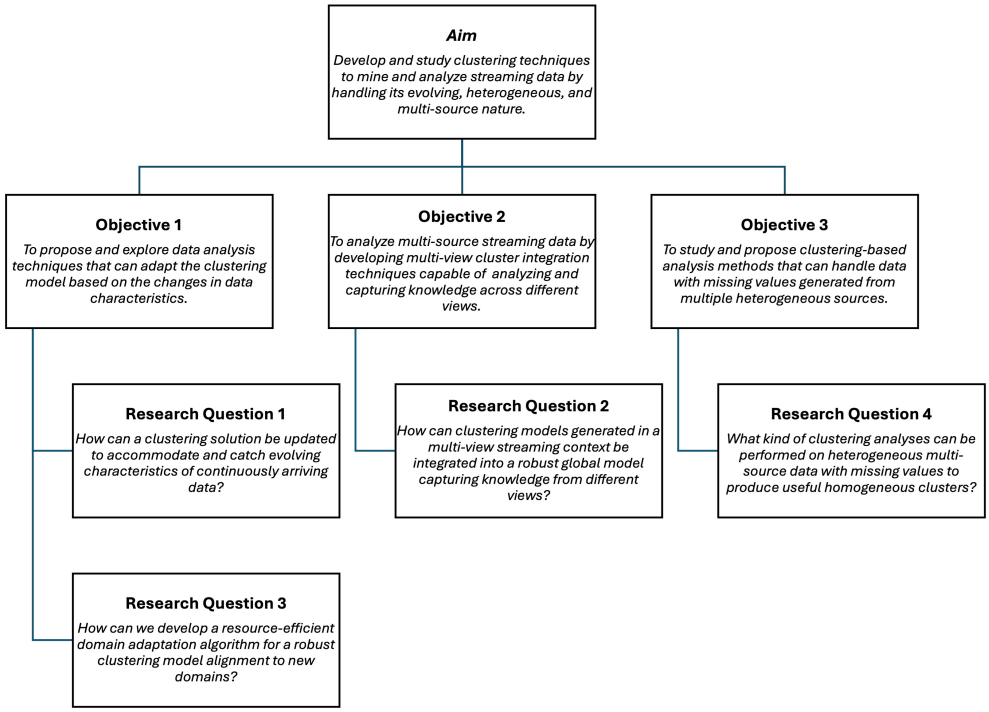


Figure 1.1: Figure visualizing the connections between the thesis's aim, objectives, and research questions.

namely Papers II, III, IV, V, and VIII, deal with multi-source data challenges, and all the papers except Paper VIII propose or use algorithms to address challenges related to the evolving nature (Papers I, II, III, IV, and V) and domain shift (when multiple devices or locations are considered; Papers VI and VII) of the streaming data. Figure 1.2 presents the research questions along with the information about which papers address them. This is followed by the text briefly summarizing the papers included in this thesis, along with their contributions.

Paper I. A novel clustering approach entitled *Split-Merge Evolutionary Clustering* is proposed. The approach integrates the clustering models of historical and newly arriving data using a bipartite graph based on the correlations between the two clustering models. An updated clustering solution is obtained by splitting or merging the clusters based on the edge connections of the bipartite graph. The algorithm is evaluated on four different data sets and compared with two other state-of-the-art algorithms.

Paper II. A multi-view clustering approach, *MV Split-Merge Clustering* based on the algorithm in Paper I (used to update the clustering solutions in each view), is proposed. It is developed to analyze multi-view streaming data and build a consensus clustering solution (global model) based on the information obtained from different views where FCA is used for the in-

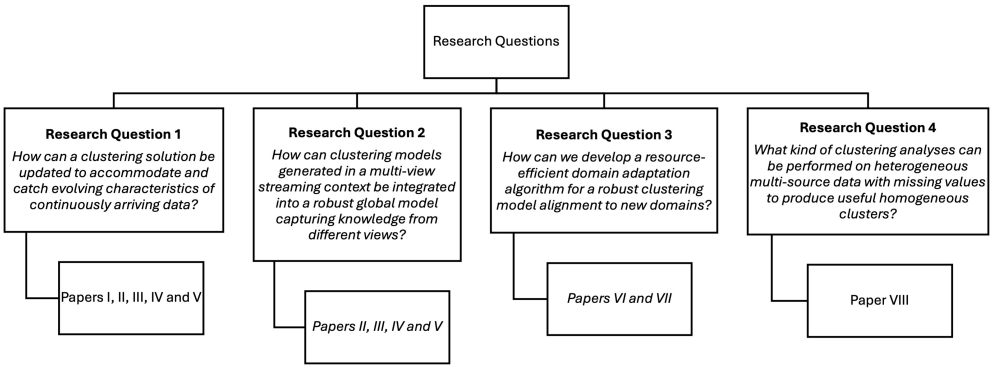


Figure 1.2: Figure visualizing the research questions and papers where they are addressed.

tegration. An initial evaluation is done, and the algorithm is compared to its batch version, where it has produced comparable results on an anthropometric data set, showcasing the algorithm's potential as a multi-view clustering solution.

Paper III. A novel multi-view clustering algorithm, entitled *MV Multi-Instance Clustering* is proposed. The new algorithm is developed to provide improved performance and interpretability of the results when compared to *MV Split-Merge Clustering*. Unlike Paper II, this paper uses a novel multi-instance learning algorithm proposed, *Bi-Correlation MI-Clustering* to update clustering solutions in each view. Closed patterns are also used to mine frequent concepts while building the global model, reducing its complexity. The results show that the proposed algorithm has performed better than the *MV Split-Merge clustering*.

Paper IV. This work studies the use of *MV Multi-Instance Clustering* algorithm for multi-view analysis of data in the smart building domain, using data provided by an industrial partner. The scenarios in which the algorithm could be used to analyze the data are presented and examined, focusing on contextual and integrated analysis of the systems. It also presents visualization techniques to showcase extracted knowledge that could be used to aid domain experts in detecting trends such as deviating behaviors. The study showed the algorithm's potential in monitoring, analyzing, and identifying deviating behaviors of sub-systems in a smart building system.

Paper V. A novel multi-view clustering algorithm entitled *MST-MVS* is proposed.

It is based on MST clustering and can be used to analyze and monitor streaming data. It is a continuous data mining approach where the integrated knowledge from the global model obtained at each data chunk is transferred to the next one through artificial nodes of the MST algorithm. This knowledge transfer, which proved to be beneficial, helps build the clustering solution in the next chunk by considering how the previous data has been grouped. A post-labeling technique is also proposed to label the chunk's data points based on the built global model. *MST-MVS* is evaluated on both real-world and synthetic data sets.

Paper VI. This work proposes a resource-efficient novel domain adaptation technique based on cluster correlation and integration, entitled *DIBCA*. It integrates knowledge from different domains, i.e., the source and target domain, by identifying their correlations. Correlations are obtained by labeling source data with the target model and target data with the source model. *DIBCA* produces an integrated model that can be used across the domains and a personalized adapted model for each domain. Its capability in automatic data labeling is showcased using the Human Activity Recognition (HAR) data set, and in addition, a real-world industrial use case of smart logistics is used to study its potential in the domain adaptation task.

Paper VII. This study proposes *DIBCA++*, a novel domain adaptation technique and an improved version of *DIBCA*. *DIBCA++* is developed to be robust to outliers compared to its predecessor. It requires a modest amount of storage and computational resources as it requires only the clusters' mean, standard deviation, and size. The algorithm's explainability aspects and applicability potential are also studied and presented. The algorithm is evaluated on a HAR data set and a smart logistics use case from an industrial partner. The experimental results showcase the better performance of *DIBCA++* over *DIBCA*. They also present the ability of *DIBCA++* to transfer knowledge between domains.

Paper VIII. In this study, a novel unsupervised data analysis method is proposed to group heterogeneous data with missing values into homogeneous groups that can be used for performance monitoring. Each group is expected to have comparable behavior, thus aiding domain experts in monitoring the performance of assets in the group. The proposed approach is developed such that it can handle missing values. The approach is based on concepts of multi-layer clustering, shared nearest-neighbor similarity, and hypergraph clustering. The proposed approach is evaluated on a real-world industrial data set of multi-source heterogeneous assets (compressors) with a substantial amount of missing values.

1.3 Thesis Structure

The rest of the thesis is structured as follows. Chapter 2 presents the background required for an easy understanding of the concepts used in different studies. Chapter 3 presents the summary of related works of the main contributing areas of the thesis. This is followed by Chapter 4, which offers an overview of the methodology used in this thesis. It covers data sets used, distance measures, evaluation measures, research methodology, and validity threats. The results of the thesis and their analysis are presented in Chapter 5, and the conclusions and future work are presented in Chapter 6. In Chapter 7, experiences and the learning outcomes identified during the PhD journey are presented. The next eight chapters consist of the research papers included in this thesis.

2 Background

This chapter briefly introduces the main research areas of the thesis, namely domain adaptation (Section 2.1), evolving (stream) clustering (Section 2.2), and multi-view stream clustering (Section 2.6) along with some concepts used in different papers included in the thesis. FCA, Section 2.3, is used as a cluster integration technique in multi-view clustering while building the global model. Section 2.4 presents different graph-based clustering techniques used in the thesis. The bipartite graph clustering and cut-clustering algorithm are used in identifying cluster structures. In one of the papers, hypergraph is used as an intermediate step to obtain a lower data granularity and help avoid the use of raw data in the later steps. MI clustering (Section 2.5) is also used to determine the cluster structures but is better at handling the ambiguity in real-world streaming data scenarios. Different subsections of this chapter are ordered alphabetically.

2.1 Domain Adaptation

ML models trained in one domain (location, device, etc.) do not always perform well when used in another domain due to changes in data characteristics. In such situations, the existing model must be adapted to the newer data characteristics. The domain where the model is initially trained is referred to as the *source*, and the one to which this model is adapted to is referred to as the *target* [19]. Domain adaptation is a sub-branch of transfer learning, where the source and target address similar problems but have different data characteristics [20]. That is, the quintessential ML assumption that the train and test data have the same data characteristics does not hold in this case [7, 21]. Papers VI and VII propose clustering-based domain adaptation algorithms for knowledge transfer between different domains.

2.2 Evolving (Stream) Clustering

Evolving clustering algorithms are designed to adapt and continuously accommodate data with evolving characteristics and are being used for data streams [22, 23]. Clustering is an unsupervised learning technique where labeled data is not required to

build a model. Similar to the traditional clustering algorithms, the main aim of these algorithms is to group the data, where data points belonging to a group are similar to each other and are different from the data points of other groups. In contrast to the traditional algorithms, in evolving clustering algorithms, all the data is not available to be handled in one batch; instead, the clustering structure evolves as new data arrives and is accommodated. According to Bouchachia [24], different phases of an evolving clustering algorithm can be categorized into matching, accommodating new data, and model refinement.

Stream clustering techniques are used to cluster evolving streaming data generated continuously without having the need to revisit or store the data [12]. Due to the large amounts of data produced, streaming data is generally not labeled. Hence, clustering is one of the most suitable techniques to analyze or mine useful information from such data [6]. While designing stream clustering algorithms, it is essential that time and memory constraints are considered for handling large amounts of data. Concept drift is a common problem that needs to be addressed by a clustering algorithm designed for streaming data [6].

Concept drift is a phenomenon where the data characteristics tend to change over time [6]. As a result, the ML model that cannot handle concept drift is considered outdated and shows degraded performance [8]. Such changes in data characteristics are common in many fields where data is generated over time. For example, if we consider consumer profiles of a clothing store, what a person buys from that store might change over a period of time due to some external factors like climate, special occasions, etc. Such changes in data characteristics are referred to as concept drifts. There are different types of concept drift scenarios introduced in the literature, such as blip, gradual, incremental, noise, recurring, or sudden [25]. In their study, Agrahari et al. [26] review the existing works on concept drift detection and identify their advantages and limitations.

The research literature does not always present a common opinion about the relationship between time-series and streaming data. Authors of [27] conclude that data streams are a special form of time series by discussing the relationship between both. Time-series data is obtained and collected chronologically over time; however, the amount of data received when considering streaming data is vast. In another work [28], in contrast to the popular opinion, the authors show that the presence of concept drift implies temporal dependence and claim about the effectiveness of stochastic gradient descent methods for learning in data streams.

2.3 Formal Concept Analysis

Formal Concept Analysis (FCA) [29] is a technique used to find relations between objects and their properties, also known as attributes in ML terminology. It has been used in areas like data mining and ML to extract useful information. Formal context

and concept lattice are two important parts of FCA. A formal context is a table where the rows and columns consist of objects and attributes, respectively. If an object possesses an attribute, the cell corresponding to this object and attribute is marked with a cross. Concept lattice, a hierarchical structure, is derived from the formal context and contains concepts that can be represented using (X, Y), where X and Y are the subsets of objects and attributes, respectively. A concept represents a group of data points all sharing the properties (attributes) present in the concept and vice-versa. In the concept hierarchy, there exists a super and sub-concept for each concept. Previously, many works such as [30–32] used FCA as a part of their clustering approaches to analyze or aggregate the clustering solutions. Papers II, III, and IV included in this thesis use FCA to integrate and build a global model from clustering solutions of different views.

2.4 Graph-Based Clustering

This section briefly reviews the graph-based algorithms used in our work. Bipartite graphs (Papers I, II, III, IV,), Cut clustering based on MST (Paper V), and hypergraph (Paper VIII), which are used as a part of our proposed algorithms or methods.

A bipartite graph is a graph with two disjoint sets of vertices U, V such that all the edges of the graph connect vertices from set U to set V. It can be noted that no two vertices from either set U or V are adjacent [33]. As stated in [34], there are many ways to cluster data from a bipartite graph. The authors define Bipartite correlation clustering as follows. They state that a bi-clique of the bipartite graph represents a cluster, and all the bi-cliques together represent the clustering solution. Ailon et al. [34] propose *PivotBiCluster*, a bipartite correlation clustering algorithm that does not need to have prior knowledge about the number of clusters. The algorithm uses merge functionality to integrate clusters from either side of the bipartite graph to obtain the final clustering solution.

Cut-clustering algorithm [35], which uses minimum cuts in a graph to cluster data, is also used in this thesis. The algorithm considers the data instances as nodes of the graph. An undirected graph with the edge weight representing the similarity between the nodes is built. This is followed by introducing an artificial node connected to all the other nodes of the graph with a constant distance α. Then, the graph's MST is computed, followed by removing the artificial node. The forest generated is the final clustering solution. Clustering algorithms proposed in [36, 37] use this *Cut-clustering* algorithm.

A hypergraph is a generalized version of a graph in which an edge can join any number of nodes. The edge of a hypergraph is also referred to as a hyperedge or a net; the nodes of an edge are referred to as the pins [38]. Hypergraphs are being used in a wide range of application scenarios like telecommunications, parallel data structures, computer science, ML, etc., [39]. Gao [40] in their work review exist-

ing works using hypergraph learning. They present different ways of hypergraph generation, learning models and applications where hypergraphs could be used. Difficulty in identifying appropriate hypergraph generation methods is highlighted and is identified as a potential area for conducting research.

2.5 Multi-Instance Clustering

Multi-Instance (MI) learning is used in papers III and IV. In MI learning, unlike traditional learning algorithms, a group of instances (also known as a bag) is considered as a single data object [41]. MI clustering is an unsupervised learning technique used to group these bags into clusters. Bags within the same clusters are similar to each other and are different from the bags in the other clusters. Like traditional clustering algorithms, MI clustering does not require labeled data and can create groups based on the built-in data structure. Even though the basic properties of MI clustering are similar to traditional clustering algorithms, it cannot be entirely treated like them as a single data instance considered here has many instances that might have different characteristics [42].

2.6 Multi-View (Stream) Clustering

Multi-view clustering techniques are used to cluster data obtained from multiple sources. Viewing information from numerous sources can sometimes provide valuable information that is not obvious when viewing it from just one source [9]. For example, a patient profile can be represented by the data collected from different smart monitoring devices, notes written by doctors, images such as x-rays, etc. A multi-view clustering technique considers information from across the views and builds a consensus or aggregated model representing this information [43].

Many challenges need to be addressed while working with data coming from multiple sources. As the data is obtained from various sources, there is a possibility that it could be heterogeneous [44]. Incomplete views, that is, the possibility of missing information from different views, is another common challenge.

Multi-view stream clustering algorithms are expected to address the challenges of both multi-view and stream clustering algorithms, as the name suggests. That is, it should be able to analyze and group streaming data generated from more than one source.

3 Related Work

This section presents the related works concerning the three main contributing areas of this thesis, namely evolving clustering, multi-source data analysis, and domain adaptation.

3.1 Evolving Clustering

The majority of applications these days generate streaming data, making the traditional clustering algorithms ineffective for analyzing and mining knowledge from such data. Data stream mining is an important research area [28]. Authors of [6, 12, 45, 46] review the existing state-of-the-art stream clustering algorithms. In [45], the authors highlight the challenges in the domain along with possible future directions and also focus on temporal aspects of data stream clustering. In [6], the common concepts related to stream clustering are introduced. This review analyses the considered algorithms in terms of computational complexity, clustering technique and accuracy. Authors of [46] classify stream clustering algorithms into two types one which has both online and offline learning phases and others which are capable of doing the job with just the online phase. In [12], the authors review 51 different stream clustering algorithms, which are categorized based on distance threshold, density grids, and statistical models.

Chakrabarti et al. [47] define evolving clustering algorithms (referred to as evolutionary clustering) as the ones that are able to deliver a clustering solution that can reflect current characteristics of evolving time series data and, at the same time, should not change drastically between two-timestamps. The authors of [47] also propose the evolutionary versions of k-means and agglomerative hierarchical clustering algorithms.

The authors of papers [48–50] propose novel clustering algorithms capable of splitting or merging the clusters based on the need to be able to accommodate new data. Lughofer in [48] proposes a dynamic clustering algorithm entitled *Dynamic split-merge*. The algorithm is designed to be used in integration with another incremental clustering algorithm. Incoming data points are initially accommodated into the existing clustering solution; this is followed by evaluating each cluster to determine if it needs to be split, merged, or retained. Fa et al. [49] in their work also

propose an algorithm with split and merge functionality, which are applied when the preconditions are satisfied.

In [22], the authors propose an evolving clustering algorithm entitled MicroTEDAclus, based on the Typicality and Eccentricity Data Analysis (TEDA) concept. The authors of [23] expand the existing evolving classifier (TEDAclass), which is also based on TEDA to clustering and regression and propose novel algorithms entitled TEDACluster and TEDAPredict, respectively.

Even though there is a lot of work being done in the field of evolving clustering and stream clustering algorithms, the intersection of this with multi-view clustering is yet to be completely explored [13, 14]. It can be noted that except in Paper I, the other evolving algorithms proposed are capable of handling multi-view data.

3.2 Multi-Source Data Analysis

Studies [10, 11] are surveys providing a good overview of the recent works in the field of multi-view clustering. Some of the novel algorithms proposed in the field are briefly presented in the following text. Wang et al. [51] in their work propose a novel algorithm entitled *MVC-LFA*. In the conducted experiments, the proposed algorithm performed better than the other considered algorithms. A multi-view clustering algorithm based on non-negative matrix factorization is proposed by Shao et al. in [14]. Zhu et al. [52] have proposed another algorithm in the field that is based on feature selection. There are various challenges that need to be considered when dealing with multi-source data, such as heterogeneity [44], missing data, or incomplete views. Authors of [14, 53–55] address incomplete views with the challenge of missing data in their works. Algorithms proposed in both [53, 54] are based on late fusion, where data in each view are initially clustered, and the results are integrated to obtain the final clustering solution. In [55], the authors propose a novel clustering method entitled SURE, which uses random sampling for the imputation of missing values, but potential inconsistencies that could occur are addressed.

Compared to the fields of multi-view and stream clustering, multi-view stream clustering is still in its infancy [13, 14]. Authors of [13, 14] have proposed novel algorithms in the field of Multi-View Stream Clustering. MVStream proposed in [13] is based on support vectors and is robust to concept drift. In [14], online multi-view clustering based on non-negative matrix factorization is proposed. In another study [56], the authors propose a multi-view representation learning method for clustering multi-view data streams. This is done by building a sparse affinity matrix based on the different views.

Papers II, III, and V of this thesis propose novel multi-view stream clustering algorithms. In Paper IV, the algorithm proposed in Paper III is further evaluated in a smart building domain. Paper VIII, unlike other included papers in this thesis, presents a multi-layered/view data analysis approach that is unsuitable for streaming

data but instead is relevant when the data set has a substantial amount of missing data.

3.3 Domain Adaptation

Studies like [19, 57, 58] present a state-of-the-art overview by summarizing the works done in the field. Authors of [57] present a survey of the state-of-the-art transfer learning algorithms suitable for the tasks of classification, regression, and clustering. The authors also discuss the relationship between transfer learning with domain adaptation, multi-task learning, sample selection bias, and covariate shift. Madadi et.al [58] present a survey studying the unsupervised domain adaptation techniques that could be used for classification tasks. In [19], the authors review homogeneous transfer learning algorithms that deal with the scenario where the source and target data sets have the same attributes.

In their work, the authors of [59] state the need for empirical results on time-series data and test the performance of two of the existing domain adaptation algorithms (based on convolutional and recurrent neural networks) on four different time series data sets. They have identified that the similarity of the source and target data sets impacts the performance. Li et al. [60], in their work, have proposed a novel domain consensus clustering algorithm to be able to handle source and target having different labels. The algorithm deals with this situation by separating the common and private clusters of the domains. It was evaluated on four different image data sets. In [61], a semi-supervised adaptive clustering algorithm based on adversarial clustering is proposed. Pseudo-labeling is used to increase the labeled instances of the target domain. Various works have been done in the field of domain adaptation, but the majority of these works are in the fields of computer vision and deep learning. There have not been many works addressing domain adaptation for the time-series data sets.

Source data privacy is another concern with many algorithms in the field; authors of [62, 63] propose source-free domain adaptation algorithms and state the importance of these. In addition, Pan et al. [57] state that unsupervised transfer learning is a less explored area and that there might be an increased research interest in this area in the future. Authors of [64] also propose an algorithm that does not require labeled data in both source and target domains and also state the importance of doing so. They propose an approach based on deep clustering, which uses data from different source domains and builds a domain-agnostic clustering model, refined in line with the target data when available.

Unlike the majority of the domain adaptation algorithms from the state-of-the-art, which are in the fields of deep learning and computer vision, the algorithms proposed in papers VI and VII are designed to be used in an unsupervised setup using clustering algorithms for time-series data. Another important aspect that these

algorithms handle well is data privacy. Both source and target data are protected as only parameters of the clustering models generated are used for knowledge transfer.

4 Methodology

This thesis is related to the areas of data mining, knowledge discovery, and ML; more specifically, it explores unsupervised learning techniques in adaptive and evolving clustering, multi-source data analysis, and domain adaptation. It aims to develop and study clustering techniques to mine and analyze streaming data by handling its evolving, heterogeneous, and multi-source nature. This chapter begins by presenting the data sets, distance measures, and evaluation measures used in different papers this thesis includes. Then, the research methodology and the challenges identified are discussed. Finally, the validity threats of the studies conducted are presented.

4.1 Data sets

The data sets used for evaluating the algorithms proposed in this thesis are presented in this section. We have used a mix of synthetic data, publicly available real-world data, and also data from real-world use cases provided by industrial partners. Ten different types of data sets are used across papers of this thesis namely, cover type [65], yeast [66], wine quality [67], anthropometric data [68], Dim32 [69], PAMAP2 [70], DaLiAc [71], real-world industrial assets data, real-time sensor data from smart building and smart logistics domains. Three of these data sets, cover type, yeast, and wine quality, are obtained from the UCI ML repository. The anthropometric data set is a publicly available data set of undergraduate students. It classifies whether a person has high blood pressure or not based on their anthropometric measures. Dim32 is a publicly available synthetic data set. PAMAP2 and DaLiAc are HAR data sets that are used to evaluate the domain adaptation algorithms. In addition to the publicly available data sets, three data sets, assets data, real-time sensor data from the smart buildings (heating and tap water sub-systems), and smart logistics domains, are obtained from our industrial partners. Table 4.1 presents the details about the data sets.

4.2 Distance measures

Most clustering techniques are distance-based. Different distance measures are usually used to measure dissimilarity between data points in order to conduct clustering

Table 4.1: Data sets used in the thesis.

Data set	Attributes	Classes	Papers
Cover-type	14	7	I, V
Yeast	8	10	I
Wine quality	12	7	I
anthropometric data	9	6	I, II, III
Sensor data - Smart buildings Domain (heating)	8	unlabeled	III, IV, V
Sensor data - Smart buildings Domain (tap water)	11	unlabeled	IV
Dim32	32	16	V
Sensor data - Smart Logistics	8	unlabeled	VI, VII
PAMAP2 (HAR)	31	17	VI
DaLiAc (HAR)	24	13	VII
Assets data	24	unlabelled	VIII

analysis. The quality of the clustering results is dependent on choosing an appropriate distance measure. It should be suitable to the data set and the used clustering algorithm [72].

Euclidean Distance (ED) is one of the most common distance measures used [72]; the distance between two data points A and B in a n-dimensional space using the ED can be calculated using Eq. 4.1:

$$ED(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}. \tag{4.1}$$

MI Clustering has its own characteristics, and the similarity measures used in single-instance clustering, such as ED, may not be appropriate. Average Hausdorff distance is used to determine the distance between the bags of objects in a MI learning problem [42]. The average Hausdorff distance between two bags of instances A, B can be computed applying Eq. 4.2, in which $d(a,b)$, is usually obtained using the ED. Another measure that can be used to measure similarity between two clusters is introduced in Paper VI. We have proposed a new range-based correlation measure to determine the closeness between two clusters represented by their minimum bounding boxes.

$$HD_{avg}(A,B) = \frac{\sum_{a\in A}\min_{b\in B} d(a,b) + \sum_{b\in B}\min_{a\in A} d(a,b)}{\mid A \mid + \mid B \mid} \tag{4.2}$$

Shared Nearest Neighbor Similarity (SNNS) [73] works on the principle that data points having a higher number of common neighbors are more likely to be similar to each other. The neighbors for each data point are obtained using k nearest neighbors. SNNS of two data points a, b can be calculated using Eq. 4.3, where $\Gamma(a)$ represents the neighborhood of a, d_{ap} is the distance between a and its neighbour p, and Shared Nearest Neighbors (SNN), $SNN(a,b) = \Gamma(a) \cap \Gamma(b)$. Inspired by this, we have modified and used SNNS to determine the similarity between different edges (hyperedges) of a hypergraph, where SNN between two edges are the common

vertices between the hyperedges. SNNS between two edges when they are present in each other's neighborhood is obtained by dividing the intersection of neighbors by the union of neighbors.

$$SNNS(a,b) = \begin{cases} \frac{|SNN(a,b)|^2}{\sum_{p \in SNN(a,b)} (d_{ap}+d_{bp})}, \text{ if } a,b \in SNN(a,b). \\ 0, \text{otherwise} \end{cases} \tag{4.3}$$

Other distance measures that have been explored in some of our studies are Hamming distance [74], which determines the similarity between two strings of equal length, and dynamic time wrapping [75] suitable for time-series data.

4.3 Evaluation measures

Different evaluation measures have been used across the papers in the thesis. Clustering models can be evaluated using either internal or external validation measures [5]. External validation measures use external information not used to build the clustering model, such as the labels to validate the model. Whereas it is the opposite for internal validation measures, they use the same information that is used to build the clustering solutions. Internal measures assess compactness, separation, connectedness, and stability aspects of clustering solutions. External validation measures are further divided into unary and binary [76], and the third category, information theory [77], is also used by some authors. Table 4.2 gives an overview of the types of clustering measures used in this thesis.

4.3.1 Internal Measures

Silhouette Index (SI): It measures the compactness and separation within and between different clusters of a clustering solution [78]. Its values range from -1 to 1, where higher values denote a good clustering solution. SI of a clustering solution C is defined as follows:

$$SI(C) = \frac{1}{m} \sum_{i=1}^{m} \frac{(b_i - a_i)}{\max\{a_i, b_i\}} \tag{4.4}$$

In the above equation, for object i, a_i is the average distance between i and other objects within the cluster, and b_i denotes the minimum average distance between distances of objects.

Davies Bouldin: The Davies Bouldin measure [79] measures the average similarity of each cluster of the clustering solution to its most similar one, where similarity is the ratio of intra-cluster (within clusters) distances to inter-cluster (between clusters)

distances. The least values represent a better clustering solution, zero being the best possible value.

Calinsiki Harabasz Index: Calinsiki Harabasz Index [80] or the Variance Ratio Criterion is the ratio of the sum of inter-cluster dispersion and intra-cluster dispersion. The formula to calculate the index is presented in Eq. 4.5, where k is the number of clusters in the clustering solution, n is the total number of data instances in the data set, BGSS and WGSS stand for between-group (cluster) sum of squared and within-group sum of squared distance.

$$CHI = \frac{BGSS/(k-1)}{WGSS/(n-1)} \tag{4.5}$$

Connectivity: Connectivity [76] uses k-nearest neighbors to determine how well the data points within a cluster are connected. It is based on the number of neighbors of k-nearest neighbors within the cluster. It ranges between 0 and infinity, and the lower the value, the better the clustering solution. The connectivity, c of a clustering solution C, is obtained using Eq. 4.6, where m_{ij} represents the jth nearest neighbor of data point i. The value of Xim_{ij} depends on whether i and m_{ij} belong to the same cluster or not. If they belong to the same cluster, the value is 0, else it is $1/j$.

$$c(C) = \sum_{i=1}^{m} \sum_{j=1}^{n_r} Xim_{ij} \tag{4.6}$$

4.3.2 External measures

Adjusted Rand Index (ARI): It is used to measure the similarity between two clustering solutions based on the number of pairs (predicted and ground truth) of samples that are categorized into similar or different clusters. ARI [81] is an adjusted for-chance version of the Rand Index (RI) [82]; adjustment is done such that it produces a score closer to 0 for random results. It ranges between -0.5 and 1.0, where 1.0 is a perfect match and closer to 0.0 for random labeling. ARI can be obtained using Eq. 4.7, where n is the number of elements; a_i and b_j are sum of the pair of samples of i and j, respectively.

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - (\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2})/(\binom{n}{2})}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - (\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2})/(\binom{n}{2})} \tag{4.7}$$

F-Measure: The F-measure [83] or F-score is used to evaluate the obtained final clustering result against the ground truth. It is a harmonic mean of precision (positive prediction quality) and recall (amount of true positives identified correctly). Given two clustering solutions, C and C' presenting the ground truth and the obtained clustering solution, respectively. F-measure of a cluster i of C'_i is calculated using Eq. 4.8, where C_j represents the cluster with most instances from C'_i. The final F-measure

value of C' is obtained using Eq. 4.9, where n is the total number of clusters in C'. F-measure ranges between 0 to 1, where 1 is the perfect value.

$$F(C'_i) = \frac{2\,|C_j \bigcap C'_i|}{|C_j| + |C'_i|}. \tag{4.8}$$

$$F(C') = \frac{1}{n}\sum_{i=1}^{n} F_i. \tag{4.9}$$

Jaccard Index: Also known as the Jaccard similarity coefficient [84], it is used to measure the similarity between two clustering solutions, i.e., the benchmark and the obtained clustering solution. It ranges between 0 and 1, where values closer to 1 present higher similarity between the clustering solutions. Jaccard Index of two clustering solutions A and B is defined in equation 4.10. The numerator presents the number of data points of same class within the same cluster of A and B, and the denominator presents the total number of data points in the same clusters of A and B.

$$JaccardIndex(A, B) = \frac{|\ A \cap B\ |}{|\ A \cup B\ |} \tag{4.10}$$

Accuracy: Accuracy presents the fraction of correct predictions made from the total predictions. It is used to evaluate the performance of the proposed clustering solution. Accuracy is calculated using Eq. 4.11, where p and n represent the correct and total predictions, respectively. It can be noted that accuracy might not be a good measure when the dataset does not have a balanced distribution of classes or labels, as the accuracy value will be high even if only one class is correctly predicted.

$$Accuracy = \frac{p}{n} \tag{4.11}$$

4.3.3 Information Theory

Adjusted Mutual Information: Adjusted Mutual Information (AMI) [85, 86] is an adjusted-for-chance version of MI. The value of MI is impacted by the number of clusters, and the higher the number of clusters, the better the value which is accounted for in AMI. AMI of two clustering solutions, A and B can be obtained using the below formula [87], where $H(.)$ represents the entropy of the clustering solutions and the expected MI between A and B is represented using $E(MI(A, B))$. It has an upper limit of one when the two clustering solutions perfectly match each other.

$$AMI(A, B) = \frac{[MI(A, B) - E(MI(A, B))]}{[avg(H(A), H(B)) - E(MI(A, B))]} \tag{4.12}$$

Homogeneity and Completeness: Homogeneity and completeness [88] are both independent of the absolute values of the cluster labels. They are used to compare two

clustering solutions (the benchmark considered and the obtained clustering solution). A perfectly homogeneous clustering solution's clusters contain instances of only one class of benchmark in each cluster of the obtained clustering solution. Whereas for perfect completeness, all instances of a class from the benchmark are in the same cluster. These measures are not symmetric, i.e., replacing benchmark labels with clustering solution labels or vice-versa will not give the same results but instead give the output of the other measure. The formula used to obtain the homogeneity and completeness of two clustering solutions is complex and out of the scope of this thesis and, hence, is not presented.

Apart from evaluating the final clustering solutions, the SI is also used in the clustering process. In papers I, III, and IV, SI is used to find the optimal number of clusters when using k-means to do the initial clustering of the data points in a chunk. In Paper VIII, SI along with Calinski Harabasz, Davies Bouldin, and Connectivity are used to determine the optimal number of clusters while using k-means to cluster the data in different layers (multi-layered clustering approach is used in this paper). In Paper V, SI is used to evaluate the quality of the local clustering solutions to decide which ones will be used for building the global model. It can be noted that in papers IV and VIII, the final results are evaluated using prior domain knowledge available with the help of experts and statistics.

Table 4.2: Evaluation measures used across papers

Evaluation measures	Papers
Internal	
Silhouette Index	I, III, IV, VIII
Davies Bouldin	VIII
Calinski Harabasz Index	VIII
Connectivity	VIII
External	
Adjusted Rand Index	II, III, V, VII
F-measure	I, VI
Jaccard Index	I
Accuracy	VI
Information theory	
Adjusted Mutual Information	V, VII
Purity/Homogeneity	II, III, V
Completeness	V

4.4 Research Methodology

Studies conducted in this thesis use research methodologies implementation, experimentation, and case study [89]. Novel clustering approaches are proposed and evaluated using experimentation in each of the studies included in this thesis, except for paper IV. In Paper IV, a case study is conducted where the algorithm proposed in III is applied and evaluated in the smart building domain. Various types of experiments are conducted to validate the algorithms using different types of data sets (more information about the data sets used is present in Section 4.1). The different ex-

perimental setups have been specially designed to study and evaluate the developed algorithms' performance and capability of capturing new data characteristics. Those are discussed in relation to the different conducted studies further in this section. The applicability of the algorithms to problems in different applied scenarios, such as use cases in logistics, smart building, and performance monitoring of assets, have also been studied, and the algorithms have been validated on real-world data sets provided by industrial partners. Other characteristics of some of the proposed algorithms that have been analyzed and explored are explainability and interpretability.

In Paper I, the proposed algorithm, *Split-Merge Evolutionary Clustering* has been compared with two other state-of-the-art algorithms, namely *PivotBiCluster* [34] and *Dynamic Split and Merge Clustering* algorithm [48] using the cover-type and wine quality data sets. Along with these comparisons, experimentation was also done to evaluate if the size of newly arriving data impacts the algorithm's performance, for which yeast and anthropometric data sets are used.

Papers II and III propose novel multi-view clustering approaches, entitled *MV Split-Merge Clustering* (based on *Split-Merge Evolutionary Clustering*) and *MV Multi-Instance Clustering*, respectively. Paper II is an initial study where the proposed algorithm is evaluated on anthropometric data. The proposed algorithm is also compared to its batch version. Paper III tries to improve the performance, understandability, and interpretability of the results compared to the algorithm proposed in Paper II and has been evaluated on both anthropometric and real-world sensor data of the heating system from the smart building domain. The results obtained on the anthropometric data are compared to those obtained in Paper II. Paper IV is an applied work where the potential of *MV Multi-Instance Clustering* is investigated in the smart building domain. Real-world sensor data of the heating and tap-water systems from the smart building systems are used in the study. Two different sets of experiments are designed and performed to showcase the algorithm's potential in context-aware modeling of system behavior and integration analysis of system performance. The study also presents different visualization techniques to aid domain experts in understanding the results and using them to analyze and monitor system performance.

In Paper V, the proposed MST-based multi-view algorithm (*MST-MVS*) has been evaluated on three data sets (Dim32, Cover-type, and Real-world sensor data of the heating system from the smart building domain), it has been compared to *MV Multi-Instance Clustering* (algorithm proposed in Paper III) using the real-world sensor data set. Two new approaches, BNodes, and LEdges, are proposed to calculate the artificial nodes that are used to transfer the knowledge extracted from the global model to cluster the newly arriving data chunks. Various experiments are conducted to evaluate and justify that different steps included in the algorithm improve the quality of the final clustering solution generated. In one experimental setup, the usefulness of the knowledge transfer is studied. Another set of experiments is done to understand how the quality of the local models affects the built global model.

Papers VI and VII propose unsupervised domain adaptation algorithms, *DIBCA*

and *DIBCA++*, respectively. Both these papers use sensor data from the smart logistics domain. Paper VI uses a publicly available HAR data set (HAR-1) to showcase the algorithm's potential in the automatic data labeling task, and sensor data is used to showcase its potential in the domain adaptation task. The sensor data used in this paper is obtained from a single device operating at two different locations. Paper VII uses another HAR data set (HAR-2) and sensor data obtained from five different devices. The sensor data is used to understand how the similarity between data from different domains (devices) affects *DIBCA++*. Using both data sets, the ability of *DIBCA++* to transfer new knowledge between the domains is showcased. *DIBCA++* is also experimentally compared to its predecessor *DIBCA* in Paper VII.

Paper VIII proposes a hybrid clustering methodology, combining multi-layer data analysis with SNNS and hypergraph clustering. The algorithm is developed to be used to interpret and analyze heterogeneous multi-source data with missing values. A real-world data set related to condition monitoring of assets is used to validate the approach. The obtained homogeneous clusters are used to derive KPIs and are evaluated on the sensor data related to these compressors.

4.4.1 Challenges

The main challenges we have faced during the implementation and evaluation of the proposed algorithms are briefly stated in this section.

Evaluating unsupervised models: Since ground truth is not available in most situations, evaluating the obtained final clustering solution can sometimes be challenging. Even though internal cluster evaluation measures are available, they might give contradictory results as they evaluate different aspects of the clustering solution. Thus, sometimes making it difficult to identify the best-performing one. This has also been highlighted by Luxburg et al. [90] in their work.

In addition, there are only a few cluster validation measures for streaming data [91]. Instead, most researchers in the field use validation measures for traditional clustering algorithms which might not be very optimal for streaming data scenarios.

Comparison with state-of-the-art: In the fields where this thesis is conducted, there are not many standardized benchmark algorithms against which the proposed algorithms can be compared and evaluated. Even when similar algorithms are identified, their implementations are unavailable in most scenarios. The alternative is to implement them by ourselves, but there might be some disparities in the implementation as there is a chance of misinterpretation of some minute details. Therefore, motivated by Luxburg et al. [90] belief that clustering should not be treated as an application-independent mathematical problem in the majority of the studies collected in this thesis the proposed algorithms/approaches are evaluated on real-world industrial prob-

lems.

4.5 Validity Threats

This section describes the various validity threats that could have occurred in the process of conducting the thesis and the measures taken to overcome them.

4.5.1 Internal Validity Threat

Internal validity addresses the concerns related to the effects of experimental setup on the results [92–94]. Multiple test sets are used to avoid selection bias while dividing the data set into different chunks. For example, 10 different test sets of the data set are used in Papers I, II, III, and V for all the conducted experiments in all experimental scenarios except when real-world sensor data is used. The average value (Paper V uses the minimum and maximum values as well) is considered as the final result.

4.5.2 External Validity Threat

External validity reflects the generalisability of the results obtained in a study [92–94]. Experiments conducted across all the papers included are designed with caution to mitigate such threats. In most papers, more than one data set is used to evaluate the proposed algorithm to avoid results specific to a particular scenario or use case. Only papers II, IV, and VIII use one type of data set for evaluation. Paper II is an initial study, and hence only a single data set is used. Paper IV uses the algorithm proposed in Paper III (where initial evaluation has been done on other data sets) in an applied scenario of smart building systems. Paper VIII is also an applied study where the proposed approach is tested on the use case related to performance monitoring of industrial assets. Testing the algorithm on more than one data set can generalize the performance of the algorithm when compared to using it just on one data set. However, there might still be some use cases or scenarios where the proposed algorithms are not appropriate. Luxburg et al. [90], in their study, highlight the need to study clustering algorithms as an application-dependent problem.

4.5.3 Construct Validity Threat

Construct validity addresses the concerns related to the results being deviated from the desired conceptual output [92–94]. Such threats could become a reality if the implemented algorithm does not produce results similar to the algorithm proposed during the development phase. To avoid such threats, the proposed algorithms, experimental scenarios, and setups are well discussed within the research group before the implementation phase begins. During the implementation phase, care is taken

by conducting periodical tests to ensure that the code's functionality is as expected. This is done to avoid logical and/or run time errors, which are difficult to identify compared to compile time errors.

Devoted experiments are conducted to select the algorithm's parameters in most cases. For Paper IV, we studied different sizes of data chunks, different techniques for finding artificial nodes, etc. Empirical methods have been used to determine the number of clusters when this information is unavailable. Papers I, III, and IV use the elbow method based on the SI to determine the k value when using the k-means algorithm; Paper VIII uses other cluster evaluation measures along with SI as stated in Section 4.3.

4.5.4 Conclusion Validity Threat

Conclusion validity deals with the effectiveness of the research in terms of treatment of data, how the experiments are conducted, and the obtained outcomes [94]. In general, all the papers have undergone a peer-review process. They are published (except Paper VII, which, as of 4^{th} April 2024, is under review) in various conferences and journals, which validates the experimental treatment used across the papers.

5 Results and Analysis

This chapter summarizes the results of the thesis. We have identified three main research domains to which the thesis has contributed: evolve clustering, multi-source data analysis, and domain adaptation. The thesis achievements are presented and discussed with respect to these domains. Furthermore, potential application scenarios are outlined for the algorithms proposed in each domain. Finally, all the research questions formulated in this thesis are stated and answered.

5.1 Evolving Clustering

The thesis's results contributing to the evolving clustering domain are presented in papers I, II, III, IV, and V. papers I, III, and V propose novel evolving clustering algorithms, namely *Split-Merge Evolutionary Clustering*, *Bi-Correlation MI-Clustering*, and *MST-MVS*. *Split-Merge Evolutionary Clustering* is also used in Paper II in a multi-view context. It can be noted that *Bi-Correlation MI-Clustering* is a part of *MV Multi-Instance Clustering* algorithm, which along with *MST-MVS* are developed to handle multi-view data in a streaming fashion. Paper IV is an applied paper using the algorithm proposed in Paper III in the smart building domain. Algorithms proposed/used in papers I, II, III, and IV use bipartite correlation clustering based on different approaches. The correlations between the existing clustering model and the clustering solution of the newly arriving data chunk are considered to determine the similarities and differences between the two clustering models. Based on the correlations, the existing clusters are either merged, split, or retained as they were, and a new, updated clustering model is obtained. Figure 5.1 (taken from Paper I) visualizes the bipartite correlation clustering or the split-merge framework of the *Split-Merge Evolutionary Clustering* where C and C' represent the existing and new clustering models, respectively.

In the conducted experiments, *Split-Merge Evolutionary Clustering* has exhibited its ability to integrate newly arriving data continuously. When compared to two other state-of-the-art algorithms, *PivotBiCluster* [34], and *Dynamic split and merge clustering* [48], it is observed that different algorithms excelled for different evaluation measures. This validates what Luxburg et al. [90] have stated, that different cluster validation measures can sometimes produce contradictory results, and it is

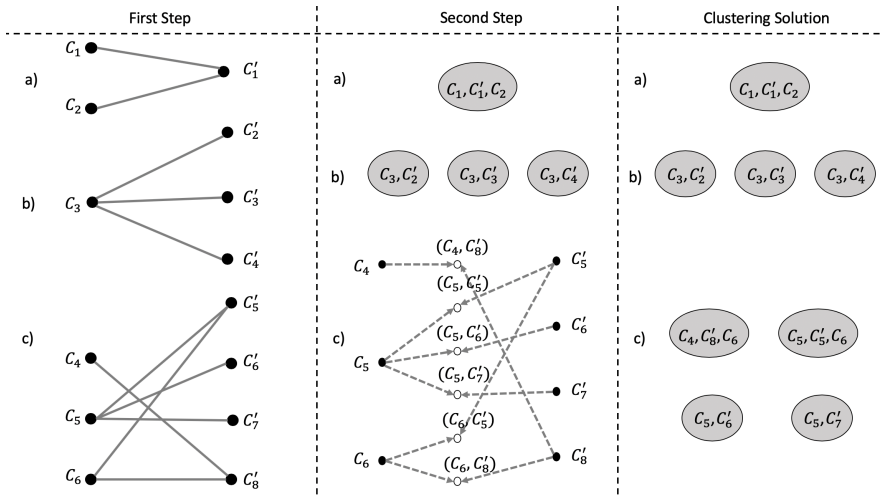


Figure 5.1: Visualization of Split-Merge framework in three different scenarios. a) bi-clique with under-clustered nodes (C_1 and C_2 correlate with C'_1); b) bi-clique with over-clustered nodes (C_3 correlates with C'_2, C'_3 and C'_4); c) a bi-clique where there is a need to be decomposed into subcomponents, the second step presents the transformation into tripartite graph with split (left) and merge (right) subcomponents. Figure is taken from Paper I.

difficult to validate how the clustering algorithm performs using these validation measures. Luxburg et al., therefore, highlight the importance of studying clustering algorithms considering their final usage and not as an application-independent problem. It is interesting to discover that the number of clusters produced by the proposed *Split-Merge Evolutionary* clustering is closer to the benchmark solution when compared to the other two algorithms.

Bi-Correlation MI-Clustering developed in Paper III is based on MI clustering and bipartite correlation clustering. In *Split-Merge Evolutionary Clustering*, only representatives from the clusters are used to find the correlations, which might sometimes impact the performance. The proposed algorithm in Paper III overcomes this by using MI-Clustering, where each cluster is treated as an individual data object (bags of instances). The experimental results also show improved performance when MI-Clustering is used (see Section 5.2). Paper V proposes yet another evolving clustering approach based on MST clustering. Unlike others, this algorithm uses knowledge from the global model (model integrating knowledge from all the views) when building the clustering solution of the next data chunk.

The algorithms proposed/used in Papers I, II, III, and IV can be categorized as the sliding window processing models and the *MST-MVS* (Paper V) as a landmark window model based on the classification of processing methods of streaming data from the literature [95]. Note that, unlike regular sliding window models, where one or more elements are processed in each fixed-element counting window, the proposed algorithms categorized into this do not have a hard restriction on the chunk sizes, and they may vary, but the algorithm always uses two chunks in each iteration (chunks t

and $t-1$).

Note that the above-discussed algorithms automatically adapt the clustering solution to handle the changes in data characteristics due to phenomena like concept drift. We want to study this aspect in future works further to be able to analyze and understand how the proposed algorithms fare in different types of concept drift scenarios, classified as sudden, incremental, gradual, recurring, blip, or noise [25]. Based on our conceptual understanding, the proposed algorithms perform better in the case of recurring, sudden, and incremental drifts. Blips or noises may not be identified due to the following reasons: (i) the underlying algorithm used to identify the initial clusters might not be able to identify the noise as a new cluster due to a low number of instances, (ii) specifically in papers III and IV only closed concepts are used while building the global model (combined model of different views obtained after using *Bi-Correlation MI-Clustering* to obtain clustering solution in each view) filtering out the noise.

Application Scenarios: These proposed algorithms could be used in a wide range of real-world application scenarios where the data is generated in a continuous fashion and characterized by an evolving nature. Some examples include (i) analysis and performance monitoring of the heating sub-system in the smart building domain; due to changes in behavioral patterns of humans and climate changes, the normal system behavior is not constant and is evolving. (ii) patient profiling and monitoring; these models could be used for precision medicine, where patients are divided into groups to provide personalized treatment. Patients in a group are expected to have similar characteristics, thus requiring similar treatment. As more information from different patients is obtained, the existing clustering model could become outdated due to factors like advancements in the medical fields, the arrival of patients with new disease characteristics, etc., and the need arises to rebuild the existing clustering model to be able to adapt to the new trends. In such situations, the proposed evolving clustering algorithms could be used. Papers III, IV, and V have evaluated the functionality of the proposed algorithms in the smart building domains.

5.2 Multi-Source Data Analysis

Papers II, III, IV, V, and VIII of this thesis contribute to the area of multi-source data analysis. Novel multi-view clustering algorithms *MV Split-Merge Clustering*, *MV Multi-Instance Clustering*, and *MST-MVS* are proposed in Papers II, III, and V, respectively. As stated before, Paper IV is an applied work where *MV Multi-Instance Clustering* is evaluated in the smart building domain. Its potential in analyzing and monitoring the sub-systems (heating and tap-water systems are considered in the study) of the smart building domain is demonstrated. Paper VIII presents a hybrid clustering workflow for analyzing incomplete heterogeneous data.

FCA is used in papers II, III, and IV to integrate knowledge from different views

into a global model consisting of a formal context and concept lattice. In addition to FCA, closed patterns are used in papers III and IV to extract frequently occurring patterns from the formal context, thereby reducing the complexity of the concept lattice. Multi-view stream clustering algorithms proposed in Papers II and III can perform both horizontal and vertical data integration. That is, the algorithms can integrate the clustering solution of newly arriving data chunks with the existing clustering solution, and they can integrate the knowledge from different views into a global model. These algorithms also provide flexibility in choosing which views to consider for building the global model. If three views are available, the algorithms can use all three or just two of them to build the global model. This type of functionality can be advantageous if information from any view is missing due to system or device failures or other similar reasons. Figure 5.2 visualizes a high-level overview of the functionality of the *MV Split-Merge Clustering* and MV Multi-Instance Clustering algorithms with three different views.

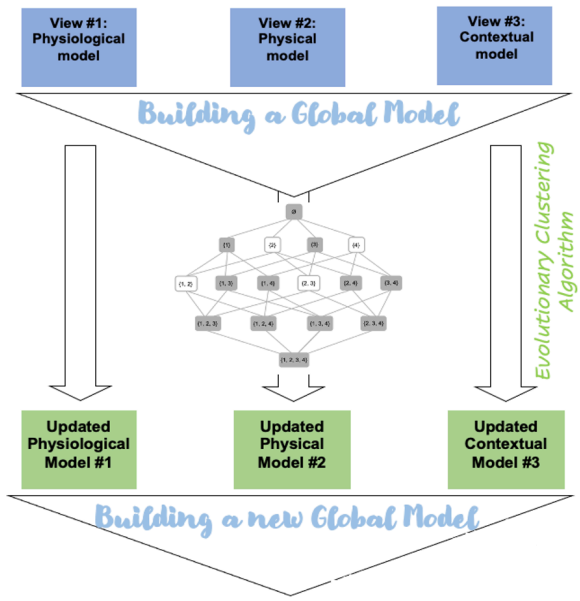


Figure 5.2: High-level overview of the functionality of *MV Split-Merge Clustering* and MV Multi-Instance Clustering algorithms where three different views are identified.

When compared to its batch version, the *MV Split-Merge Clustering* algorithm proposed in Paper II has produced comparable results. The algorithm proposed in Paper II is outperformed by *MV Multi-Instance Clustering* algorithm proposed in Paper III for the experiments conducted on the anthropometric data set. In addition, when evaluated on the real-world sensor data, the *MV Multi-Instance Clustering* algorithm (Paper III) is able to perform continuous monitoring, analysis, and mining of streaming data. When new data chunks arrive, the algorithm is able to perform stable integration. It is observed that some concepts from the previous chunk are re-

tained or expanded with new instances. The algorithm has also successfully detected deviating behavior that is already known to have occurred in the system during the considered time.

Paper IV is an applied paper where *MV Multi-Instance Clustering* is further evaluated in the smart building domain. The heating and tap-water sub-systems, which depend on each other, are studied in different scenarios. The study demonstrates the algorithm's potential in performing context-aware modeling and integrated system analysis of system behaviors. Various visualization and data mining techniques are proposed for each step of the algorithm, which can aid experts in step-by-step analysis and easy understandability of the results. The algorithm successfully detected some known and unknown deviating behaviors that have occurred in the system during the time considered for experimentation. The experimental results show the ability of *MV Multi-Instance Clustering* to monitor, analyze, and detect deviating behaviors of systems in the smart building domain. The obtained global model from different views depicted correlations between different views.

MST-MVS proposed in Paper V uses a different integration approach to obtain the global model. Cluster representatives from all the views are used in this process. The local clustering models of each view are evaluated before being used to build the global model. Only the views whose local clustering models have an SI score greater than the predefined threshold are used to build the global model. For each representative qualified to be used, its attribute values from other views are also extracted. An integrated matrix is built using this information (representatives of different clusters from each view and their corresponding attributes from other views). This is followed by obtaining a final global model by clustering the data points in the matrix using MST based clustering algorithm. The pre-evaluation of clustering models in each view is done to ensure that the quality of the local clustering model does not negatively affect the global model. It is important to maintain the quality of the global model as the knowledge obtained from this is transferred and used to seed the local clustering model of the next data chunks. A specifically devoted experiment has studied this. Figure 5.3 (taken from Paper V) illustrates how knowledge is transferred from one chunk to another in *MST-MVS* clustering algorithm.

MST-MVS has performed well on the synthetic data (Dim32) compared to other data sets considered. This behavior can be backed logically as the synthetic data is free from factors such as noise, outliers, and cluster overlap that could exist in real-world data sets. The results indicate that the transfer of knowledge from the global model to build the local clustering solutions of the next data chuck is beneficial in the considered experimental scenarios. That is, the quality of the local clustering solutions generated using the transferred knowledge is better than the one where this knowledge is not available. Apart from these, the study has also proposed a post-labeling technique to label each view's data points into different clusters obtained in the global model. This labeling technique has produced better results than the Convex Non-negative Matrix Factorization (CNMF) based labeling technique used

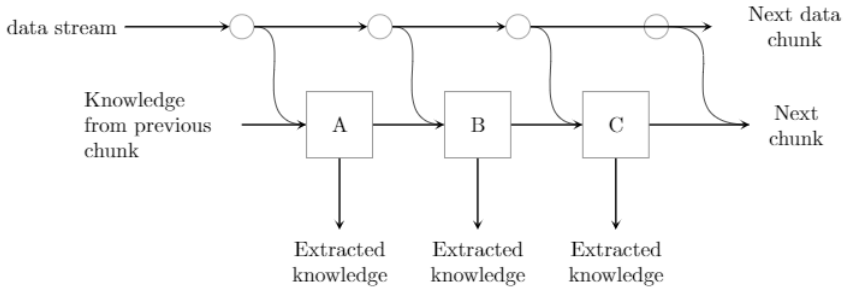


Figure 5.3: High-level overview of MST-MVS clustering algorithm for data chunks A, B, and C showcasing how knowledge from the global model of the previous chunk is used in building the clustering model of the new data chunk. Figure is taken from Paper V.

in the study.

Paper VIII proposes a data analysis approach for high-dimensional multi-source data with missing values. A multi-layered clustering approach followed by hypergraph clustering based on k-medoids and SNNS is used to achieve this. The approach is developed to be able to effectively use data with missing values, reducing information loss. This is done by using the layering approach, where each layer (similar to a view) contains a subset of features representing an aspect of the data set. Data objects having missing values in a layer are removed, but it can be noted that they are still being used in other layers where the features are available. It can be noted that the other multi-view clustering algorithms proposed in the study, except Paper V, also work with missing and heterogeneous data, but they are not evaluated concerning this aspect in those studies. The main purpose of the global models in these scenarios is to showcase the relations between the local models; this can not be achieved for the view with missing values. Figure 5.4 (taken from Paper VIII) presents the overview of the different steps of the proposed approach. The proposed approach is tested in a real-world use case related to condition monitoring of industrial assets, i.e., a fleet of compressors used in very different conditions, like factories and hospitals in this case. It is difficult to analyze and mine knowledge from such data as they have different technical specifications and other characteristics, thus making it difficult to compare. The approach helps to group these assets into homogeneous groups, thus enabling easier analysis. The study uses high-dimensional metadata with missing values to obtain these groups. The obtained groups are further validated using the time-series data generated during their operation.

Application Scenarios: The multi-source nature of data is common in fields like IoT, where data is collected using multiple sensors, or even in other scenarios where the data is collected from different sources. In this thesis, the developed data mining techniques are evaluated on real-world use cases in the areas of smart buildings and performance monitoring of industrial assets. It can further be used in domains like health care, where data is commonly heterogeneous and available in different formats

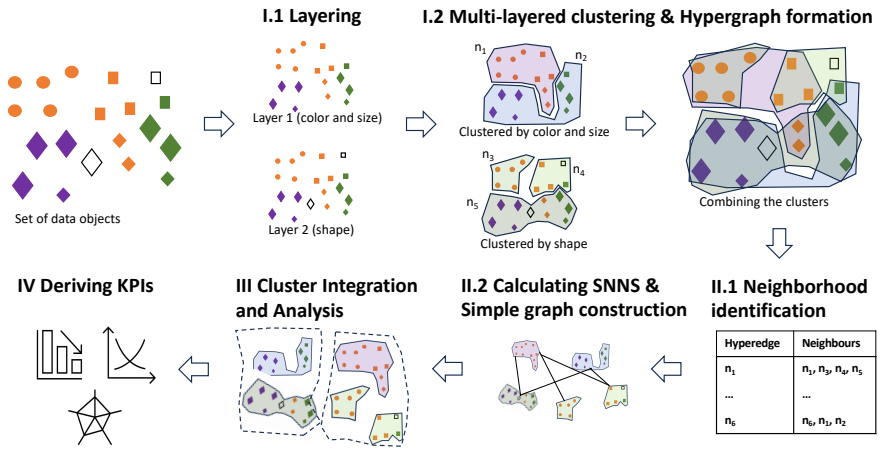


Figure 5.4: Overview of proposed data analysis approach for analyzing incomplete heterogeneous data. Figure is taken from Paper VIII.

like numerical data, images, etc. Data in this area is also prone to missing values for reasons such as manual entry. The available information (feature set) can be divided into various views, different learning algorithms suitable for the data in that view can be used, and then relationships between them or consensus knowledge can be obtained by building a global model.

5.3 Domain Adaptation

This thesis explores the area of unsupervised domain adaptation for time-series data in papers VI and VII. Novel domain adaptation algorithms based on cluster integration are proposed. Paper VI proposes *DIBCA*, which has been optimized to be robust to outliers in Paper VII with *DIBCA++*. The algorithms are developed to be able to transfer new knowledge between domains. Both algorithms adapt clustering solutions from different domains by identifying correlations across models and then performing cluster integration. Cross-labeling of models across domains (the model of one domain is applied to the representatives of the other domain) is used to identify the correlations. Two types of clusters are obtained from the algorithm: common clusters presenting the behavior present in both domains and private clusters presenting the behavior of their respective domain. An integrated model that could be used across domains is obtained by combining all the common and private clusters. Along with the integrated model, two personalized adapted models are obtained, one for each domain consisting of common clusters and their respective private clusters. Figure 5.5 (taken from Paper VI) presents the flowchart showing different steps of *DIBCA* and *DIBCA++*.

While the clustering algorithm used by *DIBCA* is based on ISM [96], which

uses the clusters' low-value and high-value vectors as representatives, *DIBCA++* is developed to be robust to outliers by using the clusters' mean, standard deviation, and size. This is because *DIBCA++* makes use of all the data instances to obtain the representatives, unlike *DIBCA* where only minimum and maximum values of each attribute are considered to obtain the representatives. Clustering done in *DIBCA++* is inspired from [97], but unlike it, where the data is assumed to be normally distributed, the proposed algorithm uses Chebyshev's inequality, which satisfies most distributions to determine the interval describing cluster boundaries. Both algorithms use only cluster representatives in their operations, thus making them resource-efficient and also preserving privacy as actual data points are not required to be disclosed.

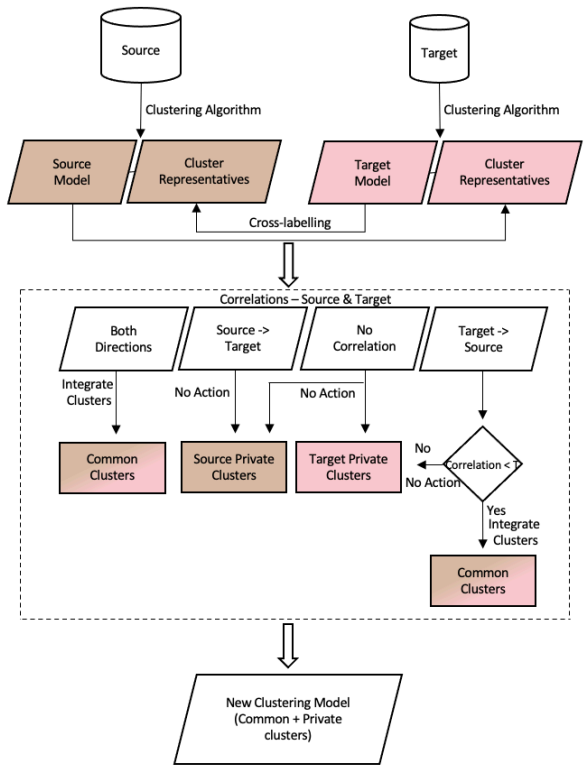


Figure 5.5: Overview of different steps of DIBCA and DIBCA++. Figure is taken from Paper VI.

DIBCA's potential in the data labeling task is evaluated using the PAMAP2 data set, where it has performed well and correctly labeled up to 91.1% of the clusters, and in the cases where the correct match is not found, they are not labeled. In the smart logistics domain, based on the obtained accuracy and F-measure values the performance of the integrated and adapted models is better or comparable to the original source and target models of the considered domain. The experimental results showcase the capability of *DIBCA++* to transfer knowledge between domains, leading to improved performance. In the smart logistics use case, the adapted source and tar-

get models produced by *DIBCA++* have performed better 70% and 80% of the time, respectively. Both the algorithms have been compared against each other using the smart logistics and DaLiAc data sets. The results demonstrate the better performance of *DIBCA++*.

Application Scenarios: Domain adaptation algorithms can be used in application scenarios where there is a requirement for the model to be adapted to be used in different (new) locations or by different people/customers that have different data characteristics from those the model has been exposed during the training. One example is the case of automatic vehicles, where the actual environment they would be used in might be different from what they have been trained in. For example, if the model is trained in a location with a hot climate, but the car needs to be used in a country with a cold climate and snowfall. The model must be adapted to the new circumstances to perform well, which can be done using domain adaptation. This thesis evaluates the proposed domain adaptation algorithms in the smart logistics use case to correctly identify and perform GNSS activation as required based on the tracker's current location. This helps reduce energy consumption and localize trackers with low energy capacities. The tracker can be used in various locations like cities and countryside with many differences like the number of cell towers, traffic, etc.; the model should be able to adapt to these new circumstances. We have also tested the algorithms in the case of HAR, where knowledge obtained from training the model on one person's data could be transferred to another.

5.4 Summary

This section presents the research questions formulated in this thesis and summarizes how each of these is addressed.

RQ1 *How can a clustering solution be updated to accommodate and catch evolving characteristics of continuously arriving data?*

In Papers I, II, III, and IV, the newly arriving data chunk is initially clustered. Then the existing clustering model is updated based on the newly arriving data by taking into account the correlations between the two clustering solutions. Therefore, the updated clustering solution is based on the data characteristics of both the existing clustering solution and the newly arriving data. Techniques like bipartite graphs and MI clustering are used to achieve this. Unlike these, the algorithm proposed in Paper V does not update the existing clustering solution when a new data chunk arrives. Instead, a new clustering solution is built for each data chunk based on the current data (landmark window model) and knowledge from the previous data chunk in the form of the artificial node used in the MST.

RQ2 *How can clustering models generated in a multi-view streaming context be*

integrated into a robust global model capturing knowledge from different views?

In Papers II, III, and IV, FCA integrates the clustering models from different views and builds a consensus clustering (global) model. The global model consists of a formal context and a concept lattice. While building the formal context, the data points are considered as the objects, and the cluster each data point belongs to in each view is represented as the object's properties. A novel integration approach has been presented in Paper V; the global model is obtained by using MST clustering on an integrated matrix. The integrated matrix is built using the representatives of each cluster from the local clustering models and their attributes from other views. It can be noted that only representatives of clusters that have passed the quality check are used. This approach successfully integrated information from different views and built a global model.

RQ3 *How can we develop a resource-efficient domain adaptation algorithm for a robust clustering model alignment to new domains?*

Algorithms *DIBCA* and *DIBCA++* are proposed to address this research question in papers VI and VII. These algorithms find correlations between the domains using cross-labeling (cluster representatives in one domain are labeled using the clustering model of the other domain), followed by obtaining common and private clusters based on the correlation obtained. Three different models are obtained using these clusters: one integrated model containing all the common and private clusters that could be used in both domains and two adapted (source/target) models for their respective domains. These algorithms use only cluster representatives in their operations, making them resource-efficient.

RQ4 *What kind of clustering analyses can be performed on heterogeneous multi-source data with missing values to produce useful homogeneous clusters?*

This research question is addressed in Paper VIII, where a novel hybrid clustering workflow is proposed. The approach uses multi-layered clustering to obtain a hypergraph, followed by hypergraph clustering using k-medoids and shared nearest-neighbor similarity. The multi-layered clustering assists in handling missing values, reducing information loss. This clustering in each layer is done separately, and data objects with missing values in a particular layer are removed but are still used in other layers where their features are available. Other multi-view clustering algorithms (except *MST-MVS*) proposed in this thesis are also capable of handling missing values, but this feature of the algorithm is not evaluated in these studies.

6 Conclusions and Future Work

This thesis has proposed and evaluated different clustering algorithms suitable for analyzing and mining evolving and heterogeneous data. Through the various phases (different papers) of the thesis, we have worked on improving the algorithms' robustness by dealing with new challenges not addressed in the earlier versions. The research results reported in this thesis have been identified to contribute to three research areas: evolving clustering, multi-source data analysis and domain adaptation. The results and contributions are summarized below.

Evolving Clustering

- Novel evolving clustering algorithms *Split-Merge Evolutionary Clustering* (Paper I), *Bi-Correlation MI-Clustering* (Paper III) and *MST-MVS* (Paper V, also capable of handling multi-view data) are proposed and evaluated.
- Results show the ability of the proposed algorithms to integrate newly arriving data chunks into the existing clustering model.

Multi-Source Data Analysis

- Multi-view clustering algorithms *MV Split-Merge Clustering* (Paper II), *MV Multi-Instance Clustering* (Paper III), and *MST-MVS* (Paper V), which are also capable of handling evolving data, are proposed and studied.
- The clustering model and relationships between different views obtained by *MV Split-Merge Clustering* are comparable to those obtained in the batch scenarios.
- *MV Multi-Instance Clustering* uses Multi Instance clustering and closed patterns, which improves the performance and interpretability of the algorithm compared to *MV Split-Merge Clustering*.
- *MV Multi-Instance Clustering* is evaluated in an industrial use case of a smart building, and the results show the algorithm's potential in monitoring and performance analysis of the system.

- *MST-MVS* is capable of transferring knowledge between consecutive data chunks, which is also enriched with a post-clustering pattern-labeling procedure. The knowledge transfer is shown to have a positive impact on the generated clustering solutions.
- A novel hybrid cluster analysis technique suitable for heterogeneous data with many missing values is proposed and is evaluated in an industrial use case of performance monitoring of assets. It is demonstrated to be capable of producing homogeneous groups of assets, suitable for continuous monitoring and deviating behavior detection.

Domain Adaptation

- *DIBCA* and *DIBCA++* are proposed and evaluated in Papers VI and VII respectively.
- *DIBCA* has performed well in automatic data labeling task on a publicly available human activity recognition data set.
- *DIBCA++* is an optimized version of *DIBCA*. It has demonstrated its capability in transferring knowledge between the domains and the need/advantage for personalizing the models for each domain.

Future Work

As a part of future work, we would like to study, analyze, and test how the evolving clustering algorithms proposed/used in papers I, II, III, IV, and V perform on different types of concept drift. According to our initial analysis, all the algorithms developed in this thesis are naturally adaptive to concept drift scenarios. The algorithm automatically adapts to new data characteristics, making it difficult to detect the occurrence of concept drift. We plan to explore this aspect and work in the direction of being able to detect concept drifts. We also plan to simulate different concept drift scenarios and test the algorithm's performance in these different scenarios.

Explainability is an important feature to consider while developing machine learning algorithms to be able to be accepted and used in a wide range of applications, especially if the models will be applied in fields like medicine or law. Two of the algorithms included in the current thesis specially study and showcase the explainability aspect of our approaches (Papers IV and VII). Our future interests are also directed to the area of studying explainable AI solutions. We are interested in exploring whether the proposed domain adaptation approaches could be combined with deep learning models to understand whether such combinations would improve the explainability of transfer learning or domain adaptation models based on deep learning.

7 Experiences and Learning Outcomes

This PhD journey has been a great learning activity and has provided a lot of new experiences. It has helped me to grow as a researcher over time. During my PhD study, I have been involved in three different research projects and had the opportunity to work with different researchers, including practitioners from the industry. I also got a chance to go on a research visit to the EluciDATA lab, Sirris in Belgium. This has exposed me to the challenge of integrating into a new research environment and has further helped me create new international collaborations. The work done as a part of this research visit has been successfully published as Paper VIII. Through my PhD, I got an opportunity to work on different interesting research problems, which are also very relevant for different application scenarios in the industry. I worked with real-world data obtained from our industrial partners in domains like smart buildings, smart logistics, and performance monitoring of industrial assets. This has provided us an opportunity to test how our proposed algorithm works on real-world data. While working with real-world data, I understood the importance of knowing the system and data well to be able to use the developed clustering models successfully. I acted as a co-supervisor for a master's thesis student, which has led to a successful publication (Paper V). It was a different experience being on the other side and added a new responsibility in guiding the student to conduct the research.

A part of the PhD was done during the COVID pandemic. The pandemic has completely changed everything and was unpredictable. Just like everyone, we were forced to adapt to new situations. Working from home, virtual meetings and conferences have become a new normal. During these tough times, I have learned to adapt to new situations and deliver. This period demanded a lot of motivation. One of the major fears that I partially overcame in this journey is public speaking. Over these last few years, there have been numerous occasions, like internal monthly seminars within our research group, workshops, and conferences where I had to present my research work. These events have helped me to overcome my fears to a certain extent. Writing has not been my forte, but I have come a long way in this as well.

Bibliography

[1] A. Bifet, B. Hammer, and F. Schleif. “Recent trends in streaming data analysis, concept drift and analysis of dynamic data sets”. In: *27th European Symposium on Artificial Neural Networks, ESANN 2019, Bruges, Belgium, April 24-26, 2019*. 2019.

[2] M. Mohammed, M. B. Khan, and E. B. M. Bashier. *Machine learning: algorithms and applications*. Crc Press, 2016.

[3] I. H. Sarker. “Machine Learning: Algorithms, Real-World Applications and Research Directions”. In: *SN computer science* 2.3 (2021), pp. 160–160. DOI: 10.1007/s42979-021-00592-x.

[4] D. Xu and Y. Tian. “A Comprehensive Survey of Clustering Algorithms”. In: *Annals of Data Science* 2 (Aug. 2015), pp. 165–193. DOI: 10.1007/s40745-015-0040-1.

[5] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. English. Englewood Cliffs, NJ: Prentice Hall, 1988, pp. xiv + 320. ISBN: 0-13-022278-X.

[6] A. Zubaroglu and V. Atalay. “Data stream clustering: a review”. In: *Artificial Intelligence Review* 54.2 (2021), pp. 1201–1236. DOI: 10.1007/s10462-020-09874-x.

[7] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia. “A Brief Review of Domain Adaptation”. In: *Advances in Data Science and Information Engineering*. Ed. by R. Stahlbock, G. M. Weiss, M. Abou-Nasr, C.-Y. Yang, H. R. Arabnia, and L. Deligiannidis. Cham: Springer International Publishing, 2021, pp. 877–894. ISBN: 978-3-030-71704-9.

[8] A. S. Iwashita and J. P. Papa. “An Overview on Concept Drift Learning”. In: *IEEE Access* 7 (2019), pp. 1532–1547. DOI: 10.1109/ACCESS.2018.2886026.

[9] B. Jiang and et al. “Evolutionary multi-objective optimization for multi-view clustering”. In: *2016 IEEE CEC 2016*. 2016, pp. 3308–3315.

[10] Y. Yang and H. Wang. “Multi-view clustering: A survey”. In: *Big Data Mining and Analytics* 1.2 (June 2018), pp. 83–107. ISSN: 2096-0654.

[11] L. Fu, P. Lin, A. V. Vasilakos, and S. Wang. “An overview of recent multi-view clustering”. In: *Neurocomputing* 402 (2020), pp. 148–161. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.02.104.

[12] M. Carnein and H. Trautmann. “Optimizing Data Stream Representation: An Extensive Survey on Stream Clustering Algorithms”. In: *Business & Information Systems Engineering* (June 2019), pp. 277–297. DOI: 10.1007/s12599-019-00576-5.

[13] L. Huang and et al. “MVStream: Multiview Data Stream Clustering”. In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3482–3496.

[14] W. Shao and et al. “Online multi-view clustering with incomplete views”. In: *2016 IEEE Int. Conf. on Big Data (Big Data)*. 2016, pp. 1012–1017.

[15] V. Wenz, A. Kesper, and G. Taentzer. “Clustering Heterogeneous Data Values for Data Quality Analysis”. In: *J. Data and Information Quality* 15.3 (2023).

[16] D. Gamberger et al. “Multilayer Clustering: A Discovery Experiment on Country Level Trading Data”. In: *Discovery Science*. Springer Int. Publ., 2014, pp. 87–98.

[17] G. Pio, F. Serafino, D. Malerba, and M. Ceci. “Multi-type clustering and classification from heterogeneous networks”. In: *Information Sciences* 425 (2018), pp. 107–126.

[18] M. C. de Goeij, M. van Diepen, K. J. Jager, G. Tripepi, C. Zoccali, and F. W. Dekker. “Multiple imputation: dealing with missing data”. In: *Nephrology Dialysis Transplantation* 28.10 (2013), pp. 2415–2420.

[19] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76. DOI: 10.1109/JPROC.2020.3004555.

[20] M. AlShehhi, E. Damiani, and D. Wang. “Toward Domain Adaptation for small data sets”. In: *Internet of Things* 16 (2021), p. 100458. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2021.100458.

[21] G. Csurka. *Domain adaptation in computer vision applications*. Cham: Springer International Publishing, 2017.

[22] J. Maia, C. A. Severiano, F. G. Guimarães, C. L. de Castro, A. P. Lemos, J. C. Fonseca Galindo, and M. Weiss Cohen. “Evolving clustering algorithm based on mixture of typicalities for stream data mining”. In: *Future Generation Computer Systems* 106 (2020), pp. 672–684. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.01.017. URL: https://www.sciencedirect.com/science/article/pii/S0167739X19312786.

[23] D. Kangin and P. Angelov. “Evolving clustering, classification and regression with TEDA”. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280528.

[24] A. Bouchachia. “Evolving clustering: an asset for evolving systems”. In: *IEEE SMC Newsletters* 36 (2011).

[25] K. Wadewale and S. Desai. “Survey on Method of Drift Detection and Classification for time varying data set”. In: *Int. Res. J. Eng. Technol.* Vol. 2. 2015, pp. 709–713.

[26] S. Agrahari and A. K. Singh. “Concept Drift Detection in Data Stream Mining : A literature review”. In: *Journal of King Saud University - Computer and Information Sciences* 34.10, Part B (2022), pp. 9523–9540. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2021.11.006. URL: https://www.sciencedirect.com/science/article/pii/S1319157821003062.

[27] J. Read, R. A. Rios, T. Nogueira, and R. F. de Mello. “Data Streams Are Time Series: Challenging Assumptions”. In: *Intelligent Systems*. Ed. by R. Cerri and R. C. Prati. Cham: Springer International Publishing, 2020, pp. 529–543. ISBN: 978-3-030-61380-8.

[28] J. Read. *Concept-drifting Data Streams are Time Series; The Case for Continuous Adaptation*. 2018. arXiv: 1810.02266 [cs.LG].

[29] B. Ganter, G. Stumme, and R. Wille. “Formal Concept Analysis: Foundations and Applications”. In: LNAI, no. 3626, Springer-Verlag, 2005.

[30] V. Boeva and et al. “Analysis of Multiple DNA Microarray Datasets”. In: *Springer Handbook of Bio-/Neuroinformatics*. Ed. by N. Kasabov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 223–234. ISBN: 978-3-642-30574-0.

[31] A. Hristoskova, V. Boeva, and E. Tsiporkova. “A Formal Concept Analysis Approach to Consensus Clustering of Multi-Experiment Expression Data”. In: *BMC Bioinformatics* 15 (May 2014), p. 151.

[32] S. K. and A. K. Ch. “Concept Lattice Simplification in Formal Concept Analysis Using Attribute Clustering”. In: *Journal of Ambient Intelligence and Humanized Computing* 10 (2018), pp. 2327–2343. ISSN: 1868-5145.

[33] A. S. Asratian, T. M. J. Denley, and R. Häggkvist. “Introduction to bipartite graphs”. In: *Bipartite Graphs and their Applications*. Cambridge Tracts in Mathematics. Cambridge University Press, 1998, pp. 7–22.

[34] N. Ailon, N. Avigdor-Elgrabli, E. Liberty, and A. van Zuylen. “Improved Approximation Algorithms for Bipartite Correlation Clustering”. In: *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*. 2011, pp. 25–36.

[35] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. “Graph clustering and minimum cut trees”. In: *Internet Mathematics* 1.4 (2004), pp. 385–408.

[36] R. Görke, T. Hartmann, and D. Wagner. “Dynamic Graph Clustering Using Minimum-Cut Trees”. In: *Algorithms and Data Structures*. Ed. by F. Dehne, M. Gavrilova, J.-R. Sack, and C. D. Tóth. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 339–350.

[37] B. Saha and P. Mitra. “Dynamic Algorithm for Graph Clustering Using Minimum Cut Tree”. In: *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW’06)*. 2006, pp. 667–671.

[38] S. Schlag, T. Heuer, L. Gottesbüren, Y. Akhremtsev, C. Schulz, and P. Sanders. “High-Quality Hypergraph Partitioning”. In: *ACM J. Exp. Algorithmics* 27 (Feb. 2023).

[39] A. Bretto. *Hypergraph Theory: An Introduction*. English. 1;2013;2013.; vol. 11. Cham: Springer International Publishing AG, 2013. ISBN: 2192-4732.

[40] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou. “Hypergraph Learning: Methods and Practices”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.5 (2022), pp. 2548–2566. DOI: 10.1109/TPAMI.2020.3039374.

[41] J. Foulds and E. Frank. “A review of multi-instance learning assumptions”. In: *Knowledge Engineering Review* 25.1 (2010), pp. 1–25. DOI: 10.1017/S026988890999035X.

[42] M. Zhang and Z. Zhou. “Multi-instance clustering with applications to multi-instance prediction”. In: *Applied Intelligence* 31 (2009), pp. 47–68.

[43] S. Huang, Z. Xu, I. W. Tsang, and Z. Kang. “Auto-weighted multi-view co-clustering with bipartite graphs”. In: *Information Sciences* 512 (2020), pp. 18–30. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.09.079.

[44] M. Bendechache and M.-T. Kechadi. “Distributed clustering algorithm for spatial data mining”. In: *2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM)*. IEEE. 2015, pp. 60–65.

[45] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, and J. Gama. “Data Stream Clustering: A Survey”. In: *ACM Comput. Surv.* 46.1 (July 2013). ISSN: 0360-0300. DOI: 10.1145/2522968.2522981. URL: https://doi-org.miman.bib.bth.se/10.1145/2522968.2522981.

[46] M. Ghesmoune, M. Lebbah, and H. Azzag. “State-of-the-art on clustering data streams”. In: *Big Data Analytics* 1.1 (2016), pp. 1–27.

[47] D. Chakrabarti, R. Kumar, and A. Tomkins. “Evolutionary clustering”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 554–560.

[48] E. Lughofer. “A dynamic split-and-merge approach for evolving cluster models”. In: *Evolving Systems* 3.3 (Sept. 2012), pp. 135–151.

[49] R. Fa and A. K. Nandi. “Smart: Novel self splitting-merging clustering algorithm”. In: *European Signal Processing Conference, Bucharest, Romania, August, 27-32*. IEEE, 2012.

[50] M. Wang, V. Huang, and A.-M. C. Bosneag. “A Novel Split-Merge-Evolve k Clustering Algorithm”. In: *IEEE 4th International Conference on Big Data Computing Service and Applications (BigDataService), Bamberg, Germany, March 26-29*. 2018.

[51] S. Wang and et al. “Multi-view Clustering via Late Fusion Alignment Maximization”. In: *Proceedings of IJCAI-19*. July 2019, pp. 3778–3784.

[52] C. Zhu. “Kappa Based Weighted Multi-View Clustering with Feature Selection”. In: *Proceedings of ICCPR 2018*. ICCPR ’18. Shenzhen, China, 2018, pp. 50–54. ISBN: 978-1-4503-6471-3.

[53] X. Liu and et al. “Late Fusion Incomplete Multi-View Clustering”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 41.10 (2019), pp. 2410–2423.

[54] Y. Ye and et al. “Incomplete Multiview Clustering via Late Fusion”. In: *Computational Intelligence and Neuroscience* 2018 (Oct. 2018), pp. 1–11.

[55] M. Yang et al. “Robust multi-view clustering with incomplete information”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 45.1 (2022), pp. 1055–1069.

[56] J. Chen, S. Yang, and Z. Wang. “Multi-view representation learning for data stream clustering”. In: *Information Sciences* 613 (2022), pp. 731–746. DOI: 10.1016/j.ins.2022.09.045.

[57] S. J. Pan and Q. Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[58] Y. Madadi, V. Seydi, K. Nasrollahi, R. Hossieni, and T. Moeslund. “Deep Visual Unsupervised Domain Adaptation for Classification Tasks: A Survey”. In: *IET Image Processing* 14.14 (2020), pp. 3283–3299. ISSN: 1751-9659. DOI: 10.1049/iet-ipr.2020.0087.

[59] S. Hundschell, M. Weber, and P. Mandl. “An Empirical Study of Adversarial Domain Adaptation on Time Series Data”. In: *Artificial Intelligence and Soft Computing*. Ed. by L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada. Cham: Springer International Publishing, 2023, pp. 39–50. ISBN: 978-3-031-23492-7.

[60] G. Li, G. Kang, Y. Zhu, Y. Wei, and Y. Yang. “Domain Consensus Clustering for Universal Domain Adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 9757–9766.

[61] J. Li, G. Li, Y. Shi, and Y. Yu. “Cross-domain adaptive clustering for semi-supervised domain adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2505–2514. DOI: 10.1109/CVPR46437.2021.00253.

[62] S. Tang, Y. Zou, Z. Song, J. Lyu, L. Chen, M. Ye, S. Zhong, and J. Zhang. “Semantic consistency learning on manifold for source data-free unsupervised domain adaptation”. In: *Neural Networks* 152 (2022), pp. 467–478. ISSN: 0893-6080.

[63] M. Zhu. “Source Free Domain Adaptation by Deep Embedding Clustering”. In: *2021 18th Int. Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. 2021, pp. 309–312. DOI: 10.1109/ICCWAMTIP53232.2021.9674068.

[64] W. Menapace, S. Lathuilière, and E. Ricci. “Learning to Cluster Under Domain Shift”. In: *Computer Vision – ECCV 2020*. Ed. by A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm. Cham: Springer International Publishing, 2020, pp. 736–752. ISBN: 978-3-030-58604-1.

[65] J. A. Blackard, D. J. Dean, and C. W. Anderson. *UCI Machine Learning Repository*. 1998. URL: http://archive.ics.uci.edu/ml.

[66] K. Nakai and M. Kanehisa. “Expert Sytem for Predicting Protein Localization Sites in Gram-Negative Bacteria”. In: *PROTEINS: Structure, Function, and Genetics* 11 (1991), pp. 95–110.

[67] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reisa. “Modeling wine preferences by data mining from physicochemical properties”. In: *Decision Support Systems* 47.4 (2009), pp. 547–553.

[68] H. F. Golino, L. S. de Brito Amaral, S. F. P. Duarte, and et al. “Predicting Increased Blood Pressure Using Machine Learning”. In: *Journal of Obesity* 2014 (2014).

[69] P. Fränti and S. Sieranoja. *K-means properties on six clustering benchmark datasets*. 2018. URL: http://cs.uef.fi/sipu/datasets/.

[70] A. Reiss and D. Stricker. *Introducing a New Benchmarked Dataset for Activity Monitoring*. 2012. URL: https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring.

[71] H. Leutheuser, D. Schuldhaus, and B. M. Eskofier. "Hierarchical, multi-sensor based classification of daily life activities: comparison with state-of-the-art algorithms using a benchmark dataset". In: *PloS one* 8.10 (2013), e75196. DOI: 10.1371/journal.pone.0075196.

[72] V. Kumar, J. K. Chhabra, and D. Kumar. "Impact of distance measures on the performance of clustering algorithms". In: *Advances in Intelligent Systems and Computing* 243 (2014), pp. 183–190. DOI: 10.1007/978-81-322-1665-0_17.

[73] R. Liu, H. Wang, and X. Yu. "Shared-Nearest-Neighbor-Based Clustering by Fast Search and Find of Density Peaks". In: *Inf. Sci.* 450.C (June 2018), pp. 200–226.

[74] R. W. Hamming. "Error detecting and error correcting codes". English. In: *Bell System Technical Journal* 29.2 (1950), pp. 147–160.

[75] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. "Experimental comparison of representation methods and distance measures for time series data". In: *Data mining and knowledge discovery* 26.2 (2013), pp. 275–309.

[76] J. Handl, J. Knowles, and D. Kell. "Computational cluster validation in post-genomic data analysis". In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.

[77] H. Van der Hoef and M. J. Warrens. "Understanding information theoretic measures for comparing clusterings". In: *Behaviormetrika* 46 (2019), pp. 353–370.

[78] P. J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65.

[79] D. L. Davies and D. W. Bouldin. "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227.

[80] T. Caliński and J. Harabasz. "A dendrite method for cluster analysis". In: *Communications in Statistics-theory and Methods* 3 (1974), pp. 1–27.

[81] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of Classification* 2.1 (Dec. 1985), pp. 193–218.

[82] W. M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. ISSN: 01621459.

[83] B. Larsen and C. Aone. “Fast and Effective Text Mining Using Linear-time Document Clustering”. In: *Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD’99. ACM, 1999, pp. 16–22.

[84] P. Jaccard. “The distribution of flora in the alpine zone”. In: *New Phytologist* 11 (1912), pp. 37–50.

[85] N. X. Vinh, J. Epps, and J. Bailey. “Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?” In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML’09. Montreal, Quebec, Canada, 2009, pp. 1073–1080.

[86] N. X. Vinh, J. Epps, and J. Bailey. “Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance”. In: *Journal of Machine Learning Research* 11.95 (2010), pp. 2837–2854. URL: http://jmlr.org/papers/v11/vinh10a.html.

[87] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[88] A. Rosenberg and J. Hirschberg. “V-measure: A conditional entropy-based external cluster evaluation measure”. In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2007, pp. 410–420.

[89] “Developing your Objectives and Choosing Methods”. In: *Thesis Projects: A Guide for Students in Computer Science and Information Systems*. London: Springer London, 2008, pp. 54–70. ISBN: 978-1-84800-009-4. DOI: 10.1007/978-1-84800-009-4_8. URL: https://doi.org/10.1007/978-1-84800-009-4_8.

[90] U. von Luxburg, R. C. Williamson, and I. Guyon. “Clustering: Science or Art?” In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Vol. 27. Proceedings of Machine Learning Research. 2012, pp. 65–79.

[91] M. Moshtaghi, J. C. Bezdek, S. M. Erfani, C. Leckie, and J. Bailey. “Online cluster validity indices for performance monitoring of streaming data clustering”. In: *International Journal of Intelligent Systems* 34.4 (2019), pp. 541–563.

[92] W. R. Shadish, T. D. Cook, and D. T. Campbell. *Experimental and quasi-experimental designs for generalized causal inference*. Houghton, Mifflin and Company, 2002.

[93] R. Feldt and A. Magazinius. “Validity Threats in Empirical Software Engineering Research - An Initial Survey”. In: *Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE’2010), Redwood City, San Francisco Bay, CA, USA, July 1 - July 3, 2010*. Knowledge Systems Institute Graduate School, 2010, pp. 374–379.

[94] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. “Planning”. In: *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 89–116. ISBN: 978-3-642-29044-2. DOI: 10.1007/978-3-642-29044-2_8. URL: https://doi.org/10.1007/978-3-642-29044-2_8.

[95] M. Bahri, A. Bifet, J. Gama, H. Gomes, and S. Maniu. “Data stream analysis: Foundations, major tasks and tools”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11.3 (2021). DOI: 10.1002/widm.1405.

[96] D. L. Iverson. “Inductive System Health Monitoring.” In: *IC-AI*. 2004, pp. 605–611.

[97] P. Davidsson. “Coin Classification Using a Novel Technique for Learning Characteristic Decision Trees by Controlling the Degree of Generalization”. In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. 1996.

Paper I

Bipartite Split-Merge Evolutionary Clustering

Veselka Boeva, Milena Angelova, Vishnu Manasa Devagiri, Elena Tsiporkova
In: Agents and Artificial Intelligence, Ed. by J. van den Herik, A. P. Rocha, and L. Steels. Cham: Springer International Publishing, 2019, pp. 204–223, DOI: 10.1007/978-3-030-37494-5_11

Abstract

We propose a split-merge framework for evolutionary clustering. The proposed clustering technique, entitled Split-Merge Evolutionary Clustering is supposed to be more robust to concept drift scenarios by providing the flexibility to consider at each step a portion of the data and derive clusters from it to be used subsequently to update the existing clustering solution. The proposed framework is built around the idea to model two clustering solutions as a bipartite graph, which guides the update of the existing clustering solution by merging some clusters with ones from the newly constructed clustering while others are transformed by splitting their elements among several new clusters. We have evaluated and compared the discussed evolutionary clustering technique with two other state of the art algorithms: a bipartite correlation clustering (PivotBiCluster) and an incremental evolving clustering (Dynamic split-and-merge).
Keywords: Data mining, Dynamic clustering, Evolutionary clustering, Bipartite clustering, Split-merge framework, Unsupervised learning.

1 Introduction

The problem addressed in this article deals with the development of evolutionary clustering algorithm that can be used to (continuously) adjust existing clustering solution to match newly arrived data. For example, in many real-world applications such as personalizing customer recommendations, the information available in the system database is periodically updated by collecting new data. The available data

elements, e.g., customers of a retailing company, are usually partitioned into a number of segments (clusters of customers with similar product preferences). As the data increases we need to re-group existing data and also accommodate new customers in the existing customer segments. However, the existing original segments (clusters) can become outdated due to shifts in preferences and characteristics of the newly attracted customers. Another example is profiling of users with wearable applications with the purpose to provide personalized recommendations. As more users get involved one needs to update the division of the initial set of users into groups of characteristic profiles and also assign new incoming users to these groups.

In the context of profiling of machines (industrial assets) for the purpose of condition (health) monitoring the existing original clusters can become outdated caused by aging of the machines and degradation of performance due to influence of changing external factors. This gradual or abrupt (e.g. due to software update) model invalidation is in fact known as a concept drift and requires that the clustering techniques, used for deriving the original machine profiles, can deal with such a concept drift and enable reliable and scalable model update.

Evolving clustering models are good candidate to tackle concept drift scenarios as discussed above. They have been designed to mine very large datasets or online continuous data streams [1] in an unsupervised learning context by grouping and summarizing data in a fast incremental manner. Evolving clustering models are also referenced as incremental or evolving (dynamic) clustering methods, because they can process data step-wise and update and evolve cluster partitions in incremental learning steps [2]. Incremental clustering methods process one data element at a time and maintain a good solution by either adding each new element to an existing cluster or placing it in a new singleton cluster while two existing clusters are merged into one [3], [4], [5]. Incremental algorithms also bear a resemblance to one-pass stream clustering algorithms [6]. Although, one-pass stream clustering methods address the scalability issues of the clustering problem, they are not sensitive to the evolution of the data, because they assume that the clusters are to be computed over the entire data stream. This implies that changes in the characteristic of newly arriving data are not well reflected while building the clustering solution.

Dynamic clustering is also a form of online/incremental unsupervised learning. However, it considers not only the incremental fashion of building the clustering model, but also self-adaptation of the built model. In that way, the incremental model construction deals with the problem of model re-training over time and memory constrains, while dynamic aspects (e.g., data behavior, clustering structure) of the model to be built can be captured via adaptation of the current model. Notice that the dynamic (evolving) clustering paradigm is also close to the ideas of stream reasoning [7]. Stream reasoning studies the application of inference techniques to data streams to perform continuous reasoning tasks. The access to the stream is managed by creating time-dependent finite views over the streams (windows) over which the tasks are performed. Window contains a portion of the input streams, i.e. a set of timestamped

data items, that represents the data needed to solve the task at the current time instant.

The clustering scenario discussed in this work is different from the one treated by incremental clustering methods. Namely, we are interested in clustering techniques that enable to compute clusters on a new portion of data collected over a defined time period (window) and to update the existing clustering solution by the computed new one. Such an updating clustering should better reflect the current characteristics of the data by being able to examine clusters occurring in the considered time period and eventually capture interesting trends in the area. In [8], we have studied two different clustering algorithms to be suited for the discussed scenario: *PivotBiCluster* [9] and *Split-Merge Evolutionary Clustering*. Both algorithms are bipartite correlation clustering algorithms that do not need prior knowledge about the optimal number of clusters in order to produce a good clustering solution. Notice that in our considerations the input graph nodes of PivotBiCluster algorithm are clusters. In the final clustering generated by the PivotBiCluster algorithm some clusters are obtained by merging clusters from both side of the graph, i.e. some of existing clusters will be updated by some of the computed new ones. However, existing clusters cannot be split by the PivotBiCluster algorithm even the corresponding correlations with clusters from the newly extracted data elements reveal that these clusters are not homogeneous. This has motivated us to develop our *Split-Merge Evolutionary Clustering* algorithm that overcomes this disadvantage. Namely, our algorithm is able to analyze the correlations between two clustering solutions and based on the discovered patterns it treats the existing clusters in different ways. Thus some clusters will be updated by merging with ones from newly constructed clustering while others will be transformed by splitting their elements among several new clusters.

An interesting dynamic clustering algorithm which is also equipped with dynamic split-and-merge operations and which is dedicated to incremental clustering of data streams is proposed by Lughofer in [10]. We have found a resemble between this algorithm, entitled *Dynamic split-and-merge algorithm*, and our Split-Merge Evolutionary Clustering. Hence, in this study the Split-Merge Evolutionary Clustering and the PivotBiCluster are further evaluated and compared against the Dynamic split-and-merge algorithm in two different experiment scenarios. Compared to the previous paper [8], the bibliography and related work section have also been extended with more recent works on the studied problem. We have also added a discussion on the computational complexity of our Split-Merge Evolutionary Clustering algorithm.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 states the problem and briefly describes the PivotBiCluster and Dynamic split-and-merge algorithms. In addition, it introduces the proposed Bipartite Split-Merge Evolutionary Clustering technique. Section 4 gives an overview of the experimental setup. Section 5 discusses the results from the evaluation of the three clustering algorithms. Section 6 is devoted to conclusions and future work.

2 Related Work

The model of incremental algorithms for data clustering is motivated by practical applications where the demand sequence is unknown in advance and a hierarchical clustering is required. Incremental clustering methods process one data element at a time and maintain a good solution by either adding each new element to an existing cluster or placing it in a new singleton cluster while two existing clusters are merged into one [4]. Incremental algorithms also bear a resemblance to one-pass clustering algorithms for data stream problems [6]. Several incremental clustering techniques have been proposed in the past [11], [12], [13], [14]. Such algorithms need to maintain a substantial amount of information so that important details are not lost. For example, the algorithm in [6] is implemented as a continuous version of k-means algorithm which continues to maintain a number of cluster centers which change or merge as necessary throughout the execution of the algorithm.

To qualify the type of cluster structure present in data, Balcan introduced the notion of clusterability [15]. It requires that every element be closer to data in its own cluster than to other points. In addition, Balcan showed that the clusterings that adhere to this requirement are readily detected offline by classical batch algorithms. On the other hand, it was proven by Ackerman [3] that no incremental method can discover these partitions. Thus, batch algorithms are significantly stronger than incremental methods in their ability to detect cluster structure. This is mainly due to the fact that the latter methods consider incrementality by dealing with the problem of model re-training over time and memory constrains, but they are not robust to the model dynamics.

Dynamic clustering is also a form of incremental unsupervised learning. However, it considers not only incrementality of the methods to build the clustering model, but also self-adaptation of the built model. Lughofer has proposed an interesting dynamic clustering algorithm which is equipped with dynamic split-and-merge operations and which is also dedicated to incremental clustering of data streams [10]. In [16] similarly to the approach of Lughofer a set of splitting and merging action conditions are defined, where optional splitting and merging actions are only triggered during the iterative process when the conditions are met. Wang et al. also propose a split-merge-evolve algorithm for clustering data into k number of clusters [17]. This algorithm has the ability to optimize the clustering result in scenarios where new data samples may be added in to existing clusters. However, a k cluster output is always provided by the algorithm, i.e. it is also not sensitive to the evolution of the data. In general, incremental and one-pass stream clustering methods address the scalability issues of the clustering problem, but they are not sensitive to the evolution of the data because they assume that the clusters are to be computed over the entire data stream.

In [18] an adaptive clustering approach that can apply to re-cluster a set of previously clustered objects when the feature set characterizing the objects increases has been proposed. The authors have developed adaptive extensions for k-means and hi-

erarchical agglomerative clustering algorithms. Further it has been shown how these extensions can be used for adjusting a clustering, that was established by applying the corresponding non-adaptive clustering algorithm before the feature set changed [18]. Such adaptive clustering techniques could be necessary in some applied scenarios, e.g., in the expertise mining context when the recently gathered information reveals that some of the known experts have expanded their competence. However, in this case the clustering scenario will be different from one considered in [18], because usually the expert expertise profiles are not presented by fixed-length feature vectors. Moreover, not all expert profiles will be affected by this expansion.

Gionis et al. proposed an approach to clustering that is based on the concept of aggregation [19]. They are interested in a problem in which a number of different clusterings are given on some data set of elements. The objective is to produce a single clustering of the elements that agrees as much as possible with the given clusterings. Clustering aggregation provides a framework for dealing with a variety of clustering problems. For instance, it can handle categorical or heterogeneous data by producing a clustering on each available attribute and then aggregating the produced clusterings into a single result. Another possibility is to combine the results of several clustering algorithms applied on the same dataset etc. Clustering aggregation can be thought as a more general model of multi-view clustering proposed in [20]. The multi-view approach considers clustering problems in which the available attributes can be split into two independent subsets. A clustering is produced on each subset and then the two clusterings are combined into a single result. Consensus clustering algorithms deal with similar problems to those treated by clustering aggregation techniques. Namely, such algorithms try to reconcile clustering information about the same data set coming from different sources [21] or from different runs of the same algorithm [22]. The both clustering techniques are not suited for our scenario, since they are used to integrate a number of clustering results generated on one and the same data set.

The idea for the proposed *Split-Merge Evolutionary Clustering* algorithm is inspired by the work of Xiang et al. [23]. They have proposed a split-merge framework that can be tailored to different applications. The framework models two clusterings as a bipartite graph which is decomposed into connected components, and each component is further decomposed into subcomponents. Pairs of related subcomponents are then taken into consideration in designing a clustering similarity measure within the framework.

3 Methods and the Proposed Solution

3.1 Problem Description

Let us formalize the cluster updating problem we are interested in. We assume that X is the available set of data points and each data point is represented by a vector of attributes (features). In addition, the data points are partitioned into k groups, i.e. $C = \{C_1, C_2, \ldots, C_k\}$ is an existing clustering solution of X and each C_i ($i = 1, 2, \ldots, k$) can be considered as a disjoint cluster. In addition, a new set X' of recently collected data elements (instances) is created, i.e. $X \cap X'$ is an empty set. Each data point in X' is again represented by a list of attributes and $C' = \{C'_1, C'_2, \ldots, C'_{k'}\}$ is a clustering solution of X'. The objective is to produce a single clustering of $X \cup X'$ by combining C and C' in such a way that the obtained clustering realistically reflects the current distribution in the domain under interest.

3.2 Pivot Bi-Clustering Algorithm

Two existing correlation clustering techniques are suitable for the considered context: correlation clustering [24] and bipartite correlation clustering [9]. The latter algorithm seems to be better aligned to our clustering scenario. In Bipartite Correlation Clustering (BCC) a bipartite graph is given as input, and a set of disjoint clusters covering the graph nodes is output. Clusters may contain nodes from either side of the graph, but they may possibly contain nodes from only one side. A cluster is thought as a bi-clique connecting all the objects from its left and right counterparts. Consequently, a final clustering is a union of bi-cliques covering the input node set. We compare our Split-Merge Evolutionary Clustering algorithm described in Section 3.4 with *PivotBiCluster* realization of the BCC algorithm [9]. The PivotBiCluster algorithm is implemented according to the original description given in [9].

Notice that in our considerations the input graph nodes of the PivotBiCluster algorithm are clusters and in the final clustering some clusters are obtained by merging clusters (nodes) from both sides of the graph, i.e. some of the existing clusters will be updated by some of the computed new ones. However, existing clusters cannot be split by the PivotBiCluster algorithm even when the corresponding correlations with clusters from the new data elements reveal that these clusters are not homogeneous.

3.3 Dynamic Split-and-Merge Clustering Algorithm

The proposed algorithm, described in Section 3.4, is also compared with the Dynamic split-and-merge clustering algorithm proposed by Lughofer in [10]. The Dynamic split-and-merge algorithm of Lughofer can be used as an extension to any existing incremental and evolutionary clustering algorithm provided it stores details regarding cluster centers, spread, elements of a cluster [10]. Once the newly arriving data points are assigned to existing clusters by applying some incremental clustering algorithm,

all the modified clusters are then examined in order to identify whether they need to be split or merged. Optional splitting and merging actions are only triggered during the iterative process if predefined action conditions are met. For example, a cluster is merged with another existing cluster if both of them are homogeneous and the clusters touch or overlap with each other. Whereas a cluster is split into two if the quality criterion of the clustering solution after the split is better than that of before it.

Although, the dynamic split-and-merge algorithm addresses the clustering dynamics, it is not very sensitive to concept drift phenomenon, because it assigns the newly arriving data points to the existing clusters in an incremental way and then improves the clustering solution by either splitting or merging the modified clusters. In comparison our split-merge clustering technique provides the flexibility to compute clusters on a new portion of data collected over a defined time period and to update the existing clustering solution by the computed new one [8]. Such an updating clustering should better reflect the current characteristics of the data by being able to examine clusters occurring in the considered time period and eventually capture interesting trends in the area.

3.4 Bipartite Split-Merge Evolutionary Clustering Algorithm

In this paper, we propose an evolutionary clustering algorithm that overcomes the above mentioned disadvantage of the two discussed state of the art algorithms. Namely, our algorithm is able to analyze the correlations between two clustering solutions C and C' and based on the discovered patterns it treats the existing clusters ($C = \{C_1, C_2, \ldots, C_k\}$) in different ways. Thus, some clusters will be updated by merging with ones from newly constructed clustering (C') while others will be transformed by splitting their elements among several new clusters. One can find some similarity between our idea and an interactive clustering model proposed in [25]. In this model, the algorithm starts with some initial clustering of data and the user may request a certain cluster to be split if it is *overclustered* (intersects two or more clusters in the target clustering). The user may also request to merge two given clusters if they are *underclustered* (both intersect the same target cluster).

As it was already mentioned in Section 2 our evolutionary clustering algorithm is inspired by a split-merge framework proposed by Xiang et al. in [23]. By modeling the intrinsic relation between two clusterings as a bipartite graph, they have designed a split-merge framework that can be used to obtain similarity measures to compare clusterings on different data sets. The problem addressed in this article is different from the one considered by Xiang et al. [23]. Namely, we concern with the development of split-merge framework that can be used to adjust the existing clustering solution to newly arrived data. Our framework also models two clusterings (the existing and the newly constructed one) as a bipartite graph which is decomposed

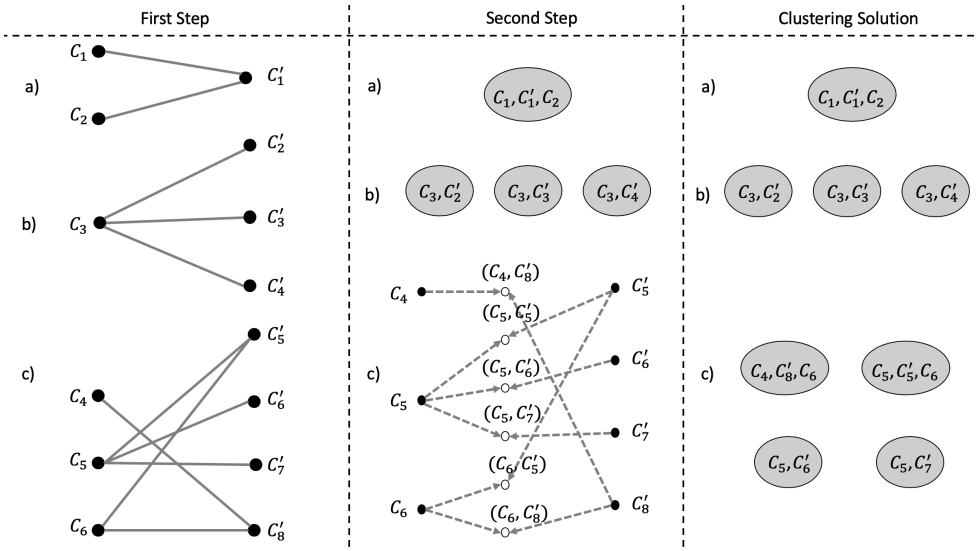


Figure 1: Split-Merge Framework: a) a bi-clique that contains underclustered nodes (C_1 and C_2 intersect C'_1); b) a bi-clique that contains an overclustered node (C_3 intersects C'_2, C'_3 and C'_4); c) a bi-clique that has to be decomposed into subcomponents in the second step of the algorithm. It is transformed into a tripartite graph that has split (left) and merge (right) subcomponents.

into connected components (bi-cliques) (see Fig. 1 (a), (b) and (c)). Each component is further analysed and if it is necessary it is decomposed into subcomponents (see Fig. 1 (c)). The subcomponents are then taken into consideration in producing the final clustering solution. For example, if an existing cluster is *overclustered* (Fig. 1 (b)), i.e. it intersects two or more clusters in the new clustering, it is split between those. If several existing clusters intersect the same new cluster, i.e. they are *underclustered* (Fig. 1 (a)), they are merged with that cluster. Notice that in comparison with the dynamic split-and-merge algorithm of Lughofer [10], the splitting and merging operations of our algorithm can be conducted on more than two clusters.

Let us formally describe the proposed Split-Merge Evolutionary Clustering algorithm. The input bipartite graph is $G = (C, C', E)$, where C and C' are sets of clusters of left and right nodes and E is a subset of $C \times C'$ that represents correlations between the nodes of two sets. The two main steps of the algorithm are as follows:

1. At the first step, all bi-cliques of G are found. Then we consider and treat three different scenarios:

   (i) If a bi-clique is an unreachable node it is made a singleton in the final clustering solution.

   (ii) If a bi-clique connects a node from the left side of G with several nodes from C' the elements of this node are split among the corresponding nodes from C' (see Fig. 1 (b)).

(iii) In the opposite case, i.e., when we have a bi-clique that connects a node from the right side of G with several nodes from left those nodes have to be merged with that node (cluster) (see Fig. 1 (a)).

All clustered nodes are removed from the graph.

2. At the second step, the remained bi-cliques are decomposed into split/merge subcomponents. Each bi-clique, which is a bipartite graph, is transformed into a tripartite graph constructed by two (split and merge) bipartite graphs. Suppose $G_i = (C_i, C'_i, E_i)$ is the considered bi-clique. Then the corresponding tripartite graph is built by the following two bipartite graphs: $G_{iL} = (C_i, E_i, E_{iL})$ and $G_{iR} = (E_i, C'_i, E_{iR})$, where C_i, C'_i and E_i are ones from G_i, E_{iL} is a subset of $C_i \times E_i$ that represents correlations between the nodes of C_i and E_i, and E_{iR} is a subset of $E_i \times C'_i$ representing correlations between the nodes of E_i and C'_i (see Fig. 1 (c)). For example, $c_i \in C_i$ will be correlated with all pairs $(c_j, c'_k) \in E_i$ such that $c_i \equiv c_j$, and $c'_i \in C'_i$ will be correlated with all pairs $(c_j, c'_k) \in E_i$ such that $c'_i \equiv c'_k$. Then splitting and merging sub-steps are sequentially conducted:

   (i) First all *overclustered* nodes of G_{iL} are *split* and new temporary clusters are formed as a result. This can be implemented, e.g., by calculating the distance between each data point of the overclustered node from C and the centroids of its adjacent nodes (cluster) from C'. Then the data point in question is assigned to the closest cluster.

   (ii) Then we perform the corresponding *merging* for all *underclustered* nodes in G_{iR}.

   For example, in Fig. 1 (c) cluster C_5 will first be split among clusters C'_5, C'_6 and C'_7, i.e. three new clusters, denoted by (C_5, C'_5), (C_5, C'_6) and (C_5, C'_7), will be obtained. Then at the next step of the algorithm clusters (C_5, C'_5) and (C_6, C'_5) will be merged together.

The pseudocode of the proposed *Split-Merge Evolutionary Clustering* algorithm is given in Algorithm I.1. In addition, the algorithm is illustrated with an example in Fig. 2. The clustering solution generated by the Split-Merge Clustering is compared to one produced by the PivotBiCluster. It is interesting to notice that the two algorithms will produce very different clustering solutions on the same input graph. For example, the Split-Merge Clustering will generate a 4-cluster solution while one obtained by the PivotBiCluster will have only 2 clusters. The latter number is quite low taking into account the number of clusters in the two input clusterings. Moreover, as it was mentioned in the previous section the PivotBiCluster algorithm cannot produce a clustering solution in which existing clusters are split among new clusters.

Algorithm I.1 Split-Merge Evolutionary Clustering Algorithm

1: **function** SPLIT-MERGE($G = (C, C', E)$)
2: **for** all nodes $c \in C \cup C'$ **do** (*First step*)
3: **if** c is an unreachable node **then**
4: Turn c into a singleton and remove it from G (*First step (i)*)
5: **end if**
6: **end for**
7: **for** all nodes $c \in C \cup C'$ **do**
8: **if** c_1 is the only node from C that takes part in a bi-clique connecting it with one or several nodes from C' **then**
9: Split c_1 among the corresponding nodes from C' (*First step (ii)*)
10: Remove the clustered nodes from G
11: **end if**
12: **end for**
13: **for** all nodes $c \in C \cup C'$ **do**
14: **if** c'_1 is the only node from C' that takes part in a bi-clique connecting it with one or several nodes from C **then**
15: Merge c'_1 with the corresponding nodes from C (*First step (iii)*)
16: Remove the clustered nodes from G
17: **end if**
18: **end for**
19: **for** all nodes $c \in C$ **do** (*Second step*)
20: Split c_1 among its adjacent nodes from C' and form new temporary clusters (*Second step (i)*)
21: **end for**
22: **for** all nodes $c' \in C'$ **do**
23: Merge c'_1 with its adjacent nodes from the built set of temporary clusters (*Second step (ii)*)
24: Remove the clustered nodes from G
25: **end for**
26: **return** all connected components (bi-cliques) as clusters of $X \cup X'$
27: **end function**

We now discuss the computational complexity of the Split-Merge Evolutionary Clustering. Suppose that n is the number of instances in the existing data set and n' ($n' < n$) is the number of instances in the new data set. In addition, we assume that the instances of the existing data set have already been grouped in k ($k << n$) categories. Initially, the new data elements have to be clustered into k' ($k' << n'$) clusters. The computational complexity of this part depends on the used clustering algorithm. It will be $O(n'k'mi)$ in case of k-means clustering algorithm [26], where m and i are the dimensionality of the learning problem and the number of iterations, respectively. According to Gan et al. [27], k-means usually converges quickly, i.e. the number of iterations is usually low and the algorithm complexity can be reduced to $O(n'k'm)$. In order to build the bipartite graph we calculate the similarity between the centroids of each pair of clusters belonging to $C \times C'$. Any pair of clusters which centroids' similarity is above a given threshold are considered connected by an edge. Hence, the computational complexity of building the bipartite graph is equal to $O(kk')$. We further focus our discussion on the computational complexity of the main steps of our algorithm, given in Algorithm I.1. The first part of the algorithm (steps 2 to 18) requires execution time that is proportional to $k + k'$. The computational complexity of the remainder part of the algorithm (from step 19 downwards) depends on the average size of clusters that have to be split. Suppose that l ($l << n$) is the average number of instances in those clusters. Then the computational complexity of this part can be approximated to $O((k + k')l)$. Finally, the total computational

complexity of the Split-Merge Evolutionary Clustering is $O(n'k'm+kk'+(k+k')+(k+k')l)$ and $n' >> (k+k')$, i.e. it can be simplified to $O(n'k'm+kk'+(k+k')l)$. In addition, $l >> k'$, i.e. we can further simplify to $O(n'k'm+(k+k')l)$ and finally reduce to $O(n'(k+k')m)$, as $n' >> l$. The latter expression is very close to the computational complexity of the Dynamic split-and-merge algorithm evaluated to $O(n'km)$ in [10]. The complexity of PivotBiCluster, commented in [9], cannot be directly compared to the complexity of our algorithm, since the former algorithm is not originally defined to work with clusters.

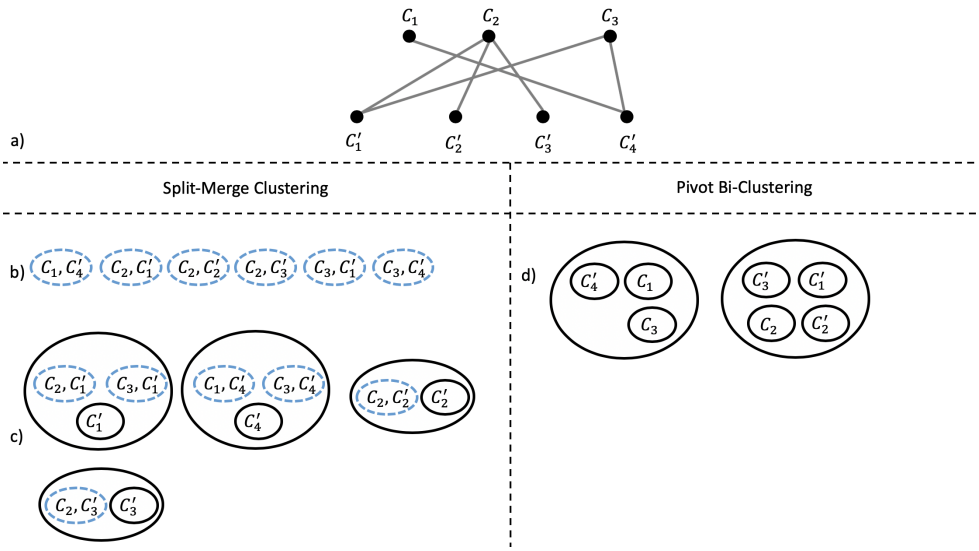


Figure 2: Clustering solutions generated by Split-Merge Clustering (left) and PivotBiCluster (right), respectively: a) the input bipartite graph; b) temporary clusters formed by Split-Merge Clustering after splitting overclustered nodes from the left set (above) ($\{C_1, C_2, C_3\}$) of the graph among corresponding nodes from the right set (below) ($\{C'_1, C'_2, C'_3, C'_4\}$); c) the final clustering solution produced by Split-Merge Clustering, d) the final clustering solution produced by PivotBiCluster.

4 Experimental Setup

In [8], we have evaluated the Split-Merge Evolutionary Clustering and PivotBiCluster algorithms in two different case studies. We have compared the performance of the algorithms in expertise retrieval domain by applying them on data extracted from PubMed repository. In addition, a case study in profiling patients in healthcare domain has been conducted. The Split-Merge Clustering algorithm has shown better performance than the PivotBiCluster in most of the studied experimental scenarios.

In the current work we further study and compare the two clustering algorithms with the Dynamic split-and-merge algorithm, proposed in [10], on four different data sets (explained in the following section) under two different experiment scenarios

(see Section 4.3).

4.1 Data

Anthropometric data set: This dataset is publicly available and published in [28]. The data contains 400 undergraduate students aged between 16 and 63 years old, where a 56.3% are women. The following features describe the data: age, obesity, BMI, WC, HC, WHR, Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), *preh* for women and *hyper* for men, where the *preh* and *hyper* are classification labels that show what kind of blood pressure the individual has (e.g., regular or hyper). According to the results published in [29] people can be grouped into six clusters depending on their blood pressure. This grouping is considered as the ground truth to benchmark the results generated by the three studied clustering algorithms.

Yeast data set: The yeast data set obtained from the UCI machine learning repository is used to predict the cellular localization site of protein [30]. Data set consists of 1484 instances of data with 8 attributes, divided into 10 classes.

Wine quality data set: The wine quality-data set obtained from the UCI machine learning repository includes two data sets, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests [31]. These data sets are labelled and can be used for classification tasks.

Cover-type data set: The cover-type data set obtained from the UCI machine learning repository is created to predict the forest cover type [32]. The data set contains cartographic values of a forest. It is a labeled data set primarily designed to validate classification algorithms. Data set consists of 581012 instances of data with 54 attributes, divided into 7 classes.

Notice that anthropometric data set has been used in [8] for our case study in healthcare domain, while cover-type data set has been used by Lughofer in [10]. The selected data sets are labelled and their characteristics are summarized in Table 1. One of the advantages of using labeled data is that the available class labels could be used as a benchmark while validating the obtained clustering solution.

Table 1: Characteristics of the used test data sets

data sets	#Instances	#Attributes	#Clusters
antropometric	400	9	6
yeast	1484	8	10
wine quality	6498	12	7
cover-type	581,012	54	7

4.2 Metrics

The data mining literature provides a range of different cluster validation measures, which are broadly divided into two major categories: *external* and *internal* [26]. External validation measures have the benefit of providing an independent assessment of clustering quality, since they validate a clustering result by comparing it to a given external standard. However, an external standard is rarely available. Internal validation techniques, on the other hand, avoid the need for using such additional knowledge, but have the alternative problem to base their validation on the same information used to derive the clusters themselves. Furthermore, internal measures can be split with respect to the specific clustering property they reflect and assess to find an optimal clustering scheme: *compactness*, *separation*, *connectedness*, and *stability* of the cluster partitions.

External validation measures can be two types: *unary* and *binary* [33]. Unary external evaluation measures take a single clustering result as the input, and compare it with a known set of class labels to assess the degree of consensus between the two. Comprehensive measures like the F-measure provide a general way to evaluate this [34]. In addition to unary measures, the data-mining literature also provides a number of indices, which assess the consensus between a produced partitioning and the existing one based on the contingency table of the pairwise assignment of data items. Most of these indices are symmetric, and are therefore equally well-suited for the use as binary measures, i.e., for assessing the similarity of two different clustering results.

In this work, we have implemented three different validation measures for estimating the quality of clusters, produced by the three studied clustering algorithms: F-measure, Jaccard Index and Silhouette Index.

We have used the F-measure as an external (unary) validation metric [35]. The *F-measure* is the harmonic mean of the precision and recall values for each cluster. Let us consider two clustering solutions $C = \{C_1, C_2, \ldots, C_k\}$ and $C' = \{C'_1, C'_2, \ldots, C'_l\}$ of the same data set. The first solution C is a known partition of the considered data set while the second one C' is a partition generated by the applied clustering algorithm. The F-measure for a cluster C'_j is then given as

$$F(C'_j) = \frac{2\left|C_i \bigcap C'_j\right|}{|C_i| + \left|C'_j\right|},$$

where C_i is the cluster that contains the maximum number of objects from C'_j. The overall F-measure for clustering solution C' is defined as the mean of cluster-wise F-measure values, i.e.

$$F(C') = \frac{1}{l}\sum_{j=1}^{l} F_j. \tag{I.1}$$

For a perfect clustering, when $l = k$, the maximum value of the F-measure is 1.

In addition, we have applied Jaccard Index (Jaccard similarity coefficient) [36] to evaluate the stability of the clustering algorithms. Given a pair of clustering solutions of the same data set, C and C', we define a as the number of data point pairs that belong to the same cluster in C as well as in C'. Let b be the number of data point pairs that belong to the same cluster in C but not in C'. Further, c is defined to be the number of data point pairs that belong to the same cluster in C' but not in C. The *Jaccard Index* (JI) between C and C' is then defined as:

$$J(C, C') = \frac{a}{a + b + c}. \tag{I.2}$$

The Jaccard Index ranges from 0 to 1, where a higher value indicates a higher similarity between cluster solutions. Jaccard Index has been used to measure the similarity between the generated clustering solutions and the corresponding benchmark partitionings of the used test data sets.

Furthermore, *Silhouette Index* (SI) has been applied as an internal measure to assess compactness and separation properties of the generated clustering solutions [37]. It is a cluster validation index that can be used to judge the quality of any clustering solution C. Suppose a_i represents the average distance of object i from the other objects of its assigned cluster, and b_i represents the minimum of the average distances of object i from objects of the other clusters. The Silhouette Index for clustering solution C of m objects is defined as:

$$s(C) = \frac{1}{m} \sum_{i=1}^{m} \frac{(b_i - a_i)}{\max\{a_i, b_i\}}. \tag{I.3}$$

The values of Silhouette Index vary from -1 to 1 and higher values indicate better clustering results.

4.3 Experiments

We have studied two different experiment scenarios. In the first scenario we compare the three clustering algorithms on cover-type and wine quality data sets described in Section 4.1. Each data set is used to generated 10 test data set couples by randomly separating the data points in two sets. One set (containing 70% of data) of each couple presents the available data set and the other one (30% of data) is the set of newly collected data objects. In that way 10 test clustering couples are created for each data set.

In the second scenario we examine whether the three studied algorithms are sensitive to the size of the new portion of data. For this purpose we use the other two data sets (anthropometric and yeast) described in Section 4.1. For each data set we produce 4 times 10 test data set couples by randomly separating its data points in two sets in a ratio 50/50, 60/40, 70/30 and 80/20, respectively.

4.4 Implementation and Availability

The three studied clustering algorithms (Split-Merge Evolutionary Clustering, PivotBiCluster and Dynamic split-and-merge) are implemented in Python. We have selected the MiniBatchKMeans algorithm available in scikit-learn library [1] as an incremental clustering used in the implementation of the Dynamic split-and-merge. F-measure, Jaccard Index and Silhouette Index (see Section 4.2) used to validate the clustering solutions generated in our experiments are also implemented in scikit-learn library.

Notice that in the experiments conducted on cover-type data set we have used only the 14 non-binary attributes from all 54 attributes of this data. We have not considered the soil type data, since they are very sparse. In addition, we have used a sample set of 50 000 instances.

Supplementary information is available at GitLab [2].

5 Results and Discussion

The results produced by the three studied clustering algorithms in the first experiment scenario are given in Table 2 and Table 3. The performance of the algorithms is studied with respect to three different cluster validation measures: Silhouette Index (SI), F-measure and Jaccard Index. The results from the evaluation of the algorithms on cover-type data set are given in Table 2. As one can see, the PivotBiCluster and Split-Merge Clustering have generated significantly higher F-measure scores than the Dynamic split-and-merge. However, the latter algorithm slightly outperforms the other two with respect to SI. Notice that the PivotBiCluster behaviors significantly better then the other two algorithms with respect to F-measure and Jaccard Index. In general, the PivotBiCluster can be considered as the best performing algorithm on cover-type data set. It is further interesting to discuss that although, the incremental algorithm (MiniBatchKMeans) used by the Dynamic split-and-merge has modified all the 7 initial clusters in each test data couple no split and merge actions have been performed. For example, if we compare it with the other two algorithms on one and the same data set couple the PivotBiCluster has performed 2 merges while the Split-Merge Clustering has done 4 merges and 7 splits. In addition, the PivotBiCluster has generated a cluster solution with 5 cluster while the clustering solution produced by the Split-Merge Clustering has 7 cluster. This supports our discussion in Section 3.3 that the Dynamic split-and-merge algorithm is not very sensitive to concept drift scenarios compared to the other two algorithms, which update the existing clustering solution by considering the clustering extracted from the new portion of data.

Table 3 contains the results obtained from the evaluation of the three clustering

[1] scikit-learn is a Python library for data mining and data analysis.

[2] `https://gitlab.com/machine_learning_vm/clustering_techniques`

Table 2: Experiment 1: Average cluster validation metrics scores generated on the clustering solutions of the 10 cover-type test data set couples.

metrics	PivotBiCluster	Split-Merge Clustering	Dynamic split-and-merge
SI	0.194	0.034	0.196
F-measure	0.903	0.759	0.376
Jaccard Index	0.231	0.021	0.161

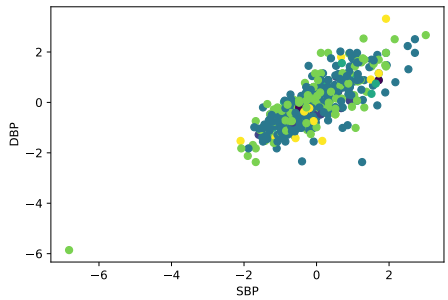
algorithms on wine quality data set with respect to the three used cluster validation criteria. The PivotBiCluster is again the best performing algorithm according to the results produced by F-measure and Jaccard Index. However, this is not supported by the generated SI scores. Namely, the Dynamic split-and-merge has the highest average SI score. However, it is outperformed by the Split-Merge Clustering with respect to Jaccard Index and F-measure. It is also interesting to observe that the number of clusters of the clustering solutions generated on the wine quality test data set couples varies from 5 to 8 for the PivotBiCluster, and between 1 and 7 (seven data set couples have generated clustering solutions with 4 or less clusters) for the Dynamic split-and-merge. This might be the main reason for the higher SI scores generated by the Dynamic split-and-merge algorithm, since the SI score generated on the benchmark clustering of wine quality data set is -0.06. In the case of the Split-Merge Clustering the data points are grouped into 7 or 8 clusters, i.e. much closer to the benchmark clustering of wine quality data set. The latter one has 7 clusters (see Table 1). This trend has been noticed also for the other three data sets (see the discussion below about the results generated on antropometric data set).

Table 3: Experiment 1: Average cluster validation metrics scores generated on the clustering solutions of the 10 wine quality test data set couples.

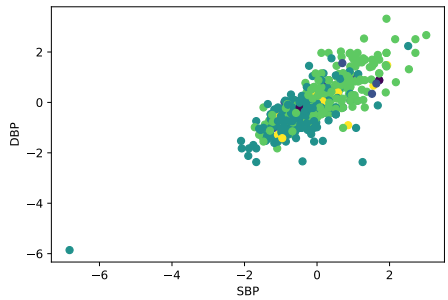
metrics	PivotBiCluster	Split-Merge Clustering	Dynamic split-and-merge
SI	-0.111	-0.129	0.143
F-measure	0.676	0.461	0.311
Jaccard Index	0.269	0.143	0.137

The results obtained in the second experiment scenario are given in Tables 4, 5 and 6. For example, Table 4 presents the evaluations of the clustering solutions generated by the three algorithms on antropometric and yeast data sets with respect to F-measure. The obtained results support the better performance of PivotBiCluster and Split-Merge Clustering compared to the Dynamic split-and-merge. The PivotBiCluster even slightly outperforms the Split-Merge Clustering with respect to this evaluation criterion (F-measure).

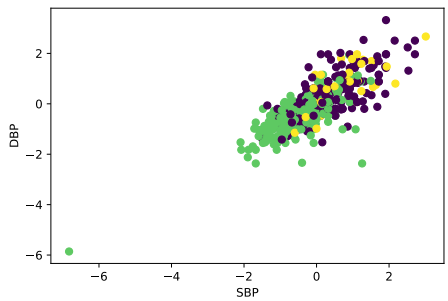
In line with the results obtained on cover-type and wine quality data sets the Dynamic split-and-merge outperforms the other two algorithms with respect to the SI evaluation criteria (see Table 5). As it was already discussed above we believe that this is due to the fact that it generates the clustering solutions with less number of clusters compared to the other two algorithms. For example, we have compared the three algorithms on the ten 60/40 test couples of antropometric data set. The number of clusters of the clustering solutions generated by the Dynamic split-and-merge



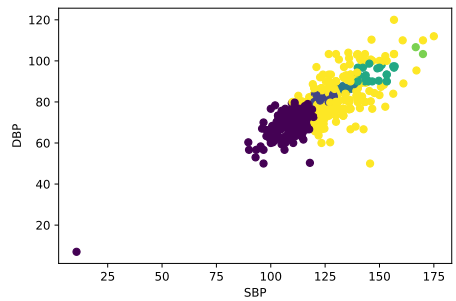
(a) *PivotBiCluster: 4 clusters, SI = - 0.23, F-measure = 0.544*



(b) *Split-Merge Clustering: 5 clusters, SI = - 0.05, F-measure = 0.594*



(c) *Dynamic split-and-merge: 3 clusters, SI = 0.24, F-measure = 0.369*



(d) *Benchmark clustering: 6 clusters, SI = 0.017*

Figure 3: Clustering solutions generated by the three studied clustering algorithms on an antropometric 70/30 test data set couple versus the benchmark clustering.

Table 4: Experiment 2: Average F-measure scores generated on the clustering solutions of the 4×10 antropometric data set couples (above) and 4×10 yeast test data set couples (below).

antropometric	50/50	60/40	70/30	80/20
PivotBiCluster	0.677	0.624	0.544	0.676
Split-Merge Clustering	0.546	0.504	0.519	0.481
Dynamic split-and-merge	0.374	0.389	0.442	0.482
yeast				
PivotBiCluster	0.700	0.710	0.821	0.858
Split-Merge Clustering	0.576	0.522	0.496	0.489
Dynamic split-and-merge	0.419	0.423	0.426	0.410

varies from 1 to 3, in the case of the PivotBiCluster all ten clustering solutions have 4 clusters, while the Split-Merge Clustering has grouped the data points into 6, 7 or 8 clusters. Evidently, the three clustering algorithms have generated clustering solutions with very different number of clusters. However, the clustering solutions of the Split-Merge Clustering are most close to the benchmark clustering of antropometric data set (Section 4.1), which has 6 clusters. This trend has been noticed also for the

other three data sets (see the discussion on wine quality data set). Figure 3 further illustrates this by plotting clustering solutions generated by the three algorithms on an antropometric 70/30 test data set couple. The corresponding benchmark clustering and its SI score are given in Figure 3d. As one can see the three algorithms have generated clustering solutions that have different number of clusters. In addition, they have produced different SI and F-measure scores. The clustering solutions produced by the Split-Merge Clustering (Figure 3b) and Dynamic split-and-merge (Figure 3c) seem close to each other and they are visually more similar to the benchmark clustering than the PivotBiCluster solution (Figure 3a). This is also supported by the calculated SI scores. We further observe that the PivotBiCluster is the worst performing algorithm according to SI on antropometric data set while its performance on yeast data set is almost comparable to that of the Dynamic split-and-merge algorithm. It is interesting to notice that the performance of the Split-Merge Clustering is influenced by the size of the new data set, while this is not clearly demonstrated by the other two algorithms, even in some experiments they have been improving their performance.

Table 5: Experiment 2: Average SI scores generated on the clustering solutions of the 4×10 antropometric test data set couples (above) and 4×10 yeast test data set couples (below).

antropometric	50/50	60/40	70/30	80/20
PivotBiCluster	-0.344	-0.327	-0.231	-0.178
Split-Merge Clustering	-0.212	-0.189	-0.108	-0.096
Dynamic split-and-merge	0.2	0.238	0.188	0.170
yeast				
PivotBiCluster	0.142	0.068	0.044	0.076
Split-Merge Clustering	-0.061	-0.061	-0.048	-0.036
Dynamic split-and-merge	0.164	0.157	0.158	0.150

The evaluations of the clustering solutions produced by the three algorithms on antropometric and yeast data sets with respect to Jaccard Index are given in Table 6. The Dynamic split-and-merge is the best performing algorithm with respect to this evaluation criterion. However, the generated values are very close to ones of the Split-Merge Clustering, particularly for the 70/30 data test couples. It is also interesting to notice that in contradiction to the behaviour of PivotBiCluster on cover-type and wine quality data sets, it is the worst performing algorithm under Jaccard Index on antropometric and yeast data sets.

Table 6: Experiment 2: Average Jaccard Index scores generated on the clustering solutions of the 4×10 antropometric data set couples (above) and 4×10 yeast test data set couples (below).

antropometric	50/50	60/40	70/30	80/20
PivotBiCluster	0.021	0.015	0.068	0.058
Split-Merge Clustering	0.077	0.164	0.107	0.074
Dynamic split-and-merge	0.156	0.199	0.119	0.094
yeast				
PivotBiCluster	0.014	0.022	0.034	0.020
Split-Merge Clustering	0.086	0.089	0.090	0.086
Dynamic split-and-merge	0.099	0.136	0.105	0.118

In [38], Luxburg et al. argue that clustering should not be treated as an application independent mathematical problem, but should always be studied in the context of its end-use. The authors further discuss that the cluster evaluation methods can produce contradictory results and often do not serve their purpose. The main point of the authors is that clustering algorithms cannot be evaluated in a problem independent way, i.e. the known cluster validation measures cannot be used to evaluate the usefulness of the clustering. However, it is still not clear how we can measure the usefulness of a newly developed clustering algorithm.

The results obtained in this study support the above mentioned arguments of Luxburg et al. [38]. Namely, the conducted experiments have not clearly pointed out an algorithm that we can consider and recommend as the best performing one compared to the other two algorithms with respect to the used cluster validation criteria. For example, SI has favoured the Dynamic split-and-merge algorithm in most of the performed experiments. On the other hand, the PivotBiCluster and Split-Merge Evolutionary Clustering have generated higher values for F-measure than the Dynamic split-and-merge algorithm. Moreover, the Split-Merge Clustering and Dynamic split-and-merge algorithms have performed better compared to the PivotBiCluster algorithm with respect to Jaccard Index on two of the used data sets. However, on the other two data sets the PivotBiCluster algorithm is the best performing one under this evaluation criterion. The PivotBiCluster has slightly outperformed the Split-Merge Evolutionary Clustering with respect to F-measure. However, the Split-Merge Evolutionary Clustering algorithm has shown to be more robust than the other two algoritms in producing clustering solutions with cluster number close to that of the benchmark clustering solutions. Evidently, the three clustering algorithms need to be further studied and validated on different applied scenarios in order to get better understanding of their specific characteristics, behaviour and further evaluate the usefulness.

6 Conclusion and Future Work

In this work, we have studied and evaluated a novel evolutionary clustering technique, entitled Split-Merge (Evolutionary) Clustering. The proposed algorithm is supposed to be more robust to concept drift scenarios by providing the flexibility to update the existing clustering solution by considering the clusters derived from a new portion of data. The proposed technique has been compared with other two state of the art clustering algorithms: PivotBiCluster and Dynamic split-and-merge. The three algorithms have been evaluated and demonstrated in two experiment scenarios on four different data sets using three cluster validation indices: Silhouette Index (SI), F-measure and Jaccard Index. The obtained results have not clearly prioritized any of the three studied clustering algorithms. The Dynamic split-and-merge algorithm has been favoured by SI in the most of conducted experiments. The Pivot-

BiCluster and Split-Merge Evolutionary Clustering have produced higher F-measure scores than the Dynamic split-and-merge algorithm in all the experiments. The PivotBiCluster algorithm has demonstrated a slightly better performance than the Split-Merge Evolutionary Clustering under this evaluation criterion. Jaccard Index has not clearly pointed out an algorithm that can be considered as the best performing one. The Split-Merge Evolutionary Clustering algorithm has shown to be more robust to producing clustering solutions that have number of clusters close to that of the benchmark clustering solutions.

Our future plans are to pursue further study and evaluation of our Split-Merge Evolutionary Clustering technique by comparing it with the other two state of the art algorithms on richer data sets and in case studies from different application domains. For example, we are currently interested in evaluating the algorithms on household electricity consumption data. We study whether they can be applied for modelling and monitoring evolving user behavior.

In a long-term perspective, we are interested in building upon the proposed split-merge evolutionary algorithm and develop measures for monitoring clusters evolution and mining changes. This might be treated as time-series forecasting problem where we need to forecast the changes in the clustering solution that might occur. Other interesting future direction is to use the proposed split-merge framework for developing a continual and shared learning technique that enable to learn from multiple data sources by continual updating and evolving of the model.

References

[1] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. “MOA: massive online analysis”. In: *Journal Machine Learning Research* 11 (2010), pp. 1601–1604.

[2] A. Bouchachia. “Evolving clustering: an asset for evolving systems”. In: *IEEE SMC Newsletters* 36 (2011).

[3] M. Ackerman and S. Dasgupta. “Incremental Clustering: The Case for Extra Clusters”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’14. 2014, pp. 307–315.

[4] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. “Incremental Clustering and Dynamic Information Retrieval”. In: *Proc. of the 29th Annual ACM Symposium on Theory of Computing*. STOC ’97. 1997, pp. 626–635.

[5] M. Zopf and et al. “Sequential Clustering and Contextual Importance Measures for Incremental Update Summarization”. In: *Proc. of COLING’2016*. 2016, pp. 1071–1082.

[6] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. "Streaming-Data Algorithms for High-Quality Clustering". In: *Proceedings of IEEE International Conference on Data Engineering*. 2001, pp. 685–694.

[7] D. Dell'Aglio, E. D. Valle, F. van Harmelen, and A. Bernstein. "Stream reasoning: A survey and outlook". In: *Data Science* 1 (1-2 2017), pp. 59–83.

[8] V. Boeva, M. Angelova, and E. Tsiporkova. "A Split-Merge Evolutionary Clustering Algorithm". In: *Proceedings of ICAART 2019*. 2019, pp. 337–346.

[9] N. Ailon, N. Avigdor-Elgrabli, E. Liberty, and A. van Zuylen. "Improved Approximation Algorithms for Bipartite Correlation Clustering". In: *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*. 2011, pp. 25–36.

[10] E. Lughofer. "A dynamic split-and-merge approach for evolving cluster models". In: *Evolving Systems* 3.3 (Sept. 2012), pp. 135–151.

[11] P. Angelov. "An approach for fuzzy rule-base adaptation using on-line clustering". In: *International Journal of Approximate Reasoning* 35 (3 2004), pp. 275–289.

[12] A. Bouchachia and C. Vanaret. "Incremental learning based on growing gaussian mixture models". In: *Proceedings of 10th international conference on machine learning and applications (ICMLA 2011), Honululu, Haweii*. 2011.

[13] D. Dovzan and I. Skrjanc. "Recursive clustering based on a Gustafson-Kessel algorithm". In: *Evolving Systems* 2 (1 2011), pp. 15–24.

[14] F. Farnstrom, J. Lewis, and C. Elkan. "Scalability for clustering algorithms revisited". In: *SIGKDD Explorations, London*. Vol. 2. 2000, pp. 51–57.

[15] M.-F. Balcan, A. Blum, and S. Vempala. "A Discriminative Framework for Clustering via Similarity Functions". In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. 2008, pp. 671–680.

[16] R. Fa and A. K. Nandi. "Smart: Novel self splitting-merging clustering algorithm". In: *European Signal Processing Conference, Bucharest, Romania, August, 27-32*. IEEE, 2012.

[17] M. Wang, V. Huang, and A.-M. C. Bosneag. "A Novel Split-Merge-Evolve k Clustering Algorithm". In: *IEEE 4th International Conference on Big Data Computing Service and Applications (BigDataService), Bamberg, Germany, March 26-29*. 2018.

[18] A. Campan and G. Serban. "Adaptive Clustering Algorithms". In: *Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006: Advances in Artificial Intelligence*. 2006, pp. 407–418.

[19] A. Gionis, H. Mannila, and P. Tsaparas. "Clustering Aggregation". In: *ACM Transaction of Knowledge Discovery Data* 1.1 (2007).

[20] S. Bickel and T. Scheffer. “Multi-View Clustering”. In: *Proceedings of the Fourth IEEE International Conference on Data Mining*. ICDM ’04. 2004, pp. 19–26.

[21] V. Boeva, E. Tsiporkova, and E. Kostadinova. “Analysis of Multiple DNA Microarray Datasets”. In: *Springer Handbook of Bio-/Neuroinformatics*. Springer Berlin Heidelberg, 2014, pp. 223–234.

[22] A. Goder and V. Filkov. “Consensus Clustering Algorithms: Comparison and Refinement”. In: *ALENEX*. 2008, pp. 109–234.

[23] Q. Xiang, Q. Mao, K. M. A. Chai, H. L. Chieu, I. W. Tsang, and Z. Zhao. “A Split-Merge Framework for Comparing Clusterings”. In: *ICML*. 2012.

[24] N. Bansal, A. Blum, and S. Chawla. “Correlation Clustering”. In: *Machine Learning* 56.1-3 (2004), pp. 89–113.

[25] P. Awasthi, M. F. Balcan, and K. Voevodski. “Local algorithms for interactive clustering”. In: *Journal of Machine Learning Research* 18.3 (2017), pp. 1–35.

[26] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. English. Englewood Cliffs, NJ: Prentice Hall, 1988, pp. xiv + 320. ISBN: 0-13-022278-X.

[27] G. Gan, C. Ma, and J. Wu. *Data clustering: theory, algorithms, and applications*. (Asa-Siam Series on Statistics and Applied Probability). Society for Industrial & Applied Mathematics, USA, 2007.

[28] H. F. Golino, L. S. de Brito Amaral, S. F. P. Duarte, and et al. “Predicting Increased Blood Pressure Using Machine Learning”. In: *Journal of Obesity* 2014 (2014).

[29] Y. Li, X. Feng, M. Zhang, M. Zhou, N. Wang, and L. Wangb. “Clustering of cardiovascular behavioral risk factors and blood pressure among people diagnosed with hypertension: a nationally representative survey in China”. In: *Sci Rep.* 6 (2016).

[30] K. Nakai and M. Kanehisa. “Expert Sytem for Predicting Protein Localization Sites in Gram-Negative Bacteria”. In: *PROTEINS: Structure, Function, and Genetics* 11 (1991), pp. 95–110.

[31] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reisa. “Modeling wine preferences by data mining from physicochemical properties”. In: *Decision Support Systems* 47.4 (2009), pp. 547–553.

[32] J. A. Blackard, D. J. Dean, and C. W. Anderson. *UCI Machine Learning Repository*. 1998. URL: `http://archive.ics.uci.edu/ml`.

[33] J. Handl, J. Knowles, and D. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.

[34] C. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann Newton. MA, USA, 1979.

[35] B. Larsen and C. Aone. “Fast and Effective Text Mining Using Linear-time Document Clustering”. In: *Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD’99. ACM, 1999, pp. 16–22.

[36] P. Jaccard. “The distribution of flora in the alpine zone”. In: *New Phytologist* 11 (1912), pp. 37–50.

[37] P. J. Rousseeuw. “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65.

[38] U. von Luxburg, R. C. Williamson, and I. Guyon. “Clustering: Science or Art?” In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Vol. 27. Proceedings of Machine Learning Research. 2012, pp. 65–79.

Paper II

Split-Merge Evolutionary Clustering for Multi-View Streaming Data

Vishnu Manasa Devagiri, Veselka Boeva, Elena Tsiporkova

In: 24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems KES 2020, Procedia Computer Science 176, 2020, pp. 460–469, DOI: 10.1016/j.procs.2020.08.048

Abstract

In this study, we propose a new multi-view stream clustering approach, called MV Split-Merge Clustering. The proposed approach is an extension of an existing split-merge evolutionary clustering algorithm (entitled Split-Merge Clustering) to multi-view data applications. The extended version can be used to integrate data from multiple views in a streaming manner and discover cluster structure for each data chunk. The MV Split-Merge Clustering can be applied for grouping distinct chunks of multi-view streaming data so that a global integrated clustering model is built on each data chunk. At each time window, an updated clustering solution (local model) is initially produced on each view of the current data chunk by applying the Split-Merge Clustering algorithm. Formal Concept Analysis is then used in order to integrate information from the multiple views (local clustering models) and generate a global model (formal concept lattice) that reveals the correlations among the clusters of the local models. The proposed MV Split-Merge Clustering has been initially evaluated on a publicly available data set. Our results show that the approach is able to identify a clustering structure and relationships among the different views comparable to those produced in a batch scenario.

Keywords: Clustering algorithms, Data stream mining, Evolutionary clustering, Multi-View clustering, Online learning

1 Introduction

These days there are a number of smart devices used in our everyday life collecting tons of information continuously. Information collected by these devices is often in-

complete and would require data from other devices to deduce useful patterns, and groups. For example, while we are performing activities in our everyday life different types of measurements are stored by different devices like smart phones, smart watches, etc. Information form different views when integrated together could discover new groups or patterns which could not have been found otherwise [1]. Bringing all this information together is a tedious task which requires a lot of computational resources as well as bandwidth [2, 3]. So, using a multi-view solution, which can integrate information from different devices (views) to a global model in an effective way would be of a great use. Moreover, the data collected in these views are seldom of the same type and requires the designed algorithms to be able to handle heterogeneous data.

In the above scenarios data also arrives continuously in a potentially infinite stream and has to be processed by a resource-constrained system. Hence, multi-view stream data mining algorithms are needed to process distributed fast generated data, fundamental properties of which may also change, i.e. the algorithms must handle concept drift. Most traditional data mining algorithms cannot meet these challenges, since they assume centralized and static data. In addition, in many smart monitoring applications the model has to be developed, trained, and evaluated with no direct access to labeled data.

To address the above-discussed challenges, we propose a Multi-View Split-Merge data stream clustering algorithm, entitled MV Split-Merge Clustering. The proposed algorithm can be applied for grouping distinct chunks of multi-view streaming data so that an integrated global clustering model is built on each data chunk. Initially, a updated local clustering model is produced on each view of the current data chunk by applying the Split-Merge evolutionary clustering algorithm (Split-Merge Clustering), published in [4]. Formal Concept Analysis (FCA) is then used in order to integrate information from the local clustering models and generate a formal concept lattice (global model) that reveals the associations among the local models. Notice that the updated local models are built by analysing and integrating the information presented in the current and previous data chunk. In that way, the proposed MV Split-Merge clustering algorithm is capable of performing vertical and horizontal data integration, i.e. over data chunks of each individual view and over all available views per chunk.

The proposed MV Split-Merge Clustering can be applied to tackle different aspects of the today smart monitoring applications such as smart healthcare, smart buildings, etc., where data collection is not centralized. For example, it can be used for continuous adaptation of the model and improvement of the monitoring as more data becomes available. Another example is modelling and detection of context, such as activity mode in health monitoring applications, from multiple sensor values. The algorithm is also able to produce a global model even when data from some sensors are missing, due to network changes (addition or removal of nodes) or degradation. The missing (undelivered) data view models can be re-constructed by using the global and local models built at the previous time window. Notice that the proposed algo-

rithm is also robust to the privacy aspects similar to some of the algorithms reviewed in [2], as actual data from the local devices is not transmitted to the global system (cloud) in order to create a global model.

The rest of the paper is structured as follows. Section 2 contains a brief description of the related work. This is followed by Section 3 which introduces required background. The proposed approach is explained in Section 4. Details about the data, experimental setting and discussion of the results obtained from initial evaluation of the proposed approach are presented in Section 5. Section 6 is devoted to conclusions and future work.

2 Related Work

Distributed data mining is an area of study that addresses the problems of large data which is not available at a single location [3]. A good summary of the published distributed data mining approaches can be found in [3]. In addition, an overview of existing distributed clustering algorithms has been published in [2]. The authors explore distributed clustering algorithms based on density, k-means, privacy protection and highlight advantages and disadvantages of the studied methods. They compare two of the distributed clustering algorithms with a centralised k-means algorithm. Their experiments have shown that the centralized algorithm has a better performance. According to the authors it might have been due to the reason that the global model does not have the complete information that is available.

There are many reviews published in the areas related to multi-view clustering [5–7]. Benezúr et al. [8], summarize the work done in the area of online learning. In [7], the authors consider five popular traditional clustering techniques and adapt them to multi-view clustering. The advantages of the multi-view adaptations in comparison with the traditional approaches have been demonstrated.

Different classifications and categorizations of the existing state-of-the art distributed data mining and multi-view clustering approaches have been proposed in the above discussed reviews. Based on [3], the algorithm proposed in this study could be assigned to meta-learning, heterogeneous data categories. Meta-learning, as our approach builds a global model by using the knowledge presented in the local clustering models. Heterogeneous data, as the algorithm is able to handle heterogeneous data across different data views.

Different multi-view clustering algorithms have been proposed in the data mining literature, e.g. [1, 9–12]. In [9, 11], the authors handle the issue of incomplete views in multi-view clustering. A kernel k-means is used in each view in [11] and information from all the views is fused to find a consensus clustering. In [10], a multi-view clustering algorithm, entitled MVC-LFA, has been proposed. The MVC-LFA has been benchmarked to eight other state of the art algorithms and shown to outperform those in the conducted evaluations. In [12], a weighted multi-view clustering with

feature selection has been proposed to use information from multiple views simultaneously to boost the clustering results. In [1] the multi-view clustering problem is dealt as a multi-objective optimisation problem.

Another interesting and slightly different problem has been addressed in [13]. The authors deal with multi-view clustering in which the views are added gradually (streaming views). Initially they have a model built based on the data from currently available views and as new views arrive the algorithm is able to integrate this information into the built clustering solution.

Although a lot of research in the area of multi-view clustering has already been published, designing multi-view clustering algorithms specially suited for streaming data is still a quite new research direction. According to the authors of [14], their work is the first one to have addressed this problem. In [14], they propose a multi-view clustering algorithm (MVStream), which is based on support vectors. The proposed algorithm is robust to concept drift and is also capable of detecting clusters of arbitrary shapes.

FCA has been used in the field of data integration, e.g. see [15–17]. A MapReduce approach for clustering of data sets generated in multiple-experiment settings is proposed in [15]. It is inspired by the map-reduce functions used in functional programming and consists of two distinctive phases: map-reduce clustering and FCA-based analysis. The latter is applied to analyse and further refine the generated clustering solution. Hristoskova et al. have proposed a generic consensus clustering technique that applies FCA to consolidate clustering solutions derived from several microarray data sets [16]. In [17], the authors proposes a FCA-based solution for data reduction. They initially group the attributes (properties in FCA terminology) and this modified set of attributes or properties are then used to describe the objects. Notice that our algorithm instead of grouping attributes, initially clusters the available objects (instances) in each view. Then the labels of the generated clusters are interpreted as properties and are used to describe the objects in the final clustering solution (global model).

3 Methods and Background

In this study, we propose an extension of the Split-Merge Evolutionary clustering algorithm (Split-Merge Clustering) published in [4] to multi-view data stream scenarios. The proposed algorithm, entitled MV Split-Merge Clustering, can be used to integrate data from multiple views in a streaming manner. The MV Split-Merge Clustering can be applied for grouping distinct chunks of multi-view streaming data so that a global clustering model is built on each data chunk. At each time window, a updated clustering solution (local model) is initially produced on each view of the current data chunk by applying the Split-Merge Clustering. In that way updated local models reflecting the information presented in the current and previous data chunk

are obtained. FCA is then used in order to integrate information from the local clustering models and generate a global model that reveals the relationships among the local models.

3.1 Split-Merge Clustering

Split-Merge Clustering has been proposed in [4]. It is an evolutionary clustering algorithm designed to be robust to concept drift scenarios. The algorithm can be applied to update an existing clustering solution when new data arrives. The Split-Merge Clustering accommodates newly arrived data into the existing clustering solution thus also reducing the amount of computations required to build a new clustering solution on the expanded data. Namely, it enables to compute clusters on a new portion of data collected over a defined time window (or a given data chunk) and to update the existing clustering solution by the computed new one. The existing and the newly constructed clusterings are initially modelled as a bipartite graph which is further decomposed into connected components (bi-cliques). Each component is further analysed and if it is necessary it is decomposed into sub-components. The sub-components are then taken into consideration in producing the final clustering solution. For example, if an existing cluster is overclustered, i.e. it intersects two or more clusters in the new clustering, it is split between those. If several existing clusters intersect the same new cluster, i.e. they are underclustered, they are merged with that cluster.

A detailed description of the Split-Merge Clustering and its pseudo code can be found in [4].

3.2 Formal Concept Analysis

Formal Concept Analysis (FCA) [18] is a data analysis method which enables to discover hidden knowledge existing in data. It allows to extract implicit relationships between objects described through a set of attributes. Central to FCA is the notion of a formal context. Namely, FCA derives a concept lattice from a formal context constituted of a set of objects O, a set of attributes A, and a binary relation defined on the Cartesian product $O \times A$. The context is described as a table, the rows correspond to objects and the columns to attributes or properties and a cross in a table cell means that "an object possesses a property". FCA can be used for a number of purposes among which knowledge formalization and acquisition, ontology design, and data mining are some.

The *concept lattice* is composed of formal concepts, or simply concepts, organized into a hierarchy by a partial ordering (a subsumption relation allowing to compare concepts). Intuitively, a concept is a pair (X, Y) where $X \subseteq O$, $Y \subseteq A$, and X is the maximal set of objects sharing the whole set of attributes in Y and vice-versa.

The set X is called the *extent* and the set Y the *intent* of the concept (X, Y). The subsumption (or sub-concept – super-concept) relation between concepts is defined as follows: $(X_1, Y_1) \prec (X_2, Y_2) \Leftrightarrow (X_1 \subseteq X_2 \text{ or } Y_2 \subseteq Y_1)$. Relying on this subsumption relation $\prec$, the set of all concepts extracted from a context is organized within a complete lattice, that means that for any set of concepts there is a smallest super-concept and a largest sub-concept, called the *concept lattice*.

FCA is capable of discovering hidden knowledge by structuring the data. Namely, the concept lattice organizes the data into concepts (formal abstractions) that are described through attributes and in that way allowing meaningful human-understandable interpretation [18]. Thus FCA can be seen as a conceptual clustering technique as it provides descriptions for the abstract concepts it produces.

4 Proposed Multi-View Split-Merge Clustering

In this section the proposed *MV Split-Merge Clustering* algorithm is further discussed by formally describing the problem and introducing the main operations and steps of the algorithm.

Assume that a particular phenomenon (patient, physical object etc.) is monitored under n different circumstances (views) in a streaming fashion. In addition, the data arrives over time in chunks and each chunk can contain different number of data points. Each chunk t can be represented by a list of n different data matrices $D_t = \{D_{t1}, D_{t2}, \ldots, D_{tn}\}$, one per view. Each matrix i ($i = 1, 2, \ldots, n$) contains the information about the data points in the current chunk with respect to the corresponding view. Assume that chunk t contains N_t data points.

Initially, n clustering models, one per view, can be built on available multi-view historical data or on the initial data chunk. Let $C_t = \{C_{t1}, C_{t2}, \ldots, C_{tn}\}$ be a set of clustering solutions (local models), such that C_{ti} ($i = 1, 2, \ldots, n$) represents the grouping of the data points in tth chunk with respect to ith view, i.e. a local model built on data set D_{ti}. When new data chunk arrives, i.e. the new data matrices $D_{t+1} = \{D_{(t+1)1}, D_{(t+1)2}, \ldots, D_{(t+1)n}\}$ are available the main challenge in the described context is how to update the current local clustering models and build a global model. The global model may be useful to combine and find existing implicit relationships between the local clustering models.
Basic operations conducted by our MV Split-Merge Clustering algorithm on each data chunk t are explained below:

1. **Input:** Existing n local clustering models C_t built on tth data chunk and newly arrived data D_{t+1}.

2. **Adapting step:** Updating the existing local cluster models by applying the Split-Merge Clustering.

(a) Build a clustering solution on each new data matrix $D_{(t+1)i}$ $(i = 1, 2, \ldots, n)$ of $(t + 1)$th chunk.

(b) Use the Split-Merge Clustering to update the local clustering models C_t. In order to build the updated local models we consider only the data points from chunks t and $(t + 1)$.

(c) Generate updated local clustering models $C_{t+1} = \{C_{(t+1)1}, C_{(t+1)2}, \ldots, C_{(t+1)n}\}$. Each clustering solution $C_{(t+1)i}$ $(i = 1, 2, \ldots, n)$ distributes the data points into k_i disjoint clusters.

3. **Integration step:** Use FCA to integrate the local clustering models C_{t+1} into a global model $G_{(t+1)}$.

   (a) Create a formal context by using information about the data points' distribution from the local clustering models. In our case a formal context consists of the set of $(N_t + N_{t+1})$ data points, the set of K $(K = k_1 + k_2 + \ldots + k_n)$ clustering labels of C_{t+1} and an indication of which data points are associated with which clusters (properties). Thus the context is described as a matrix, with the data points corresponding to the rows and the cluster labels corresponding to the columns of the matrix, and a value 1 in cell (i, j) whenever data point i belongs to cluster C_j $(j = 1, 2, \ldots, K)$. Note that in each row we will have exactly n values of 1.

   (b) Generate a formal concept lattice (i.e. a global model $G_{(t+1)}$) by using the built formal context.

4. **Output:** Updated local clustering models C_{t+1} and integrated global model $G_{(t+1)}$.

Figure 1 illustrates the basic operations of the MV Split-Merge Clustering with a two-view scenario example. We have 24 data points in chunk t that are represented by two different views. These data points are grouped into two clusters w.r.t. view 1 and into three clusters w.r.t. view 2, respectively. In chunk $(t + 1)$ the Split-Merge Clustering is initially applied to update the two local clustering models of chunk t. As one can notice one of the clusters of the view 2 clustering model is split between the other two clusters thus resulting in two clustering model in chunk $(t+1)$. FCA is then used to integrate the two local clustering models of chunk $(t + 1)$ into a global model, i.e. the corresponding concept lattice revealing the relationships between the two clusterings. For example, it can be seen that the data objects of blue triangle cluster of view 1 are associated with the objects of blue square cluster of view 2.

In addition to the above description Figure 2 shows a diagram of main steps of the MV Split-Merge Clustering algorithm. As one can notice three main steps can be recognized: constructing a clustering solution on each view of the newly arrived

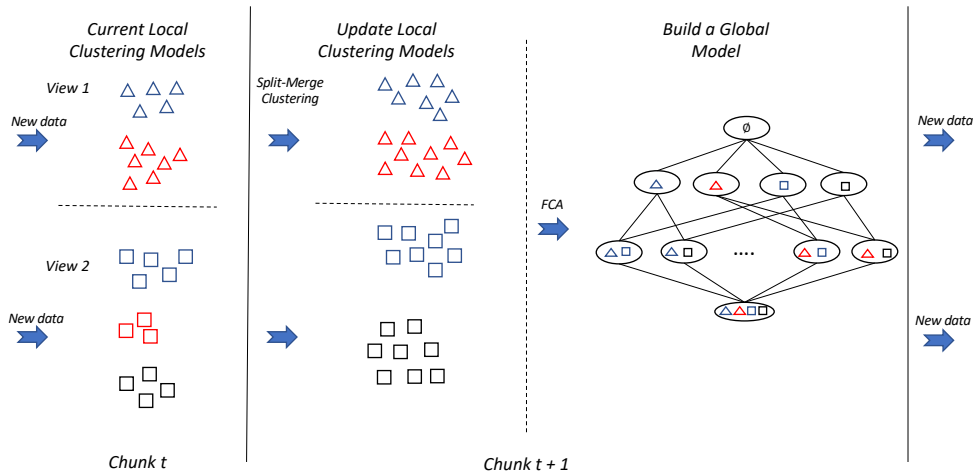


Figure 1: A two-view example of the MV Split-Merge Clustering operations

chunk; updating the local clustering models; building a global model on the current chunk. In that way on each data chunk a clustering solution is initially generated on each view and it is further used to update the corresponding local clustering model built on the previous data chunk. The Split-Merge Clustering algorithm, introduced in [4], is used for updating the existing clustering solution by the computed new one. This algorithm is more robust to concept drift scenarios by being able to examine clusters generated on each view of the new data chunk and eventually capture new trends in each view. The algorithm produces a updated local clustering model on each view by merging some clusters with ones from the newly constructed clustering on this view while others are transformed by splitting their elements among several new clusters. The produced local clustering models are finally integrated into a global model by applying FCA method. FCA allows to derive a concept lattice from a formal context. In our scenario, the formal context constitutes of the set of data points and the set of clustering labels (see 3(a) in the above description). Notice that on one hand the built formal context can be utilized to abstract the summary statistics of the historical multi-view data points. On the other hand the generated concept lattice (global model) provides information about implicit relationships (concepts) among the data objects described through different view clustering labels.

According to [4], the computational complexity of the Split-Merge Clustering is $O(N(k + k')m)$, where m is the dimensionality of the learning problem, N is the number of instances in the newly arriving data, k and k' are number of clusters in the current and new data chunk, respectively. This represents the cost of building the local clustering models at each data chunk. The number of views n usually does not have a significant effect on the cost, since $n << N$ and $n << m$. Furthermore,

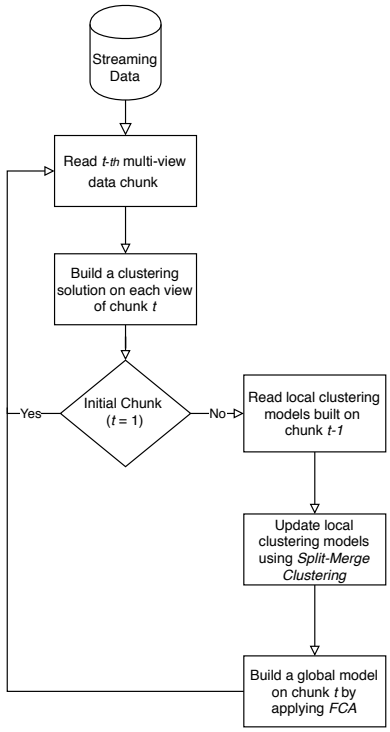


Figure 2: Flowchart diagram of the MV Split-Merge Clustering

the Python implementation of FCA, we have used [19], is based on the algorithm proposed in [20]. According to the authors its time complexity is $O(L^2)$, where L is the number of lattice concepts.

5 Initial Evaluation and Results

This section provides information about the used data, experimental settings, conducted experiments, and discusses the obtained initial results.

5.1 Data and Experimental Setup

For initial evaluation of the proposed algorithm we have used a publicly available data set [21] that describes the medical conditions of undergraduate students based on their anthropometric data. The data contains information about 399 undergraduate students aged between 16 and 63 years old, where 56.3% are women. The following features describe the data: age, obesity, body mass index (BMI), waist circumference (WC), hip circumference (HC), and waist hip ratio (WHR), Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), *preh* for women and *hyper* for men, where

the *preh* and *hyper* are classification labels that show what kind of blood pressure the individual has (e.g., regular or hyper).

In order to adapt the data set to a multi-view streaming scenario we have divided the available features into three groups (views). The first view (v_1) consists of age and sex; the second view (v_2) includes BMI, WC, HC, WHR; and the third view (v_3) contains details about blood pressure, i.e. SBP and DBP. The streaming scenario has been modelled by dividing the data set into two parts, where one part consisting of 70% of the data represents the current data chunk and the other part consisting of the remain 30% of the data represents the newly arriving data chunk.

The data set is further used to generate 10 test data set couples by randomly separating the individual profiles in two sets, as it was explained above. Thus the first set (279 patients) of each couple presents the current data chunk of individual profiles, and the other one (120 individuals) is the new chunk of patients' profiles. In that way we have created 10 test data set couples. In addition, each data set has three different representations (views), i.e. it is separated into three different view sets (v_1, v_2, v_3). v_1 records the patient age and gender. v_2 has information about the patient anthropometric features, e.g., such as BMI, WC, HC and WHR. Finally, v_3 contains information about the patient's systolic and diastolic blood pressure levels.

The above multi-view data context can be used to study and associate different age categories with the patient anthropometric measurements and persons' increased risk for cardiovascular disease, e.g., hypertension. For example, it can be useful to find the correlations between an age category with specific anthropometric measurements and high blood pressure levels and further study and try to understand these correlations. Therefore the patient profiles can initially be grouped separately for each view and then the groups from the different views can be associated. However, when new patients data becomes available the groups obtained in the different views are necessary to be adapted in order to reflect the new information. In addition, the new patients data may also affect the groups' associations and even new groups may appear in some views. In this scenario it may be useful to have a technique that is able to continuously adapt the grouping in different views as well as the associations between views when new data arrives.

Table 1: Cluster categories in view 1

Cluster label	Cluster description	Cluster size
v_{10}	Adolescence, male (age $<$ 20)	44
v_{11}	Adolescence, female (age $<$ 20)	63
v_{12}	Early adulthood, male (20 $\leq$ age $\leq$ 39)	124
v_{13}	Early adulthood, female (20 $\leq$ age $\leq$ 39)	157
v_{14}	Adulthood, male (age $>$ 39)	7
v_{15}	Adulthood, female (age $>$ 39)	4

The patients' profiles in v_1 can be initially grouped in six categories based on the individuals' age and sex (see Table 1). In the second view (v_2) the patients are distributed in four body weight categories with respect to their BMI levels. The body weight categories are defined according to WHO BMI cut-offs (see Table 2). In v_3

the values of SBP and DBP are used to group the data into six categories labeled as Level 1, Level 2,. . ., Level 6 as in [22] (see Table 3).

The *MV Split-Merge Clustering* algorithm has been implemented in Python version 3.7. Prepossessing and metrics libraries, NearestCentroid have been used from the Scikit-learn library. In addition, Python library Concepts has been used for FCA [19]. The executable of the proposed algorithm and the experimental results are available on GitLab[1].

Table 2: Cluster categories in view 2

Cluster label	Cluster description	Cluster size
v_{20}	underweight (BMI $\leq$ 18.49)	21
v_{21}	normal weight (18.50 $\leq$ BMI $\leq$ 24.99)	234
v_{22}	overweight (25.00 $\leq$ BMI $\leq$ 29.99)	113
v_{23}	obese (BMI $\geq$ 30.00)	31

Table 3: Cluster categories in view 3

Cluster label	Cluster description	Cluster size
v_{30}	Level 1 (SBP $<$ 120 and DBP $<$ 80)	141
v_{31}	Level 2 (120 $\leq$ SBP $\leq$ 129 and/or 80 $\leq$ DBP $\leq$ 84)	83
v_{32}	Level 3 (130 $\leq$ SBP $\leq$ 139 and/or 85 $\leq$ DBP $\leq$ 89)	67
v_{33}	Level 4 (140 $\leq$ SBP $\leq$ 159 and/or 90 $\leq$ DBP $\leq$ 99)	80
v_{34}	Level 5 (160 $\leq$ SBP $\leq$ 179 and/or 100 $\leq$ DBP $\leq$ 109)	23
v_{35}	Level 6 (SBP $\geq$ 180 and/or DBP $\geq$ 110)	5

5.2 Results and Discussion

We have used the whole data set, explained in Section 5.1, to build a global model that links the three data views in a batch data fashion. This model will be used as a baseline and the clustering solutions generated by applying the proposed multi-view streaming algorithm will be benchmarked to it. Three local clustering models are created for the three data views as it was explained in Section 5.1 (see Table 1, Table 2 and Table 3, respectively). Then a formal context presented by a 399×16 matrix, with the individuals corresponding to the rows and the cluster category labels corresponding to the columns is built. Finally, a formal concept lattice for the built context is generated. It produces a lattice of 147 non-empty concepts where each one represents a subset of individuals who belong to a number of cluster categories. We are interested in those concepts that link clusters from all the three views, e.g., $\{v_{12}$, v_{22}, $v_{34}\}$ is a concept that contains all overweight adult males (concept 10 in Table 4) having Level 5 of the blood pressure category. 74 such concepts are generated. We are interested in studying about people with increased risk for cardiovascular disease. Therefore we have selected concepts with level of the blood pressure above 3, i.e. Level 4, 5 and 6, and cardinality above 2. We have made an exception only for Level 6, since it does not have any concepts with more than 2 instances. These concepts are listed in Table 4.

[1]https://gitlab.com/vishnu.manasa26/mv_splitmerge_algorithm

Table 4: Concepts from the global clustering model generated in the batch data scenario

Blood Pressure Level	S/N	Concept	Concept Size	Blood Pressure
Level 4	1	v_{10}, v_{21}, v_{33}	7	Hyper and Regular
	2	v_{12}, v_{21}, v_{33}	14	Hyper and Regular
	3	v_{13}, v_{21}, v_{33}	15	Pre
	4	v_{10}, v_{22}, v_{33}	3	Hyper
	5	v_{11}, v_{22}, v_{33}	6	Pre
	6	v_{12}, v_{22}, v_{33}	18	Hyper and Regular
	7	v_{10}, v_{23}, v_{33}	3	Hyper and Regular
	8	v_{12}, v_{23}, v_{33}	3	Hyper and Regular
	9	v_{13}, v_{23}, v_{33}	3	Pre
Level 5	10	v_{12}, v_{22}, v_{34}	4	Hyper and Regular
	11	v_{12}, v_{23}, v_{34}	3	Hyper
Level 6	12	v_{13}, v_{21}, v_{35}	2	Pre

Our experiments are carried out according to the experimental setup described in Section 5.1. The streaming scenario is modelled by dividing the data set in two parts: the current data chunk and the newly arriving one. Thus the whole data set is used to generate 10 such test data set couples. The three local clustering models are initially built on the first data set (current chunk) of each test couple as it was described in Section 5.1. The Split-Merge Clustering is then applied to update the clustering solution on each view of the first data set of each test couple by the newly arrived data chunk (its second data set). Hence three updated local clustering models are produced for each test data couple. FCA is then used to integrate these local clustering solutions into a global clustering model of each test data couple.

Out of the ten experimental iterations, explained above, we have selected the results of one to be presented and discussed in detail further in this section. The lattice produced in this iteration has a total of 160 non-empty concepts. Out of these 82 concepts link clusters from all the three views.

Table 5 presents how the concepts in Table 4 are modified or remained unchanged in the stream data scenario, i.e. when the MV Split-Merge Clustering has been used to generate the global clustering model. Note that concepts with cardinality of 2 and above are only listed. It can be observed that there is not much difference in the number of concepts linking clusters from all the three views in the benchmark (74 concepts) and the ones generated by the MV Split-Merge Clustering (82 concepts). In addition, the number of clusters of the three updated local models are the same as those of the corresponding local clustering solutions produced in the batch scenario. It can also be noticed that some concepts have remained unchanged in Table 5, e.g. 11 and 12. In fact, all the concepts of Level 6 are the same as those in the benchmark solution (not shown in the tables). Some concepts however, are partitioned into a few in the concept lattice generated in the stream data scenario. For example, instances of concept 1 in Table 4 are distributed into two different Level 4 concepts in Table 5. There are also concepts that contain instances from other blood pressure levels besides the three studied, e.g. concepts 3, 6 and 9. This is mostly due to the split-merge operations used to update the local clustering models of the first data chunk.

It is interesting to notice that some of patients grouped into concepts with high

Table 5: Concepts from the global clustering model produced in the stream data scenario

Blood Pressure Level	S/N	Concept	Concept Size	Blood Pressure
Level 4	1	v_{10}, v_{21}	2	1 Hyper, 1 Regular
Level 4	1	$v_{10}, (v_{20} \wedge v_{21})$	4	2 Hyper, 2 Regular
Level 4	2	$v_{12}, (v_{20} \wedge v_{21})$	6	4 hyper, 2 Regular
Level 4, 5	2	v_{12}, v_{21}	4	4 Hyper
Level 4	3	$v_{13}, (v_{20} \wedge v_{21})$	6	Pre
Level 3, 4	3	v_{13}, v_{21}	5	Pre
Level 3, 4	4	v_{10}, v_{22}	6	2 Hyper, 4 Regular
Level 4	5	v_{11}, v_{22}	6	Pre
Level 3, 4	6	v_{12}, v_{22}	12	6 Hyper, 6 Regular
Level 4	7	v_{10}, v_{23}	2	Hyper, Regular
Level 4	8	v_{12}, v_{23}	2	Hyper, Regular
Level 1, 2, 4	9	v_{13}, v_{23}	4	Pre
Level 5	11	v_{12}, v_{23}, v_{34}	3	Hyper
Level 6	12	v_{13}, v_{21}, v_{35}	2	Pre

level of blood pressure have a Regular class label, e.g. some overweight (v_{22}) and underweight individuals (v_{20}). This might indicate that those people have implicitly a higher risk factor. Other interesting aspect observed when comparing the results of selected concepts with the benchmark solution is that the proposed algorithm is able to distinguish people who have same level of SBP and DBP with people who have different SBP and DBP levels to a large extent.

The above has motivated us to evaluate additionally the two clustering scenarios with respect to the purity of the generated clustering solutions. For this purpose we first consider how the four main classes (Regular male, Regular female, Hyper and Pre) are distributed among the clusters. The score obtained for the benchmark clustering is 0.89. The average value calculated on the ten conducted iterations of our algorithm is 0.76. We have also evaluated a scenario with the six blood pressure levels (i.e., Level 1 - Level 6) as main classes, where naturally the score generated by the benchmark solution is 1 versus 0.65 for our algorithm. The obtained results are logical taking into account that our algorithm does not have access to the complete information initially. Its performance however, is not very distant from that of the batch scenario. In addition, we have used the adjusted Rand Index [23] to determine the similarity between the partitions generated in the two clustering scenarios as a function of positive and negative agreements in pairwise cluster assignments. It produces a value in the interval [-1, 1], where 1 means that the clustering solutions are identical. It has been calculated for each iteration of our experiment and the average score is 0.44.

The time required to build three local models and a global model for the considered 10 iterations are 5.06 and 16.67 seconds, respectively. Consequently, for a single run using *2.7 GHz Quad-Core Intel Core i7* processor these are 0.51 and 1.67 seconds, respectively. It is interesting to notice that the time needed for building a global model is three times the time for constructing three local models. This is due to the fact that the number of concepts L ($\approx$160) in the conducted experiments is higher than the number of instances N (120) in the new data chunk (see the discussion about the algorithm time complexity in Section 4).

We may conclude that the results obtained by the MV Split-Merge Clustering are comparable to that generated in the batch data scenario. Our algorithm is able to discover an underlying clustering structure and implicit relationships among the clusters of local clustering models. It is also worth to notice that the proposed algorithm is an evolutionary approach designed for stream data scenarios and would require a lot of less computational resources in the long run.

6 Conclusions and Future work

We have proposed a new multi-view stream clustering approach, which is an extension of an existing split-merge evolutionary clustering algorithm to multi-view data applications. The proposed algorithm is shown to be robust to processing, analysing and integrating multi-view streaming data. It is capable of performing vertical and horizontal data integration, i.e. over data chunks of each individual view and over all available views per chunk. In that way the learning model is continuously adapted and improved as more data becomes available. Initial evaluation of the proposed algorithm has been done on a public medical data. The obtained results have been benchmarked against a clustering model generated in a batch data scenario. The clustering structure and discovered relationships among the local clustering models produced by the proposed algorithm are comparable to those identified in the batch scenario.

Our future plans are to pursue further study and evaluate the algorithm potential on richer data sets and in case studies from different application domains. For example, we are interested in evaluating the algorithm on real-world data coming from a smart application monitoring blood glucose level. We also plan to compare the proposed algorithm with some of the other state of the art algorithms applicable to multi-view stream data scenarios.

Acknowledgements

This work is part of the research project ”Scalable resource efficient systems for big data analytics” funded by the Knowledge Foundation (grant: 20140032) in Sweden.

We would like to thank *Milena Angelova*, for her valuable feedback and support during the implementation phase.

References

[1] B. Jiang and et al. “Evolutionary multi-objective optimization for multi-view clustering”. In: *2016 IEEE CEC 2016*. 2016, pp. 3308–3315.

[2] M. Hai and et al. “A Survey of Distributed Clustering Algorithms”. In: *Int. Conf. on Ind. Control and Electronics Engineering*. Aug. 2012, pp. 1142–1145.

[3] L. Zeng and et al. “Distributed data mining: a survey”. In: *Information Technology and Management volume* 13 (2012), pp. 403–409.

[4] V. Boeva and et al. “Bipartite Split-Merge Evolutionary Clustering”. In: *Agents and AI*. Ed. by J. van den Herik and et al. Springer, 2019, pp. 204–223. ISBN: 978-3-030-37494-5.

[5] D. Padmanabhan and A. Jurek-Loughrey. “Multi-View Clustering”. In: *Linking and Mining Heterogeneous and Multi-view Data*. Unsupervised and Semi-supervised Learning. Springer, Dec. 2018, pp. 27–53.

[6] Y. Yang and H. Wang. “Multi-view clustering: A survey”. In: *Big Data Mining and Analytics* 1.2 (June 2018), pp. 83–107. ISSN: 2096-0654.

[7] F. Ye and et al. “New Approaches in Multi-View Clustering”. In: *Recent Applications in Data Clustering*. Aug. 2018.

[8] A. A. Benczúr, L. Kocsis, and R. Pálovics. “Online Machine Learning in Big Data Streams”. In: *CoRR* abs/1802.05872 (2018).

[9] X. Liu and et al. “Late Fusion Incomplete Multi-View Clustering”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 41.10 (2019), pp. 2410–2423.

[10] S. Wang and et al. “Multi-view Clustering via Late Fusion Alignment Maximization”. In: *Proceedings of IJCAI-19*. July 2019, pp. 3778–3784.

[11] Y. Ye and et al. “Incomplete Multiview Clustering via Late Fusion”. In: *Computational Intelligence and Neuroscience* 2018 (Oct. 2018), pp. 1–11.

[12] C. Zhu. “Kappa Based Weighted Multi-View Clustering with Feature Selection”. In: *Proceedings of ICCPR 2018*. ICCPR ’18. Shenzhen, China, 2018, pp. 50–54. ISBN: 978-1-4503-6471-3.

[13] P. Zhou and et al. “Incremental multi-view spectral clustering”. In: *Knowledge-Based Systems* 174 (2019), pp. 73–86. ISSN: 0950-7051.

[14] L. Huang and et al. “MVStream: Multiview Data Stream Clustering”. In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3482–3496.

[15] V. Boeva and et al. “Analysis of Multiple DNA Microarray Datasets”. In: *Springer Handbook of Bio-/Neuroinformatics*. Ed. by N. Kasabov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 223–234. ISBN: 978-3-642-30574-0.

[16] A. Hristoskova, V. Boeva, and E. Tsiporkova. “A Formal Concept Analysis Approach to Consensus Clustering of Multi-Experiment Expression Data”. In: *BMC Bioinformatics* 15 (May 2014), p. 151.

[17] S. K. and A. K. Ch. “Concept Lattice Simplification in Formal Concept Analysis Using Attribute Clustering”. In: *Journal of Ambient Intelligence and Humanized Computing* 10 (2018), pp. 2327–2343. ISSN: 1868-5145.

[18] B. Ganter, G. Stumme, and R. Wille. “Formal Concept Analysis: Foundations and Applications”. In: LNAI, no. 3626, Springer-Verlag, 2005.

[19] S. Bank. *Formal Concept Analysis with Python*. 2019. (Visited on 03/18/2020).

[20] C. Lindig. “Fast concept analysis”. In: *Working with Conceptual Structures – Contributions to ICCS 2000*. Shaker Verlag, 2000, pp. 152–161.

[21] H. F. Golino and et al. “Predicting Increased Blood Pressure Using Machine Learning”. In: *Journal of Obesity* 2014 (2014).

[22] Y. Li and et al. “Clustering of cardiovascular behavioral risk factors and blood pressure among people diagnosed with hypertension: a nationally representative survey in China”. In: *Sci Rep.* 6 (2016).

[23] W. M. Rand. “Objective Criteria for the Evaluation of Clustering Methods”. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. ISSN: 01621459.

Paper III

A Multi-View Clustering Approach for Analysis of Streaming Data

Vishnu Manasa Devagiri, Veselka Boeva, Shahrooz Abghari

In: Artificial Intelligence Applications and Innovations, Ed. by I. Maglogiannis, J. Macintyre; L. Iliadis. Cham: Springer International Publishing, 2021, pp.169–183, DOI: 10.1007/978- 3-030-79150-6_14

Abstract

Data available today in smart monitoring applications such as smart buildings, machine health monitoring, smart healthcare, etc., is not centralized and usually supplied by a number of different devices (sensors, mobile devices and edge nodes). Due to which the data has a heterogeneous nature and provides different perspectives (views) about the studied phenomenon. This makes the monitoring task very challenging, requiring machine learning and data mining models that are not only able to continuously integrate and analyze multi-view streaming data, but also are capable of adapting to concept drift scenarios of newly arriving data. This study presents a multi-view clustering approach that can be applied for monitoring and analysis of streaming data scenarios. The approach allows for parallel monitoring of the individual view clustering models and mining view correlations in the integrated (global) clustering models. The global model built at each data chunk is a formal concept lattice generated by a formal context consisting of closed patterns representing the most typical correlations among the views. The proposed approach is evaluated on two different data sets. The obtained results demonstrate that it is suitable for modelling and monitoring multi-view streaming phenomena by providing means for continuous analysis and pattern mining.

Keywords: Multi-View Clustering, Multi-Instance Learning, Closed Patterns, Streaming data, Formal Concept Analysis.

1 Introduction

In recent years, the amount of data being generated in areas such as web, social media, IoT, and smart monitoring applications is increasing rapidly. Data generated in most of these areas is usually heterogeneous as the data is generally collected at different locations using variety of devices (e.g., mobile devices, edge nodes, sensors in IoT networks) and/or streaming in nature as new data is continuously produced. Another common factor of the data generated in streaming scenarios is its evolving nature. Change of data characteristics over a period of time, known as concept drift, is an important challenge to be addressed when dealing with streaming data.

Clustering techniques are well-known tools and broadly used for analysis and extraction of interesting patterns from unlabeled data sets. Traditional clustering algorithms however, are not suitable and cannot deal with the data generated in today's smart monitoring applications due to characteristics already mentioned above like heterogeneity, streaming nature, concept drift [1]. There is a need for new clustering algorithms that are able to address these challenges. Data stream mining is an area dealing with the challenges concerning analysis and understanding of streaming data scenarios. Multi-view clustering, a distributed clustering technique, is capable of analysing heterogeneous data that are generated by different sources and represents different views or perspectives about the studied phenomenon. In multi-view clustering scenarios different views, contexts or interpretations of the data bringing complementary information (e.g., numerical reports of a patient and reports like ECG), are analysed in order to extract meaningful correlations among the different views. Although many research studies have been conducted and published in both data stream mining and multi-view clustering fields, the area of multi-view stream clustering is still in its infancy and there is a need for clustering techniques addressing and analysing streaming data in a multi-view fashion [2, 3]. Some of the major challenges of multi-view stream clustering techniques are data heterogeneity [4], incomplete views [3, 5, 6] and evolving nature of the data [2].

In this work, we propose a multi-view clustering algorithm, entitled MV Multi-Instance Clustering, that can be used for monitoring and continuous analysis of streaming data scenarios. The proposed algorithm allows for parallel monitoring of the individual view clustering models and analyzing the views' correlations revealed by the integrated (global) clustering model. The individual view clustering models at each data chunk are initially updated when new data arrives by applying multi-instance clustering. Then, a global model can be built at each data chunk as a formal concept lattice generated by a formal context. The latter consists of selected closed patterns presenting the most typical correlations among the different views. Such a hierarchical global model allows to analyse and compare the views' correlations derived by two consecutive data chunks. Note that the local models' data values are not needed in order to build the global model which supports data privacy and lowers the required memory for data processing. In addition, if there are missing data in some

of the views the previously extracted correlations among the views could be used to reconstruct the missing values.

2 Related Work

Distributed clustering techniques can deal with large, unlabelled and heterogeneous data sets which cannot be gathered centrally [7–10]. Characteristics of distributed data like heterogeneity, scalability, security, etc., demand novel robust clustering algorithms to address these challenges [8]. While some researchers [7] have tried to tackle various challenges in the field, others [8–10] have proposed an overview of the research being done. Gan et al. [8] discuss various challenges and provide a summary on the state-of-the-art distributed clustering techniques. The authors cover various important concepts in the field of data mining like frequent itemset mining, frequent sequence mining, frequent graph mining, clustering and privacy for distributed context. In [9, 10], a comparative study on the various state-of-the-art distributed clustering techniques has been done. Bendechache and Kechadi [7] propose an algorithm, entitled Distributed Dynamic Clustering algorithm, which is based on k-means for spatial data that is distributed and heterogeneous.

Multi-view clustering deals with clustering techniques in which same data is available in different perspectives or views complementing each other [3]. Studies published in [1, 4], provide an overview and analysis of different multi-view clustering techniques proposed. Fu et al. [1] evaluate the selected multi-view algorithms on seven real-world data sets using cluster validation metrics like accuracy, purity, and normalized mutual information. In [4], the authors have reviewed available multi-view clustering algorithms by grouping them into five categories. In series of papers [3, 5, 6], the authors address the challenges of incomplete views, where data in some views maybe missing. Shao et al. [3] develop an algorithm, entitled Online Multi-View clustering, based on non-negative matrix factorization for large scale incomplete distributed data sets.

It is interesting to note that in [11], the authors treat multi-view clustering as a multi-objective optimization problem. In [12], a multi-view clustering approach based on non-negative matrix factorization and probabilistic latent semantic analysis is proposed to obtain common consensus clustering across views. Research in [2, 3] deals with streaming data in multi-view scenarios. Huang et al. [2] propose a novel multi-view clustering approach for streaming data.

In the current state-of-the-art algorithms for multi-view clustering there are not many solutions dealing with monitoring and analysis of streaming data and the challenges that come along with it. The proposed MV Multi-Instance Clustering algorithm address these challenges.

3 Background

3.1 Multi-Instance Clustering and Hausdorff Distance

Multi-Instance (MI) clustering is an unsupervised learning process, where the data objects are bags of instances and there is no information about the labels of bags [13]. This is a typical setting for many real world application scenarios in which, it is costly and even in many cases impossible to obtain labeled data.

Multi-Instance clustering algorithms are supposed to partition a set of unlabeled bags into a number of groups on the basis of a similarity measure. However, the task of distributing objects into clusters is more difficult in the multi-instance context, since the ambiguity due to the fact that the objects are bags of unlabeled often related instances. In this sense, the similarity measures used in single-instance clustering may not be appropriate for multi-instance clustering scenarios. *Maximal Hausdorff distance* has been proposed in [14] to measure the distance between two bags and later successfully applied to the standard multi-instance learning problem [15]. However, in [13] the maximal Hausdorff distance has been found to not work well in the generalized multi-instance learning problems due to its sensitivity to outliers. Therefore, the authors have proposed another distance called *average Hausdorff distance*.

In this paper, we use average Hausdorff distance to measure the distance between two bags A and B, since the preliminary experiments with this distance have generated better results than the ones produced by the maximal Hausdorff distance. Formally, given two bags of data instances A and B, the *average Hausdorff distance* is defined by Eq. III.1, where $dist(a, b)$ is the distance between instances $a \in A$ and $b \in B$, which usually takes the form of Euclidean distance, and $\mid . \mid$, represents the set cardinality.

$$H(A, B) = \frac{\sum_{a \in A} \min_{b \in B} dist(a, b) + \sum_{b \in B} \min_{a \in A} dist(a, b)}{\mid A \mid + \mid B \mid}. \qquad \text{(III.1)}$$

3.2 Formal Concept Analysis

Formal Concept Analysis (FCA) [16] is a mathematical apparatus for deriving a concept hierarchy from a collection of objects and their properties. FCA allows to generate and visualize the concept hierarchies. FCA is used for data analysis, information retrieval, and knowledge discovery. In addition, it can be understood as conceptual clustering method, which clusters simultaneously objects and their descriptions.

FCA derives a concept lattice from a formal context constituted of a set of objects O, a set of attributes A, and a binary relation defined on the Cartesian product $O \times A$. The context is described as a table, the rows correspond to objects and the columns to attributes or properties and a cross in a table cell means that "an object possesses a property". The *concept lattice* is composed of formal concepts organized into a hierarchy by a partial ordering (a subsumption relation allowing to compare

concepts). Intuitively, a concept is a pair (X, Y) where $X \subseteq O$, $Y \subseteq A$, and X is the maximal set of objects sharing the whole set of attributes in Y and vice-versa. Relying on the subsumption relation, the set of all concepts extracted from a context is organized within a complete lattice, which means that for any set of concepts there is a smallest super-concept and a largest sub-concept, called the *concept lattice*.

3.3 Closed Patterns

Sequential pattern mining is the problem of finding interesting frequent ordered patterns from a sequence database [17]. Given a sequence database $\mathcal{T}$ and a pattern α the support for α is the number of sequences in $\mathcal{T}$ that contain α as a sub-sequence. The pattern α is called frequent if its support is equal or greater than a user-specified support threshold. Mining frequent patterns in big databases can lead to generating a large number of patterns. In order to mitigate this problem, one can only extract frequent closed sequential patterns. A pattern α is closed when none of its super patterns has the same support as α.

In this study, we apply BIDE [18], which is a famous frequent closed sequential pattern mining algorithm, to extract patterns. The Python implementation of BIDE is adopted from prefixspan library.

4 MV Multi-Instance Clustering using Closed Patterns

In [19], an extension of the Split-Merge Evolutionary Clustering algorithm (abbreviated Split-Merge Clustering) [20] for multi-view data streaming scenarios has been introduced. The introduced algorithm, MV Split-Merge Clustering, has been demonstrated to be able to integrate data from multiple views in a streaming manner. The algorithm can be applied for grouping distinct chunks of multi-view streaming data so that a global clustering model is built on each data chunk. Initially, an updated clustering solution (local model) is produced on each view of the current data chunk by applying the Split-Merge Clustering. In that way updated local models reflecting the information presented in the current and previous data chunks are obtained. FCA is then used in order to integrate information from the local clustering models and generate a global model that reveals the relationships among the local models.

We have recognized two main limitations of the MV Split-Merge Clustering [19]. First, the Split-Merge Clustering algorithm [20], used for updating the local clustering models, needs to find the cluster centroids in order to integrate the local models of two consecutive data chunks. Our proposed MV Multi-Instance Clustering algorithm overcomes this by interpreting the integration of two local models as a Multi-Instance clustering problem, i.e. each cluster (bag) is regarded as an atomic object. Evidently,

by exploiting Multi-Instance clustering analysis, we enable to improve the performance of the algorithm and also handle the ambiguity which is typical for real-world streaming data. For example, we would be able to model semi-supervised learning scenarios where some bags may be labeled. Second, the MV Split-Merge Clustering [19] builds a global model by using all the identified correlation patterns among the views. This leads to the generation of a large and complex concept lattice that is not easy to be interpreted and analysed. In comparison, our MV Multi-Instance clustering algorithm uses closed patterns, which considers the most typical correlations among the views, to create a global clustering model. In this way, unimportant concepts are excluded and do not complicate the understanding and analysis of the built global model. In addition, there is an opportunity to obtain even a smaller set of the most frequent (top-ranked) patterns based on the frequency or support score associated with each closed pattern.

Let us formally describe our MV Multi-Instance Clustering algorithm. We consider a streaming scenario where a particular phenomenon (physical object, biological process, machine asset, patient etc.) is monitored under n different circumstances (views). We further assume that the data arrives over time in chunks. Each chunk t can contain different number of data points and can be represented by a list of n different data matrices $D_t = \{D_{t1}, D_{t2}, \ldots, D_{tn}\}$, one per view. Each matrix D_{ti} ($i = 1, 2, \ldots, n$) contains the information about the data points in the current chunk t with respect to the corresponding view i. Assume that chunk t contains N_t data points. In addition, n clustering models, one per view, can be built on each data chunk. Let $C_t = \{C_{t1}, C_{t2}, \ldots, C_{tn}\}$ be a set of clustering solutions (local models), such that C_{ti} ($i = 1, 2, \ldots, n$) represents the grouping of the data points in tth chunk with respect to ith view, i.e. a local model built on data set D_{ti}.

On each data chunk, the proposed algorithm conducts two main operations. They are described in Algorithms III.2 and III.3. The local models built on the current chunk C_t are first updated by analysing the newly arrived data D_{t+1}. Clustering solutions C_{t+1} are initially built on the new data chunk $t + 1$ and correlated with ones of chunk t in order to generate updated clustering models $C_t^{'}$ with respect to $t + 1$. Then, these local models $C_t^{'}$ are used to build a global model that consists of three parts providing information about different aspects of the studied phenomenon. Namely, the model includes the formal context, closed patterns and concept lattice. The latter two are generated based on the built formal context. The formal context F_t consists of the set of $(N_t + N_{t+1})$ data points, the set of K ($K = k_1 + k_2 + \ldots + k_n$) clustering labels of $C_t^{'}$ and an indication of which data points are associated with which clusters. Thus the context is described as a matrix, with the data points corresponding to the rows and the cluster labels corresponding to the columns of the matrix, and a value 1 in cell (i, j) whenever data point i belongs to cluster $C_j^{'}$ ($j = 1, 2, \ldots, K$). Evidently, the formal context F_t contains all view correlation patterns supported by the local clustering models. The set of closed patterns, denoted by F_t^c, contains the most typical correlations that exist among the views. Finally, the

concept lattice provides description of the hierarchical organisation of the concepts it produces.

The operations for updating the local clustering models on data chunk t are given in Algorithm III.1.

Algorithm III.1 Use Bi-Correlation MI-Clustering to update the local clustering models on data chunk t

Input local clustering models C_t and newly arrived data D_{t+1}
for each view i $(i = 1, 2, \ldots, n)$ **do**
  Build a clustering model $C_{(t+1)i}$
  Bi-Correlation MI-Clustering $(C_{ti}, C_{(t+1)i})$ (Algorithm III.2)
end for

Algorithm III.2 describes Bi-Correlation MI-Clustering that is applied for updating the local clustering models on data chunk t. Average Hausdorff distance (see Section 3.1) is used to find the correlations between the two clustering solutions C_{ti} and $C_{(t+1)i}$ for each view i $(i = 1, 2, \ldots, n)$. Global threshold T_i (see Eq. III.2) is calculated for each $\mid C_{ti} \mid \times \mid C_{(t+1)i} \mid$ adjacency matrix as follows:

$$T_i = \frac{\sum_{p \in C_{ti}} \min_{q \in C_{(t+1)i}} H(p,q) + \sum_{q \in C_{(t+1)i}} \min_{p \in C_{ti}} H(p,q)}{\mid C_{ti} \mid + \mid C_{(t+1)i} \mid}, \qquad \text{(III.2)}$$

where $H(p,q)$ is the average Hausdorff distance (see Eq. III.1) between a cluster $p \in C_{ti}$ and a cluster $q \in C_{(t+1)i}$. T_i averages the Hausdorff distances between each cluster in C_{ti} and its nearest cluster in $C_{(t+1)i}$ and vice-versa. Evidently, T_i measures the average Hausdorff distance between two clustering solutions.

Algorithm III.2 Bi-Correlation MI-Clustering of C_{ti} and $C_{(t+1)i}$

Input local clustering models C_{ti} and $C_{(t+1)i}$
Build a $\mid C_{ti} \mid \times \mid C_{(t+1)i} \mid$ adjacency matrix based on Hausdorff distance (Eq. III.1)
Calculate global threshold T_i (Eq. III.2)
Remove edges in the adjacency matrix for which $H(p,q) > T_i$
for each uniformly random cluster p in C_{ti} **do**
  Find average distance of adjacent nodes, denoted by T_i^p
  Remove edges in adjacency matrix for which $H(p,q) > T_i^p$
  Find neighbours of p in $C_{(t+1)i}$, denoted by $N^p_{(t+1)i}$
  Find neighbours of each $q \in N^p_{(t+1)i}$ in C_{ti}, denoted by N^q_{ti}
  Create cluster $C'_p = \{p\} \cup N^p_{(t+1)i} \cup_{q \in N^p_{(t+1)i}} N^q_{ti}$
  $C_{ti} = C_{ti} \setminus \{p\}$
end for

Algorithm III.3 Use FCA and closed patterns to build a global model on data chunk t

Input updated local clustering models $C_t^{'}$
Build a formal context, denoted by F_t. F_t is a $(N_t + N_{t+1}) \times K$ binary matrix that indicates for each data point belonging to $D_t \cup D_{t+1}$ which clusters of $C_t^{'}$ it is associated with
Derive closed patterns, denoted by F_t^c ($F_t^c \subset F_t$), from the set of all built patterns of F_t
Produce a formal concept lattice from F_t^c

The adjacency matrix can also be visualized as a bipartite graph to illustrate how the clusters are correlated. The nodes on the left side of the graph represent clustering solution of chunk t, i.e. C_{ti}, and those on the right hand side represents new clustering solution i.e. $C_{(t+1)i}$. T_i is used to filter out the edges between clusters which are far apart and thus avoiding considering too many clusters to decide which ones to merge. The average local distance T_i^p could be considered as the local threshold for each cluster p in C_{ti} and it is used to find its closest clusters in $C_{(t+1)i}$. The motivation of using T_i^p is that, it facilitates identifying new trends in the scenarios of concept drift, where a group of data points can form a new cluster by slowly moving away from their current cluster at each data chunk. By considering the average local distance as a merging condition, we avoid early merging which allows such new clusters to naturally form.

5 Evaluation

5.1 Data Sets and Experimental Setup

5.1.1 Anthropometric data:

Initial analysis is done on a comparatively small public data set [21] that describes the medical conditions of 399 undergraduate students based on their anthropometric data. Each student is described by the following features: age, obesity, body mass index (BMI), waist circumference (WC), hip circumference (HC), and waist hip ratio (WHR), Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), *preh* for women and *hyper* for men, where the *preh* and *hyper* are classification labels that show what kind of blood pressure the individual has (e.g., regular or hyper). In order to mimic the streaming data scenario required for the proposed algorithm, the data set is divided into historical and newly arriving data. The historical data set is composed of the 70% of total data and the remaining 30% is treated as the newly arriving data. The features of the data set are divided into three views, where view 1 (v_1) contains details about age and gender, view 2 (v_2) contains details about BMI, WC, HC, WHR, and view 3 (v_3) presents information about blood pressure (SBP,

DBP). Initial grouping of data points in each view is given in Table 1.

Table 1: Cluster categories in the views of Anthropometric data set

Label	Cluster description	Size
v_{10}	Adolescence, male (age $<$ 20)	44
v_{11}	Adolescence, female (age $<$ 20)	63
v_{12}	Early adulthood, male (20 $\leq$ age $\leq$ 39)	124
v_{13}	Early adulthood, female (20 $\leq$ age $\leq$ 39)	157
v_{14}	Adulthood, male (age $>$ 39)	7
v_{15}	Adulthood, female (age $>$ 39)	4
v_{20}	underweight (BMI $\leq$ 18.49)	21
v_{21}	normal weight (18.50 $\leq$ BMI $\leq$ 24.99)	234
v_{22}	overweight (25.00 $\leq$ BMI $\leq$ 29.99)	113
v_{23}	obese (BMI $\geq$ 30.00)	31
v_{30}	Level 1 (SBP $<$ 120 and DBP $<$ 80)	141
v_{31}	Level 2 (120 $\leq$ SBP $\leq$ 129 and/or 80 $\leq$ DBP $\leq$ 84)	83
v_{32}	Level 3 (130 $\leq$ SBP $\leq$ 139 and/or 85 $\leq$ DBP $\leq$ 89)	67
v_{33}	Level 4 (140 $\leq$ SBP $\leq$ 159 and/or 90 $\leq$ DBP $\leq$ 99)	80
v_{34}	Level 5 (160 $\leq$ SBP $\leq$ 179 and/or 100 $\leq$ DBP $\leq$ 109)	23
v_{35}	Level 6 (SBP $\geq$ 180 and/or DBP $\geq$ 110)	5

Our objective is to use this data set to build controlled and easy to interpret experimental multi-view streaming scenarios for studying and comparing the two multi-view clustering algorithms described in Section 4. In this setup two experiments are conducted to evaluate the algorithms. Bi-Correlation MI-Clustering step of the proposed algorithm is initially compared with Split-Merge Clustering [19] for updating the local models. We also analyse how different views are related to each other using the closed patterns derived from the global model.

5.1.2 Real-world sensor data:

The potential of the proposed approach is also demonstrated on a real-world data set from a company in the smart building domain. The data has been used in [22] for analysing and monitoring the control valve system behaviour. In smart building domain different types of metrics are collected from a wide range of sensors available for systems such as heating, ventilation, air conditioning, and refrigeration. Data covering a year period (Jan 1^{st} 2019 till Dec 27^{th} 2019) is used in the current study. The eight features listed in Table 2, seven of which also considered in [22], are used in our experiments.

Table 2: Features included in the real-world sensor data set

View	Id	Acronyms	Feature name	Units
Operation	1	SST	Secondary Supply Temperature	°C
	2	SRT	Secondary Return Temperature	°C
	3	PHL	Primary Heat Load	kW
Performance	4	VOM	Valve Openness Mean	%
	5	VOS	Valve Openness Standard Deviation	%
	6	SE	Sub-station Efficiency	%
Context	7	OTM	Outdoor Temperature Mean	°C
	8	OTS	Outdoor Temperature Standard Deviation	°C

The available data features are analysed and partitioned in three distinctive views:

system operational behaviour parameters, performance indicators and contextual factors. The features SST, SRT, and PHL are selected to model the system typical operational behaviour. The system performance can be evaluated by these three indicators: VOM, VOS, and SE. Finally, the contextual factors are represented by the features: OTM and OTS.

5.2 Results and Discussion

5.2.1 Anthropometric data:

This data can be used to study and associate different age categories with the patients' anthropometric measurements to identify patients with increased risk for cardiovascular disease, e.g., hypertension. The data set is used to generate 10 test data set couples by randomly separating the individual profiles into two sets, as it was explained in Section 5.1.1. Thus the first set (279 patients) of each couple presents the current data chunk of individual profiles, and the other one (120 individuals) is the new chunk of patients' profiles. In that way, we have created 10 test data set couples.

MV Split-Merge Clustering and MV Multi-Instance Clustering are applied and compared on the built 10 test data sets. For MV Multi-Instance Clustering, we have additionally studied and conducted the experiments with two different (maximal Hausdorff versus average Hausdorff) distance measures in order to select the better one, i.e. we have done 20 experiments in total. The average Hausdorff distance has outperformed the maximal Hausdorff distance on all the 10 test data sets. This confirms the discovery in [13], hence we have chosen to use this distance measure in the definition of Algorithm III.2 and discuss its experimental results further in this section.

Out of the ten experimental iterations of the proposed approach, we have selected the results of one of the iterations (same as the one in [19]) to be presented and discussed in detail further in this section. The lattice produced in this iteration by MV Split-Merge Clustering has a total of 160 non-empty concepts. Out of these, 82 concepts link clusters from all the three views. The lattice size generated by MV Multi-Instance Clustering on the built formal context is very similar, namely it has 165 non-empty concepts, out of which 83 concepts link clusters from all three views. In the considered iteration, the local models generated in the three views have $6, 5$ and 7 clusters, respectively. In view 1, the clusters presented in Table 1 are retained, i.e. the same six age categories. In view 2, the cluster presenting all individuals with obese weight (v_{23}) has been split into two different clusters and similarly with the individuals having blood pressure Level 5 (v_{34}) in view 3. It can be observed that most of the original clustering structure is retained with the proposed MV Multi-Instance Clustering.

We compare MV Split-Merge Clustering [19] and MV Multi-Instance Clustering algorithms with respect to the purity of the produced clustering solutions. For this

Table 3: Closed patterns showing correlations between all three views (support 10)

Blood Pressure Level	S/N	Concept	Size	Blood Pressure
Level 1	1	v_{13}, v_{21}	53	Regular
(v_{30})	2	v_{11}, v_{21}	26	Regular
	3	v_{13}, v_{22}	15	Regular
	4	v_{12}, v_{21}	12	Regular
Level 2	5	v_{13}, v_{21}	22	Regular, Pre
(v_{31})	6	v_{12}, v_{21}	15	Regular
Level 3	7	v_{12}, v_{21}	18	Regular
(v_{32})	8	v_{12}, v_{22}	12	Regular
	9	v_{13}, v_{21}	10	Pre, 1 Regular
Level 4	10	v_{12}, v_{22}	18	Regular, Hyper
(v_{33})	11	v_{13}, v_{21}	15	Pre
	12	v_{12}, v_{21}	14	Regular, Hyper

purpose, we first consider how the four main classes (Regular male, Regular female, Hyper and Pre) are distributed among the clusters. The average value calculated on the ten conducted iterations of MV Split-Merge Clustering algorithm is 0.76. While the corresponding value generated by the proposed MV Multi-Instance Clustering is 0.895. We have also evaluated the two algorithms with the six blood pressure levels (Levels 1 to 6) as main classes, where the score generated by the MV Multi-Instance Clustering is 1.0 versus 0.65 for the MV Split-Merge Clustering algorithm. The MV Multi-Instance Clustering has demonstrated a better performance in the both evaluation scenarios, i.e. it is able to detect more efficiently the correlations between the current and new incoming data chunks. We have further used the adjusted Rand Index [23] to determine the similarity between the partitions generated by MV Multi-Instance clustering algorithm and benchmark clustering (used in [19]) as a function of positive and negative agreements in pairwise cluster assignments. The average score produced by the proposed algorithm is 0.99 versus 0.44 for the MV Split-Merge Clustering.

We are interested in discovering the relationships among the views. Hence we have specially studied patterns with length 2 or 3 in order to reveal correlations that exist between two or three views. Closed patterns (see Section 3.3) have been used for this purpose. We have generated closed patterns with support 10 (patterns that cover $\approx 2.5\%$ of the data set) which resulted in 43 concepts of which 12 patterns show the relationship between all the three views. Table 3 lists all 12 derived concepts from the retrieved closed patterns as examples to study the relations among the views. The generated closed patterns do not contain concepts where the blood pressure levels are either 5 or 6. This could be due to the less number of instances in these clusters which are 23 and 5, respectively (see Table 1).

It is interesting to notice that the first three top frequent patterns among the three views (see rows 1, 2 and 5 in Table 3) represent typical categories in female population: females of age between 20 and 39 (early adulthood) with Level 1 blood pressure and normal weight; females in the same blood pressure and weight group, but in adolescent age category (age less than 20); and females again in early adulthood and

normal weight category, but Level 2 blood pressure. In comparison with these categories, the three least frequent concepts (see rows 4, 8 and 9 in Table 3) can also be considered. For example, rows 8 and 9 represent respectively, overweight males and normal weight females in early adulthood age category with Level 3 blood pressure. One can get further insight into the discussed concepts by analyzing frequent concepts that connect two views (not included in Table 3). For example, it can be observed that females in their early adulthood typically have normal weight. This is demonstrated by a concept with support 104. Females in this age group are less likely to be overweight (31 individuals) or underweight (11 individuals) since only small size concepts supports this. In addition, the individuals in this female age category are less likely to be obese as there is no concept with size above 10 to support this.

5.2.2 Real-world sensor data:

This data can be used for modelling, understanding and monitoring the control valve system behaviour. For example, it would be useful if one can link or trace back certain performance to specific operational modes by taking into account the influence of contextual factors (e.g., outdoor temperature). Initially, the available data features are partitioned in three distinctive views as it is explained in Section 5.1.2. For each view averaged daily values of the corresponding features are calculated to build daily profiles. The created daily profiles (361 in total) are then split into two parts in order to simulate two data chunks: the initial one with 243 daily profiles (January - August) used to build the system behaviour model and the new data chunk used for the model update contains 118 daily profiles (September - December).

Table 4: Closed patterns correlating all three views after the new data chunk is added

S/N	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Month	Size
1	2.42	26.24	26.01	0.03	±0.06	58	21.34	±0.48	6, 7, 8	36
2	4.39	27.41	26.61	3.25	±1.13	68	17.77	±0.46	6, 7, 8	55
3	5.54	28.13	26.93	4.76	±1.16	76	16.13	±0.35	9	12
4	7.45	29.70	28.15	6.34	±0.99	77	15.45	±0.50	5	14
5	**3.00**	31.35	29.30	7.05	±1.09	86	13.48	±0.56	4	9
6	13.26	35.90	32.36	11.87	±0.65	87	10.65	±0.44	9	11
7	15.65	37.01	33.58	12.18	±0.41	92	9.48	±0.30	10, 11	13
8	16.74	37.81	33.61	12.85	±0.60	91	9.17	±0.49	5	12
9	**3.36**	41.61	35.33	14.99	±0.63	95	6.19	±0.44	3, 4	40
10	20.93	43.13	37.86	13.26	±0.45	95	5.30	±0.34	10, 11	32
11	20.75	43.68	38.36	13.23	±0.45	95	4.81	±0.30	12	16
12	37.45	47.36	38.29	17.37	±0.42	96	1.17	±0.40	3	11
13	42.54	48.20	38.49	18.04	±0.54	96	0.46	±0.33	1,2	54
									Total	315

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 2. Row enumerations in bold italic represent patterns repeated from the initial chunk. The bold in PHL column represents deviating behavior.

Initial clustering in views 1 and 2 is done by applying k-means. Silhouette index and elbow method are used to find the optimal number of initial clusters. In the initial data chunk, the optimal number of clusters in these two views are 3 and 4, respectively. In the second data chunk, the optimal number of clusters for view 1 is

4 while for view 2 is the same as in the initial chunk, i.e. 4. In view 3, the data points are grouped into 4 clusters according to the yearly seasons based on [24], i.e. the context view has the following four clusters: December to February (winter); March, April, October and November (early spring, late autumn); May and September (late spring, early autumn); June to August (summer). After applying MV Multi-Instance clustering algorithm to update local models the number of clusters in the three views are 5, 5 and 6, respectively.

In order to analyse how the correlations among the views are updated, the global model built on the initial data chunk is compared with the one produced on the updated clustering solutions when the new data chunk is added. The lattice built on the initial local models generated 32 non-empty concepts and 14 concepts connecting all the three views. The new global model produced on the updated local models, after the new data chunk has arrived, contains 59 non-empty concepts from which 26 concepts connect all three views. We further compare the sets of closed patterns produced by the corresponding formal contexts using one and the same support ($\approx 2.5\%$ of the data set). The latter gives 18 concepts (support 6) connecting two or three views for initial data chunk and 32 concepts (support of 9), respectively on the second formal context. Table 4 lists all 13 concepts linking three views extracted after adding the new data chunk. Each concept is presented by its mean vector and additionally, the concepts are grouped into two groups with respect to the contextual view, i.e. average outdoor temperature above and below 10 °C. It is interesting to notice that 8 (rows 1, 2, 4, 5, 8, 9, 12 and 13) of these 13 concepts have been discovered by analysing the initial data chunk and they are the only discovered concepts with the same support linking the three views. By considering the concepts linking two views we observe that they are retained in the global model built on the new data chunk and are further expanded with data points from its. Evidently, the integration procedure of our algorithm demonstrates to have a stable behaviour with respect to discovered patterns. Five new patterns presented in the new data chunk have also been extracted, i.e. the proposed algorithm can be used as a continuous data mining technique. The newly discovered patterns may be labelled with the expected performance under a particular context. In addition, our results are comparable to the ones reported in [22], where 49 days in March and April have been marked as having deviating behaviour. Our algorithm presents those with two different concepts (5 and 9 in Table 4), both of which show sudden drop in PHL with respect to the other concepts in the same contextual group.

6 Conclusion and Future Work

In this study, we have proposed a novel multi-view clustering approach, entitled MV Multi-Instance Clustering, that uses average Hausdorff distance, closed patterns and Formal Concept Analysis for analysis of streaming data. The MV Multi-Instance

Clustering allows for parallel monitoring of the individual view clustering models and analysing view correlations in the global model generated at each data chunk.

The proposed algorithm has been evaluated on two different data sets. In addition its performance has been benchmarked to MV Split-Merge Clustering. The MV Multi-Instance Clustering has outperformed the latter algorithm in the studied evaluation scenarios. In general, the obtained results have demonstrated that the proposed algorithm is a robust technique for modelling and continuous analysis and mining of streaming data.

The potential of the MV Multi-Instance Clustering has been demonstrated on real-world data from smart building domain. Our future aim is to pursue further evaluation and study whether the proposed approach is fit for other real-world distributed streaming scenarios.

References

[1] L. Fu, P. Lin, A. V. Vasilakos, and S. Wang. "An overview of recent multi-view clustering". In: *Neurocomputing* 402 (2020), pp. 148–161. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.02.104.

[2] L. Huang and et al. "MVStream: Multiview Data Stream Clustering". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3482–3496.

[3] W. Shao and et al. "Online multi-view clustering with incomplete views". In: *2016 IEEE Int. Conf. on Big Data (Big Data)*. 2016, pp. 1012–1017.

[4] Y. Yang and H. Wang. "Multi-view clustering: A survey". In: *Big Data Mining and Analytics* 1.2 (June 2018), pp. 83–107. ISSN: 2096-0654.

[5] X. Liu and et al. "Late Fusion Incomplete Multi-View Clustering". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 41.10 (2019), pp. 2410–2423.

[6] Y. Ye and et al. "Incomplete Multiview Clustering via Late Fusion". In: *Computational Intelligence and Neuroscience* 2018 (Oct. 2018), pp. 1–11.

[7] M. Bendechache and M.-T. Kechadi. "Distributed clustering algorithm for spatial data mining". In: *2015 2nd IEEE ICSDM* (2015).

[8] W. Gan and et al. "Data mining in distributed environment: a survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7.6 (2017).

[9] M. Hai and et al. "A Survey of Distributed Clustering Algorithms". In: *2012 Int. Conf. on Industrial Control and Electronics Engineering*. 2012, pp. 1142–1145.

[10] D. Singh and A. Gosain. “A Comparative Analysis of Distributed Clustering Algorithms: A Survey”. In: *2013 Int. Symp. on Comp. and Business Intellig.* 2013, pp. 165–169.

[11] B. Jiang and et al. “Evolutionary multi-objective optimization for multi-view clustering”. In: *2016 IEEE CEC 2016*. 2016, pp. 3308–3315.

[12] J. Liu and et al. “Multi-view clustering via joint non-negative matrix factorization”. In: *Proceedings of the 2013 SIAM International Conference on Data Mining, SDM 2013*. 2013, pp. 252–260.

[13] M. Zhang and Z. Zhou. “Multi-instance clustering with applications to multi-instance prediction”. In: *Applied Intelligence* 31 (2009), pp. 47–68.

[14] G. Edgar. *Measure, Topology, and Fractal Geometry, 3rd*. edn. Springer, Berlin, 1995.

[15] J. Wang and J.-D. Zucker. “Solving the multiple-instance problem: a lazy learning approach”. In: *Proc. of the 17th ICML*. 2000, pp. 1119–1125.

[16] B. Ganter, G. Stumme, and R. Wille. “Formal Concept Analysis: Foundations and Applications”. In: LNAI, no. 3626, Springer-Verlag, 2005.

[17] R. Agrawal and R. Srikant. “Mining sequential patterns”. In: *Proc. of the 11th Int. Conf. on Data Engineering*. IEEE. 1995, pp. 3–14.

[18] J. Wang and J. Han. “BIDE: efficient mining of frequent closed sequences”. In: *Proceedings of the 20th International Conference on Data Engineering*. 2004, pp. 79–90.

[19] V. M. Devagiri, V. Boeva, and E. Tsiporkova. “Split-Merge Evolutionary Clustering for Multi-View Streaming Data”. In: *Procedia Computer Science* 176 (2020), pp. 460–469.

[20] V. Boeva and et al. “Bipartite Split-Merge Evolutionary Clustering”. In: *Agents and AI*. Ed. by J. van den Herik and et al. Springer, 2019, pp. 204–223.

[21] H. F. Golino, L. S. de Brito Amaral, S. F. P. Duarte, and et al. “Predicting Increased Blood Pressure Using Machine Learning”. In: *Journal of Obesity* 2014 (2014).

[22] A. Eghbalian and et al. “Multi-view Data Mining Approach for Behaviour Analysis of Smart Control Valve”. In: *Proc. of 19th IEEE ICMLA*. 2020, pp. 1238–1245.

[23] W. M. Rand. “Objective Criteria for the Evaluation of Clustering Methods”. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. ISSN: 01621459.

[24] H. Gadd and S. Werner. “Heat load patterns in district heating substations”. In: *Applied Energy* 108 (2013), pp. 176–183. ISSN: 0306-2619.

Paper IV

Multi-View Data Analysis Techniques for Monitoring Smart Building Systems

Vishnu Manasa Devagiri, Veselka Boeva, Shahrooz Abghari, Farhad Basiri, Niklas Lavesson
In: Sensors 21, 2021, DOI: 10.3390/s21206775

Abstract

In smart buildings, many different systems work in coordination to accomplish their tasks. In this process, the sensors associated with these systems collect large amounts of data generated in a streaming fashion, which is prone to concept drift. Such data are heterogeneous due to the wide range of sensors collecting information about different characteristics of the monitored systems. All these make the monitoring task very challenging. Traditional clustering algorithms are not well equipped to address the mentioned challenges. In this work, we study the use of MV Multi-Instance Clustering algorithm for multi-view analysis and mining of smart building systems' sensor data. It is demonstrated how this algorithm can be used to perform contextual as well as integrated analysis of the systems. Various scenarios in which the algorithm can be used to analyze the data generated by the systems of a smart building are examined and discussed in this study. In addition, it is also shown how the extracted knowledge can be visualized to detect trends in the systems' behavior and how it can aid domain experts in the systems' maintenance. In the experiments conducted, the proposed approach was able to successfully detect the deviating behaviors known to have previously occurred and was also able to identify some new deviations during the monitored period. Based on the results obtained from the experiments, it can be concluded that the proposed algorithm has the ability to be used for monitoring, analysis, and detecting deviating behaviors of the systems in a smart building domain.
Keywords: evolutionary clustering, multi-view clustering, multi-instance learning, closed patterns, streaming data, formal concept analysis, smart buildings

1 Introduction

The domain of smart buildings is growing rapidly these days. Today's buildings are equipped with various smart and automated systems such as heating, ventilation, and air conditioning; smart lighting; shading; etc. In order to accomplish the desired functionality, these systems work collectively. Buildings in today's urban societies generate up to 40% of the total carbon dioxide emissions [1, 2]. Along with helping us in our everyday activities, smart systems also play an essential role in energy-saving [1]. The majority of the energy used in these systems is wasted due to operational problems [3]. It therefore becomes appealing to be able to understand, analyze, and monitor the behavior of such systems. Smart buildings are equipped with multiple sensors used to facilitate operations and monitor the systems continuously. These sensors collect a large number of heterogeneous data, which becomes difficult to interpret and analyze. The heterogeneous nature of the data can be linked to a wide range of sources or sensors from which they are collected.

There have been many studies, for example, as shown by the reviews of Farzaneh et al. [1] for a wider area of smart buildings and by Mbiydzenyuy et al. [4] in district heating (DH), that utilize data mining and Machine Learning (ML) to analyze and interpret this large amount of data. Data generated from the sensors have specific characteristics such as heterogeneity, streaming nature, concept drift, etc. that need to be considered during analysis. In addition to the data characteristics stated above, another common hindrance for interpreting real-world data is that they are often not labeled. In order to cope with unlabeled data, unsupervised ML techniques such as clustering analysis can be used. Clustering analysis is the task of grouping data instances based on their similarities into few clusters. Traditional clustering algorithms do not address all the challenges stated, and this calls for a need to develop novel hybrid approaches.

This study demonstrates how the MV Multi-Instance Clustering (MV-MIC) algorithm [5] can be successfully used to analyze and interpret the data in the smart building domain. As the name suggests, the proposed algorithm can address challenges that both multi-view (MV) and stream clustering algorithms can. An MV clustering analysis approach is used to find a consensus clustering solution to group data from across different sources, where each source represents a different perspective or view of the monitored phenomenon. The MV essence of the used stream clustering algorithm helps in addressing challenges such as heterogeneity, streaming nature of sensor data, concept drift phenomenon, and non-labeled data. As stated before, many smart building systems work in coordination, and it is essential to analyze them together. The proposed algorithm can analyze a single system and preform an integrated analysis of two or more systems. Therefore, the flexibility of the algorithm makes it suitable for monitoring different contexts.

The main contributions of this study can be summarized as follows. This study proposes and analyzes different approaches for successful monitoring and analysis

of smart building systems. The proposed context-aware approach analyzes and monitors the systems by taking the contextual circumstances into account. In comparison, the other proposed technique, namely integrated system analysis, allows monitoring and analyzing a system in integration with other systems, which is very useful in a domain such as smart buildings where different systems work in coordination. The benefits of using visual data mining are also highlighted. Different visualization techniques for easy interpretation of results generated at each stage of the algorithm are proposed. These visualizations help domain experts to understand the results, monitor any changes in the system, identify the correlations between different views, and detect system failures. Based on the results obtained from the experiments, it can be stated that the proposed approach can be used for continuous monitoring and analysis of the systems in the smart building domain.

The rest of the paper is organized as follows. Section 2 gives an overview of the concepts and methods used in the study. It is followed by Section 3, which presents a brief overview of the related research. Section 4 describes the data used and an MV data analysis approach and introduces visualization techniques proposed in the article. This is followed by Section 5, which provides information about data preparation, experiments, and results. Finally, Section 6 presents the applicability and limitations of the study, and Section 7 is devoted to the conclusion and future work.

2 Background

This section introduces and presents a brief overview of various concepts used in the study and the areas related to them. Sections 2.1 and 2.2 describe clustering algorithms under which the proposed algorithm can be classified. In Section 2.3, Multi-Instance learning, a type of learning technique that is used in the study, is introduced. This is followed by providing an introduction to distance measures in Section 2.4, used to measure how close two clustering solutions are. Finally, formal concept analysis and closed patterns, which are used in the final phase of the algorithm, are described in Sections 2.5 and 2.6 respectively.

2.1 Multi-View Clustering

In MV data, a single data point can be represented in different perspectives or views [6]. Such data are generally acquired from a wide range of sources, are heterogeneous, and usually complement each other. For example, an image and a text describing the same situation can be considered as its two views. Alternatively, a single operation of a system in a smart building domain can be analyzed using different perspectives such as contextual conditions, performance indicators, and operational characteristics. MV clustering is a technique in which the complementary knowledge from different views is extracted, and a model representing all the views is obtained [7].

2.2 Stream Clustering Algorithms

Clustering techniques have been traditionally used to categorize data with similar characteristics into a group. Data points belonging to the same cluster are identical to each other and different from those grouped into other clusters.

Stream clustering algorithms belong to a sub-branch of clustering algorithms that deal with streaming data. Stream refers to infinite, non-stationary data that are continuously generated. As the data generation occurs at a high pace, it is impossible to do random access or store all the incoming data [8]. Streaming data are generally not labeled, and hence clustering is one of the most suitable learning techniques for it [9]. Due to the nature and volume of the data generated, stream clustering algorithms should be capable of performing the task considering the memory and time constrains. Apart from these, concept drift is one of the main challenges to be addressed in data stream clustering [9], a phenomenon where the data characteristics tend to change over time. A stream clustering algorithm should be able to adapt to these changes for better results. According to Wadewale et al. [10], drifts can be categorized into six different types, namely sudden, incremental, gradual, recurring, blip, or noise.

2.3 Multi-Instance Learning

Multi-Instance (MI) learning is a learning technique in which each data object is a bag consisting of a set of data instances, unlike the traditional learning techniques, where a data object is one data instance [11]. In supervised MI learning, the whole bag is labeled. For example, a picture consisting of both water and sand is labeled as a beach. MI learning has various applications ranging from image classification based on the content [12] to the diagnosis of a disease based on images [13].

Multi-Instance Clustering

MI clustering is considered as an unsupervised MI learning, where the data objects are unlabelled bags of instances. Two main advantages of MI clustering over supervised MI learning are that (1) data obtained in many real-world scenarios are not labeled, and it is generally costly to obtain the labels for bags; (2) just like traditional clustering, it is capable of detecting the inherent structure of the data [14]. Even though MI clustering has some similarities with the general clustering algorithms, it cannot be viewed entirely like them. Unlike traditional algorithms, where a single instance is considered as a data object, MI clustering considers a bag of instances. As different instances might show distinct functionalities, it is essential to take into consideration the behavior and relationships of the instances in the bags while grouping the bags into clusters [14].

2.4 Distance Measures

In clustering analysis, different types of distance measures are applied during the modeling phase to find the similarity between the instances. Some of these, such as Euclidean and Manhattan distances, are commonly used for traditional single instance clustering methods. These metrics are not suitable for MI learning as the instances are handled as a set of bags. A new distance metric, Hausdorff distance, is proven to be more efficient in these scenarios [14–16].

Hausdorff Distance

Maximal Hausdorff distance [15], minimal Hausdorff distance [16], and average Hausdorff distance [14] are different kinds of Hausdorff distances presented in the literature. Maximal Hausdorff distance is initially used to measure the distance between two bags in [15] and later on applied to MI learning. Zhang et al. [14] propose average Hausdorff distance, as the other two metrics have not proved to work well in most MI learning problems. Outliers can affect the maximum Hausdorff distance, while minimal Hausdorff distance may be sensitive to the distance between the nearest pair of instances in two bags [14]. Therefore, average Hausdorff distance is considered in the study. Average Hausdorff distance can be calculated using the following formula:

$$H(I,J) = \frac{\sum_{i \in I} \min_{j \in J} dist(i,j) + \sum_{j \in J} \min_{i \in I} dist(i,j)}{\mid I \mid + \mid J \mid}. \qquad \text{(IV.1)}$$

In the above equation, I and J represent bags of instances, and $dist(i,j)$ is the Euclidean distance between instance i from bag I and instance j from bag J.

2.5 Formal Concept Analysis

Formal Concept Analysis (FCA) [17] is a process for extracting concept hierarchy from a set of objects described by their properties. FCA supplies the user with means for building and visualizing the concept hierarchies, which groups objects and properties concurrently. Hence, it can be considered a conceptual clustering method and is used in various fields of data mining, information retrieval, and knowledge discovery.

FCA builds a *formal context* and derives a *concept lattice* from it. The formal context is a table where the rows correspond to the set of objects O and the columns correspond to the set properties or attributes A that the objects can possess. If an object possesses a property, then it is represented by a cross in the table. It is a binary relation defined on the Cartesian product $O \times A$.

The *concept lattice* is a hierarchical structure composed of formal concepts. Each concept is a pair of objects and the properties shared by them. A concept can be

represented using a pair (X, Y), where X is a subset of objects and Y is a subset of attributes (properties); the objects in the concept share these properties and vice versa. In the concept lattice hierarchy, for each concept, there exists a super-concept and sub-concept. Concepts[1], a python module containing the implementation of FCA, is used.

2.6 Closed Patterns

Given a sequence database, sequential pattern mining is defined as the problem of finding regularly reoccurring ordered patterns [18]. A pattern is frequent if it has a support greater than or equal to the chosen support threshold. For a sequential database $\mathcal{T}$ and pattern P, support of P is the number of times P occurs in $\mathcal{T}$. In larger databases, there might be many frequent patterns that can be difficult to analyze. In such cases, it is advantageous to use closed patterns. A pattern P is said to be a closed pattern if it satisfies the following two criteria:

1. It is a frequent pattern.
2. There is no super pattern with the same support as P.

BIDE [19], a famous frequent closed sequential pattern mining algorithm, is used to extract patterns. The python module prefixspan[2], which has the BIDE implemented, is used in the study.

3 Related Work

There are many recent studies in the field of MV clustering. These include surveys [6, 20, 21] that summarize the work done or articles proposing novel algorithms [7, 22–25] to address the challenges in the field. While Fu et al. [6] have compared the performance of the selected MV clustering algorithms on real-world data sets, Yang et al. [21] and Chao et al. [20] have categorized the MV clustering algorithms into different categories. In [21], categorization is based on principles and mechanisms used in the algorithms, whereas in [20], clustering algorithms are grouped into either generative or discriminative clustering. In addition, in [20], the authors have also connected MV clustering to other related areas and listed out potential open problems in the area. Huang et al. [7] in their recent study propose a novel MV clustering algorithm based on co-clustering and bipartite graphs. The authors of [22, 23] have proposed algorithms that are capable of handling incomplete or missing data in the views. Jiang et al. [24] in their work have considered MV clustering as a multi-objective optimization problem and compared how five multi-objective evolutionary algorithms work

[1] https://pypi.org/project/concepts/0.9.1/, accessed on 30 July 2021
[2] https://pypi.org/project/prefixspan/, accessed on 30 July 2021

for the considered problem. In cite [25], non-negative matrix factorization is used to cluster data across the views.

Compared to the field of MV clustering, MV stream clustering is still in its early stages [26, 27]. However, it is gaining more prominence due to large amounts of streaming data being generated across diverse fields. MV stream clustering algorithms are designed to address streamed and multi-viewed data challenges, such as data generation at a high pace and volume, heterogeneity, and concept drift. The authors of [26, 27] propose novel algorithms to address the challenges in the field. In [27], the authors use non-negative matrix factorization for incomplete data sets, whereas [26] propose an algorithm based on support vectors.

Artificial intelligence methods, especially ML techniques, are used in the smart building domains for various monitoring, analysis, prediction, and outlier detection tasks to achieve the desired final outcome in terms of energy efficiency, cost reduction, etc. Jafari-Marandi et al. [2] in their work propose a self-organizing map clustering algorithm, distributed decision model, and a homogeneity index used to evaluate the clusters. The algorithm clusters buildings based on their energy profiles. Such clustering can help reduce primary energy consumption. They use the distributed decision model for operational decisions on the building clusters.

Several studies are focused on the DH network, where the produced heat from the primary side is used for heating and supplying domestic hot water to buildings connected to the network [4, 28, 29]. For example, Mbiydzenyuy et al. [4] review the current state-of-the-art works related to the use of ML in the DH domain. The authors highlight the need for ML to plan, monitor, optimize, and control these systems. They have also listed some goals that needs to be accomplished to increase the impact of ML in the DH domain. Barriers that can hinder the achievement of these goals and ways to overcome them are also stated. Abghari et al. [28] propose an MV clustering approach to identify sub-optimal behaviors of DH substations by considering their geographical locations. The proposed method offers two different analyses, namely step-wise and parallel-wise MV clustering. The step-wise analysis performs a hierarchical clustering based on different feature sets considered in each view. Whereas in the parallel-wise analysis, clustering solutions built upon two views are compared to determine the similarities and variations. Theusch et al. [29] present an ML pipeline using clustering and regression analyses for monitoring and fault detection in DH substations using smart meter data. The authors also identify two key performance indicators, primary return temperature, and the difference of primary supply and return temperatures.

Eghbalian et al. [30] study the heating system, which is part of a Heating, Ventilation, Air Conditioning, and Refrigeration (HVAC&R) system in a smart building domain. They propose an MV data analysis method for monitoring the smart control valve system, which is a part of the heating system. The proposed approach was able to detect deviating behavior successfully. Shchetinin [31] uses clustering to be able to forecast electricity consumption by building consumer profiles. Smart meter data

are used to build the profiles.

There have not been many works using MV stream clustering to analyze and/or monitor system behavior in the domain of smart buildings. This work attempts to bridge this gap.

4 Materials and Methods

This section describes data analysis, visualization, and pattern mining methods proposed in this article. Section 4.1 provides information about the data used for evaluation purposes. The remaining two subsections are devoted to advanced methods for MV integration analysis and continuous pattern mining of smart systems sensor data. Section 4.2 discusses a method of context-aware modeling of system behavior and integration analysis of its performance, while Section 4.3 deals with the visualization and pattern mining.

4.1 Data

In this study, real-world sensor data are used to evaluate the performance of the MV-MIC algorithm, [5] in the field of smart buildings. Data used are obtained from a company based in Stockholm, Sweden. In the smart buildings domain, many systems work together. One of these systems, HVAC&R, is considered in the study. The primary focus is on the heating and tap-water sub-systems. The data used in this study belong to a health-care building located in Stockholm, Sweden. It is a three-storey building which has approximately $2100\ \mathrm{m}^2$ floor area. More details about the sensors from which the data are collected and the features included in the study can be found in Section 5.2, where the experimental scenarios are discussed.

4.2 Multi-View Data Analysis Approach

The work of smart systems is often monitored by multiple sensors, each capturing a different factor of operational or contextual circumstances, due to which the data have a heterogeneous nature and provide different perspectives (views) about the studied system. Modeling the behavior and analyzing the performance of such smart systems are often very complex and can be computationally demanding. All these make the monitoring task very challenging, requiring ML and data mining models that are not only able to continuously integrate and analyze MV streaming data, but also are capable of adapting to concept drift scenarios of newly arriving data.

4.2.1 Context-Aware Modeling of System Behavior

MV Multi-Instance Clustering is an MV stream clustering algorithm that is proposed in [5]. The MV-MIC algorithm can simultaneously monitor the local clustering mod-

els in each view and build an integrated global model, which can be used to find the correlations between the views. The proposed algorithm can continuously monitor and analyze the streaming data. When a new data chunk arrives, the clustering models in each view are updated using Bi-correlation MI clustering. This is followed by building a global model that consists of formal context and a formal concept lattice. MV-MIC takes advantage of extracting closed patterns to obtain the most frequent correlations between the views. The global model built by using FCA helps in analyzing and comparing the correlations between different views of consecutive data chunks.

In this section, we consider and demonstrate how the MV-MIC can be used for context-aware modeling and analysis of smart building system behavior and performance. Each system of the smart building can be monitored from different perspectives or views. Sensors are used to collect a large volume of valuable data about the system operation, context, and performance. For example, different contextual factors such as outdoor temperature, the social behavior of people, etc., can influence the system's operating modes and performance. Evidently, to get a realistic evaluation of the system's behavior, its performance should be assessed by analyzing its operation under different contextual factors, i.e., different perspectives should be studied and linked. Such a division of the system's characteristics will facilitate the domain experts in better understanding how the system's performance correlates with its operating modes and is affected by different contextual circumstances.

Let us consider a streaming scenario where data are analyzed in chunks. That is, each data chunk t contains N most relevant data characteristics of the monitored system. These characteristics (attributes) are selected via a preliminary discussion with domain experts. Hence, the data chunk corpus consists of N different data sets, one per monitored characteristic, and each data set contains n_t daily time-series profiles (measurements). Note that the data chunks can have different sizes; i.e., they can contain different numbers of daily profiles. The available data attributes are further analyzed together with the domain experts and are separated into different views with respect to the information they provide about the monitored system. For example, some parameters may be related to the system operating behavior (operational parameters); others can present different kinds of contextual factors or define system performance indicators. Note that in our description hereafter, system's operational, contextual, and performance characteristics are denoted by views 1, 2, and 3, respectively.

At each newly arrived data chunk t, the approach performs three distinctive steps as illustrated in Figure 1:

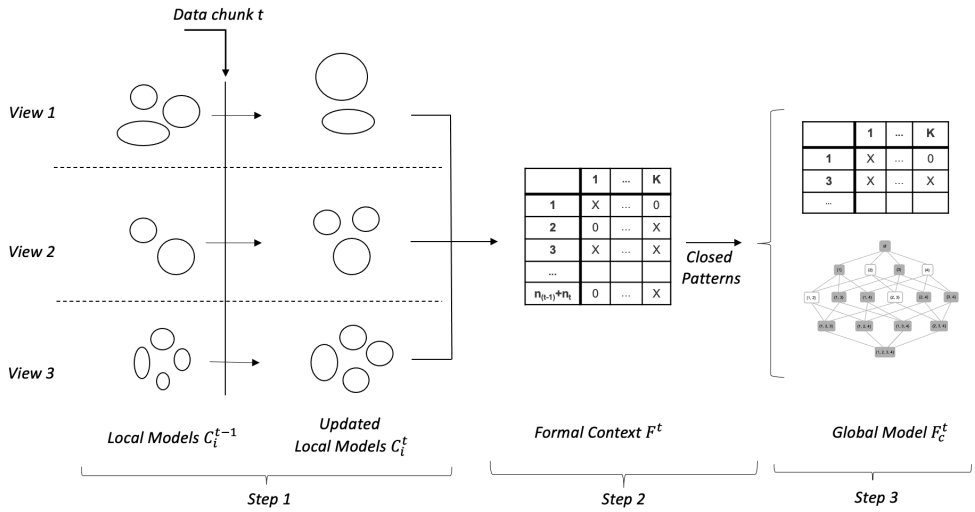


Figure 1: A schematic illustration of different steps of the MV-MIC algorithm.

Step 1: Update local clustering models

a) Initially, for each view i ($i = 1, 2, 3$), integrated daily profiles are created by using the view attributes (N_i in total); i.e., each integrated daily profile is a N_i-dimensional vector that consists of the aggregated values of the features (attributes) of view i ($i = 1, 2, 3$).

b) A local clustering model C_i^t ($i = 1, 2, 3$) is produced at each data view.

c) A clustering model C_i^{t-1} ($i = 1, 2, 3$) built at previous data chunk $t - 1$ is updated by the corresponding model (C_i^t) produced on the new data chunk by applying the Bi-correlation MI clustering algorithm [5].

Step 2: Build a formal context

A formal context matrix F^t, which consists of all MV patterns supported by the updated clustering solutions, is built. Each row j ($j = 1, 2, \ldots, (n_{t-1}+ n_t)$) of F^t is a K-length binary vector, where $K = k_1 + k_2 + k_3$, and k_i ($i = 1, 2, 3$) is the number of cluster labels in the updated clustering solution of view i.

Step 3: Generate a global model

a) Closed (most frequent) patterns, denoted by F_c^t, are extracted from F^t. These patterns present most typical current correlations among the three views, i.e., those that are supported by the chunks $t - 1$ and t.

b) The set of the closed patterns F_t^c is used to generate a concept lattice that describes the hierarchical organization of the identified concepts.

The fourth step of the proposed MV analysis approach can be considered as the post-analysis that can be conducted on the results produced by its application.

Step 4: Analysis

The produced global models can be used to study and analyze the system behavior and performance, e.g., by conducting some of the following:

a) Analysis of system behavior:
The extracted closed patterns (F_c^t) can be studied to understand the current behavior and performance of the system.

b) Identifying deviating behavior:

i. The patterns belonging to F_c^t can be benchmarked to those available in F_c^{t-1} (i.e., the most typical patterns identified at the previous data chunk $t-1$) to discover deviating or unseen behavioral modes.

ii. The hierarchical relationships revealed by the concept lattices built on F_c^t and F_c^{t-1} can also be studied for gaining additional insight into the temporal behavior of the system.

c) Tracking back system behavior:
The system behavior can be studied and tracked back for a longer period than two consecutive chunks by analyzing, e.g., the sets of extracted closed patterns $\{F_c^p \mid p = t, t-1, \ldots, t-q\}$ ($q \geq 2$) covering the studied period.

4.2.2 Integration Analysis of System Performance

A smart building uses various technologies to optimize the building's performance and energy efficiency by sharing information about what goes on in the building between systems. This information is used to automate various processes, from HVAC&R to lighting and security. The most fundamental feature of a smart building is the interconnections of its core systems.

In this section, we propose and discuss the extension of the MV-MIC approach to conduct integration analysis of few different systems that together realize the functionalities of a smart building system. The ability to conduct integrated analysis benefits the understanding of the correlations between different views of the systems involved and how these affect the performance or behavior of a larger system (for ex: HVAC&R) in consideration.

As stated, a smart building has various systems that work both individually and in integration with other systems. Each system can further have multiple sub-systems performing their own designated task or collaborate with other systems. For example, in the HVAC&R, which is responsible for heating, including hot water system, ventilation, air conditioning, and refrigeration, a designated sub-system is responsible for each of these. The contextual conditions such as outdoor temperature and

inhabitant behavior are the same for all these systems, influencing their performance. Note that more number of inhabitants yields increased hot water consumption and demands for higher ventilation. As an illustration of how the systems work in coordination with the selected HVAC&R system, let us consider the operation of the hot tap-water system (hear after referred to as tap-water system), which is linked to the heating system. Hot water returned from the heating system is transferred to the tap-water system through a heat exchanger and used to heat the tap water. Suppose this temperature received from the heat exchanger is insufficient. In that case, the tap-water system opens one of its valves to release some more hot water to obtain the desired temperature. Such interlinks can be seen among other systems as well. As the systems work in coordination with each other, a better understanding of the systems' operation can be obtained by applying flexible data analytic algorithms capable of conducting integrated MV analysis of multiple systems. This integrated analysis can help to identify behaviors that are challenging to detect when the systems are analyzed individually.

The proposed MV-MIC algorithm perfectly fits into this role of an integrated system analysis tool. It can be used to analyze each sub-system individually or in combination with sub-systems with respect to different perspectives or views such as operating modes, context, and performance. Figure 2 shows a high-level overview of how the systems work in integration in a smart building. It can be seen that the systems of a smart building can be represented using a hierarchical structure. After the local models in each view are updated, one can dynamically decide and select views to be analyzed further and find correlations between them. Then, the global model is built using these chosen views. Note that Figure 2 is included only to illustrate how the systems of a smart building can be grouped together in a hierarchical structure. Experiments in the current study are only conducted on the heating and tap-water systems shown in the figure.

Assume that at data chunk t, the operation and performance of two linked systems, e.g., heating and tap-water (or heating and ventilation), are studied. These systems work under the same exogenous circumstances. In this context, a few different scenarios can be analyzed to understand and gain better insight into the systems' integrated behavior:

A. Meta-integration analysis of two systems:
   The MV-MIC algorithm, described in Section 4.2.1, can be applied individually to each system's data. As a result, two sets of closed patterns linking the three views' of each system are extracted. These can be separately analyzed for each system to understand the system behavior. In addition, the two sets of extracted closed patterns can be aligned to each other with respect to the features in the contextual view in order to identify some interconnections between the systems affecting and explaining their performance.

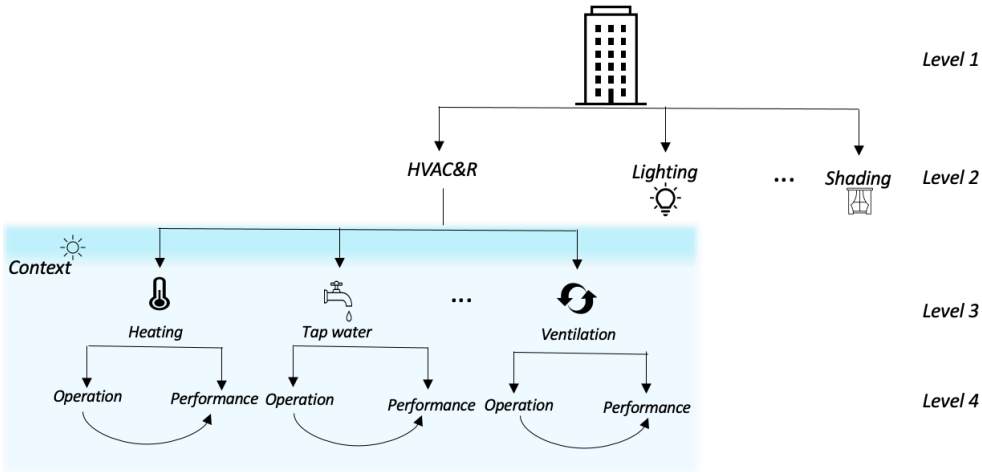


Figure 2: Example of integrated systems in a smart building.

B. Inter-study integration analysis of two systems:
The two systems' data views can be considered together, and the MV-MIC algorithm can be used to analyze the systems. In this scenario, the extracted closed patterns will represent integrated systems' profiles linking operating modes and performance of the two systems under the monitored contextual conditions.

C. Integration analysis of selected views:
Selected views from the two systems can be analyzed together, e.g., only performance indicators of heating and tap-water may be considered. Such analysis may reveal, e.g., how the performance of the tap-water system is correlated to the heating system.

There are various advantages of using integration analysis. One can initially analyze the sub-systems and then integrate them to obtain a full overview of the whole system. It will enable us to utilize the benefits provided by the MV clustering completely. Considering sub-systems helps in identifying hidden cluster groups. Note that there might be difficulty identifying proper clusters when the complete system is studied due to the increased complexity of having many different types of heterogeneous features.

The proposed approach is capable of performing both horizontal and vertical integration of the knowledge obtained through the analysis of the systems. Vertical integration of the knowledge can be done when a new data chunk arrives; Bi-correlation MI clustering is used to update the existing local clustering solutions based on the newly arriving data. Horizontal integration can be done between different views and sub-systems as per the requirement, e.g., see integration scenarios B and C above.

4.3 Data Visualization and Analysis

This section presents how the MV-MIC can be combined with different data visualization techniques for facilitating continuous monitoring, analysis, and pattern mining of smart building system behavior and performance. We demonstrate how different visualization techniques could be used to better understand and monitor the system performance and operation. The proposed new approach of continuous visual tracking for each data chunk helps to quickly identify and track the changes over time.

The amount of data generated by the smart building systems is enormous and difficult to interpret; this can be addressed using visual data mining [32]. Visualization can be considered as a part of data mining and knowledge discovery [33]. It helps in representation and eases the task of understanding a huge volume of data that are difficult to present and analyze in a textual form [32, 34]. One of the significant advantages of visual exploration, as stated by Keim [34] is that it is capable of handling heterogeneous and noisy data better than what statistical or ML techniques have to offer. The paper proposes a visual data mining based approach to detect abnormal behavior based on historical operational data.

In the current study, visualization is used to represent the outcomes of different stages of the proposed algorithm. These visualizations can be used by domain experts to better understand the operation and performance of the system.

4.3.1 Visualization of Results

As stated before, visualization increases the understandability of complex results. In [32], the authors highlight the need for continuous visualization in a data mining process. Inspired by this, we propose suitable visualizations of the results obtained in each phase of the MV-MIC algorithm. This visualization facilitates the domain expert in analyzing and better understanding the system behavior and performance by presenting details that are not visible in the end results.

1. Visualization of local models:
   Each local clustering model produced at Step 1 of the algorithm captures information about the most typical working/performing modes or conditions of the studied system in the respective view. In order to facilitate the perception of the information summarized in each local clustering model C_i^t ($i = 1, 2, 3$), it can be visualized by a $k_i \times N_i$ matrix that contains average values of the corresponding view's attributes w.r.t. the identified typical working scenarios (clusters). The matrix can be colored by using different color intensities for each row by taking into account the size of the cluster it presents; e.g., see Table 3. Such colored matrices can be used to visually inspect the system's working scenarios in the different perspectives considered. This can further facilitate the detection of deviating/underperforming scenarios by comparing the local models built on two consecutive data chunks of the system. The data presented in each matrix can also be sorted by a selected attribute, which can

further facilitate the analysis and understanding of the system behavior. For example, Tables 3 and 8 can be ordered w.r.t. PHL and then aligned to each other.

2. Visualization of formal context:
   A formal context F^t is built at Step 2 of the algorithm using the data points of the current and previous chunks. Each data point of F^t can be represented (labeled) by a three-length vector (string) that contains the respective labels from the three views' local models. This can be used to visualize an extract of m data points using an $m \times 3$ matrix (3 represents the number of views). Each column of this matrix can be colored by the respective view color used in the visualization of the view's local model. In addition, each cell may have different color intensity, similar to the local model matrices. Such visualization can be used to study a specific period of the system work by allowing them to conduct higher-order comparison and analysis of the system's daily behavior and performance under the different contextual conditions in the studied period. For example, two tables presenting, respectively, two consecutive weeks of the system work can be visualized and studied by request, e.g., see the two tables in Table 6. These can reveal, e.g., that the system has moved in a different operating mode in the second studied period (week), although the contextual conditions and performance measures have not changed from the first week. This may be an indication of a problem and can be further studied by the domain experts. Note that the above tables can also be sorted w.r.t. the labels in a column by choice, which can further facilitate the comparison.

3. Visualization of concepts linking three views:
   A set of closed patterns F_c^t is extracted from F^t at Step 3 of the proposed approach. This set of closed patterns is used to build a concept lattice (global model) at chunk t. The concepts linking three views can be presented in a table similar to the one used for local clustering models, but containing average values of the attributes from all three views; e.g., see Table 7. In addition to this table, a tripartite graph can be created to visualize the correlations between the three views' clusters revealed by those concepts. A tripartite graph $G_t = (V_1^t, V_2^t, V_3^t, E_{12}^t, E_{23}^t)$ can be created for each data chunk t, where the first three components (V_1^t, V_2^t and V_3^t) are, respectively, the three vertex sets of the graph and the remaining two (E_{12}^t and E_{23}^t) are the edge sets. Note that V_i^t ($i = 1, 2, 3$) is the set of cluster labels of clustering solution C_i^t ($i = 1, 2, 3$). Furthermore, $E_{12}^t \subset V_1^t \times V_2^t$ and $E_{23}^t \subset V_2^t \times V_3^t$ present, respectively, the links between local models C_1^t and C_2^t and between C_2^t and C_3^t, which are revealed by the formal concepts of the global model. These edges, when considered together, also present the links between the local models of all three views. For example, see Figure 4: the edge connecting nodes A_{O0}, A_{C0}, and A_{P0} in

all three views. i.e., V_1^t, V_2^t, and V_3^t represent the correlation between these three clusters from these views. The graph edges can have a different thickness that reflects the size of the concept they present. Furthermore, the nodes in the vertex sets can be colored by using the same color visualization idea as the one applied for the local models. Such a graph visualization can facilitate the domain experts in getting an overall (at a higher level) understanding of the system behavior and performance w.r.t. different contextual scenarios. The comparison of two graphs produced on the data of two consecutive chunks (e.g., G_{t-1} and G_t) can provide information about newly appeared correlations among the three main characteristics of the system.

In addition to the table and tripartite graph, each concept can be visualized by plotting its performance mode feature values in a spider chart and further labeling the chart by selected parameters from the other two views (context and/or operation). Such plots will facilitate the visual comparison of the different concepts, e.g., on domain expert request, all heating season concepts can be plotted and inspected.

4. Visualization of concept lattice:
   The set of closed patterns F_c^t can be used at Step 3 of the proposed approach to generate a concept lattice. The latter can present a complex hierarchical structure. Therefore, it is not considered very useful to visualize the whole lattice. If needed, a sub-lattice can be visualized to illustrate further, e.g., the links of specific three-view concepts with two-view concepts.

5 Experimentation and Analysis

This section presents the experimental scenarios investigated in this study along with the results obtained, followed by a discussion. Section 5.1 presents the data preprocessing steps. Section 5.2 describes the used experimental setup and discusses the obtained results. Experimentation is done by considering the heating and tapwater systems independently and together in an integrated scenarios.

5.1 Data Preparation

This section describes the data pre-processing steps used before the proposed MVMIC algorithm is applied. Each of these steps is applied to the newly arriving data chunks.

5.1.1 Outlier Removal

Real-world data often contain data points with a deviating behavior known as outliers or noise. Building a model with such data might negatively impact the performance

of the model. Sudden spikes or drops in the measurements can be smoothened using data smoothing techniques belonging to the Median Absolute Deviation (MAD) family. In this study, a Hampel filter [35] is used to replace such outliers with a local median of a sliding window of size k (7 in this case). Python module for Hampel filter[3] is used.

5.1.2 Data Cleaning

Some features in the data have missing values. In the heating or tap-water systems, the values of each feature depends on various factors such as the current outdoor temperature, building occupancy, ventilation, and tap-water usage, to name a few, making it difficult to estimate the missing values correctly. Hence, the rows with missing values are removed.

5.1.3 Data Division

Data over a period of two years (1 January 2019 to 31 December 2020) are used to conduct the experiments. The streaming data set is divided into three chunks. One year of data, that is, from 1 January 2019 until 31 December 2019 are considered as chunk 1. The second-year data are divided into two chunks, i.e., 1 January 2020 until 30 of June 2020 is considered as chunk 2 and 1 July 2020 until 31 of December 2020 is considered as chunk 3. Daily profiles are used in the study as the hourly data are sparse for some features.

5.1.4 Standardization

Each data chunk is standardized using z-score. Standardization of the features is done by subtracting the mean value of the feature from each sample and dividing it by the standard deviation of the feature. Equation IV.2 is used to calculate the z-score, where x is the sample, u is the mean, and s is the standard deviation. StandardScaler from the preprocessing module of python Scikit-learn [36] library is used to perform the standardization.

$$z = \frac{x - u}{s}. \tag{IV.2}$$

5.1.5 Estimation of the Number of Clusters

In this study, k-means is used to initially cluster the data points in different views. This partitioning algorithm requires k, the number of clusters to be known in advance. Therefore, we take advantage of the Silhouette Index cluster validation method for identifying the optimal number of clusters k.

[3] https://github.com/MichaelisTrofficus/hampel_filter, accessed on 5 July 2021

The Silhouette Index (SI) for clustering solution C of n objects is defined as:

$$s(C) = \frac{1}{n} \sum_{i=1}^{n} \frac{(b_i - a_i)}{\max\{a_i, b_i\}}, \tag{IV.3}$$

where a_i represents the average distance of item i from all the other items in the cluster to which the item i is assigned, and b_i represents the minimum of the average distances of item i from items of the other clusters.

5.2 Experimental Setup and Results

The proposed algorithm is suitable for various continuous data mining tasks such as monitoring and pattern extraction at multiple levels (see Figure 2) of the smart building system. In this section, we demonstrate how the proposed algorithm could be used for an MV data analysis of independent systems, scenario A, (Section 5.2.1), as well as for an integrated system, scenario B, (Section 5.2.2) using different experimental settings. More specifically, data from heating and tap-water sub-systems that are part of the HVAC&R system are used for different experiments conducted in the study. Figure 3 illustrates the HVAC&R system schematics, including the tap-water (dashed purple rectangle) and heating (dashed blue rectangle) sub-systems.

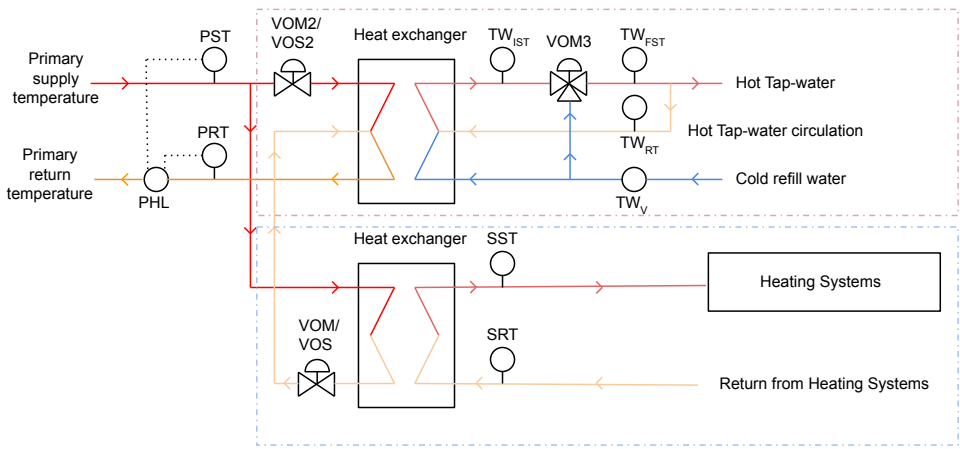


Figure 3: HVAC&R system schematics: dashed purple rectangle represents the tap-water sub-system, and dashed blue rectangle represents heating systems.

5.2.1 Individual Analysis of Heating and Tap-Water Systems

In this set of experiments, the heating and the tap-water systems are individually analyzed to observe their performance independently. These are used to find correlations within the sub-system that are difficult to identify when analyzing an integrated system.

Experiments on the Heating System This experimental scenario monitors the heating system, which is part of an HVAC&R system. The experiment is designed to help domain experts to better understand how the operational, performance, and contextual characteristics affect each other. The system contains various sensors, which are continuously collecting information. These metrics are classified into three views, representing the operation, performance, and context of the system. Details about the features included in each of these views are presented in Table 1.

Table 1: Features used for monitoring the heating system.

View	Id	Feature Name	Acronyms	Units
Operation	1	Secondary Supply Temperature	SST	°C
		Secondary Return Temperature	SRT	°C
		Primary Heat Load	PHL	kW
Performance	2	Valve Openness Mean	VOM	%
		Valve Openness Standard Deviation	VOS	%
		Sub-station Efficiency	SE	%
Context	3	Outdoor Temperature Mean	OTM	°C
		Outdoor Temperature Standard Deviation	OTS	°C

Note. Sub-station Efficiency (SE) is computed as the difference between primary supply temperature (PST) and primary return temperature (PRT) divided by the difference between PST and secondary return temperature (SRT).

The system's operational parameters include the secondary supply and return temperatures, and primary heat load. The average valve openness and its standard deviation together with sub-station efficiency are considered for measuring the performance. For the contextual parameters, average outdoor temperature, along with its standard deviation, are considered.

The proposed algorithm requires that the data in each chunk are initially clustered. Initial clustering in the operating and performance views, that is, views 1 and 2, is done using k-means. The optimal value for k is identified using SI. For the contextual view (view 3), initial clustering is performed based on the seasons in a year as proposed by [37]. The data are divided into four clusters, namely winter (December to February), early spring and late summer (March, April, October, and November), late spring and early autumn (May and September), and summer (June to August). Details about the number of initial clusters in each chunk for different views can be seen in Table 2.

Table 2: Initial number of clusters in three chunks for each view---Heating System.

Chunk	View 1	View 2	View 3
1	5	3	4
2	7	3	4
3	4	3	4

Considering the chronological order in which the data arrive, MV-MIC is initially applied on chunks 1 and 2. First, the local models in each view are updated using

the Bi-Correlation MI-Clustering, which produced $9, 4$, and 5 clusters for views 1, 2, and 3, respectively. Overview of each of these clusters from different views can be seen from Tables 3, 4, and 5. As stated in Section 4.3.1, these tables are colored based on the size of the cluster and can be used for visualizing different working or contextual modes in each of the views.

Table 3: Operating modes identified after receiving chunk 2.

Operating Modes (OM)	PHL (kW)	SST (°C)	SRT (°C)	Size
A_{O0}	24.16	43.88	38.03	118
A_{O1}	12.27	34.48	32.23	80
A_{O2}	4.83	28.41	26.78	160
A_{O3}	44.38	49.52	39.55	49
A_{O4}	3.33	42.51	36.16	34
A_{O5}	19.07	45.28	39.47	12
A_{O6}	18.29	39.68	36.12	18
A_{O7}	29.29	47.46	39.63	16
A_{O8}	10.35	31.11	29.68	26

Table 4: Performance modes identified after receiving chunk 2.

Performance Modes (PM)	SE (%)	VOM (%)	VOS (%)	Size
A_{P0}	96	14.04	± 1.16	338
A_{P1}	70	0.95	± 0.08	45
A_{P2}	75	7.46	± 4.81	98
A_{P3}	76	6.97	± 4.43	32

Table 5: Contextual conditions identified after receiving chunk 2.

Contextual Conditions (CC)	OTM (°C)	OTS (°C)	Size
A_{C0}	2.57	± 1.18	131
A_{C1}	6.53	± 1.97	168
A_{C2}	19.41	± 2.91	122
A_{C3}	13.20	± 2.39	61
A_{C4}	11.73	± 3.27	31

After building a local model for each view, FCA is used to integrate them and build the global model, which contains a formal context and concept lattice. The concept lattice has 69 non-empty concepts. Among these, only 32 concepts connect all three views. As stated in Section 4.3.1, visualization of the formal context can be used to compare the system behavior at higher granularity, e.g., a day. To illustrate this, two weeks of data are considered, where week one represents normal system behavior, while week two contains abnormal and sub-optimal behavior. A gap of one week is given in between to make sure that the normal and abnormal behaviors do not coincide. Table 6 presents two weeks of data, that is, from 1 March 2019 to 7 March 2019 and 15 March 2019 to 21 March 2019, representing the system performance in March-2019. From these tables, it can be observed that during week two, the operating mode is always A_{O4}, representing the deviating behavior, and a sudden drop in PHL down to 3.33 kW (see Table 3). In comparison, the operating modes for normal behavior during this time are either A_{O0} with an average PHL equal to 24.16 kW or A_{O3} with an average PHL equal to 44.38 kW (the operating

modes identified between 1 March 2019 to 7 March 2019). The performance and contextual modes during these weeks are identical. Such tables can help the domain expert identify the issues (here with the features related to the operating mode) and take timely action.

Table 6: Patterns representing the behavior of heating system between 1 to 7 March 2019 and 15 to 21 March 2019.

Date	OM	PM	CC	Date	OM	PM	CC
1 March 2019	A_{O3}	A_{P0}	A_{C1}	15 March 2019	A_{O4}	A_{P0}	A_{C1}
2 March 2019	A_{O0}	A_{P0}	A_{C1}	16 March 2019	A_{O4}	A_{P0}	A_{C1}
3 March 2019	A_{O0}	A_{P0}	A_{C1}	17 March 2019	A_{O4}	A_{P0}	A_{C1}
4 March 2019	A_{O3}	A_{P0}	A_{C1}	18 March 2019	A_{O4}	A_{P0}	A_{C1}
5 March 2019	A_{O3}	A_{P0}	A_{C1}	19 March 2019	A_{O4}	A_{P0}	A_{C1}
6 March 2019	A_{O3}	A_{P0}	A_{C1}	20 March 2019	A_{O4}	A_{P0}	A_{C1}
7 March 2019	A_{O0}	A_{P0}	A_{C1}	21 March 2019	A_{O4}	A_{P0}	A_{C1}

Next, closed patterns are used to extract the most common or frequent behavioral patterns. Support of ≈2.5% is used in this process, i.e., patterns that cover at least 2.5% of the data are considered to be frequent. There are 513 daily profiles in total when both chunks 1 and 2 are considered. That is, concepts with a frequency of at least 13 are considered. This gave us 31 concepts linking any two views and 11 concepts linking all three views. These 11 concepts are represented in Table 7 and Figure 4. In Table 7, the data are sorted based on the OTM.

Table 7: Closed patterns from global model showing correlations between all three views after receiving chunk 2.

S/N	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Size	Months
6	3.37	26.85	26.60	0.00	± 0.00	70	22.12	± 3.01	40	[6--8]
10	4.99	29.14	26.68	6.53	± 4.96	71	18.23	± 2.59	62	[6--8]
1	4.75	26.59	26.25	5.60	± 4.90	78	17.88	± 3.79	19	[6]
2	6.6	29.75	27.02	8.47	± 4.61	78	16.68	± 2.19	21	[5, 9]
3	12.95	33.73	32.09	11.65	± 1.30	88	11.12	± 2.55	27	[5, 9]
5	12.9	35.28	32.87	11.87	± 1.04	92	9.68	± 1.96	34	[3, 4, 10, 11]
4	3.33	42.51	36.16	15.53	± 1.55	96	5.47	± 2.01	34	[3, 4, 11]
9	24.68	44.00	37.95	14.21	± 0.99	99	4.41	± 1.05	60	[1, 2, 12]
8	23.61	43.93	38.26	13.78	± 0.88	97	4.40	± 1.33	54	[3, 4, 10, 11]
0	29.62	47.69	39.75	15.16	± 1.23	104	1.02	± 1.85	14	[1, 2]
7	45.02	49.58	39.54	18.28	± 1.61	100	−0.59	± 1.20	42	[1, 2, 12]

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature, see Table 1.

From Table 7 it can be observed that there is a sudden drop in the PHL for concept 4, implicating a deviating behavior that matched with the prior information we had regarding issues in the system during March and April 2019. This showcases that the proposed algorithm is capable of detecting abnormal or deviating behaviors. Concepts 9 and 8 are very similar but could have been interpreted as different concepts because of the clustering in view 3, since the months in these concepts belong to different initial clusters.

Figure 4 is a tripartite graph representing all 11 concepts linking three views that are obtained after using closed patterns. This figure showcases the links between

all three views and gives the observer an easy understanding of how the views are correlated. The edges of the graph are of varied thickness representing the size of the concept. Concepts with greater size are represented by thicker lines showing a stronger correlation between views, whereas the lighter lines imply that the considered concept is only supported by a few daily profiles, i.e., that the correlation is not strong. For example, the first link in the figure represents a concept linking clusters A_{O0}, A_{C0}, and A_{P0} in views 1, 2, and 3, respectively. This is supported by a size of 60 daily profiles. The link between clusters A_{O7}, A_{C0}, and A_{P0} in views 1, 2, and 3, on the other hand, is supported by a concept of size 14. One can also observe from the tripartite graph that some views' clusters do not take part in any of the three-view concepts, e.g., clusters A_{O5}, A_{O6}, A_{O8}, and A_{C4}. These might be involved in two-view concepts. It is interesting to notice that A_{O5} and A_{C4} are the smallest clusters among the others in their local clustering models. A_{O6} and A_{O8} are also of the smallest clusters in view 1.

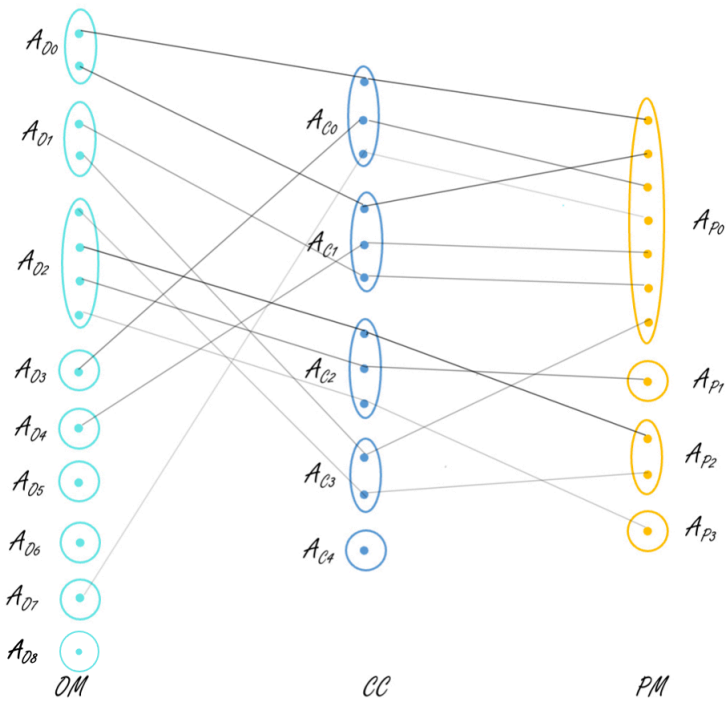


Figure 4: Tripartite graph representing the relationships between different concepts after receiving chunk 2. Color thickness of the edges correlates to the size of the concept.

When chunk 3 arrives, the local clustering models in each view are updated again. The number of clusters in the updated local models are 8, 3, and 6 clusters in views 1, 2, and 3, respectively. These can be viewed in Tables 8, 9, and 10. When comparing the operating modes clusters between Tables 3 and 8, it can be observed that in Table 3, operating mode A_{O4}, which represents deviating behavior, can be con-

sidered as an additional mode. All the other modes except A_{O3} from Table 3 can be compared to one of the modes listed in Table 8. That is, clusters B_{O2}, B_{O3}, and B_{O5} from Table 8 are similar to clusters A_{O5}, A_{O7}, and A_{O8}, respectively, from Table 3. Clusters B_{O0}, B_{O1}, and B_{O4} in Table 8, on the other hand, are close to clusters A_{O0}, A_{O2}, and A_{O1}, respectively, from Table 3. It can be stated that when the local models are updated, some clusters are retained while some are updated. For the performance modes, the number of clusters are not evenly distributed, and a majority of the instances are grouped into cluster B_{P0} (see Table 9). In view 3, as stated before, the clustering is done based on the seasons of the year, there are some new clusters, some of them are retained while others are updated.

Table 8: Operating modes identified after receiving chunk 3.

Operating Modes (OM)	PHL (kW)	SST (°C)	SRT (°C)	Size
B_{O0}	21.17	42.22	36.73	59
B_{O1}	4.03	26.69	25.57	137
B_{O2}	19.07	45.28	39.47	12
B_{O3}	29.29	47.46	39.63	16
B_{O4}	13.45	34.78	32.54	17
B_{O5}	10.35	31.11	29.68	26
B_{O6}	0.70	33.99	30.02	33
B_{O7}	0.00	44.00	35.98	29

Table 9: Performance modes identified after receiving chunk 3.

Performance Modes (PM)	SE (%)	VOM (%)	VOS (%)	Size
B_{P0}	-23	9.12	± 1.33	308
B_{P1}	-2785	13.61	± 1.23	10
B_{P2}	2894	14.14	± 1.32	11

After the global model is built, initially, there are 48 non-empty concepts; among these, 23 concepts connect all the three views. As the number of instances in chunks 2 and 3 combined is 329, support of 8 is used when extracting closed patterns. When closed patterns are used to extract the most frequent patterns, 32 concepts connecting any two views and 14 concepts connecting all the three views are obtained. The latter are represented in Table 11. From this table, deviating behavior is seen in concepts 3, 4, 9, and 10, where the PHL shows a significant difference from its original pattern. Furthermore, it is interesting to note that all these concepts show deviating behavior with respect to SE. As one can observe, SE in concepts 3, 9, and 10 is negative, while in concepts 4 and 11, it has unexpectedly high (2852%) and low (21%) values, respectively. These results were discussed with the domain expert and it was identified that there were in fact some issues in the system from the end of September till mid-December 2020. This once again showcases the potential of the proposed algorithm in identifying new trends in the data, long-term fault within the system in this scenario.

Figure 5 represents the links between different views of the concepts present in Table 11, which are obtained after using closed patterns. For example, the link

Table 10: Contextual conditions identified after receiving chunk 3.

Contextual Conditions (CC)	OTM (°C)	OTS (°C)	Size
B_{C0}	3.99	±1.07	75
B_{C1}	19.61	±3.02	86
B_{C2}	6.93	±2.52	46
B_{C3}	11.73	±3.27	31
B_{C4}	14.65	±1.98	30
B_{C5}	8.58	±1.20	61

Table 11: Closed patterns from global model showing correlations between all three views after receiving chunk 3.

S/N	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Size	Months
13	4.25	26.58	25.67	1.78	±1.27	74	19.76	±3.02	80	[6--8]
1	6.89	28.16	26.73	8.30	±4.32	75	15.47	±3.66	8	[5]
11	5.23	25.99	24.66	9.77	±2.89	21	14.65	±1.98	30	[9]
9	0.00	27.63	26.11	11.16	±0.50	−293	12.38	±0.90	19	[10, 11]
7	10.23	31.46	29.67	10.63	±1.66	81	11.36	±3.26	15	[5]
5	10.52	30.62	29.69	11.05	±1.20	90	11.19	±3.16	11	[4]
0	13.81	35.43	32.29	11.17	±1.18	89	8.69	±2.89	8	[5]
10	0.00	33.82	30.68	12.09	±0.95	−506	8.46	±1.38	22	[10, 11]
2	13.13	34.20	32.77	11.47	±1.16	90	8.0	±3.00	9	[4]
8	21.05	41.07	36.72	13.17	±0.93	100	5.59	±2.2	18	[3, 4]
3	0.00	41.43	34.89	13.45	±1.16	−787	5.27	±1.71	10	[10, 11]
12	21.22	42.73	36.73	13.22	±0.99	99	5.13	±0.96	41	[1, 2, 12]
4	0.0	44.98	36.76	14.14	±1.33	2852	3.75	±0.58	10	[12]
6	29.62	47.69	39.75	15.16	±1.23	104	1.02	±1.85	14	[1, 2]

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 1.

between B_{O1}, B_{C1}, and B_{P0} supported by 80 instances is the strongest correlation and represents concept 13.

In order to get further insight into the differences between the concepts and/or to track any potential drifts, visualization techniques such as the ones shown in Figures 6–8 can be used. Figure 6 presents concepts from both the iterations, i.e., one after receiving chunk 2 and the other after receiving chunk 3 when OTM is in the range, 10 °C < OTM < 15 °C. It can be observed that the graphs highlight two extremes, *Iteration2 - Concept 9* ($SE = -293\%$) and *Iteration 2 - Concept 11* ($SE = 23\%$), where the SE (SE ranges from 0 up to 100%. However, due to the generation of hot tap water, it can rise up to 120%.) shows deviation from the other concepts. Similarly, Figure 7 presents concepts when OTM < 10 °C and highlights three deviations, *Iteration 2 - Concept 4* ($SE = 2852\%$), *Iteration 2 - Concept 3* ($SE = -789\%$), and *Iteration 2 - Concept 10* ($SE = -506\%$). Note that except for the mentioned concepts, all the others in the figure overlap, showing the similarity among them. Figure 8 presents concepts of both the iterations after removing the deviating concepts when OTM < 10 °C. It can be clearly observed that all concepts are close to one another. As demonstrated, these graphs can be used by domain experts to see the similarities or changes between different concepts, which can help them to identify the changes in the behavior of the system. In the long run, they can also have one such graph for each smaller temperature range (say, for example, 1 or 2 °C). It is

expected that concepts in the same temperature range should be similar, so even a small deviation in behavior (gradual concept drift) can also be observed.

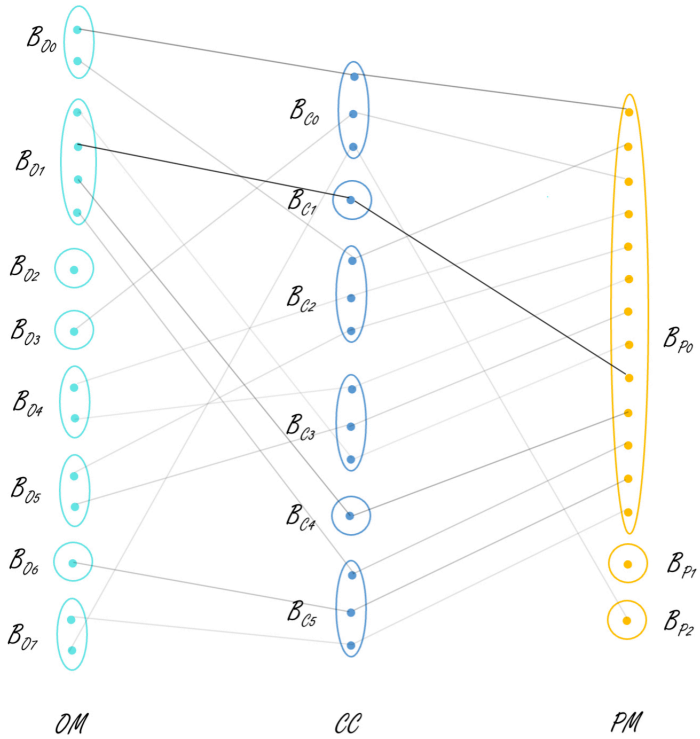


Figure 5: Tripartite graph representing the relationships between different concepts after receiving chunk 3. Color thickness of the edges correlates to the size of the concept.

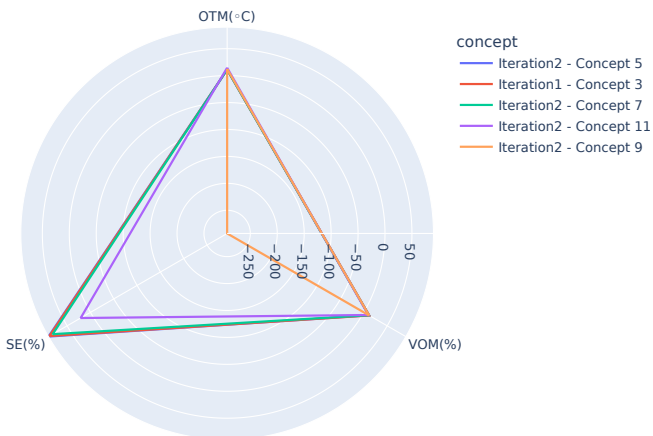


Figure 6: Comparing concepts of both the iterations (after receiving chunk 2 and 3, Iteration 1 and 2, respectively) using the relationship between OTM, VOM, and SE when 10 °C < OTM < 15 °C. Note that some of the concepts in the legend are not visible in the image due to overlap.

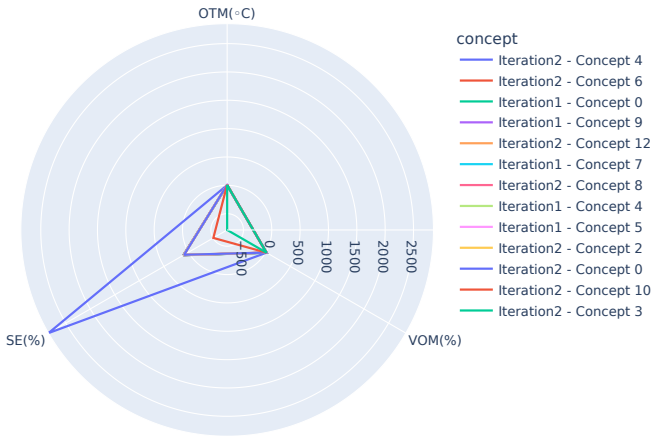


Figure 7: Comparing concepts of both the iterations (after receiving chunk 2 and 3, Iteration 1 and 2, respectively) using the relationship between OTM, VOM, and SE, when OTM < 10 °C. Note that some of the concepts in the legend are not visible in the image due to overlap.

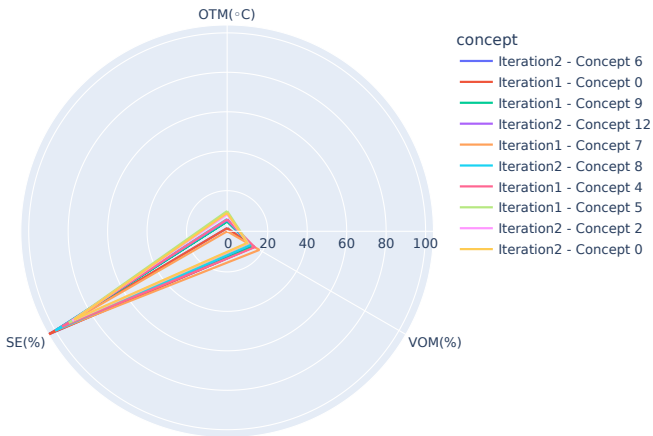


Figure 8: Comparing concepts of both the iterations (after receiving chunk 2 and 3, Iteration 1 and 2, respectively) using the relationship between OTM, VOM, and SE, when OTM < 10 °C and after removing deviating concepts 3, 4, and 10 of iteration 2. Note that some of the concepts in the legend are not visible in the image due to overlap.

Experiments on the Tap-Water System Similar to the experiments conducted for the heating system, individual system analysis is also performed on the tap-water system. Based on the discussion and feedback received from the domain expert, the features characterizing the tap-water behavior are divided into three views, namely operation, performance, and context, as shown in Table 12.

The features measuring the system's operational parameters include the primary heat load, volume of water used during the day, supply, and return temperatures. For measuring the performance, openness of both the valves used by the tap-water system along with the primary delta (difference between the primary supply and primary return temperatures) are considered. Among the two valves, VOM3, a three-way

valve, is responsible for regulating the hot tap-water temperature to be around $60\ ^{\circ}$C. In order to maintain this temperature, the valve might sometimes allow cold water to be mixed with hot water. Hence, its standard deviation is not considered as it is not varied often. The outdoor temperature and the openness of the valve from the heating system are considered as the contextual parameters. The valve openness of the heating system is included as a contextual parameter as it impacts the tap-water system. The hot water obtained from the primary network first goes through the heating system to heat the room, and after that, the water is used by the tap-water system. The valve in the heating system lets out this water and hence can be considered as a context from the tap-water system point of view. It can also be noted that during the non-heating seasons, that is, when the outdoor temperature is above 17°C, the heating system valve is completely closed and the heat obtained from the primary network is only used to heat the tap-water.

Table 12: Features used for monitoring the tap-water system.

View	Id	Feature Name	Acronyms	Units
Operation	4	Tap-water Initial Supply Temperature	TW_{IST}	$^{\circ}$C
		Tap-water Final Supply Temperature	TW_{FST}	$^{\circ}$C
		Tap-water Return Temperature	TW_{RT}	$^{\circ}$C
		Primary Heat Load	PHL	kW
		Tap-water Volume Consumed	TW_V	m^3
Performance	5	Tap-water Valve Openness Mean 2	VOM2	%
		Tap-water Valve Openness Standard Deviation 2	VOS2	%
		Tap-water Valve Openness Mean 3	VOM3	%
		Primary Delta	ΔP	$^{\circ}$C
Context	6	Outdoor Temperature Mean	OTM	$^{\circ}$C
		Valve Openness Mean (Heating system)	VOM	%

Initial clustering in all the chunks and for all the three views is done using k-means clustering, for which SI is used to determine the optimal number of clusters. Table 13 presents details about the number of initial clusters considered for each of the data chunks.

Table 13: Initial number of clusters in three chunks for each view---Tap-water System

Chunk	View 4	View 5	View 6
1	6	5	5
2	7	5	4
3	3	6	7

When the global model is updated after the arrival of chunk 2, the concept lattice generated contained 132 non-empty concepts, of which 59 concepts connected all three views. After using the closed patterns, the model has 48 concepts connecting any two views of the local models and 16 connecting all three views. Table 14 represents all these 16 concepts.

As explained earlier, VOM3 is the valve openness mean of a three-way valve used in the tap-water system. It has an opening for letting in the cold water when the

temperature of the water is above 60 °C. Opening this valve for letting in cold water is not the desired function, as it leads to energy waste, i.e., the water is initially heated and then cooled down. So, in the desired functionality, the valve of VOM3 should be close to 100, representing that the valve only allows hot water to go through. It can be observed from Table 14 that the model was able to categorize the concepts (10, 13, 14, and 15), where the average values for VOM3 are a lot less than 100, implicating that the valve was opened to let in cold water to maintain the water temperature, which is not desired. This can help the domain experts to analyze the identified situation and detect what went wrong. It is interesting to note that two out of these four (concepts 15 and 14) occurred when hot water consumption was high. All the four concepts occurred during the heating season, that is, when the outdoor temperature is below 15 °C. This reflects the heating system's impact on the tap-water system as discussed previously when explaining categorizing the features into different views.

The global model is again updated after receiving chunk 3. This time, the generated concept lattice has 63 non-empty concepts, of which 35 concepts link all three views. After the closed patterns are used, there are 35 concepts linking any two views and 18 concepts linking all three views. Table 15 presents all these 18 concepts. Similar to what is observed for the model generated on the first two chunks (Table 14), the average VOM3 values are not close to 100 during the heating season in four concepts (0, 12, 16, and 17). This is explainable as there are influences from the heating system when it is running. It can also be observed in these concepts that the supply temperature (TW_{IST}) of the tap-water system is over the natural threshold (55 °C), which is expected. Furthermore, the values for TW_{FST} are high when compared with other values.

5.2.2 Integrated Analysis of Heating and Tap-Water Systems

For the third experimental scenario, data from both heating and tap-water sub-systems, which are part of an HVAC&R system, are considered. Based on the experiments performed on the tap-water system, it is already observed that the tap-water system is influenced by the heating system, especially during the heating season. Therefore, the following experiment is performed to get a deeper insight of into how these systems work in coordination. Along with being able to highlight the correlations between different views of both systems, this experiment also showcases the flexibility of the proposed algorithm. The local models produced during the experiments of the heating and tap-water systems can be directly used to build a new global model representing the relations between all six views (note that the number of views selected could be dynamically changed based on the requirement).

Two global models are built to find the correlations between both systems, one after receiving chunk 2 and the other after receiving chunk 3. In the first iteration, the global model has 892 non-empty concepts, while in the second iteration, the number of concepts is reduced to 382. This could have also been due to fewer instances

Table 14: Closed patterns from global model showing correlations between all three views after receiving chunk 2. Values are sorted based on OTM followed by TW_V.

S/N	PHL	$\mathbf{TW_{IST}}$	$\mathbf{TW_{FST}}$	$\mathbf{TW_{RT}}$	$\mathbf{TW_V}$	VOM2	VOS2	VOM3	ΔP	OTM	VOM	Size	Months
4	2.58	57.09	56.33	53.97	0.30	5.06	±10.37	98.51	33.26	22.81	0.00	16	[6–9]
8	5.21	56.93	56.24	53.80	0.30	5.18	±11.30	99.01	35.64	18.15	7.31	24	[5–8]
2	5.22	56.7	55.94	53.50	0.68	8.62	±13.77	98.77	39.54	18.17	7.17	15	[5–9]
12	5.55	55.55	54.87	52.47	0.55	8.82	±12.33	99.85	34.08	17.47	7.44	28	[5–9]
3	5.04	55.02	54.38	51.96	0.62	17.07	±9.18	99.99	38.77	16.95	6.22	15	[5, 6]
7	11.56	55.72	55.05	52.68	0.71	10.49	±13.41	99.86	38.83	11.94	11.32	22	[4–6, 9, 10]
1	9.38	57.06	56.51	53.85	0.24	6.02	±12.02	98.24	41.14	11.60	11.27	15	[1, 4, 5, 7, 9]
9	13.60	56.57	55.87	53.43	0.84	8.92	±13.35	99.31	40.17	10.70	11.60	25	[4, 5, 7, 9, 10]
10	11.52	57.08	56.20	53.50	0.85	11.22	±14.21	93.58	51.20	10.26	11.44	26	[2–5]
11	26.48	56.04	55.75	52.67	0.95	11.13	±13.91	99.62	39.40	5.18	15.02	28	[1–3, 5, 10, 12]
0	16.05	56.98	56.30	53.77	0.24	6.33	±10.97	98.36	43.71	4.99	13.72	13	[1–4, 10–12]
6	21.86	56.97	56.12	53.68	0.85	7.59	±11.95	98.58	40.24	4.71	13.60	17	[1–3, 10–12]
15	14.14	56.80	56.23	53.32	0.95	11.48	±15.02	96.45	49.05	4.48	14.20	30	[2–4 11, 12]
13	22.04	58.9	57.60	54.92	0.60	4.60	±10.18	94.78	43.8	3.87	13.11	29	[1–3, 5, 10–12]
5	37.24	56.43	56.11	52.73	0.81	9.35	±14.07	99.47	38.38	2.09	17.43	17	[1–3, 12]
14	46.81	58.64	57.57	53.92	1.00	5.84	±11.87	95.83	42.53	−1.66	18.42	30	[1–3, 12]

Note. The units for PHL and TW_V are kw and m^3, respectively. The unit for TW_{IST}, TW_{FST}, TW_{RT}, ΔP, and OTM is °C. VOM, VOM2, VOS2, and VOM3 are expressed in %. For the full form of each feature see Table 12.

Table 15: Closed patterns from global model showing correlations between all three views after receiving chunk 3. Values are sorted based on OTM followed by TW_V.

S/N	PHL	$\mathbf{TW_{IST}}$	$\mathbf{TW_{FST}}$	$\mathbf{TW_{RT}}$	$\mathbf{TW_V}$	VOM2	VOS2	VOM3	ΔP	OTM	VOM	Size	Months
6	3.34	55.21	54.66	52.31	0.48	6.97	±8.52	100.00	32.55	23.03	0.00	11	[6–8]
15	3.99	55.34	54.55	52.20	0.56	16.39	±10.71	100.00	34.21	22.33	0.26	23	[6–8]
14	4.30	54.90	54.24	51.77	0.64	17.10	±9.73	100.00	35.87	18.24	0.01	20	[7, 8]
7	4.00	54.96	54.28	51.98	0.59	10.39	±10.65	100.00	38.30	17.99	0.00	11	[7, 8]
11	5.04	55.02	54.38	51.96	0.62	17.07	±9.18	99.99	38.77	16.95	6.22	15	[5, 6]
8	7.14	54.85	54.44	51.62	0.72	9.10	±9.02	100.00	50.96	14.63	10.31	11	[9]
4	0.00	54.88	54.49	51.58	0.65	16.14	±9.03	100.00	−13.48	14.25	10.61	10	[9, 10]
13	0.00	54.84	54.49	51.64	0.77	12.10	±10.35	100.00	−21.36	9.88	11.53	16	[10, 11]
16	13.75	56.78	55.88	53.35	0.74	10.89	±13.18	94.62	49.60	9.27	11.59	33	[1, 2, 4, 5, 9]
12	13.11	59.49	57.53	54.81	0.57	7.01	±12.30	90.68	47.41	9.03	11.35	16	[2, 4, 5]
1	0.00	54.94	54.57	51.63	0.61	7.05	±7.30	100.00	−17.17	8.92	11.52	8	[10, 11]
2	0.00	54.85	54.45	51.69	0.90	11.76	±10.99	100.00	−22.76	6.94	13.35	9	[11, 12]
9	0.00	54.90	54.42	51.67	0.57	7.09	±8.73	100.00	−18.08	5.97	13.04	12	[10–12]
5	20.65	54.93	54.50	51.68	0.64	9.62	±9.65	100.00	47.74	5.48	12.97	10	[12]
0	20.27	58.80	57.93	55.14	0.70	5.56	±12.97	93.90	46.09	3.97	12.98	8	[1–3]
10	25.04	55.58	54.78	52.39	0.99	15.93	±12.43	99.12	42.64	3.95	14.60	13	[1–4]
3	0.00	54.85	54.42	51.68	0.82	10.45	±10.84	100.00	−21.17	3.34	14.15	9	[11, 12]
17	24.33	56.37	55.48	53.18	0.79	10.62	±12.41	96.69	45.29	2.77	13.97	38	[1–4, 12]

Note. The units for PHL and TW_V are kw and m^3, respectively. The unit for TW_{IST}, TW_{FST}, TW_{RT}, ΔP, and OTM is °C. VOM, VOM2, VOS2, and VOM3 are expressed in %. For the full form of each feature see Table 12.

available when chunks 2 and 3 are combined compared to the combination of chunks 1 and 2.

Tables 16 and 17 present the concepts retained after using the closed patterns (8, 10 concepts after chunk 2 and 3 have arrived, respectively). These concepts present the correlation between all six views from both the systems. Note that all the features available in the global model are not presented in the tables, as they are too many. Some interesting features that represent the relations between the considered systems are selected.

From Table 16, a deviating behavior can be observed for concept 4. VOM deviates from the patterns seen in other concepts. In addition, it is interesting to see that VOM3 has the greatest drop (6.03%) in value from 100% when compared to other concepts. That is, the valve lets in cold water to reduce the hot tap-water temperature, which is not a desired function. Concept 4 also shows a sudden drop in the trends of the TW_V. Based on the observed patterns, it is expected to have an increase in hot water usage as the temperature decreases, but this is not the case for this concept.

Concept 0 has a deviating behavior with respect to PHL, i.e., 3 kW (note that this was also identified when analyzing the heating sub-system individually). If one takes a closer look at the features considered for the tap-water system, it can be observed that the ΔP and TW_V are the highest during this period. It is interesting to observe this, as having higher ΔP is considered a desired functionality since the system is consuming the energy provided. However, when the raw data are investigated it is noticed that the primary supply and return temperatures are constant at 89 and 38, respectively, leading to a value of 51 for ΔP. This represents a potential fault in these sensors. Based on these for concept 0, it is concluded that there could have been some issues with the sensors collecting the PHL and primary supply and return temperatures data.

For concept 2, it can be observed that the openness of VOM2 (17.43%) is unusually high compared to the other concepts and can be interpreted as a deviating behavior. It is interesting to note that the same concept is retained even after receiving chunk 3 (concept 7 from Table 17), implicating that the data characteristics do not match with any of the new data, which further solidifies that it might be a deviating concept. Interestingly, when the domain expert investigates the system to find the actual cause, it is found to be strange but expected behavior. This concept is easily identified in the integrated scenario compared to the results only from the tap water system as there is more than one concept with similar behavior. This demonstrates that the integrated scenario can help identify trends not visible in individual system analysis. The influence of these systems on each other can reveal hidden patterns and deviating behaviors.

In Table 17, concept 0 has a deviating behavior with respect to both the PHL (0 kW) and ΔP (-19.18 °C). It also has a negative SE value (-420%) which is out of range of the normal SE values; hence it can be concluded that there were issues with the system during this time (this was also identified while analyzing the heating

Table 16: Closed patterns from global model showing correlations between all six views after receiving chunk 2. Values are sorted based on OTM.

S/N	PHL	SST	SSR	VOM	SE	OTM	$\mathbf{TW_{IST}}$	$\mathbf{TW_{FST}}$	$\mathbf{TW_{RT}}$	$\mathbf{TW_{V}}$	VOM2	VOM3	$\mathbf{\Delta P}$	Size	Months
1	2.66	27.02	26.53	0.00	67	23.15	56.96	56.29	53.95	0.29	5.36	98.50	33.24	14	[6–8]
6	4.83	29.51	26.83	7.23	70	18.19	56.89	56.26	53.74	0.30	5.03	99.16	35.51	21	[7, 8]
3	5.40	29.38	26.86	7.25	69	17.95	55.69	55.00	52.64	0.45	7.71	99.72	31.52	15	[6–8]
2	5.03	26.57	26.08	6.16	80	16.92	55.01	54.36	51.94	0.61	17.43	100.00	39.14	14	[6]
0	3.00	42.12	36.19	14.83	97	5.60	56.49	56.47	52.95	0.97	11.79	98.57	51.00	14	[3, 4]
5	26.56	43.09	36.82	15.09	100	5.23	55.88	55.73	52.53	0.95	11.34	99.72	39.07	19	[1, 2, 12]
4	22.52	44.61	39.25	13.12	96	3.85	58.85	57.47	54.84	0.69	4.57	93.97	43.93	16	[10, 11]
7	47.60	50.89	40.04	18.58	99	−1.70	58.64	57.58	53.90	1.00	5.78	95.62	42.42	27	[1, 2]

Note. The units for PHL and TW_V are kw and m^3, respectively. The unit for TW_{IST}, TW_{FST}, TW_{RT}, ΔP, and OTM is °C. VOM, VOM2, and VOM3 are expressed in %. For the full form of each feature see Tables 1 and 12.

Table 17: Closed patterns from global model showing correlations between all six views after receiving chunk 3. Values are sorted based on OTM.

S/N	PHL	SST	SSR	VOM	SE	OTM	$\mathbf{TW_{IST}}$	$\mathbf{TW_{FST}}$	$\mathbf{TW_{RT}}$	$\mathbf{TW_{V}}$	VOM2	VOM3	$\mathbf{\Delta P}$	Size	Months
4	3.34	27.20	26.10	0.00	70	23.03	55.21	54.66	52.31	0.48	6.97	100.00	32.55	11	[6–8]
9	4.05	26.47	26.14	0.27	74	22.41	55.36	54.56	52.21	0.56	16.45	100.00	34.56	22	[6–8]
1	3.92	26.72	24.94	0.00	75	18.08	54.92	54.24	51.94	0.57	10.33	100.00	38.39	10	[7, 8]
8	4.39	26.32	24.83	0.01	74	18.53	54.91	54.25	51.77	0.65	16.46	100.00	36.39	16	[7, 8]
7	5.03	26.57	26.08	6.16	80	16.92	55.01	54.36	51.94	0.61	17.43	100.00	39.14	14	[6]
5	7.14	25.82	24.26	10.31	83	14.63	54.85	54.44	51.62	0.72	9.10	100.00	50.96	11	[9]
0	0.00	32.00	29.86	11.71	−420	9.10	54.83	54.46	51.6	0.75	11.81	100.00	−19.18	8	[10, 11]
2	20.65	42.07	35.52	12.97	97	5.48	54.93	54.50	51.68	0.64	9.62	100.00	47.74	10	[12]
6	24.32	44.84	37.66	13.79	101	3.40	55.92	55.15	52.70	0.73	9.80	97.95	46.02	12	[1, 2, 12]
3	29.51	47.75	39.80	15.10	104	0.87	56.48	55.47	53.14	1.03	12.71	95.69	46.33	11	[1, 2]

Note. The units for PHL and TW_V are kw and m^3, respectively. The unit for TW_{IST}, TW_{FST}, TW_{RT}, ΔP, and OTM is °C. VOM, VOM2, and VOM3 are expressed in %. For the full form of each feature see Tables 1 and 12.

sub-system).

Similar to what was observed in the tap-water system, one can see that the VOM3 has deviated from the desired average value of 100% (concepts $4, 7$ from Table 16 and concepts $3, 6$ from Table 17) mainly during the heating seasons and when the outdoor temperature is close to 0 °C. It is interesting to note that the SE and VOM both show acceptable values for this period.

The above analysis shows that the integrated global models built using the sub-systems' local models can also represent the deviating behaviors observed while analyzing each system individually. This provides the opportunity to have a high-level overview of correlations between both considered systems and helps identify deviating concepts that were not so obvious to identify when only a single system is considered.

6 Applicability and Limitations

In this study, we investigate the use of the MV Multi-Instance Clustering approach proposed in [5] for monitoring smart building systems' sensor data. Two data mining techniques are developed by applying this approach and are studied in this paper. Those can be used for multi-view analysis, mining, and visualization of sensor data to assist domain experts in monitoring and analyzing different systems' behavior. One of the techniques considers contextual factors in the analysis of system behaviour and performance. The other focuses on dealing with integrated systems, such as those available in the smart building domain. The proposed MV approach additionally allows the domain experts to set the threshold (support) used to identify frequent patterns based on their interests. Such flexibility enables the domain experts to monitor different sub-systems based on various criteria and objectives. The conducted experiments demonstrate that the proposed data mining techniques are capable of identifying deviating behaviors. In general, the presented data analytic tools may be used in other similar applied scenarios relying on static sensor networks for system monitoring.

In addition to the applicability, we identified three limitations in the current study. First, the study mainly focuses on the sub-systems of the HVAC&R system of a specific building. In the future, we plan to explore and evaluate the algorithm's performance on other systems and different types of buildings. The second limitation concerns the studied contextual conditions. Currently, only two contextual factors, namely outdoor temperature and the effect of the heating system on the tap-water system, are considered. Other complex parameters representing the social behavior of the people living in the building can be included in the model. One such example is dividing a day into parts representing people's typical daily activities, e.g., morning, afternoon, evening, and night, or including the day category, i.e., weekday or weekend. The third identified limitation is related to the concept drift. As stated in

Section 2.2, there are six different types of concept drifts. The current study does not perform explicit experimentation to test the proposed approach's ability to detect these drift scenarios. Based on the experiments and results obtained, one can conclude that the approach is able to identify frequent deviating behavior groups (based on the user-defined threshold). However, further analysis needs to be performed to determine the algorithm's performance in identifying different concept drift types.

7 Conclusion and Future Work

In this study, we have demonstrated how our multi-view stream clustering algorithm, entitled MV Multi-Instance Clustering algorithm [5], can be used to analyze and monitor different systems present in a smart building environment. The approach considers the multi-source nature of the smart building data and provides individual context-aware and integrated tools of modeling and analyzing the system behavior. We propose various visualization and data mining techniques that can be used at each step of the proposed algorithm. These visualizations facilitate further perception and understanding of the obtained results and can be used by the domain experts in step-by-step analysis of the system behaviour and performance.

Our multi-view stream clustering algorithm perfectly suits the multi-source nature of the data in the smart building domain usually collected from multiple systems. It can be used to analyze these systems due to its flexible character; i.e., it can dynamically select the views used to build the global model to analyze single or multiple systems together as per the need. This flexibility is demonstrated in our work by analyzing the heating and tap-water systems individually and together. The obtained results have shown that our algorithm has the potential to be used in the smart building domain for monitoring and analyzing system behavior and performance. The approach has successfully identified new trends and deviating or non-desired behavioral modes. The built global model has also showcased various correlations between different views considered. The proposed algorithm can facilitate the domain experts in obtaining more profound insights into systems' performances and at the same time be able to identify and analyze deviating behavior.

Our future plans include exploring other smart building systems and richer contextual conditions. For example, the ventilation sub-system, which is also a part of the HVAC&R system, could be included in the analysis as heating, tap-water, and ventilation sub-systems affect one another. In addition, in order to reduce the effects of social behavior of people on the analysis, we are interested in studying contextual factors. Note that each building has a unique and recurring social behavior patterns and energy usage. Furthermore, the ability of the algorithm in identifying different types of concept drift will be investigated. Finally, we plan to work in the direction of building a user-friendly prototype of the algorithm with the proposed visualizations at each phase so that the domain experts can directly use it in their regular day-to-day

analysis of the systems.

Abbreviations

The following abbreviations are used in this manuscript:

CC	Contextual Conditions
DH	District Heating
HVAC&R	Heating, Ventilation, Air Conditioning and Refrigeration
MAD	Median Absolute Deviation
MI	Multi-Instance
ML	Machine Learning
MV	Multi-View
MV-MIC	MV Multi-Instance Clustering
OM	Operating Modes
OTM	Outdoor Temperature Mean
OTS	Outdoor Temperature Standard Deviation
PM	Performance Modes
PHL	Primary Heat Load
SRT	Secondary Return Temperature
SST	Secondary Supply Temperature
SI	Silhouette Index
SE	Sub-station Efficiency
TW_{FST}	Tap-water Final Supply Temperature
TW_{IST}	Tap-water Initial Supply Temperature
TW_{RT}	Tap-water Return Temperature
TW_V	Tap-water Volume Consumed
VOM	Valve Openness Mean
VOS	Valve Openness Standard Deviation

References

[1] H. Farzaneh, L. Malehmirchegini, A. Bejan, T. Afolabi, A. Mulumba, and P. P. Daka. “Artificial intelligence evolution in smart buildings for energy efficiency”. In: *Applied Sciences (Switzerland)* 11.2 (2021), pp. 1–26.

[2] R. Jafari-Marandi, M. Hu, and O. Omitaomu. “A distributed decision framework for building clusters with different heterogeneity settings”. In: *Applied Energy* 165 (2016), pp. 393–404. DOI: 10.1016/j.apenergy.2015.12.088.

[3] M. Hu, J. D. Weir, and T. Wu. “Decentralized operation strategies for an integrated building energy system using a memetic algorithm”. In: *European Journal of Operational Research* 217.1 (2012), pp. 185–197. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2011.09.008.

[4] G. Mbiydzenyuy, S. Nowaczyk, H. Knutsson, D. Vanhoudt, J. Brage, and E. Calikus. “Opportunities for Machine Learning in District Heating”. In: *Applied Sciences* 11.13 (2021). ISSN: 2076-3417. DOI: 10.3390/app11136112. URL: https://www.mdpi.com/2076-3417/11/13/6112.

[5] V. M. Devagiri, V. Boeva, and S. Abghari. “A Multi-view Clustering Approach for Analysis of Streaming Data”. In: *Artificial Intelligence Applications and Innovations*. Ed. by I. Maglogiannis, J. Macintyre, and L. Iliadis. Cham: Springer International Publishing, 2021, pp. 169–183. ISBN: 978-3-030-79150-6.

[6] L. Fu, P. Lin, A. V. Vasilakos, and S. Wang. “An overview of recent multi-view clustering”. In: *Neurocomputing* 402 (2020), pp. 148–161. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.02.104.

[7] S. Huang, Z. Xu, I. W. Tsang, and Z. Kang. “Auto-weighted multi-view co-clustering with bipartite graphs”. In: *Information Sciences* 512 (2020), pp. 18–30. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.09.079.

[8] M. Ghesmoune, M. Lebbah, and H. Azzag. “State-of-the-art on clustering data streams”. In: *Big Data Analytics* 1.1 (2016), pp. 1–27.

[9] A. Zubaroglu and V. Atalay. “Data stream clustering: a review”. In: *Artificial Intelligence Review* 54.2 (2021), pp. 1201–1236. DOI: 10.1007/s10462-020-09874-x.

[10] K. Wadewale and S. Desai. “Survey on Method of Drift Detection and Classification for time varying data set”. In: *Int. Res. J. Eng. Technol.* Vol. 2. 2015, pp. 709–713.

[11] J. Foulds and E. Frank. “A review of multi-instance learning assumptions”. In: *Knowledge Engineering Review* 25.1 (2010), pp. 1–25. DOI: 10.1017/S026988890999035X.

[12] Y. Chen, J. Bi, and J. Wang. “MILES: Multiple-instance learning via embedded instance selection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.12 (2006), pp. 1931–1947. DOI: 10.1109/TPAMI.2006.248.

[13] M. Kandemir and F. Hamprecht. “Computer-aided diagnosis from weak supervision: A benchmarking study”. In: *Computerized Medical Imaging and Graphics* 42 (2015), pp. 44–50. DOI: 10.1016/j.compmedimag.2014.11.010.

[14] M. Zhang and Z. Zhou. “Multi-instance clustering with applications to multi-instance prediction”. In: *Applied Intelligence* 31 (2009), pp. 47–68.

[15] G. Edgar. *Measure, Topology, and Fractal Geometry, 3rd*. edn. Springer, Berlin, 1995.

[16] J. Wang and J.-D. Zucker. “Solving the multiple-instance problem: a lazy learning approach”. In: *Proc. of the 17th ICML*. 2000, pp. 1119–1125.

[17] B. Ganter, G. Stumme, and R. Wille. “Formal Concept Analysis: Foundations and Applications”. In: LNAI, no. 3626, Springer-Verlag, 2005.

[18] R. Agrawal and R. Srikant. “Mining sequential patterns”. In: *Proc. of the 11th Int. Conf. on Data Engineering*. IEEE. 1995, pp. 3–14.

[19] J. Wang and J. Han. “BIDE: efficient mining of frequent closed sequences”. In: *Proceedings of the 20th International Conference on Data Engineering*. 2004, pp. 79–90.

[20] G. Chao, S. Sun, and J. Bi. “A survey on multi-view clustering”. In: *IEEE Transactions on Artificial Intelligence* (2021), pp. 1–1. DOI: 10.1109/TAI.2021.3065894.

[21] Y. Yang and H. Wang. “Multi-view clustering: A survey”. In: *Big Data Mining and Analytics* 1.2 (June 2018), pp. 83–107. ISSN: 2096-0654.

[22] X. Liu and et al. “Late Fusion Incomplete Multi-View Clustering”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 41.10 (2019), pp. 2410–2423.

[23] Y. Ye and et al. “Incomplete Multiview Clustering via Late Fusion”. In: *Computational Intelligence and Neuroscience* 2018 (Oct. 2018), pp. 1–11.

[24] B. Jiang and et al. “Evolutionary multi-objective optimization for multi-view clustering”. In: *2016 IEEE CEC 2016*. 2016, pp. 3308–3315.

[25] J. Liu and et al. “Multi-view clustering via joint non-negative matrix factorization”. In: *Proceedings of the 2013 SIAM International Conference on Data Mining, SDM 2013*. 2013, pp. 252–260.

[26] L. Huang and et al. “MVStream: Multiview Data Stream Clustering”. In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3482–3496.

[27] W. Shao and et al. “Online multi-view clustering with incomplete views”. In: *2016 IEEE Int. Conf. on Big Data (Big Data)*. 2016, pp. 1012–1017.

[28] S. Abghari, V. Boeva, J. Brage, and H. Grahn. “Multi-view clustering analyses for district heating substations”. In: 2020, pp. 158–168.

[29] T. Felix, K. Patrick, B. Ralph, W. Wolfgang, B. Wolfgang, and W. Adrian. “Fault Detection and Condition Monitoring in District Heating Using Smart Meter Data”. In: *Proceedings of the 6th European Conference of the Prognostics and Health Management Society* (2021), pp. 407–417.

[30] A. Eghbalian, S. Abghari, V. Boeva, and F. Basiri. “Multi-view Data Mining Approach for Behaviour Analysis of Smart Control Valve”. In: 2020, pp. 1238–1245. DOI: 10.1109/ICMLA51294.2020.00195.

[31] E. Shchetinin. “Improving the efficiency of energy consumption in smart grids with application of artificial intellect”. In: vol. 2267. 2018, pp. 313–317.

[32] C. Zhang, Y. Zhao, T. Li, X. Zhang, and M. Adnouni. “Generic visual data mining-based framework for revealing abnormal operation patterns in building energy systems”. In: *Automation in Construction* 125 (2021), p. 103624. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2021.103624. URL: https://www.sciencedirect.com/science/article/pii/S0926580521000753.

[33] S. J. Simoff, M. H. Böhlen, and A. Mazeika. “Visual Data Mining: An Introduction and Overview”. In: *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*. Ed. by S. J. Simoff, M. H. Böhlen, and A. Mazeika. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–12. ISBN: 978-3-540-71080-6. DOI: 10.1007/978-3-540-71080-6_1. URL: https://doi.org/10.1007/978-3-540-71080-6_1.

[34] D. Keim. “Information visualization and visual data mining”. In: *IEEE Transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 1–8. DOI: 10.1109/2945.981847.

[35] F. R. Hampel. “A General Qualitative Definition of Robustness”. In: *The Annals of Mathematical Statistics* 42.6 (1971), pp. 1887–1896. ISSN: 00034851. URL: http://www.jstor.org/stable/2240114.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[37] H. Gadd and S. Werner. “Heat load patterns in district heating substations”. In: *Applied Energy* 108 (2013), pp. 176–183. ISSN: 0306-2619.

Paper V

A Graph-based Multi-view Clustering Approach for Continuous Pattern Mining

Christoffer Åleskog, Vishnu Manasa Devagiri, Veselka Boeva

In: Recent Advancements in Multi-View Data Analytics, Ed. by W. Pedrycz and SM. Chen. Cham: Springer International Publishing, 2022, pp. 201–237, DOI: 10.1007/978-3- 030-95239-6_8

Abstract

Today's smart monitoring applications need machine learning models and data mining algorithms that are capable of analysing and mining the temporal component of data streams. These models and algorithms also ought to take into account the multi-source nature of the sensor data by being able to conduct multi-view analysis. In this study, we address these challenges by introducing a novel multi-view data stream clustering approach, entitled MST-MVS clustering, that can be applied in different smart monitoring applications for continuous pattern mining and data labelling. The proposed approach is based on the Minimum Spanning Tree (MST) clustering algorithm. This algorithm is applied for parallel building of local clustering models on different views in each chunk of data. The MST-MVS clustering transfers knowledge learnt in the current data chunk to the next chunk in the form of artificial nodes used by the MST clustering algorithm. These artificial nodes are identified by analyzing multi-view patterns extracted at each data chunk in the form of an integrated (global) clustering model. We further show how the extracted patterns can be used for post-labelling of the chunk's data by introducing a dedicated labelling technique, entitled Pattern-labelling. We study and evaluate the MST-MVS clustering algorithm under different experimental scenarios on synthetic and real-world data.

Keywords: Data stream, Clustering analysis, Pattern mining, Minimum spanning tree

1 Introduction

Data collected today in smart monitoring applications have a heterogeneous nature due to the fact that they are originated from multiple sources, more often providing information about different perspectives, views or perceptions of the monitored phenomenon, physical object, system. In addition, in many real-world applications (e.g., learning from sensor data streams) the availability of relevant labelled data is often low or even non-existing. In this context, one challenge is how the newly arriving information can be taken into account in the learning or monitoring phase and how the built model can be used for analysis and pattern mining of the temporal component of data streams. This problem however, cannot be considered and solved individually. It is also necessary to take into account the multi-source nature of the sensor data by developing machine learning (ML) techniques that can distribute model training and evaluation among multiple data sources or views. Some recent works in the data mining field address these challenges [1], [2], [3]. For example, MV Multi-Instance Clustering [1] is a novel multi-view stream clustering algorithm that handles the multi-source nature of today's data by clustering each data view separately, thereby inducing a parallel part to the process. The knowledge from different views is integrated using Formal Concept Analysis (FCA) [4]. MVStream [3], is another multi-view stream clustering approach. It combines the data points from different views by transferring them into a common kernel space and identifying the common support vectors for all views. More related studies are discussed in Section 2.3.

Inspired by the above discussed challenges and current need in the area, we propose a novel multi-view data stream clustering approach that can be applied in different smart monitoring applications for continuous pattern mining and data labelling. This approach is based on the Minimum Spanning Tree (MST) clustering algorithm, which is used at each data chunk for parallel building of local clustering models on different data views. Knowledge between chunks is transferred in the form of artificial nodes used by the MST clustering algorithm. The artificial nodes are identified by analyzing the global model that is built at each data chunk by integrating view patterns extracted from the local clustering models. The proposed approach, entitled MST-MVS clustering algorithm, can be considered as a communication-efficient and privacy-preserving solution, since it is not the chunk's entire data that is transferred for building the global model. The latter one consists of integrated multi-view patterns that can be used for post-labelling of the current chunk's data.

Different configurations of the MST-MVS clustering algorithm are studied and evaluated under different experimental scenarios on two types of data: synthetic and real-world. We study two different approaches for identifying artificial nodes that are used to seed the MST algorithm producing local clustering models at the next data chunk. In addition, we investigate how the knowledge transfer affects the performance of the proposed MST-MVS algorithm by comparing it with an algorithm version that does not use artificial nodes to seed the clustering at the next data chunk.

We also propose a pattern-based labelling (Pattern-labelling) technique, which is evaluated and benchmarked to an approach relying on Convex Non-negative Matrix Factorization (CNMF). Finally, the patterns extracted by the proposed algorithm from the sensor dataset used in our experiments are benchmarked to the results produced by other state-of-the-art algorithm, MV Multi-Instance Clustering, on the same dataset.

The evaluation of the MST-MVS clustering algorithm demonstrates a higher performance on the synthetic data than on the real-world data. This result is logical, since in comparison with the synthetic data, real-world data usually does not present a perfect clustering structure, but often has outliers and overlapping clusters. In addition, the labelled real-world dataset used in our experiments is not specifically designed for multi-view scenarios and this might have affected the performance of our multi-view clustering algorithm. The transfer of knowledge feature is shown to have a positive effect on the performance of MST-MVS algorithm. The proposed Pattern-labelling technique outperforms the CNMF-labelling algorithm in the conducted experiments both in terms of computational time and results obtained.

The rest of the paper is organised as follows. Section 2 presents the recent work done in the areas of multi-view clustering and stream clustering followed by the background in Section 3, which introduces the main concepts and methods used in the study. The proposed MST-MVS clustering algorithm is presented in Section 4. Then the data used along with the experimental settings are described in Section 5. This is followed by results and discussion in Section 6, and conclusion and future work in Section 7.

2 Related Work

2.1 Multi-view Clustering Algorithms

Multi-view clustering algorithms have been intensively studied in the recent years and many different algorithms and concepts have been proposed [5], [6], [7], [8]. Most of these algorithms cluster data from different views often into a consensus model while holding all data in memory.

Many research studies have proposed multi-view clustering algorithms based on Non-negative Matrix Factorization (NMF) [9], [6], [10], [11]. For example, Liu et al. combine the objective functions of different views (minimization problems) and add a consensus part to it [6]. They propose a novel way to use the l_1 normalization to solve the problem of multiple views with NMF by imposing a new constraint to the objective function. In [10], the authors propose a co-regularized multi-view NMF approach with correlation constraint for non-negative representation learning. The proposed approach imposes correlation constraint on the low-dimensional space to learn a common latent representation shared by different views.

Peng et al. [7] propose a novel multi-view clustering algorithm without parameter

selections, entitled COMIC. The proposed algorithm projects data points into a space where the two properties, geometric consistency and cluster assignment consistency, are satisfied. The data are clustered without any parameters by enforcing a view consensus onto a connecting graph.

Bendechache and Kechadi [5] propose a distributed clustering algorithm using k-means. It clusters each view applying the k-means algorithm and merges overlapping clusters of different views. This is done multiple times with the same data until a specified level has been reached. This level is one of the algorithm parameters, additionally to the number of clusters for each view needed by the k-means algorithm.

Two graph-based multi-view clustering techniques have been recently published [12, 13]. The first study has proposed an affinity graph representation learning framework for modelling multi-view clustering tasks [12]. The approach consists of two separate stages. Namely, a robust graph Laplacian from each view is initially constructed and then those are fused using the proposed Consistent Affinity Graph Learning algorithm. In [13], multi-view subspace clustering networks using local and global graph information are introduced. The autoencoder networks are performed on different views simultaneously. In that way latent representations that conform to the linear subspace model can be achieved. Finally, a representation shared among the views is obtained by integrating the underlying graph information into the self-expressive layer of autoencoder networks.

2.2 Stream Clustering Algorithms

Ghesmoune et al. [14] discuss previous works on data stream clustering problems, and highlight the most relevant algorithms proposed in the literature to deal with these problems [15], [16], [17], [18].

Coa et al. [17] propose an iterative clustering algorithm with NMF, entitled ONMF. It uses the property in [19], that states that NMF is equivalent to k-means if the orthogonality constraint $HH^T = I$ is included in the minimization problem. The ONMF algorithm clusters temporally changing data streams, divided into chunks, where each chunk is clustered by applying the orthogonal NMF. Then the results are propagated to the next chunks' calculations to retain the knowledge found in the previous chunks. This process is continued until all data has been clustered.

In [16], the evolution of the data is addressed by dividing the clustering process into two components: online and offline. The online component periodically stores detailed summary statistics, while the offline component generates the final clusters. Similar solutions are proposed in [15], [18].

Wang et al. [20] has proposed an algorithm, entitled SVStream. It is a predecessor of MVStream [3], discussed in Section 2.3, and is similar to its successor by applying the Support Vector Clustering (SVC) to cluster the data. The difference in this algorithm when compared with its successor, excluding the multi-view part, is how

SVStream merges the spheres created by the current and next data chunks. SVStream updates the kernel space, where the sphere is clustered after each new data chunk. It removes old support vectors that could affect the clustering and merging, and updates spheres accordingly, thereby transferring knowledge of the view's correlation to the next data chunk.

2.3 Multi-view Stream Clustering Algorithms

As it is discussed in the introduction, the research field that applies multi-view clustering analysis on streaming data is still relatively new. The study of Shao et al. published in [21] is seen as one of the first articles that has discussed the idea of sectioning data into different chunks for conducting multi-view clustering. In [22], NMF is used and showed that under certain constraints it can be used for clustering. A detail description can be found in [19]. The proposed version of the NMF algorithm is combined with itself to create a new minimization problem, where different views could be inputted, and a consensus matrix describing the clustering structure could be generated. The views are divided into chunks, and for each new chunk, the algorithm generates a new consensus matrix with the information from the consensus matrix of the previous chunk [19].

A new multi-view stream clustering algorithm, entitled MVStream, has been proposed in [3]. It is similar to the approach of Shao et al. [21], but uses the SVC algorithm [23], adjusted to unsupervised data with the Position Regularized Support Vector Domain Description (PSVDD) [24]. It works by combining and transforming all data points from the different views into a common kernel space (global model) and, from this space finding the common support vectors for all views. These support vectors are transformed back to each view's space, resulting in the contours of different arbitrary clusters. The algorithm also transfers knowledge between chunks by incorporating the previous chunks support vectors as data points in the current chunks views, thereby retaining the view's correlations in the previous chunks.

Devagiri et al. [2], [1] have proposed two of the recent algorithms in the field of multi-view stream clustering. The first algorithm [2], is entitled MV Split-Merge Clustering and is based on a previous work of the authors developing a stream clustering algorithm, called Split-Merge Clustering [25]. The MV Split-Merge Clustering algorithm works by updating the local (views') models in each chunk by applying the Split-Merge Clustering on the local models produced at the previous and current data chunks. Then the updated local models are integrated into a global model by applying Formal Concept Analysis (FCA) [4]. The algorithm proposed in [1], entitled MV Multi-Instance Clustering, is more effective by addressing the limitations of [2]. The MV Multi-Instance Clustering uses Bipartite Multi-Instance clustering algorithm to update the local models which proved to provide better results. In addition, it also uses closed patterns to mine the most frequent patterns or concepts from the formal

context which is obtained using FCA. This reduces the size of the lattice generated in the global model making it easy to interpret and analyse.

3 Background

3.1 Minimum Spanning Tree Clustering

The *Cut-Clustering* algorithm is a graph-based clustering algorithm, based on minimum cut tree algorithms to cluster the input data [26]. It has been used in other clustering algorithms, e.g., [27],[28]. The input data is represented by a similarity (an undirected) graph, where each node is a data point and two nodes are connected if the similarity between the corresponding data points is positive, and the edge is weighted by the corresponding similarity score. The algorithm works by adding an artificial node to the existing graph and connecting it to all nodes in the graph with a value α. A minimum spanning tree is computed, and the artificial node is removed. The clusters consist of the nodes connected after the artificial node has been removed. The pseudo-code of Minimum Spanning Tree (MST) clustering algorithm is presented in Algorithm V.1.

Algorithm V.1 Minimum Spanning Tree Clustering

1: **procedure** MSTCLUSTERING($G(V, E), \alpha$)
2: Let $V' := V \cup t$.
3: **for** $v \in V$ **do**
4: Connect t to v with edge of weight α
5: **end for**
6: Let $G'(V', E')$ be the expanded graph after connecting t to V
7: Calculate the minimum spanning tree T' of G'
8: Remove t from T'
9: **return** All connected components as the clusters of G
10: **end procedure**

Lv et al. [29] propose another MST-based clustering algorithm, entitled CciMST. CciMST starts by finding a MST and calculating pairwise Euclidean and geodesic distances of all pairs of data points. The graph and distances are then used for finding the cluster centers based on the density and variances of the data points in the graph. The algorithm ends by determining which edges in the graph should be cut, producing and comparing two results, choosing the one where the clusters have a bigger gap between them.

Many different algorithms for finding a minimum spanning tree (MST) of the complete graph exist. The two well-known greedy algorithms for computing MST are Kruskal's and Prim's algorithms. The algorithm we use in our study is Kruskal's

algorithm [30]. It works by sorting all the graph edges in increasing order of weights and choosing the shortest edge. The two nodes connected by the shortest edge are combined to the tree. The process repeats by finding a new shortest edge that does not create a cycle. These steps are continued until $m-1$ (m is the number of graph nodes) edges are selected, i.e. a MST is found.

The value of α is an input parameter for the Cut-Clustering algorithm and it plays a crucial role in the quality of the produced clusters. Namely, the value of this parameter has an impact on how many clusters the algorithm will produce. It can be observed that as α goes to infinity, the Cut-Clustering algorithm will produce only one cluster, namely the entire graph. On the other extreme, as α goes to 0, the clustering algorithm will produce m trivial clusters (all singletons), where m is the number of graph nodes. For values of α between these two extremes the number of clusters will be between 1 and m, but the exact value depends on the graph structure and the distribution of the weights over the edges. What is important though, is that there is no direct correlation between the number of clusters and α value.

There are different ways to identify the α value. For example, a binary-like search algorithm is used in [26] to determine the optimal value for α. This is done by executing the algorithm for each loop and choosing the most stable α value. In this study, we use the mean α value that is calculated by the formula:

$$\alpha = \frac{1}{m} \sum_{i=1}^{m} \sum_{j \neq i}^{m} \frac{w_{ij}}{deg(v_i)}, \tag{V.1}$$

where m is the number of nodes in the graph, w_{ij} is the weight of the edge connecting nodes i and j, and $deg(v_i)$ is the degree of node i.

3.2 Non-negative Matrix Factorization

Traditional NMF [22] approximates two non-negative matrices $W \in \mathbb{R}_{+}^{n \times k}$ and $H \in \mathbb{R}_{+}^{k \times m}$ into a non-negative matrix $X \in \mathbb{R}_{+}^{n \times m}$. This results in a minimization problem with the objective:

$$X \approx WH. \tag{V.2}$$

Formerly used to save space in memory, the algorithm can also be used for clustering according to Ding et al. [19]. It is equivalent to k-means if the constraint $HH^T = I$ is additionally imposed on the objective function. Therefore, when NMF is used for clustering, k is the specified number of clusters to be found, columns of W holds the centroids of the clusters, and the rows with the maximum value in the columns of H describe which cluster a data point belongs to.

Convex Non-negative Matrix Factorization (CNMF) [31] is a modification of the traditional NMF that results in better centroids in W. CNMF approximates a mixed-sign data matrix X into a matrix W and non-negative matrix H. W is also divided into one mixed-sign data matrix S and a non-negative matrix L, where L is

the labelling matrix and S is a data matrix. An extensive explanation of S and how it can be used for missing data can be found in [32]. However, since missing data is not considered in this study, $S \equiv X$. Then the factorization is in the following form:

$$F = X_{\pm}L_{+} \text{ and } X_{\pm} \approx FH_{+}. \tag{V.3}$$

CNMF imposes the constraint $\|L\|_1 = 1$ to lift the scale indeterminacy between L and H so that F_i is precisely a convex combination of the elements in X. The approximation is quantified by the use of a cost function that is constructed by distance measures. Usually the square of the Euclidean distance, also known as the Frobenius norm, is used [33]. The objective function is therefore, as follows:

$$\min_{W,H} \mathbb{L} = \|X - XLH^T\|_F^2 \text{ such that } L \geq 0, H \geq 0, \|L\|_1 = 1, \tag{V.4}$$

where $X \in \mathbb{R}^{n \times m}$ is the data to be clustered, $L \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times m}$ are non-negative matrices, n is the dimensionality of the feature space, m is the number of data points, and k is the dimension to reduce to. The symbols $\|.\|_F^2$ and $\|.\|^1$ denotes the Frobenius norm and Manhattan norm [34] of the expression it encases, respectively.

3.3 Cluster Validation Measures

Cluster validation measures have a very important role in cluster analysis by providing means for validation of clustering results. A range of different cluster validation measures are published in the data mining literature [35], [36]. They can be divided into two major categories: *external* and *internal* [37]. External validation measures have the benefit of providing an independent assessment of the clustering quality, since they evaluate the clustering result with respect to a pre-specified structure. Internal validation techniques on the other hand avoid the need for using such additional knowledge. Namely, they base their validation on the same information used to derive the clusters themselves. Internal measures can be split with respect to the specific clustering property they reflect and assess to find an optimal clustering scheme: *compactness*, *separation*, *connectedness*, and *stability* of the cluster partitions.

External validation measures can be of two types: *unary* and *binary* [38]. Some authors consider a third type of external validity approaches, namely *information theory* [39]. Unary external evaluation measures take a single clustering result as the input, and compare it with a known set of class labels to assess the degree of consensus between the two. Comprehensive measures like the F-measure provide a general way to evaluate this [40]. In addition to unary measures, the data-mining literature also provides a number of indices, which assess the consensus between a produced partitioning and the existing one based on the contingency table of the pairwise assignment of data items. Most of these indices are symmetric, and are therefore

equally well-suited for the use as binary measures, i.e., for assessing the similarity of two different clustering results. Probably the best known such index is the Rand Index [41], which determines the similarity between two partitions as a function of positive and negative agreements in pairwise cluster assignments. A third class of indices is based on concepts from information theory [42]. Information theoretic indices assess the difference in shared information between two partitions. One of commonly used information theoretic indices is the adjusted mutual information [43].

The clustering algorithms usually do not perform uniformly well under all scenarios. Therefore, it is more reliable to use a few different cluster validation measures in order to be able to reflect various aspects of a partitioning. In this study, we use five different cluster validation measures to evaluate the clustering results generated in our experiments: one binary external measure (Adjusted Rand Index), three external measures related to information theory (Adjusted Mutual Information, Homogenity and Completeness) and one internal cluster validation measure (Silhouette Index). The definitions of the used evaluation measures are given hereafter.

3.3.1 Silhouette Index

Silhouette Index (SI), introduced in [44], is a cluster validation index that is used to judge the quality of any clustering solution $C = C_1, C_2, \ldots, C_k$. Suppose a_i represents the average distance of object i from the other objects of its assigned cluster, and b_i represents the minimum of the average distances of object i from objects of the other clusters. Subsequently the Silhouette Index of object i can be calculated by:

$$s(i) = \frac{(b_i - a_i)}{\max\{a_i, b_i\}}. \tag{V.5}$$

The overall Silhouette Index for clustering solution C of m objects is defined as:

$$s(C) = \frac{1}{m} \sum_{i=1}^{m} \frac{(b_i - a_i)}{\max\{a_i, b_i\}}. \tag{V.6}$$

The values of Silhouette Index vary from -1 to 1, and higher values indicate better clustering results, while a negative value shows that there are wrongly placed data points within the clustering solution. As the Silhouette Index compares the distances from instances to its respective cluster against the distance to the nearest cluster, it assesses the separation and compactness between clusters.

The proposed MST-MVS clustering algorithm applies the SI analysis at each data chunk for assessing the quality of generated local clustering models. Only the clustering models that satisfy some preliminary defined threshold are used for building the integrated matrix of multi-view patterns (see Section 4).

3.3.2 Adjusted Rand Index

Adjusted Rand Index (ARI) is used for evaluating a clustering solution and is adjusted for chance [45]. It works by calculating the Rand Index (RI) [41], i.e., computing the similarity measure between two clustering solutions by counting pairs of samples assigned to the same or different clusters. Essentially, it calculates the number of agreeing pairs divided by the total. Then adjusting it for chance, as shown in Eq. V.7. A value of 1 indicates that the two clusterings are identical, while close to 0 shows random labeling independently of the number of clusters and samples.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n_{ij}}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n_{ij}}{2}}, \tag{V.7}$$

where $\sum_{ij}\binom{n_{ij}}{2}$ is the RI of samples i and j, a_i is the sum of the pair of samples w.r.t sample i, the same for b_j but for sample j. The minuend of the denominator is the expected RI, while the subtrahend is the maximum RI of samples i and j.

3.3.3 Adjusted Mutual Information

Adjusted Mutual Information (AMI) score is an adjustment accounting for chance of the Mutual Information (MI) [43]. It is a measure of similarity between two clustering solutions, and the adjustment for it is similar to that of ARI. Replacing RI of samples i and j by MI, and the maximum value of RI by the maximum value of the entropy of the predicted and true labels. The calculation of the expected value is pretty complex, and therefore, it is not provided herein. A detailed description can be found in [43]. AMI scores a clustering solution from 0 to 1, where 1 stands for a perfect match and 0 shows a random partition.

3.3.4 Homogeneity and Completeness

Homogeneity and *Completeness* are two other external cluster validation metrics, both components of the V-Measure [46]. The latter one is an entropy-based external cluster evaluation measure that introduces these two complementary criteria to capture desirable properties in clustering tasks. Homogeneity is satisfied by a clustering solution if all of its clusters consists of only data points which are members of a single class. Opposite, a clustering solution satisfies Completeness if all the data points that belong to a given class are assigned to the same cluster. Note that increasing the Homogeneity of a clustering solution often results in decreasing its Completeness. The calculations of these concepts are complex, and therefore, are not provided herein. A detailed explanation of how they work can be found in [46].

4 MST-MVS Clustering Algorithm

In this work, we study a streaming scenario, where a particular phenomenon (machine asset, patient, system etc.) is monitored under n different circumstances (views). Let us assume that the data arrives over time in chunks, i.e. a landmark window model according to the classification in [47]. In the landmark window model, the whole data between two landmarks form the current chunk are included in the processing. In our considerations each data chunk (window) t can contain different number of data points. Assume that chunk t contains N_t data points. In addition, in our multi-view context each chunk can be represented by a list of n different data matrices $D_t = \{D_{t1}, D_{t2}, \ldots, D_{tn}\}$, one per view. Each matrix D_{ti} $(i = 1, 2, \ldots, n)$ contains information about the data points in the current chunk t with respect to the corresponding view i. Further assume that $C_t = \{C_{t1}, C_{t2}, \ldots, C_{tn}\}$ is a set of clustering solutions (local models), such that C_{ti} $(i = 1, 2, \ldots, n)$ represents the grouping of the data points in tth chunk with respect to ith view, i.e. a local model built on data set D_{ti}.

The proposed MST-MVS algorithm's main idea is schematically illustrated in Figure 1. It has an initialization phase that is performed on the available historical data or on the first data chunk. Then on each subsequent data chunk the algorithm goes through a sequence of six different operational stages. At **Stage 1** the MST clustering algorithm is applied for parallel building of local clustering models on different data views. Then at the **Stage 2** the produced clustering solutions are evaluated and only ones satisfying a predefined evaluation criteria are considered in **Stage 3** for extracting individual view patterns. This makes our algorithm more robust to noisy and low-quality data, since the low-evaluated views' clustering solutions do not contribute to the global clustering model. The extracted patterns are used at the next stage to build an integrated matrix of multi-view profiles (**Stage 4**). In the proposed algorithm knowledge between chunks is transferred in the form of artificial nodes used by the MST clustering algorithm. The artificial nodes are identified at **Stage 6** by analyzing the global model that is built at **Stage 5** by using MST clustering algorithm on the integrated matrix. The six stages are described in more details hereafter and also illustrated by an example in Figure 2.

The MST-MVS clustering algorithm goes through the following six operation stages (see Figure 2) for each data chunk t $(t > 1)$:

> **Input:** Newly arrived data D_t, set of artificial nodes A_{t-1} identified at chunk $t-1$, clustering quality threshold Θ_t.

1. **Views' clustering:** Produce a clustering solution C_{ti} on each data matrix D_{ti} $(i = 1, 2, \ldots, n)$ of t^{th} chunk by applying MST clustering algorithm with a parameter $A_{(t-1)i}$, where $A_{(t-1)i}$ are artificial nodes' values corresponding to

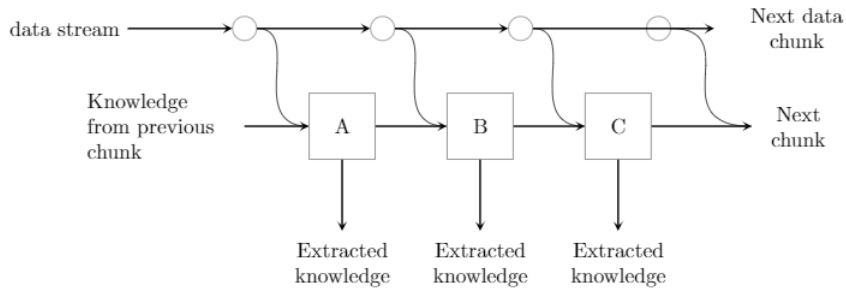


Figure 1: A high level overview of the proposed MST-MVS clustering algorithm for three data chunks A, B and C. The figure depicts how the knowledge extracted from the previous data chunk is used in analyzing a new data chunk.

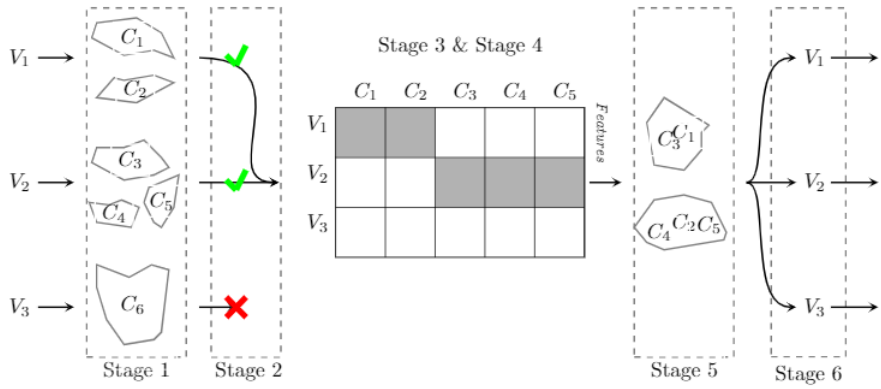


Figure 2: A schematic illustration of different stages of the proposed MST-MVS clustering algorithm. The example presents a three-view scenario, where V_1, V_2, V_3 represent the views, and C_1, C_2, C_3, C_4, C_5, C_6 represent different clusters across views. Local clustering models in each view are built in Stage 1, followed by evaluating and selecting the local models in Stage 2, these are then used for pattern extraction and building integrated matrix in Stages 3 & 4. A global model is built in Stage 5 followed by identifying artificial nodes in Stage 6.

view i and chunk $t-1$.

2. **Cluster models' evaluation:** At this stage the quality of each produced clustering solution C_{ti} $(i = 1, 2, \ldots, n)$ is evaluated by applying SI analysis, i.e. $s(C_{ti})$ is calculated for each view $i = 1, 2, \ldots, n$. A set of selected clustering solutions $C_t^{'} = \{C_{ti} \mid s(C_{ti}) > \Theta_t\}$ $(C_t^{'} \subseteq C_t)$ satisfying the given threshold Θ_t is then built. Note that other internal cluster validation measures or an ensemble of few measures can be used in this stage.

3. **Pattern extraction from the individual views:** Consider only local clustering models which satisfy the given threshold Θ_t, i.e. the set $C_t^{'}$ built at the

previous stage. The medoids of each $C_{ti} \in C_t^{'}$ are identified and extracted.

4. **Integrate the individual views' patterns:** The patterns extracted from the local models of $C_t^{'}$ are used at this stage to build an integrated matrix, denoted by M_t, that contains the multi-view profiles of the identified medoids. A detail explanation of the integration stage can be found in Section 4.1.

5. **Build a global model:** In order to build a global model the multi-view data points of M_t are clustered by applying MST clustering algorithm. The built global model, denoted by C_t^M, consists of a number of clusters of similar multi-view data points. Each cluster, as seen in Figure 2, may be interpreted as a multi-view pattern that presents a relationship among individual views' patterns (further information about multi-view pattern extraction can be found in Section 4.2). This information, as it will be discussed further in this study, can be used for post-labelling of D_t (see Section 4.5).

6. **Identify artificial nodes:** The integrated matrix M_t and the built global model C_t^M are used to identify the set of artificial nodes A_t that will be used to seed the MST clustering algorithm at chunk $t + 1$. Two different techniques for finding the artificial nodes are developed and studied. Their descriptions and pseudo-codes can be found in Section 4.3.

Output: The set of multi-view patterns C_t^M, and the set of artificial nodes A_t.

The following sub-sections describe some of the complex steps of the algorithm, followed by an estimation of the computational complexity of the algorithm in Section 4.6. Section 4.1 explains the multi-view integration procedure conducted in stage 4 of the MST-MVS algorithm. Sections 4.2 and 4.3 supply with additional details about stages 5 and 6, respectively. Sections 4.4 and 4.5 are devoted to the two labelling (CNMF-based and Pattern-based) algorithms studied in this work and used in the evaluation of the algorithm performance.

4.1 Multi-view data integration

The integrated matrix M_t of chunk t ($t = 1, 2, \ldots$) is built using the medoids of the approved views' clustering solutions, i.e. ones in set $C_t^{'}$ built at stage 3 of the MST-MVS algorithm. A medoid is seen as a representative of a cluster and each clustering solution is summarized by its medoids. In that way, the proposed algorithm can be considered as a communication-efficient and privacy-preserving solution, since we

do not need to transfer complete data from each view in order to build the global model C_t^M. The medoids of each clustering solution in $C_t^{'}$ are only transferred, i.e., privacy is preserved. The proposed algorithm can also be interpreted as a distributed clustering, since it clusters and evaluates multiple data views (sources) in parallel.

The integrated matrix M_t ($t = 1, 2, \ldots$) is built using the extracted data points from the views, where each column in the matrix is a medoid of a clustering solution in $C_t^{'}$. As one can see in Figure 2, each medoid takes up one column, and each row presents one feature vector. For example, if C_1 in Figure 3 is a medoid extracted from view 1 (the gray area in Figure 3), the rest of the column (view 2 and view 3) contains the remaining features of this data point, namely ones extracted from the other two views. In general, if a medoid is extracted from one view, the data points corresponding to that medoid in the other views are also used to build the multi-view data point, i.e. to fill in the respective column in M_t. Evidently, all features of a medoid (in this case C_1) are placed in the same column. As we can see in Figure 2 the third view (V_3) does not supply any medoids, since the view has not passed the evaluation criteria in stage 2.

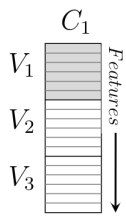


Figure 3: Example of one column from the integrated matrix, where a multi-view data point (Column C_1 from Figure 2) is built up by using the features' values from all the views that correspond to that data point.

4.2 Extraction of multi-view patterns

Remember that the matrix M_t of chunk t ($t = 1, 2, \ldots$) integrates the medoids of approved views' clustering solutions that are in $C_t^{'}$. This matrix is used in stage 5 of the MST-MVS clustering algorithm to build the global model C_t^M by clustering M_t with MST clustering algorithm. The built global model consists of clusters of similar multi-view data points. Note that each data point in M_t is a medoid in a view's clustering solution belonging to $C_t^{'}$, i.e. it is a pattern extracted from the respective view. Therefore, the clusters of the global model C_t^M present multi-view patterns. The latter ones may be interpreted as most typical relationships among the views supported by the data of the current chunk.

As it is mentioned above, MST clustering algorithm is used for clustering the integrated matrix M_t. However, one problem arises from this approach, namely, what value should α take? Note that the integrated matrix is usually very small, built just from the medoids of the approved views' clustering solutions. This implies that

each cluster in the global model would often have only one medoid in it, representing a cluster from one view. As it was stated in Section 3.1, mean α values can be chosen. However, this choice depends more on the medoids themselves than the results from each view. Mean α is the average value α can take, producing often wrong numbers of clusters when there is an unbalance of the weights between nodes, focusing on the overall average.

In this work, we propose an alternative of the α value solution, entitled *Longest Edges* (LEdges). LEdges makes use of how the MST algorithm works. Instead of calculating α for an artificial node's edges, it builds a MST on the data points of the integrated matrix M_t and generates the global clustering model C_t^M by cutting the longest edges of the built tree. A detail explanation of LEdges is provided in the forthcoming section.

4.3 Transfer of knowledge through artificial nodes

At the initialization phase, first (or historical) data chunk of each view is clustered by applying the MST clustering algorithm with α as an input parameter. At each subsequent chunk t ($t > 1$) the integrated matrix M_t and the built global model C_t^M are used to identify data points that are transferred to data chunk $t + 1$. These are used as artificial nodes by the MST clustering algorithm that is applied to build the local clustering models of chunk $t + 1$. This knowledge transferred in the form of artificial nodes influences the $t+1$ chunk's local clustering models. Particularly, the transferred artificial nodes seed and guide the clustering of the $t + 1$ chunk to be close/similar to the previous chunk's results, but still the clustering solutions can be modified since they are influenced by the data points in the $t + 1$ data chunk. In that way, the knowledge learnt through the processing of current data chunk (t) is used to support the clustering of the next chunk data ($t + 1$).

In the process of algorithm design, two different techniques for finding the artificial nodes are developed and studied: Boundary Nodes (BNodes) and LEdges. The main difference between BNodes and LEdges is the strategy that is applied for calculating the artificial nodes.

BNodes uses the centroids of global clustering model (C_t^M) built at stage 5 of the MST-MVS algorithm to calculate the set of artificial nodes (A_t). An artificial node is placed between two cluster centroids, i.e. it can be interpreted as a boundary node between two clusters. The pseudo-code of BNodes algorithm is shown in Algorithm V.2. Note that this strategy of finding artificial nodes does not guarantee the optimal solution for all possible scenarios. For example, it can happen three clusters centroids to be in a straight line. Then one of the resulting artificial nodes will be placed on the middle centroid, splitting the central cluster in two. In practice, the possibility of this scenario to happen is very low.

As the name suggests, LEdges uses the longest edges of the MST generated from

Algorithm V.2 Boundary Nodes

1: **procedure** BNODES(C_t^M)
2: $A_t := \varnothing$.
3: $Pairs\_of\_centroids :=$ All pairs of centroids of C_t^M
4: **for** $(u, v) \in Pairs\_of\_centroids$ **do**
5: $A_t \leftarrow$ Average(u, v)
6: **end for**
7: **return** A_t
8: **end procedure**

the integrated matrix (M_t) that is created at stage 4 of the MST-MVS algorithm. The artificial nodes are the middle points of the longest edges. However, this technique requires to know in advance how many longest edges to use. Each data view is already clustered and this information can be used to determine the number of artificial nodes, denoted by k_A. For example, it can be equal to the average number of clusters in the selected views, i.e. $k_A = \sum_{C_{ti} \in C_t'} \mid C_{ti} \mid / \mid C_t' \mid$, where $\mid . \mid$ represents the set cardinality. The LEdges algorithm pseudo-code is given in Algorithm V.3.

As one can see in Algorithm V.2, BNodes returns the artificial nodes by using the centroids of the global model C_t^M. The latter is built by clustering the integrated matrix M_t. In comparison LEdges, as a variation of MST clustering algorithm, finds not only the artificial nodes, but also clusters the integrated matrix M_t. Hence stages 5 and 6 are merged in the configuration of the MST-MVS algorithm using LEdges technique.

The artificial nodes returned by LEdges and BNodes are split into components corresponding to different views, e.g., the features belonging to view 1 are extracted from the artificial nodes and used in the clustering of view 1's data points in the next chunk. This allows MST clustering algorithm to be seeded with artificial nodes in each view.

4.4 CNMF-based labelling algorithm

In this work, we propose and study two different techniques for post-labelling of the data points in chunk D_t ($t = 1, 2, \ldots$). The first technique uses CNMF, with fixed X and F matrices (see Eq. V.3). CNMF is chosen over the traditional NMF as it allows negative values [31]. Let F consists of the centroids of the global model C_t^M built at stage 5 and $X \equiv D_t$ ($t = 1, 2, \ldots$). Then by applying Eq. V.4, the rows with the maximum values of a column in H gives the label of each data point (columns in H).

Note that the CNMF method is slow and is not very accurate. CNMF initiates two matrices L and H with random values, resulting in different approximations for each execution. To rectify this, the CNMF algorithm is executed multiple times, and the

Algorithm V.3 Longest Edges

procedure LEDGES(M_t, k_A)
  $A_t := \varnothing$
  $removed\_edges := \varnothing$
  $G(V, E) :=$ complete graph of M_t, i.e. $V \equiv M_t$
  $MST :=$ Kruskal(G)
  sorted(E) := sort edges of G in decreasing order
  for ($e \in$ sorted(E)) $\wedge$ ($\mid removed\_edges \mid \leq k_A$) **do**
    remove e from MST
    $removed\_edges \leftarrow e$
  end for
  for $e \in removed\_edges$ **do**
    $A_t \leftarrow$ Average(u, v)
  end for
  Build global clustering model C_t^M from MST
  return (A_t, C_t^M)
end procedure

instance which gives the lowest Frobenius norm $|X \approx FH_+^T|$ is used. However, this increases the time needed for labelling of the data points. Therefore, we propose in Section 4.5 a new algorithm for labelling the data chunk D_t, entitled Pattern-labelling. This labelling technique uses the cluster patterns identified by the global model C_t^M, and maps data points to the patterns they match.

4.5 Pattern-based labelling algorithm

In this study, we propose a new algorithm for post-labelling of the data points in the current data chunk, entitled Pattern-labelling. The proposed algorithm uses the extracted patterns at each data chunk, which are the ones determined by the clusters of the global model. These patterns are mapped to the chunk's data points to identify matches. Each pattern's index (cluster label) is used as the label for all data points that match it.

In order to execute the Pattern-labelling algorithm on data chunk D_t it is necessary to format the global clustering model C_t^M. Remember that each cluster in C_t^M can be considered as a multi-view pattern and all data points in dataset D_t that match to it have to get its cluster label. Therefore each multi-view pattern (cluster) of C_t^M has to be presented as a sequence of its views' clustering labels. The Pattern-labelling algorithm pseudo-code is shown in Algorithm V.4. Note that it is also necessary to translate each data point $D_t[i]$ ($i = 1, 2, \ldots, N_t$) into a list of its views' clustering labels. In addition, assume that each multi-view pattern is denoted by $C_t^M[j]$, where $j = 1, 2, \ldots, \mid C_t^M \mid$.

Algorithm V.4 Pattern-labelling algorithm

procedure PATTERN_LABELLING(C_t^M, D_t)
  $D_t^{labelled}$:= each data point in D_t is initialized to label -1
  for $i \in \{1, 2, \ldots, N_t\}$ **do**
    $D_t[i]$ is translated into a sequence of its views' clustering labels
    for $j \in \{1, 2, \ldots, |\ C_t^M\ |\}$ **do**
      $C_t^M[j]$ is presented by a sequence of its views' clustering labels
      if Match($D_t[i]$, $C_t^M[j]$) $\neq$ -1 **then** $D_t^{labelled}[i]$:= Match($D_t[i]$, $C_t^M[j]$)
      end if
    end for
  end for
  return $D_t^{labelled}$
end procedure

If a data point does not match any of the patterns, it is seen as an outlier in the data. The definition of an outlier in this study is a data point that is hard to be grouped in any of the clusters. One view might assign it to one global cluster, and another view assigns the data point to another global cluster. For this reason, the data point will not match any pattern and is considered as an outlier, hence it is assigned the label -1.

4.6 Computational Complexity

The computational complexity of the proposed MST-MVS clustering algorithm is not easy to estimate due to its complexity involving different stages, variety of views, and dimensionality of the data. Therefore, we propose a separate estimation of the computational complexity for each stage of the core algorithm.

1. **Views' clustering:** The clustering of the views have a computational time complexity of $O(nN_t^2 + nN_t + nN_t \log N_t)$, where N_t is the number of data points in data chunk t and n is the number of views. The first part, $O(nN_t^2)$, is the identification of the medoids for all views. The second part, $O(nN_t)$, presents the creation of the complete graphs for all views. The third part stands for the complexity of Kruskal's algorithm for all views, i.e. $O(nN_t logN_t)$. The computational complexity of this stage of the algorithm can be approximated to $O(N_t^2)$, since $n << N_t$ and $N_t < N_t \log N_t < N_t^2$.

2. **Cluster models' evaluation:** The SI has a quadratic computational time complexity, which means that stage 2 of the algorithm has a computational time complexity of $O(nN_t^2)$. It can be approximated to $O(N_t^2)$, since $n << N_t$.

3. **Build of the integrated matrix:** This includes stages 3 and 4 of the algorithm. The computational complexity of these stages depend on how the creation of the integrated matrix is implemented. For the implementation used in this study, the computational time complexity is $O(n'n\sum_{i=1}^{n}(n_i \mid C_{ti} \mid))$, where n' is the number of approved views, $\mid C_{ti} \mid$ is the number of clusters in the local model of view i, and n_i is the number of features characterizing view i.

4. **Clustering of the integrated matrix:** The clustering of matrix M_t has different time complexity depending on which of the two methods (BNodes and LEdges) is used.

   - If BNodes is used then the computational time complexity is

     $$O(k_t^{(k_t-2)} + \mid M_t \mid + \mid M_t \mid log \mid M_t \mid)),$$

     where k_t is the number of centroids (clusters) of the global model C_t^M, and $\mid M_t \mid$ is the number of multi-view data points (columns) of matrix M_t. The first part in the above expression ($k_t^{(k_t-2)}$) presents the time complexity of BNodes, $\mid M_t \mid$ is proportional to the time needed for the creation of the complete graph, and the third part ($\mid M_t \mid log \mid M_t \mid$) stands behind the calculations of MST.

   - If LEdges is used then the computational time complexity is

     $$O(\mid M_t \mid + (\mid M_t \mid log \mid M_t \mid) + avg\_k_t),$$

     where avg_k_t is the average number of clusters found in each local model for chunk t. The above expression can be simplified to $O(\mid M_t \mid log \mid M_t \mid)$.

5. **Knowledge transfer:** The computational time complexity of stage 6 is $O(n)$. Due to the linearity of stage 6, it will not be included in the overall complexity evaluation.

6. **Pattern-labelling:** The computational complexity of the Pattern-labelling algorithm is approximated to

   $$O(\sum_{i=1}^{n} \mid C_{ti} \mid + \sum_{i=1}^{k_t} k_t^{(i)} + N_t \sum_{i=1}^{k_t} k_t^{(i)}),$$

   where $\mid C_{ti} \mid$ is the number of clusters in the local model of view i, k_t is the number of clusters in the global model, and $k_t^{(i)}$ is the number of data points in global cluster i. The first two parts in the above expression assess the mapping of the extracted patterns to the real medoids in each view, while the third part presents the matching of patterns for all data points.

Based on the above analysis the overall computational complexity of MST-MVS algorithm in a configuration using LEdges and Pattern-labelling is approximately (assuming all views are approved in stage 3):

$$O(N_t^2 + n^2 \sum_{i=1}^{n} n_i \mid C_{ti} \mid + \lambda(2 + N_t + log\lambda)),$$

where n is the number of views, $\lambda = \sum_{i=1}^{n} \mid C_{ti} \mid$, and N_t is the number of data points in chunk t. The highest complexity is when all views are approved, resulting in $\sum_{i=1}^{n} \mid C_{ti} \mid = \sum_{i=1}^{k_t} k_t^{(i)} =\mid M_t \mid$.

5 Data and Experimental Settings

5.1 Data

We study and analyse the performance of the proposed MST-MVS clustering algorithm on synthetic and real-world data. We have used four different datasets: one synthetic and three real-world. These are listed in Table 1. A detailed description of the datasets used as well as their interpretation in a multi-view context are given below.

Table 1: Datasets used in the experiments.

Dataset	Type	#Samples	#Attributes	#Classes
Dim32	synthetic	1024	32	16
Cover-type	real-world	50000	14	7
One-year Sensor	real-world	8664	8	non-labelled
Two-year Sensor	real-world	17544	8	non-labelled

5.1.1 Dim32 dataset

A synthetic dataset, Dim32 [48], is built to be used for evaluation of clustering algorithms. Each cluster in Dim32 dataset is well separated, even in higher dimensions. Initially, this dataset is used in [49] with variations of the number of attributes. The Dim32 dataset used in our work has 1024 instances with 32 features, and is divided into 16 different Gaussian clusters. This dataset is chosen to provide a controlled and well-known environment for studying different configurations of the algorithm. The experiments conducted on this dataset are presented in Section 5.3.

Dim32 has been interpreted as a two-view dataset in our experiments. The first 16 attributes are assigned to the first view and the rest to the second view.

5.1.2 Forest Cover-type dataset

The second dataset used in our experiments is a subset of the Cover-type dataset [50], available at the UCI repository [51]. The main motivation for selecting this dataset

is the fact that it is well-known and used in many machine learning papers studying new clustering algorithms [25], [3], [52]. It has been created for classification tasks with 581012 instances and 54 attributes, divided into 7 clusters. Cover-type is an unbalanced dataset with 283301 instances for the biggest cluster and 2747 instances for the smallest. From the total 54 attributes, 44 are binary values, and 40 of these indicate soil types, the remaining 4 indicate wilderness areas. These 44 binary attributes make the data sparse, since each instance has only 2 values of these 44, rest of them are 0.

In accordance to the experiments conducted in [25], the 40 binary soil types in the Cover-type dataset are not considered in the experiments due to their sparsity. In addition, a sample set of 50000 instances of the Cover-Type dataset is used in the experiments and is divided into four views. The first three attributes, Elevation, Aspect, and Slope, are assigned to the first view. The next three with the ninth attribute, i.e. horizontal and vertical distance to nearest surface water features, horizontal distance to nearest roadways, and wildfire ignition points are assigned to the second view. The third view includes attributes six, seven and eight in the dataset, corresponding to the hill shade at 9 am, noon, and 3 pm, respectively. The last view has been assigned to the remaining four binary attributes indicating Rawah, Neota, Comanche Peak, and Cache la Poudre wilderness areas.

5.1.3 Real-world sensor datasets

The potential of the proposed algorithm is also demonstrated on real-world data from a company in the smart building domain. We have used two datasets: one covering a year (Jan 1^{st} 2019 till Dec 27^{th} 2019) and the other one containing measurements from two years (Jan 1^{st} 2019 till Dec 31^{st} 2020). The one-year dataset has been used in [53] for analysing and monitoring the control valve system behaviour. It has also been used in the evaluation of the MV Multi-Instance Clustering algorithm in [1].

In the smart building domain different types of metrics are collected from a wide range of sensors available for systems such as heating, ventilation, air conditioning, and refrigeration. The eight features listed in Table 2, used in [1] and seven of which also considered in [53], are used in our experiments.

Table 2: Features included in the real-world sensor dataset

View	Id	Acronyms	Feature name	Units
Operation	1	SST	Secondary Supply Temperature	°C
	2	SRT	Secondary Return Temperature	°C
	3	PHL	Primary Heat Load	kW
Performance	4	VOM	Valve Openness Mean	%
	5	VOS	Valve Openness Standard Deviation	%
	6	SE	Sub-station Efficiency	%
Context	7	OTM	Outdoor Temperature Mean	°C
	8	OTS	Outdoor Temperature Standard Deviation	°C

The available data features are analysed and partitioned in three distinctive views: system operational behaviour parameters, performance indicators and contextual fac-

tors. The features SST, SRT, and PHL are selected to model the system typical operational behaviour. The system performance can be evaluated by these three indicators: VOM, VOS, and SE. Finally, the contextual factors are represented by the features: OTM and OTS. For each view, after removing the outliers using Hampel filter, averaged daily values of the corresponding features are calculated to build daily profiles.

5.2 Data Preparation

In this study, to identify and remove the outliers in each feature of the two Sensor datasets, Hampel filter [54], a method based on median absolute deviation (MAD) estimation is applied. In addition, each feature of the four used datasets are standardized using the z-score. Namely, each feature is subtracted by their mean value ($\bar{x}$) and divided by the standard deviation (σ), i.e.

$$z = \frac{x - \bar{x}}{\sigma}.$$

Ten different experimental datasets are generated from each of Cover-Type and Dim32 datasets and used in the conducted experiments. The obtained results reported and analysed in Section 6, are averaged values of the ones produced on these experimental datasets. These datasets are generated by randomly shuffling rows in the data, thereby generating new versions of the datasets. This is done to mitigate the biases in the results due to the data and in that way to provide with more objective evaluation of algorithm performance.

5.3 Experiments and Validation

The conducted experiments are categorised into three groups: (i) algorithm configuration, (ii) tuning of algorithm parameters, and (iii) evaluation of algorithm performance. In the first group, we have performed a series of experiments on Dim32 dataset to study algorithm properties and find its optimal configuration. This configuration, called standard configuration hereafter, has been used in our experiments evaluating algorithm performance on real-world datasets, i.e. in the third group of experiments. We have also conducted a number of experiments to fine tune the algorithm's parameters for each of the used dataset.

5.3.1 Algorithm configuration

As described in Section 4.3, two different algorithms for calculating artificial nodes have been designed: BNodes and LEdges (see Algorithms V.2 and V.3, respectively). Two configurations of MST-MVS clustering algorithm, respectively using BNodes and LEdges to identify artificial nodes, are studied and compared on the Dim32 experimental datasets.

In this work, we have also proposed two different labelling techniques: CNMF-based labelling and Pattern-labelling (see Sections 4.4 and 4.5, respectively). Those are also studied and validated on the experimental datasets of Dim32. Since Dim32 dataset is labelled, the results generated in both experiments are evaluated by the four external cluster validation measures discussed in Section 3.3.

5.3.2 Tuning of algorithm parameters

We have conducted an experiment to find the optimal value of the threshold Θ_t, an algorithm parameter for each dataset used. We have explored each dataset by applying the MST-MVS algorithm on the dataset and plotting together the SI scores of the views' clustering solutions produced in each data chunk. The threshold is set as a trade-off value among the plotted scores.

The data chunk size is usually problem specific. However, Dim32 and Cover-type datasets are not considered in a concrete applied context. Therefore, we have performed an experiment to study and tune the chunk size of Cover-type dataset. The experiment executes the algorithm three times, each time using different chunk sizes. The data chunk size is chosen based on evaluating the results generated by the algorithm using the four external cluster validation measures described in Section 3.3.

In case of Dim32 dataset we have determined its data chunk size through reasoning, since it is a comparatively small dataset and an experiment is not needed. The Dim32 dataset is divided into two chunks containing 614 and 410 instances, respectively. As one can observe the first chunk is larger than the second. This is motivated by the algorithm working mechanism. Namely, the algorithm is initialized by mean α and transfers over knowledge from the current data chunk to the next. The quality of the initial clustering will have a significant impact on the upcoming data chunks. Therefore, the first chunk of Dim32 dataset is bigger, approximately 60% of the whole dataset, while the second have the remaining 40%.

Note that an experiment is not conducted for the Sensor datasets to determine the chunks' sizes. In case of one-year Sensor dataset we have simulated the experimental scenario used in the evaluation of MV Multi-Instance Clustering [1] to be able to compare the results. Thus the one-year Sensor dataset is divided into two chunks, 243 and 118, respectively. The first chunk contains the measures gathered during the months January to August, while the second chunk covers the period from September to December.

The two-year Sensor dataset is divided into two chunks, each one containing measurements from a year period. Our motivation behind this is to be able to compare the patterns extracted from two data chunks which are expected to capture similar seasonal behaviour modes. Note that there are missing values for a few days in the second year data. These have been removed from the dataset. Due to this the chunk sizes result in 363 and 334, respectively.

5.3.3 Evaluation of algorithm performance

The standard configuration of the proposed MST-MVS algorithm has been studied and evaluated in a real-world context by using Cover-type and Sensor data. The standard configuration of the proposed algorithm is executed on the experimental datasets of Cover-type data and the obtained results are evaluated with the four external cluster validation measures discussed in Section 3.3.

We have investigated how the knowledge transfer affects the performance of the proposed MST-MVS algorithm by comparing it with an algorithm version that does not use artificial nodes to seed the clustering at the upcoming data chunk. This experiment has been conducted on the experimental datasets of Cover-type data. The generated results are evaluated with the four external cluster validation measures. In addition, the number of clusters of the global model have been compared to the ground-truth number of clusters to facilitate the interpretation of the results.

Our MST-MVS algorithm evaluates the quality of the clustering solutions produced on different views' data at each chunk and select the best ones to build the global clustering model. In order to study how quality of the data chunks impact the global result we have conducted an experiment on Cover-type data in which we skip the evaluation phase and build the global model using all views' clustering solutions and then compare the results with the one produced by the original version of the algorithm.

The one-year Sensor dataset is used to study and benchmark the patterns extracted by the proposed algorithm to ones identified by MV Multi-Instance Clustering in [1]. We have further studied and analysed the potential of MST-MVS algorithm in the pattern mining task by applying it on the two-year Sensor dataset.

In order to study how the algorithm's performance is affected by the quality of knowledge transferred we have applied the MST-MVS algorithm on the one-year Sensor dataset in a circular mode, i.e. the algorithm has been executed in 10 consecutive iterations by seeding the first data chunk with the artificial nodes extracted by the last chunk. The results are evaluated by calculating SI scores of the views' clustering models generated in each data chunk.

5.4 Implementation and Availability

The proposed MST-MVS clustering algorithm is implemented in Python [55]. The following libraries have been used:

- Scikit-Learn [56] for z-score, and external cluster validation measures: ARI, AMI, Completeness and Homogeneity.
- NetworkX [57] for Kruskal's algorithm, and creation of the complete graph.
- Pandas [58] for reading dataset from disk and manipulations of data.

- NumPy [59] for fast array manipulations, and operations.
- SciPy [60], utility functions for distance calculations, in this case, Euclidean.
- PyMF [61] for solving the CNMF algorithm, its implementation is based on [31].

Dim32 [48] and Cover-type [50] datasets are public. The Sensor datasets are provided by a company and are not publicly available. The algorithm code can be provided on request.

6 Results and Discussion

6.1 Algorithm configuration

The Pattern-labelling technique introduced in Section 4.5 is benchmarked to CNMF-labelling[1] (see Section 4.4) on Dim32 dataset. As one can observe in Table 3, the Pattern-labeling has produced better results than the CNMF-labelling. This is logical and can be predicted due to the randomness embedded into the CNMF algorithm. The latter algorithm randomly initiates the H matrix and is executed multiple times to generate a good approximation. In comparison to it the Pattern-labelling is consistent, since randomness is not used in its calculations, as this is described in Section 4.5. Therefore, Pattern-labelling is chosen to be used in our experiments studying the MST-MVS algorithm performance.

Table 3: The cluster validation measures' scores generated on Dim32 dataset by Pattern-labelling (PL) and CNMF-labelling (CNMF). MST-MVS algorithm is executed with a threshold $\Theta = 0.0$ and LEdges method. The dataset is divided into two chunks containing 614 and 410 instances, respectively.

Method	Minimum				Maximum				Mean			
	ARI	H	C	AMI	ARI	H	C	AMI	ARI	H	C	AMI
CNMF	0.82	0.91	0.98	0.94	0.84	0.92	0.98	0.94	0.83	0.92	0.98	0.94
PL	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Note. H and C stand for Homogenity and Completeness, respectively.

In addition, we have compared the two algorithms for calculating artificial nodes (see Algorithms V.2 and V.3) on Dim32 dataset. As it can be observed in Table 4 both methods (BNodes and LEdges) have similar performance w.r.t. the used metrics. Note that BNodes, as stated before, has a small shortcoming, namely it would split a cluster if the clusters are ordered in a straight line. On the other hand, LEdges is more efficient by combining stages 5 and 6 of the algorithm. Due to this and the close similarity between the results generated by the two methods LEdges is selected to be used in the experiments studying algorithm performance.

[1] Note that in our experiments we have used a CNMF-based, instead of NMF-based, version of the CNMF-labelling technique to allow for negative data.

Table 4: The cluster validation measures' scores generated on Dim32 dataset by LEdges and BNodes. MST-MVS algorithm is executed with a threshold $\Theta = 0.0$ and CNMF-labelling method. The dataset is divided into two chunks containing 614 and 410 instances, respectively.

Method	Minimum				Maximum				Mean			
	ARI	H	C	AMI	ARI	H	C	AMI	ARI	H	C	AMI
BNodes	**0.83**	0.91	**0.99**	0.94	0.84	0.92	**0.99**	**0.95**	**0.84**	0.92	**0.99**	**0.95**
LEdges	0.82	0.91	0.98	0.94	0.84	0.92	0.98	0.94	0.83	0.92	0.98	0.94

Note. H and C stand for Homogenity and Completeness, respectively.

6.2 Tuning of algorithm parameters

The only parameter used in the algorithm, threshold Θ_t, is defined based on data exploration. This is done by running the MST-MVS algorithm on each used dataset and plotting the SI values of the views' clustering models calculated in each data chunk. For example, in case of Dim32 dataset, the threshold is set to 0.5. It could theoretically be higher due to the high SI scores produced on the views' clustering solutions, this can be observed in Table 5.

Table 5: The average SI scores of the views' clustering models produced in the two chunks of Dim32 dataset by applying the MST-MVS algorithm configuration using LEdges and CNMF-labelling.

Chunk	Avg. SI	
	View 0	View 1
1	0.91	0.94
2	0.93	0.94

In case of Cover-type data we plot the SI values found for each view in Figure 4. As one can observe they are relatively low, and most of the time below 0. Therefore, -0.2 is used as the threshold in the experiments conducted on Cover-type dataset. It could be argued that a positive threshold should be used, as negative SI denotes a poor clustering solution. However, most of the clusters of this dataset are overlapping and this affects the quality of the views' clustering solutions. For example, only the first chunk has produced positive SI values for all views. Hence, the threshold -0.2 is set to be a close average of the three lowest scoring views (views 0, 1, 2).

Table 6: The SI scores of the views' clustering models produced in the two chunks of one-year Sensor dataset by applying the MST-MVS algorithm configuration using LEdges and Pattern-labelling.

Chunk	Avg. SI		
	View 0	View 1	View 3
1	-1.00	0.39	-1.00
2	-0.10	0.63	-0.11

Similar to Cover-type data, the threshold on the Sensor dataset is set below 0 to include more than one view in the calculation of the global model. Specifically -0.4, a close average of the lowest and highest SI values with an added margin for error, as it can be observed in Table 6. Our motivation for this is the high difference between the values generated on view 2 and the other two views' values.

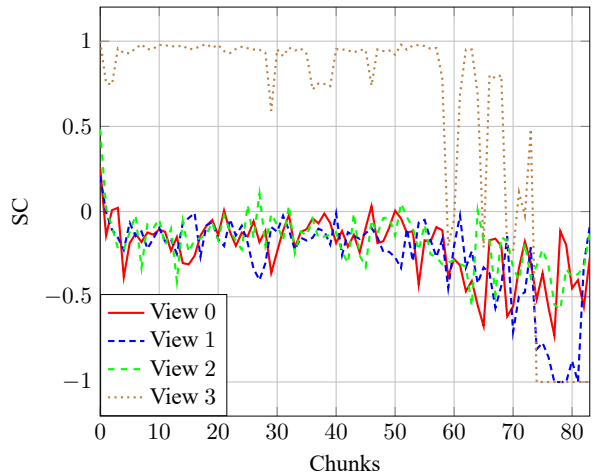


Figure 4: The SI scores of the views' clustering models produced on Cover-Type dataset by applying the MST-MVS algorithm configuration using LEdges and Pattern-labelling.

In order to determine the chunk size for the Cover-type dataset we have conducted experiments by studying different chunk sizes. Table 7 presents the produced cluster validation measures' scores for three different chunk sizes. As one can see they are very close. Therefore, we have selected 600 to be the chunk size, since the smallest size results in experimenting with more chunks. The latter provides more results for discussion and facilitates to get better insight into the working mechanism of the algorithm.

Table 7: Average cluster validation measures' scores generated on the Cover-type experimental datasets using different chunk sizes.

Chunk size	ARI	H	C	AMI
600	0.27	0.21	0.46	0.27
1000	0.25	0.20	0.49	0.26
2000	0.28	0.22	0.50	0.29

Note. H and C stand for Homogeneity and Completeness, respectively.

6.3 Evaluation of algorithm performance

Based on the experiments discussed in Sections 6.1 and 6.2 a standard configuration of MST-MVS algorithm has been determined for each dataset. These standard configurations are used in the experiments discussed in this section. The standard configurations of the used datasets are shown in Table 8.

Note that in case of Cover-Type, the SI analysis for some chunks fails for all views, resulting in a skipped chunk. When plotting the results, the skipped chunk is not shown. The results from the last processed chunk are used instead. This can be observed in certain places in the graphs depicting the metrics' scores, e.g., see last

Table 8: MST-MVS algorithm standard configurations for the used datasets.

Dataset	Global model method	Labeling Method	Threshold	Chunk size(s)
Dim32	LEdges	Pattern-labelling	0.5	614(410)
Cover-Type	LEdges	Pattern-labelling	- 0.2	600
One-year Sensor	LEdges	Pattern-labelling	-0.4	243(118)
Two-year Sensor	LEdges	Pattern-labelling	-0.4	363(334)

chunks in Figure 5, where the ARI scores plotted for MST-MVS algorithm (solid black line) between chunk 70 and 82 are constants, due to skipped chunks.

6.3.1 Cover-Type dataset

The performance of the MST-MVS algorithm's standard configuration is evaluated on the Cover-Type dataset. The obtained results are reported in Table 9. It can be observed that the transfer of knowledge does have a positive impact on the resulting clustering solution's properties evaluated by the four external cluster validation measures. Note that the maximal values generated by the two versions of the algorithm are very similar. However, the version implementing the knowledge transfer generates higher minimal and average values then the other version.

Table 9: The cluster validation measures' scores generated on Cover-type dataset by the MST-MVS algorithm standard configuration with and without knowledge transfer between two consecutive data chunks.

	Minimum				Maximum				Average			
Transfer	ARI	H	C	AMI	ARI	H	C	AMI	ARI	H	C	AMI
Yes	**0.10**	**0.12**	**0.11**	**0.12**	0.50	**0.48**	0.87	0.61	**0.28**	**0.23**	**0.50**	**0.29**
No	-0.02	0.00	0.03	0.00	0.50	0.46	**0.91**	0.61	0.17	0.14	0.45	0.19

Note. H and C stand for Homogenity and Completeness, respectively.

It should also be noted that the evaluation of the views' clustering solutions does have a positive impact on the global models' clustering solutions, as supported by the results presented in Table 10. The difference between the obtained results, with and without evaluations of local clustering solutions is not significant, but in all conducted experiments except one (the minimum value of Completeness) the scores generated in the former scenario, i.e. when the local clustering solutions are evaluated, are higher.

Table 10: The cluster validation measures' scores generated on Cover-type dataset by the MST-MVS algorithm standard configuration with and without evaluation of the views clustering solutions (Stage 2).

	Minimum				Maximum				Average			
Evaluation	ARI	H	C	AMI	ARI	H	C	AMI	ARI	H	C	AMI
Yes	**0.10**	**0.12**	0.11	**0.12**	**0.50**	**0.48**	**0.87**	**0.61**	**0.28**	**0.23**	**0.50**	**0.29**
No	-0.00	0.00	**0.19**	-0.00	0.38	0.37	0.81	0.48	0.24	0.19	0.48	0.25

Note. H and C stand for Homogenity and Completeness, respectively.

In order to get further insight into the results reported in Table 9, we have depicted the calculated ARI scores in Figure 5. This plot indicates that the instability in the

results is around and after the 58^{th} chunk, i.e., something is different in the data after that chunk. Therefore, to better understand the data structure the number of clusters of the clustering solutions produced by the two versions are further benchmarked to that of the ground-truth clustering of the Cover-type dataset in Figure 6.

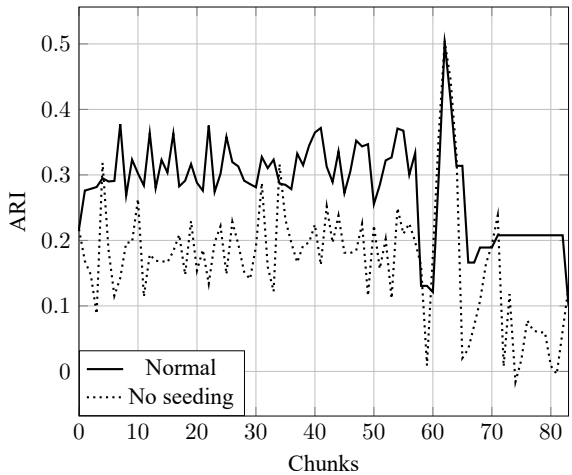


Figure 5: ARI scores generated on Cover-type dataset by the MST-MVS algorithm standard configuration with and without knowledge transfer between two consecutive data chunks.

As one can observe in Figure 6 the number of clusters fluctuates after the 58^{th} chunk, resulting in the instability in the algorithm's results after it. If the results after the 58^{th} chunk are excluded, then as can be seen in Figure 5, the knowledge transfer between chunks do have a positive impact on the results. This in turn indicates that the artificial nodes from the previous chunk really guide the clustering of the next chunk.This is additionally demonstrated in Figure 6, where the two versions show different behaviour signatures w.r.t. the number of clusters. Despite the fact that the number of clusters produced by the version without implementing knowledge transfer is closer to the ground-truth number of clusters, its behaviour is more fluctuating in comparison to that of the other version.

6.3.2 Sensor datasets

Initially, the MST-MVS clustering algorithm has been applied on the one-year Sensor dataset by simulating the same experimental scenario as the one used in [1] to evaluate MV Multi-Instance Clustering. Namely, as described in Section 5.3, the created daily profiles (361 in total) are split into two parts in order to simulate two data chunks: the first one with 243 daily profiles (January - August) and the second chunk with 118 daily profiles (September - December).

Tables 11 and 12 present the clustering solutions generated by the MST-MVS algorithm on the first and second data chunks of the one-year Sensor dataset, respectively. These are benchmarked to 13 cluster concepts identified by MV Multi-

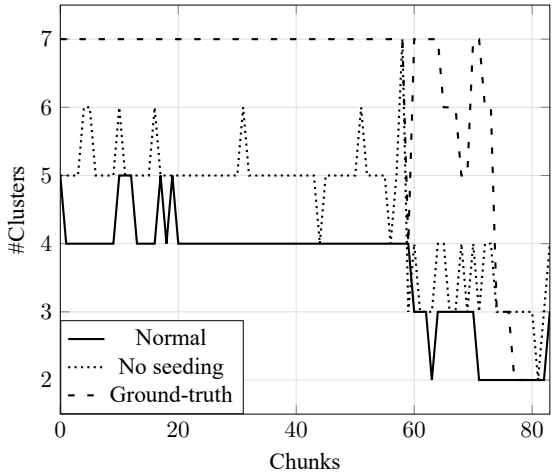


Figure 6: Comparison of the number of clusters produced on Cover-type dataset by the MST-MVS algorithm standard configuration with and without knowledge transfer between two consecutive data chunks with the ground-true number of clusters.

Instance Clustering algorithm in [1]. The latter algorithm has extracted 8 concepts that links three views from the first chunk and 5 additional concepts from the second chunk. The last column *Concept*, in both Table 11 and Table 12 present the concepts reported in [1] that based on our analysis match the clusters identified by applying the MST-MVS clustering algorithm.

Table 11: Summary of the identified clusters in the first chunk (January - August) of the one-year Sensor dataset.

Cluster	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Months	Size	Concept
0	15.15	36.33	31.95	9.83	±0.65	81	10.71	±0.43	1 - 8	240	Unite concepts that share common trends for both heating and non-heating season
1	36.92	52.13	39.99	19.32	±3.18	96	-2.89	±0.36	1, 3	2	Concepts 12 & 13
2	11.42	34.33	31.43	11.77	±7.28	84	11.29	±0.44	5	1	Deviant behaviour
									Total	243	

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 2.

It can be observed that two concepts (concepts 12 and 13) extracted by MV Multi-Instance Clustering are very similar to cluster 1 in the first data chunk (see Table 11). Concepts 12 and 13 present the system behaviour typical for the heating season when the average outdoor temperature is close to zero. Cluster 1 differs from these concepts on having a negative mean value for the outdoor temperature (OTM), i.e. it groups together the profiles of the only two days in January and March when the temperature was below zero. Cluster 0 unites many of the concepts produced by the MV Multi-Instance Clustering in [1]. Evidently, it represents trends common for both heating and non-heating season. This result is mainly due to missing of previ-

ous knowledge. This is also confirmed by the improved results obtained when the algorithm is executed iteratively (see Figure 7, and Tables 13 and 14). Cluster 2 is a singleton and can be considered as a pattern hinting deviating behaviour (from the one presented by cluster 0). It indicates a large deviation in the opening and closing of the valve in May. Therefore, it is worth to be further analysed and discussed with domain experts. In general, as it will be seen further in this section, clustering solutions produced by the proposed algorithm are more compact than the ones generated by MV Multi-Instance Clustering. This is due to the different working mechanisms of the two algorithms. Namely, the MV Multi-Instance Clustering algorithm integrates the clustering solutions that have been produced independently on two consecutive data chunks. Then it applies FCA analysis, and closed patterns to extract the integrated clustering result. This certainly implies the generation of clustering solutions that have more smaller-size clusters. In comparison with MV Multi-Instance Clustering, the proposed algorithm clusters each data chunk separately by applying a MST clustering algorithm seeded with artificial nodes from the previous chunk.

Table 12: Summary of the identified clusters in the second chunk (September - December) of the one-year Sensor dataset.

Cluster	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Months	Size	Concept
0	14.27	36.86	33.44	10.62	±0.59	87	9.79	±0.30	9 - 12	65	Concept 6
-1	20.90	42.61	37.12	13.32	±0.56	92	5.01	±0.40	9 - 12	26	Concepts 10 & 11
2	20.51	43.48	38.26	13.25	±0.45	94	5.00	±0.35	10 - 12	23	Concepts 10 & 11
1	25.49	42.76	36.35	14.36	±0.53	95	5.56	±0.46	10	3	Concept 10
3	23.22	44.87	38.76	13.32	±1.80	96	3.90	±0.32	12	1	Concept 11
									Total	118	

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 2.

As one can see in Table 12, all clusters are linked to concepts identified by MV Multi-Instance Clustering in the second chunk of one-year Sensor dataset. We can notice the same trend as the one discussed above. Namely, more compact solution than the one produced by MV Multi-Instance Clustering. The five clusters (0, -1, 1, 2 and 3) identified by the MST-MVS algorithm are linked only to three concepts. It should be noted that cluster -1 indicates "outliers", i.e. data points that do not match any of the patterns extracted by the MST-MVS algorithm from the second data chunk. In comparison with clusters 1 and 2, cluster -1 seems to be a combination of them both. This could therefore, be the reason why these data points are labelled as "outliers" by the algorithm. In addition, clusters 1 and 3 contain data points presenting behaviour deviating from the one modeled by cluster 2. This phenomenon is also noticed in the clustering solution summarized in Table 11 (see cluster 2) and clustering solutions generated on two-year Sensor data chunks (see Tables 15 and 16).

In order to study how the algorithm performance is affected by the quality of transfer knowledge we have applied the MST-MVS algorithm on the one-year Sensor dataset in a circular mode. This means that after the second chunk the algorithm is run

on the first chunk again, but now it is seen as the next data chunk. This is repeated 10 times to observe the changes in SI score over time, see Figure 7. It can be observed in the figure that the SI values stabilize after chunk 10 (i.e. after the 5^{th} iteration), which results in an average SI value of 0.25 for all views' clustering solutions. In addition to this, we notice a significant increase in SI score in comparison to the starting value. All these indicate that the quality of the clustering solution of the initial chunk has an impact on the successive chunks' results, i.e. the quality of knowledge transferred is important.

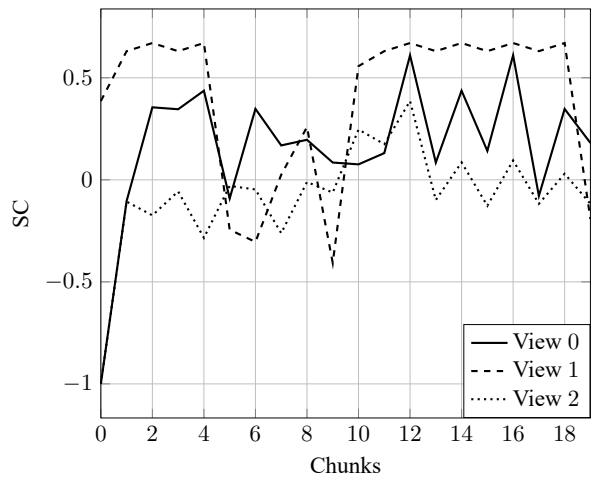


Figure 7: The SI scores of local clustering models generated by the MST-MVS algorithm standard configuration applied 10 times on the one-year Sensor dataset in a circular mode. The threshold value is set to -0.4.

Table 13: Summary of the identified clusters in the first chunk (10^{th} iterative chunk) after 5^{th} iteration of the algorithm on the one-year Sensor dataset.

Cluster	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Months	Size
0	39.75	47.37	38.17	17.57	±0.53	96	1.20	±0.35	1 - 3, 5	73
3	19.81	50.32	39.17	18.59	±4.18	98	-1.55	±0.41	3	1
-1	6.59	37.38	32.89	12.43	±0.73	89	9.34	±0.44	3 - 7	66
1	3.40	27.89	26.96	2.65	±0.72	67	18.31	±0.49	3 - 8	101
4	11.42	34.34	31.44	11.77	±7.28	85	11.30	±0.44	3	1
2	10.40	32.73	30.08	11.18	±1.09	81	13.00	±0.47	3	1
									Total	243

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 2.

We further study the clustering solutions produced on the first and second chunk after the 5^{th} iteration of our algorithm on one-year Sensor dataset. They are presented in Tables 13 and 14, respectively. In comparison with the results reported in Tables 11 and 12 these clustering solutions obtained better capture the valve system behaviour. We can observe that the two clustering solutions have a very similar structure. Namely, they both have two bigger clusters (clusters 0 and 1 in Table 13 and

Table 14: Summary of the identified clusters in the second chunk (11^{th} iterative chunk) after 5^{th} iteration of the algorithm on the one-year Sensor dataset.

Cluster	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Months	Size
0	5.47	28.38	27.24	5.22	±0.98	74	16.17	±0.36	9	17
-1	11.28	34.37	31.41	11.54	±0.65	85	11.76	±0.36	9 - 10	10
1	12.45	34.94	32.08	11.81	±0.47	88	11.10	±0.32	9 - 10	10
2	21.11	43.21	37.88	13.29	±0.47	95	4.94	±0.33	9 - 12	80
3	23.23	44.88	38.76	13.33	±1.80	97	3.90	±0.32	12	1
									Total	118

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 2.

clusters 2 and 0 in Table 14) that model the system behaviour typical for heating and non-heating seasons, respectively; a single cluster (cluster -1 in both clustering solutions) that presents behaviour fluctuating between heating and non-heating modes, i.e. the data points assigned to this cluster do not match any of the extracted patterns; and finally, a few smaller clusters that demonstrate behaviour deviating from the two typical modes. For example, clusters 3 and 4 in Table 13 are singletons that present two days in March with a large deviation in the valve openness (VOS) in heating and non-heating season, respectively. Interestingly, the only day of cluster 2 is not assigned to cluster 4, mainly due to the fact that its VOS behaviour is normal. Evidently, the generated clustering models provide a lot of useful information that can facilitate the domain experts in analyzing of the system behaviour.

Table 15: Summary of the identified clusters in the first chunk (first year) of the two-year Sensor dataset.

Cluster	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Months	Size
0	17.41	38.73	33.87	11.66	±0.71	87	8.36	±0.39	1 - 12	328
1	11.42	34.34	31.44	11.77	±7.28	85	11.30	±0.44	5	1
2	2.43	26.27	26.07	0.00	±0.00	58	21.50	±0.47	6 - 9	34
									Total	363

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 2.

Table 16: Summary of the identified clusters in the second chunk (second year) of the two-year Sensor dataset.

Cluster	PHL	SST	SRT	VOM	VOS	SE	OTM	OTS	Months	Size
1	11.67	38.63	33.80	12.46	±0.57	-28	6.89	±0.33	1 - 12	207
0	10.01	38.28	33.25	12.57	±0.51	-31	7.23	±0.42	2, 11	2
2	3.31	26.77	25.58	2.86	±0.44	13	18.20	±0.45	5 - 11	123
3	4.32	26.14	25.94	1.63	±1.02	76	16.77	±0.52	6	1
									Total	334

Note. The unit for PHL is kW and for SST, SRT, OTM, and OTS is °C. VOM, VOS, and SE are expressed in %. For the full form of each feature see Table 2.

Tables 15 and 16 show the clustering solutions generated by the MST-MVS algorithm on the two chunks of the two-year Sensor dataset, respectively. We can see that

there is a similarity between the two clustering solutions. In addition, the clustering produced on the first chunk (see Table 15) is similar to the one-year Sensor dataset's clustering results (see Tables 11 and 12). Namely, clusters 0 in the one-year Sensor dataset from both chunks have been combined to produce cluster 0 in the two-year dataset. Furthermore, cluster 1 from the two-year dataset is exactly the same as the deviant behaviour cluster from the one-year dataset (cluster 2 in Table 11). An interesting behaviour is presented by cluster 2 generated on the first chunk of the two-year Sensor dataset (see Table 15). We can see that all data points with VOM and VOS of 0.00 in a context of average outdoor temperature (OTM) above 20°C have been grouped together.

In Table 16, containing the clustering solution produced on the second chunk of two-year Sensor dataset, we can observe some interesting cluster patterns. First, clusters 0 and 1 in Table 16 have both negative values for the substation efficiency (SE). During a span in the second data chunk, that is from 23^{rd} September 2020 till 14^{th} December 2020 the system had faulty behaviour. The negative values for SE in these clusters are due to this. The primary supply temperature during these days is less than the primary return temperature which gave large negative SE values. Second, cluster 2 from the first chunk (see Table 15) seems to have been split into clusters 2 and 3 in the second chunk (see Table 16). Finally, clusters 0 and 1 contain the heating season patterns while clusters 2 and 3 present the two modes during the non-heating period. We can notice the clustering has the same structure as that of the clustering solutions presented in Tables 13 and 14. Namely, clusters 0 and 3 contain just two and one data points, respectively. These can be interpreted as ones presenting behaviours deviating from the typical for heating and non-heating season, respectively.

7 Conclusion and Future Work

In this work, we have proposed a novel multi-view graph-based clustering algorithm, entitled MST-MVS clustering, suitable for modelling multi-view streaming scenarios. We have developed different configurations of the MST-MVS clustering algorithm. They have been evaluated under different experimental scenarios on two types of data sets: synthetic and real-world. We have studied two different approaches for identifying artificial nodes that are used to seed the MST algorithm producing local clustering models at the upcoming data chunk. We have also investigated how the knowledge transfer affects the performance of the proposed MST-MVS algorithm. Finally, we have proposed two post-labelling techniques: Pattern-labelling and CNMF-labelling.

The MST-MVS clustering algorithm logically has shown a higher performance on the synthetic data than on the real-world data. This is due to the fact that synthetic data usually does not have the features which are characteristic for real-world data such as noise, missing values and overlapping clusters. The transfer of knowl-

edge feature has shown to have a positive effect on the performance of MST-MVS algorithm. The Pattern-labelling technique has been demonstrated to outperform the CNMF-labelling algorithm in the conducted experiments both in terms of computational time and labelling accuracy. Our future plans include further evaluation of the proposed MST-MVS clustering algorithm on richer real-world datasets in different applied scenarios.

Acknowledgements

We would like to thank *Farhad Basiri* for providing us the data from the smart building domain.

References

[1] V. M. Devagiri, V. Boeva, and S. Abghari. "A Multi-view Clustering Approach for Analysis of Streaming Data". In: *Artificial Intelligence Applications and Innovations*. Ed. by I. Maglogiannis, J. Macintyre, and L. Iliadis. Cham: Springer International Publishing, 2021, pp. 169–183. ISBN: 978-3-030-79150-6.

[2] V. M. Devagiri, V. Boeva, and E. Tsiporkova. "Split-Merge Evolutionary Clustering for Multi-View Streaming Data". In: *Procedia Computer Science* 176 (2020), pp. 460–469.

[3] L. Huang, C. .-. Wang, H. .-. Chao, and P. S. Yu. "MVStream: Multiview Data Stream Clustering". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3482–3496.

[4] C. Lindig. "Fast concept analysis". In: *Working with Conceptual Structures-Contributions to ICCS* (2000), pp. 152–161.

[5] M. Bendechache and M.-T. Kechadi. "Distributed clustering algorithm for spatial data mining". In: *2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM)*. IEEE. 2015, pp. 60–65.

[6] J. Liu, C. Wang, J. Gao, and J. Han. "Multi-view clustering via joint nonnegative matrix factorization". In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 252–260.

[7] X. Peng, Z. Huang, J. Lv, H. Zhu, and J. T. Zhou. "COMIC: Multi-view clustering without parameter selection". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5092–5101.

[8] Y. Yang and H. Wang. “Multi-view clustering: A survey”. In: *Big Data Mining and Analytics* 1.2 (2018), pp. 83–107.

[9] Z. Akata, C. Thurau, and C. Bauckhage. “Non-negative matrix factorization in multimodality data for segmentation and label prediction”. In: *16th Computer vision winter workshop*. 2011.

[10] W. Ou, F. Long, Y. Tan, S. Yu, and P. Wang. “Co-regularized multiview non-negative matrix factorization with correlation constraint for representation learning”. In: *Multimedia Tools and Applications* 77.10 (2018), pp. 12955–12978.

[11] J. Wang, F. Tian, H. Yu, C. H. Liu, K. Zhan, and X. Wang. “Diverse non-negative matrix factorization for multi-view data representation”. In: *IEEE transactions on cybernetics* 48.9 (2017), pp. 2620–2632.

[12] P. Jing, Y. Su, Z. Li, and L. Nie. “Learning robust affinity graph representation for multi-view clustering”. In: *Information Sciences* 544 (2021), pp. 155–167. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.06.068. URL: https://www.sciencedirect.com/science/article/pii/S0020025520306575.

[13] Q. Zheng, J. Zhu, Y. Ma, Z. Li, and Z. Tian. “Multi-view subspace clustering networks with local and global graph information”. In: *Neurocomputing* 449 (2021), pp. 15–23. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.03.115. URL: https://www.sciencedirect.com/science/article/pii/S0925231221005075.

[14] M. Ghesmoune, M. Lebbah, and H. Azzag. “State-of-the-art on clustering data streams”. In: *Big Data Analytics* 1.1 (2016), pp. 1–27.

[15] R. M. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler. “StreamKM++: A clustering algorithm for data streams”. In: *ACM Journal of Experimental Algorithmics* 17.1 (2012), pp. 173–187.

[16] C. Aggarwal, J. Han, J. Wang, and P. Yu. “A framework for clustering evolving data streams”. In: *VLDB. Berlin: VLDB Endowment*. Vol. 7. 2003, pp. 81–92.

[17] B. Cao, D. Shen, J.-T. Sun, X. Wang, Q. Yang, and Z. Chen. “Detect and Track Latent Factors with Online Nonnegative Matrix Factorization.” In: *IJCAI*. Vol. 7. 2007, pp. 2689–2694.

[18] P. Kranen, I. Assent, C. Baldauf, and et al. “The ClusTree: indexing micro-clusters for anytime stream mining”. In: *Knowledge Information Systems* 29 (2011), pp. 249–272.

[19] C. Ding, X. He, and H. D. Simon. “On the equivalence of nonnegative matrix factorization and spectral clustering”. In: *Proceedings of the 2005 SIAM international conference on data mining*. SIAM. 2005, pp. 606–610.

[20] C.-D. Wang, J.-H. Lai, D. Huang, and W.-S. Zheng. “SVStream: A support vector-based algorithm for clustering data streams”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.6 (2011), pp. 1410–1424.

[21] W. Shao, L. He, C. Lu, and P. S. Yu. “Online multi-view clustering with incomplete views”. In: *2016 IEEE International Conference on Big Data (Big Data)*. 2016, pp. 1012–1017.

[22] D. D. Lee and H. S. Seung. “Learning the parts of objects by non-negative matrix factorization”. In: *Nature* 401.6755 (1999), pp. 788–791.

[23] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. “Support vector clustering”. In: *Journal of machine learning research* 2.Dec (2001), pp. 125–137.

[24] C.-D. Wang and J. Lai. “Position regularized support vector domain description”. In: *Pattern Recognition* 46.3 (2013), pp. 875–884.

[25] V. Boeva, M. Angelova, V. M. Devagiri, and E. Tsiporkova. “Bipartite Split-Merge Evolutionary Clustering”. In: *Agents and Artificial Intelligence*. Ed. by J. van den Herik, A. P. Rocha, and L. Steels. Cham: Springer International Publishing, 2019, pp. 204–223.

[26] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. “Graph clustering and minimum cut trees”. In: *Internet Mathematics* 1.4 (2004), pp. 385–408.

[27] R. Görke, T. Hartmann, and D. Wagner. “Dynamic Graph Clustering Using Minimum-Cut Trees”. In: *Algorithms and Data Structures*. Ed. by F. Dehne, M. Gavrilova, J.-R. Sack, and C. D. Tóth. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 339–350.

[28] B. Saha and P. Mitra. “Dynamic Algorithm for Graph Clustering Using Minimum Cut Tree”. In: *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW’06)*. 2006, pp. 667–671.

[29] X. Lv, Y. Ma, X. He, H. Huang, and J. Yang. “CciMST: A clustering algorithm based on minimum spanning tree and cluster centers”. In: *Mathematical Problems in Engineering* (2018).

[30] J. B. Kruskal. “On the shortest spanning subtree of a graph and the traveling salesman problem”. In: *Proceedings of the American Mathematical society* 7.1 (1956), pp. 48–50.

[31] C. H. Ding, T. Li, and M. I. Jordan. “Convex and semi-nonnegative matrix factorizations”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.1 (2008), pp. 45–55.

[32] R. Hamon, V. Emiya, and C. Févotte. “Convex nonnegative matrix factorization with missing data”. In: *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2016, pp. 1–6.

[33] P. Paatero and U. Tapper. “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values”. In: *Environmetrics* 5.2 (1994), pp. 111–126.

[34] S. Craw. “Manhattan Distance”. In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by C. Sammut and G. I. Webb. Boston, MA: Springer US, 2017, pp. 790–791.

[35] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. “On clustering validation techniques”. In: *J Intell Inf Syst* 17.2-3 (2001), pp. 107–145.

[36] L. Vendramin, R. Campello, and E. Hruschka. “Relative clustering validity criteria: A comparative overview”. In: *Statistical Analysis and Data Mining* 3 (2010), pp. 209–235.

[37] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. English. Englewood Cliffs, NJ: Prentice Hall, 1988, pp. xiv + 320. ISBN: 0-13-022278-X.

[38] J. Handl, J. Knowles, and D. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.

[39] H. Van der Hoef and M. J. Warrens. “Understanding information theoretic measures for comparing clusterings”. In: *Behaviormetrika* 46 (2019), pp. 353–370.

[40] G. Schwarz. “Estimating the Dimension of a Model”. In: *The Annals of Statistics* 6.2 (1978), pp. 461–464.

[41] W. M. Rand. “Objective Criteria for the Evaluation of Clustering Methods”. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. ISSN: 01621459.

[42] T. J. Cover TM. *Elements of information theory*. Schilling D (ed) Wiley series in telecommunications. Wiley, New York, 1991, pp. 12–49.

[43] N. X. Vinh, J. Epps, and J. Bailey. “Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?” In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML’09. Montreal, Quebec, Canada, 2009, pp. 1073–1080.

[44] P. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[45] L. Hubert and P. Arabie. “Comparing partitions”. In: *Journal of Classification* 2.1 (Dec. 1985), pp. 193–218.

[46] A. Rosenberg and J. Hirschberg. “V-measure: A conditional entropy-based external cluster evaluation measure”. In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2007, pp. 410–420.

[47] A. Zubaroglu and V. Atalay. “Data stream clustering: a review”. In: *Artificial Intelligence Review* 54.2 (2021), pp. 1201–1236. DOI: 10.1007/s10462-020-09874-x.

[48] P. Fränti and S. Sieranoja. *K-means properties on six clustering benchmark datasets*. 2018. URL: http://cs.uef.fi/sipu/datasets/.

[49] P. Fränti, O. Virmajoki, and V. Hautamäki. “Fast agglomerative clustering using a k-nearest neighbor graph”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 28.11 (2006), pp. 1875–1881.

[50] J. A. Blackard, D. J. Dean, and C. W. Anderson. *UCI Machine Learning Repository*. 1998. URL: http://archive.ics.uci.edu/ml.

[51] S. Hettich and S. Bay. *The UCI KDD Archive. University of California, Department of Information and Computer Science, Irvine, CA*. 1999.

[52] E. Lughofer. “A dynamic split-and-merge approach for evolving cluster models”. In: *Evolving Systems* 3.3 (Sept. 2012), pp. 135–151.

[53] A. Eghbalian and et al. “Multi-view Data Mining Approach for Behaviour Analysis of Smart Control Valve”. In: *Proc. of 19th IEEE ICMLA*. 2020, pp. 1238–1245.

[54] F. R. Hampel. “A General Qualitative Definition of Robustness”. In: *The Annals of Mathematical Statistics* 42.6 (1971), pp. 1887–1896. ISSN: 00034851. URL: http://www.jstor.org/stable/2240114.

[55] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[57] A. Hagberg, P. Swart, and D. S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[58] W. McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by S. van der Walt and J. Millman. 2010, pp. 56–61.

[59] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. "Array programming with NumPy". In: *Nature* 585 (2020), pp. 357–362.

[60] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272.

[61] J. Erler, M. E. Ramos-Ceja, K. Basu, and F. Bertoldi. "Introducing constrained matched filters for improved separation of point sources from galaxy clusters". In: *ArXiv e-prints* (2018). eprint: `1809.06446`.

Paper VI

Domain Adaptation Through Cluster Integration and Correlation

Vishnu Manasa Devagiri, Veselka Boeva, Shahrooz Abghari

In: 2022 IEEE International Conference on Data Mining Workshops (ICDMW). 2022, pp. 1–8, DOI: 10.1109/ICDMW58026.2022.00025

Abstract

Domain shift is a common problem in many real-world applications using machine learning models. Most of the existing solutions are based on supervised and deep-learning models. This paper proposes a novel clustering algorithm capable of producing an adapted and/or integrated clustering model for the considered domains. Source and target domains are represented by clustering models such that each cluster of a domain models a specific scenario of the studied phenomenon by defining a range of allowable values for each attribute in a given data vector. The proposed domain integration algorithm works in two steps: (i) cross-labeling and (ii) integration. Initially, each clustering model is crossly applied to label the cluster representatives of the other model. These labels are used to determine the correlations between the two models to identify the common clusters for both domains, which must be integrated within the second step. Different features of the proposed algorithm are studied and evaluated on a publicly available human activity recognition (HAR) data set and real-world data from a smart logistics use case provided by an industrial partner. The experiment's goal on the HAR data set is to showcase the algorithm's potential in automatic data labeling. While the conducted experiments on the smart logistics use case evaluate and compare the performance of the integrated and two adapted models in different domains.

Keywords: domain adaptation, clustering techniques, data integration

1 Introduction

Machine learning (ML) models, such as clustering models used in many real-time applications, may experience a drop in performance over time or in a new environ-

mental context. One of the reasons is the change in data characteristics of the environment in which the model is used compared to the one it is trained on [1]. A natural solution to this problem could be to rebuild the clustering solution, but this is not ideal due to the challenges that come along with it, like the number of computational resources required to rebuild the model from scratch and the lack of proper data. This creates the need to use already available knowledge to build a newly updated clustering solution that works in the new environment (domain). This model adaptation to new domains is known as domain adaptation, a sub-branch of transfer learning. The domain on which the model is initially trained is the source domain, and the one to which it is adapted to is the target domain [2]. In the context of domain adaptation, the source and target domains address the same problem and are related but can have different data distributions [3]. Some examples include autonomous vehicles, tracking of goods, personalized healthcare, and customer recommendations like spam detection. In the cases of autonomous vehicles and monitoring of goods, different environments (locations) can be considered as the domains, whereas in the case of personalized recommender systems like healthcare, users with distinct behaviors are considered as different domains.

Most of the works in the field of domain adaptation propose deep learning-based techniques in computer vision and image analysis, some of which are described in [4, 5]. The current study instead proposes a clustering-based domain adaptation approach. Compared to deep learning algorithms, clustering algorithms require fewer computational resources and are transparent and understandable. We explore the area of domain adaptation by proposing a domain integration bi-correlation clustering algorithm (DIBCA). The proposed algorithm integrates the clustering models describing the source and target domains to obtain an integrated clustering model that can be used in both domains. In addition, the integrated model is composable and can be split into two adapted models, one per domain. The latter two models are smaller than the integrated model and are also expected to better reflect their domains' specificity. In general, DIBCA is designed considering the edge devices' computational and memory constraints, making it suitable for use in situations with fewer computational or memory resources. DIBCA can be segregated into two major steps, cross-labeling and integration. In the cross-labeling phase, cluster representatives of each of the source and target domains are labeled using the other clustering model, which is used to find the correlations between clusters of different domains. This is followed by the integration phase, where the correlated clusters are merged, and the rest are retained.

The proposed algorithm properties have been studied and evaluated in two different use cases: real-world data from smart logistics and publicly available Human Activity Recognition (HAR) data. The conducted experiments in the real-world data context have confirmed our expectation that the performance of the integrated and two adapted models produced by the algorithm is comparable to that of the two initially built (source and target) models. In addition, experimental results on the HAR

data have shown that DIBCA has the potential to be used for automatic data labeling tasks.

2 Related Work

Pan *et al.* [1] and Zhuang *et al.* [2] present comprehensive surveys of transfer learning. Pan *et al.* [1] review transfer learning approaches in the fields of clustering, classification, and regression, whereas the second study [2] focuses more on homogeneous transfer learning, i.e., where the source and target have the same feature space. Adversarial clustering is used in many unsupervised domain adaptation algorithms like [6, 7] to reduce the domain shift between the source and target domains. Deng *et al.* [8] propose a new cluster alignment technique with a teacher for unsupervised domain adaptation. The algorithm uses discriminative clustering loss to align similar classes across domains.

In [9], the authors propose domain consensus clustering. The algorithm can transfer knowledge from the source to the target domain when the labeled space is not the same. Common clusters from both domains, as well as private clusters which are specific to one of the domains, are identified. Class-aware alignment technique is used on the common clusters to minimise distribution shift between different domains. Li *et al.* [7] propose a semi-supervised domain adaptive clustering algorithm entitled Cross-Domain Adaptive Clustering. Adversarial clustering loss is introduced to group features of unlabelled data, followed by cluster-wise feature alignment in both domains. The proposed algorithm takes advantage of pseudo labeling to increase labeled instances in the target domain.

A survey published in [4] presents an overview of unsupervised domain adaptation algorithms suitable for classification problems. The algorithms are grouped into the following five different categories based on the adopted technology, namely, discrepancy, adversarial, reconstruction, representation, and attention, based methods. Lu *et al.* [10] focus on domain adaptation in the field of HAR. The authors have proposed an algorithm using which knowledge obtained from one HAR data set could be transferred and used in another similar target data set. Xu *et al.* [11] use domain adaptation with meta-learning for cross-domain recommendations. The proposed algorithm helps to avoid cold-start in recommender systems. Gunasekara *et al.* [12] propose a novel method to deal with constant changes in data characteristics and catastrophic forgetting. They create a pool of learners entitled Online Domain Incremental Pool, containing different learners from which a trained task predictor selects the appropriate one.

Tang *et al.* [13] and Zhu *et al.* [14] propose and highlight the advantages of not using actual data from the source domain. The source model and the target data are used by the algorithm in the adaptation process. As such a solution protects data privacy, there is a rise in demand for similar works [13].

3 Problem Statement

Applications on smart or edge devices require continuous learning and updating of their ML models as data characteristics change over time due to changes in the device's environmental context, data collection from a new user or device, etc. This change in data characteristics between different domains, referred to as domain shift, drops the ML model's performance and requires model adaptation to the new circumstances. These devices usually have limited energy and memory capacities which should be considered while designing algorithms to be used by them. In addition, the availability of labeled data is limited, requiring unsupervised learning techniques like clustering to analyze, interpret and act on the data.

Clusters are traditionally represented by their centroids (e.g., mean, median, or medoid). However, such representatives cannot always represent cluster-specific characteristics completely. In a clustering model, knowing the range of values each feature could take for each cluster is more valuable information that is used in this study. Namely, each cluster could be represented as a range between which the data points of the cluster lie, an idea used in the Inductive System Health Monitoring (ISM) method introduced in [15]. Similarly, in the current study, each cluster of a model is represented by its low and high value vectors i.e., the minimum and maximum values of each feature representing the data set, respectively. In addition, the mean vector can be derived from the low and high vectors as needed. This idea can be illustrated as follows. Suppose a clustering model represents different outdoor scenarios where each cluster models a specific outdoor environment based on the cellular network radio signals. Each of these clusters can be represented using the range of low and high radio signal strengths of the considered features.

Domain shift phenomenon for clustering can cause appearing, disappearing, or change in the ranges of the clusters. This study proposes a resource-efficient novel domain adaptation algorithm using cluster representatives of low and high value vectors. The algorithm is based on cluster integration and can be used in supervised, unsupervised and semi-supervised setups to integrate source and target models. In case of a supervised setup, both domain models consist of labeled classes and each class can additionally be represented by a group of clusters. In an unsupervised setting source and target models are presented by clustering solutions, while in case of semi-supervised scenario, only source model is based on labeled data. In that way, the source model can consist of classes of clusters, while the target model contains just non-labeled clusters.

4 Proposed Domain Integration Clustering Algorithm

4.1 Range-based correlation measure

This work introduces a range-based correlation measure that has been inspired by interval decision-making, where the available information is usually presented by suitable intervals and comparisons. More specifically, we have studied how the overlap of the interval alternatives' evaluations have been used to express valued preferences among the alternatives, introduced in [16].

Formally, given two sets of data instances A and B, the *range-based correlation* R is defined by Eq. VI.1, where each set, e.g., A, is represented by two vectors, denoted by l_A and h_A, of its low and high attributes' values, respectively. In addition, a special data vector m_A is constructed (or selected) by applying some preliminarily defined criterion. For example, in our experiments, we use the mean vector of l_A and h_A, but it can also be a specific data instance in the set (e.g., its medoid). Notice that $d(.,.)$ is the Euclidean distance between two data vectors. To simplify the definition of R given in Eq. VI.1, we use the abbreviation d_{AB} to denote $d(l_A, m_A) + d(m_B, h_B)$.

$$R(A,B) = \begin{cases} 1 & \textit{if } d_{AB} < d(m_A, m_B) \\ \frac{d(m_A,m_B)}{d_{AB}} & \textit{otherwise}\,. \end{cases} \tag{VI.1}$$

Note that R is non-negative and symmetric for any pair of data sets. In addition, if $A = B$, then $R(A,B) = 0$.

4.2 The proposed algorithm

This section formally introduces the proposed DIBCA algorithm. The algorithm is used to integrate the clustering models of different but related domains. Assume that C^1 and C^2 are respectively clustering models of the source and target domains, represented as $C^1 = \{C_1^1, C_2^1, \ldots, C_m^1\}$ and $C^2 = \{C_1^2, C_2^2, \ldots, C_n^2\}$, where $\mid C^1 \mid = m$ and $\mid C^2 \mid = n$. Each cluster from these models can be represented by three value vectors, low, mean, and high. For example, the representatives of a cluster C_i^1 can be presented as $l_{C_i^1}$, $m_{C_i^1}$, $h_{C_i^1}$. The algorithm uses the mean of each of these clusters to determine the alignments between the clustering solutions of source and target domains by cross-labeling, where the source model is applied on the target representatives and the target model on the source representatives. The alignments identified are of three types: (i) both directions, (ii) source to target, and (iii) target to source. When the association is in both directions, this implies that the clusters are similar and are correlated well. In the second case, i.e., when the source representatives are labeled using the target model, no action is performed as the target model is not rich. For the third case, the range-based correlation measure described in Section 4.1 (Eq. VI.1) is used to find the correlations between the aligned clusters. Only if the

value obtained is less than the threshold (0.45 in this case), the clusters are considered to be similar. This is followed by the integration phase, where the identified similar (strongly correlated) clusters from both domains are merged to obtain common clusters. The remaining clusters which did not undergo integration are left as private clusters in their respective domains. As a result, the algorithm obtains two different clustering models for each domain, an integrated model and an adapted source/target model. The integrated clustering model consists of common clusters and private clusters of both source and target. The obtained integrated clustering model is composable and can be split into smaller adapted models consisting of private clusters of the respective domain and common clusters. These models are respectively denoted as adapted source and adapted target models and can be easily generated in addition to the integrated model. The idea of common and private clusters is inspired from [9]. Figure 1 presents a high-level visual overview of the proposed DIBCA algorithm.

The pseudo-code of the algorithm is broken down into Algorithm VI.1 and Algorithm VI.2. Algorithm VI.1 presents the overview of the main steps of the proposed DIBCA algorithm, whereas Algorithm VI.2 overviews the integration procedure, i.e., it describes how two closely correlated clusters are integrated.

In the case of labeled data, the source and target models are built by applying the ISM algorithm on the instances of each class separately and merging together the clustering models obtained on the different classes. In that way, each cluster will be marked by a pair of labels representing its class and cluster, respectively. The DIBCA algorithm can be used on such built models to identify the correlated clusters in order to produce an integrated model. It is worth mentioning that if cross-class correlated clusters are detected, this may be an indication of mislabeled instances which can lead to further analysis of the correlated classes.

The algorithm is designed to be used for data with numerical features, categorical features such as ordinal and nominal should be transformed into numerical values to be used by the algorithm. While designing the algorithm, a conscious effort has been made to make sure that DIBCA is resource efficient. It can be noted that the algorithm performs all its operations on the cluster representatives instead of using all the cluster data points, which significantly reduces the amount of computational power and memory required. This makes the algorithm suitable to be run on edge devices, known as ones with resource restrictions. Another characteristic of the algorithm that is worth noting is that data privacy is not compromised as the algorithm uses only the cluster representatives, which are artificial nodes.

The computational time complexity of the algorithm is approximated separately for the two main parts of the algorithm: identifying cluster correlations and cluster integration. $O(mnn_f)$ is the time complexity of the part that finds the correlations between the source and target clustering models, where m and n are the number of clusters in source and target domains, n_f is the number of the data set features. In the integration phase, low and high value vectors of length n_f are computed for each of the common clusters. These vectors are obtained by comparing the feature values of

Algorithm VI.1 Domain integration clustering algorithm for updating the clustering solution to be adapted to both source and target domains.

Input: Clustering models C^1 (source model) and C^2 (target model), where each cluster C_i^j is represented by $l_{C_i^j}$ and $h_{C_i^j}$

for each $C_i^j \in C^1 \cup C^2$ **do**
  find $m_{C_i^j}$
end for
Label $m_{C_i^1}$, $(i = 1, \ldots, m)$ using C^2
Label $m_{C_i^2}$, $(i = 1, \ldots, n)$ using C^1
if $(m_{C_i^1} \in C_i^2) \wedge (m_{C_i^2} \in C_i^1)$ **then**
  Correlation(C_i^2, C_i^1), Eq. VI.1 $= 0$
end if
for each $C_i^1 \in C^1$ **do**
  for each $C_i^2 \in C^2$ **do**
    if Correlation(C_i^2, C_i^1), Eq. VI.1 $<$ threshold (0.45) **then**
      ClusterList.add(C_i^2)
    end if
  end for
  if ClusterList $\neq \emptyset$ **then**
    Integration(C_i^1, ClusterList), Algorithm VI.2
  end if
end for
Clusters not integrated are private in their respective domains.

Algorithm VI.2 Integrating clusters to build common clusters.

Input: Cluster representatives of C_i^1 and ClusterList = $\{C_j^2, C_k^2 \ldots\}$, i.e., $l_{C_i^1}$, $h_{C_i^1}$; $l_{C_j^2}$, $h_{C_j^2}$; $l_{C_k^2}$, $h_{C_k^2}$; …, respectively

for i in the range(length($l_{C_i^1}$)) **do**
  low of union.append($\min(l_{C_i^1}, l_{C_j^2}, l_{C_k^2} \ldots)$)
  high of union.append($\max(h_{C_i^1}, h_{C_j^2}, h_{C_k^2} \ldots)$)
end for
low, high value vectors of the new integrated cluster = (low of union, high of union)

the clusters to be integrated and finding the minimum and maximum value for each feature; this can be approximated to $O(n_c n_f)$, where n_c is the number of identified final correlations between the source and target. Summing both expressions, the computational time complexity of the algorithm would be $O(mnn_f + n_c n_f)$. Finally,

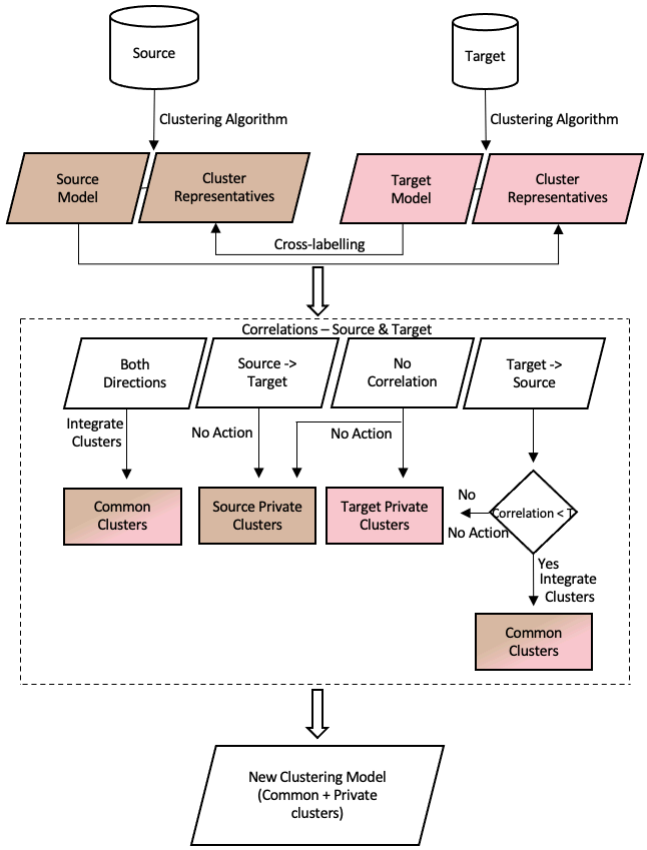


Figure 1: A high-level overview of the proposed Domain Integration Bi-correlation Clustering Algorithm. Initially, the source and target clustering models are crossly applied to each other in order to identify correlations. This is followed by the integration phase, where similar clusters from both domains are merged to obtain common clusters or retained as private clusters of the respective domains.

it can be approximated to $O(mnn_f)$, since $n_c < mn$.

5 Experimentation

The properties of DIBCA algorithm are studied and evaluated on two data sets (real-world and publicly available data) under a few different experimental setups. The public data set is associated to multi-class classification tasks and is used to explore the potential of the proposed algorithm in data labeling. Details about the experiments are presented in Sub-section 5.1.1. The real-world data set contains measurements from trackers used in smart logistics for monitoring and tracking goods. This data set is used to evaluate the potential of the algorithm in domain adaptation tasks, we compare the performance of the three models (i.e., integrated, adapted source, and adapted target models) generated by the DIBCA algorithm with the original source

and target models. Further details about the experiments conducted using the real-world data set are presented in Sub-section 5.2.1.

5.1 Public data

We use a publicly available real-world HAR data set, PAMAP2 [17], taken from the UCI machine learning repository, to evaluate the algorithm's performance concerning labeling in semi-supervised contexts. The data set contains regular activity of nine subjects, eight male and one female, with an average age of 27.22. It presents 18 different activities and contains 52 attributes and $3,850,505$ instances. Out of the available 52 attributes, 31 are used in the study. Features related to the orientation and one of the accelerometers are not considered as they are invalid and not properly calibrated, respectively as explained in the data description.

5.1.1 Experimental Setup

In the preprocessing phase, initially, the data set is cleaned. All the data points belonging to activity 0 are dropped as this activity corresponds to transient activities, and the data set description suggests to discard them. The column corresponding to the heart rate is dropped as most of the rows have missing values for this feature. This is followed by removing rows containing missing values. Similar to what is done in [10, 18], the three dimensional data collected from each of the accelerometer, gyroscope, and magnetometer at each measurement unit are aggregated using $a = \sqrt{x^2 + y^2 + z^2}$ formula. Once the data is divided into the source and target, each part of the data set is standardized separately using the z-score, $z = (x - \mu)/s$. Labeled data is the most sort after for many data analysis tasks but is sparsely available. Labeling data set is often costly [2], requiring the manual work of many domain experts and users. The PAMAP2 [17] data set is used to illustrate how the proposed algorithm could be used to label the data in a semi-supervised context where the labels are known for source data but not for the target. Assuming that the target data is grouped into different clusters, each representing one class of data points. The proposed algorithm can be used to find the correlations between the source and target, thus enabling the labeling of the clusters and, in turn, the data points assigned to them.

To showcase the algorithm's potential in data labeling, two experimental setups, namely A.I and A.II are studied. In both A.I and A.II, the experiment is conducted on all the users one by one in the following way. Data is initially divided between source and target domains, during which it is made sure that each domain contains all the activities of a user. In experiment A.I, sampling with replacement is used while dividing the data into source and target domains. For each user activity, the source and target domains have 80% and 20% of sampled data with replacement, respectively. In this experimental setting, the source and target clusters may have a

closer resemblance to each other, as there could be a possibility of having overlapping data points. In experiment A.II, sampling without replacement is used. That is for each user activity, after considering a sample of 80% for the source domain, rest of the 20% data is used to represent the target domain. Once divided, data in both source and target domains are clustered based on the class labels. The proposed algorithm is then used to find the correlations between the source and target domain in experiments A.I and A.II, which can help in data instance labeling.

5.2 Real-world use case

The proposed algorithm is also evaluated in a use case from smart logistics, provided by our industrial partner. Namely, the algorithm is applied to integrate Global Navigation Satellite Systems (GNSS) activation models built on different domains (devices/locations) into an integrated model. GNSS is the positioning technique used for detecting the tracker's current position, which is known to perform well in open-sky environments. However, the trackers may be in any place, such as open outdoors, crowded city areas, indoors etc. Therefore it is necessary to perform context-aware control of GNSS activation by automatically and accurately detecting indoor/outdoor localization of trackers with low battery consumption. This is based on the use of radio signals received from Long-Term Evolution (LTE) base stations to detect the environment (indoor/outdoor). The data set used contains radio signal measurements collected by five smart devices (trackers) and their geographical location in various environmental scenarios. The devices collected various parameters like cell Id, GNSS location, device number, etc., out of which eight features, depicting the signal strength from the cells and details about the number of cells receiving the signals are used to build the GNSS component activation model.

We have studied two experimental domain integration scenarios (B.I and B.II) in which the source and target domains represent two different locations based in South Sweden. Both experiments are performed with data from a single device. The source domain is based on data collected from a device used in a comparatively big city, while the target domain uses data covering a nearby rural area.

5.2.1 Experimental Setup

Before conducting the experiments, the data set is standardized using the z-score in both the experimental scenarios B.I and B.II. In the experimental setting B.I, the training and test data sets are standardized separately, whereas, in the experimental setting B.II, the data with and without GNSS location are standardized separately. Clustering of source and target domains in experiments B.I and B.II is done by using the algorithm proposed in [19], which is based on [15]. By using the DIBCA algorithm, three new updated models are generated, an integrated model and two adapted models, one for the source domain and one for the target domain. These models can

contain two types of clusters, private and common. Private clusters are the remaining original clusters, which did not undergo integration. The source and target domains have their own private clusters. Common clusters are those obtained after integrating two or more clusters from the source and target clustering models. The integrated model contains all the source and target private clusters and common clusters. In comparison, the adapted source and target models contain the common clusters, and private clusters of either source or target, respectively.

In all the experimental scenarios, the three newly generated models, along with the initial source and target models, are used to analyze how each model performs on the source and target domains. Therefore, the models are evaluated on the test data from both domains. For the experimental setting B.I, models of both source and target domains are built using the first 80% of the data with GPS coordinates. The remaining 20% of the data with GPS coordinates and all the data points without GPS coordinates are used for testing purposes. The experiment aims to analyze how the algorithm performs when notion of time is considered. Only a single iteration of the experiment is conducted as the chronological order of data is taken into account. For the experimental setting B.II, the source and target models are built using 90% of the data with GPS coordinates. The remaining 10% of the data with GPS coordinates and all the data points without GPS coordinates are used for testing purposes. Ten fold cross-validation is used to conduct the experiments and split the data with GPS location into training or test data sets.

6 Results and Discussion

6.1 Public data

The results of experiments A.I and A.II conducted on the PAMAP2 HAR data set are presented in Tables 1 and 2, respectively. As stated in Sub-section 5.1.1, the data of each user is divided into two parts, source and target, using sampling with and without replacements in experiments A.I and A.II, respectively. In the tables, the first column presents the User Id, followed by information about the model and data set used (Model-Data). M_t, M_s, D_t, and D_s stand for target model, source model, target data and source data, respectively. Next column presents information about how each activity performed by the respective user are correlated between the source and target domains. Cells are colored green, red, or grey depending on if the correlation has been identified to the same class cluster, different class cluster, or no correlation, respectively. This is followed by a numerical summary of the correlations in columns 4 and 5. The last column presents the final result, i.e., the information about what percentage and number of target clusters are labeled correctly.

In Table 2, activity 13 for user 4 is labeled incorrectly when the target model is used on source data, thus correlating it to a different cluster. This is the only

Table 1: *Experiment A.I:* Labeling of target data in semi-supervised scenario using sampling with replacement - (80%) source and (20%) target data for each user

U Id	Model-Data	Activities												Correct corr.	No corr.	Clusters Labeled Correctly
		1	2	3	4	5	6	7	12	13	16	17	24	(%)	(%)	**in Target** (%, Number)
U1	$M_t - D_s$													91.6	8.3	100.0 (12)
	$M_s - D_t$													100.0	0.0	
U2	$M_t - D_s$													83.3	1.7	83.3 (10)
	$M_s - D_t$													83.3	1.7	
U3	$M_t - D_s$													87.5	12.5	87.5 (7)
	$M_s - D_t$													87.5	12.5	
U4	$M_t - D_s$													90.9	9.1	81.8 (9)
	$M_s - D_t$													90.0	10.0	
U5	$M_t - D_s$													91.6	8.3	91.7 (11)
	$M_s - D_t$													91.6	8.3	
U6	$M_t - D_s$													50.0	50.0	66.7 (8)
	$M_s - D_t$													66.7	33.3	
U7	$M_t - D_s$													81.8	18.2	90.9 (10)
	$M_s - D_t$													90.9	9.1	
U8	$M_t - D_s$													83.3	16.7	75.0 (9)
	$M_s - D_t$													75.0	25.0	
U9	$M_t - D_s$													0.0	100.0	100.0 (1)
	$M_s - D_t$													100.0	0.0	
Avg														**80.3**	**18.0**	**86.3**
std														**22.6**	**23.3**	**10.4**

Note. Activities 9-11, 18-20 are missing for all users, so these are removed from the table.

case where a correlation between two different activities of the source and target is misidentified, and hence, the cell is marked in red. However, this did not impact the final result of labeling. The target cluster is labeled correctly since the algorithm gives higher weight to correlation obtained using source model (which is built on a richer data set) on target data.

From both the tables, it can be seen that the algorithm has performed decently well in correctly labeling the target instances. In Experiment A.I, on an average 86.3% of the clusters are correctly labeled; this value has gone up to 91.1% in experiment A.II, displaying the algorithm's ability in data labeling. It can be noted that even though the algorithm was unable to identify labels for some clusters, none of them were incorrectly labeled, which is a positive sign and displays the algorithm's potential in being used for automatic data labeling. It is worth to further study this property of the algorithm by evaluating the algorithm capacity in the annotation of data in different applied contexts and data sets.

6.2 Real-world use case

Figures 2a and 2c present the performance of different clustering models in experimental scenario B.I, with respect to accuracy and F-measure, respectively. In this experimental scenario, even though there is a drop in the overall performance of all models compared to experiment B.II, both integrated and adapted source/target models perform better based on accuracy than the initial source and target models. In addition, their performance is comparable to that of the initial models based on the F-measure. This is a promising result, as the time series data is obtained and analyzed chronologically in real-world scenarios.

The performance of the target model is the least in terms of accuracy and F-measure on both source (47.77%, 0.16) and target data (35,03%, 0.44). This could be due to insufficient data in the target domain to detect various cellular network scenarios. The figures show that both integrated and adapted source/target models have similar performance; in such cases, using adapted models saves resources as private clusters of other domains are not used in these models.

The results obtained by experiment B.II are presented in Figures 2b and 2d, in terms of average accuracy and average F-measure, respectively. The plots show that the best-performing models are the source model applied to the source data (68.68%, 0.49) and the target model applied to the target data (92.96%, 0.95). In terms of both measures, the adapted source/target model performs better when compared to the integrated model. These results are logical as the adapted model does not contain private clusters of the other domain. With respect to experiment B.II, it is worth noting that even though the performance of the adapted source/target model and the integrated model are not the best, they demonstrate equally good performance on both source and target data, unlike the initial source and target models, which have

Table 2: *Experiment A.II:* Labeling of target data in semi-supervised scenario using sampling without replacement - (80%) source and (20%) target data for each user

U Id	Model-Data	Activities												Correct corr.	No corr.	Clusters Labeled Correctly
		1	2	3	4	5	6	7	12	13	16	17	24	(%)	(%)	**in Target**(%, Number)
U1	$M_t - D_s$													91.6	8.3	100.0 (12)
	$M_s - D_t$													100.0	0.0	
U2	$M_t - D_s$													75.0	25.0	83.3 (10)
	$M_s - D_t$													83.3	16.7	
U3	$M_t - D_s$													75.0	25.0	100.0 (8)
	$M_s - D_t$													100.0	0.0	
U4	$M_t - D_s$													72.7	18.2	81.8 (9)
	$M_s - D_t$													90.0	10.0	
U5	$M_t - D_s$													83.3	16.7	100.0 (12)
	$M_s - D_t$													100.0	0.0	
U6	$M_t - D_s$													50.0	50.0	66.7 (8)
	$M_s - D_t$													66.7	33.3	
U7	$M_t - D_s$													90.9	9.1	100.0 (11)
	$M_s - D_t$													100.0	0.0	
U8	$M_t - D_s$													83.3	16.7	88.3 (10)
	$M_s - D_t$													83.3	16.7	
U9	$M_t - D_s$													0.0	100.0	100.0 (1)
	$M_s - D_t$													100.0	0.0	
Avg														**80.3**	**19.2**	**91.1**
std														**23.5**	**23.5**	**11.5**

Note. Activities 9-11, 18-20 are missing for all users, so these are removed from the table.

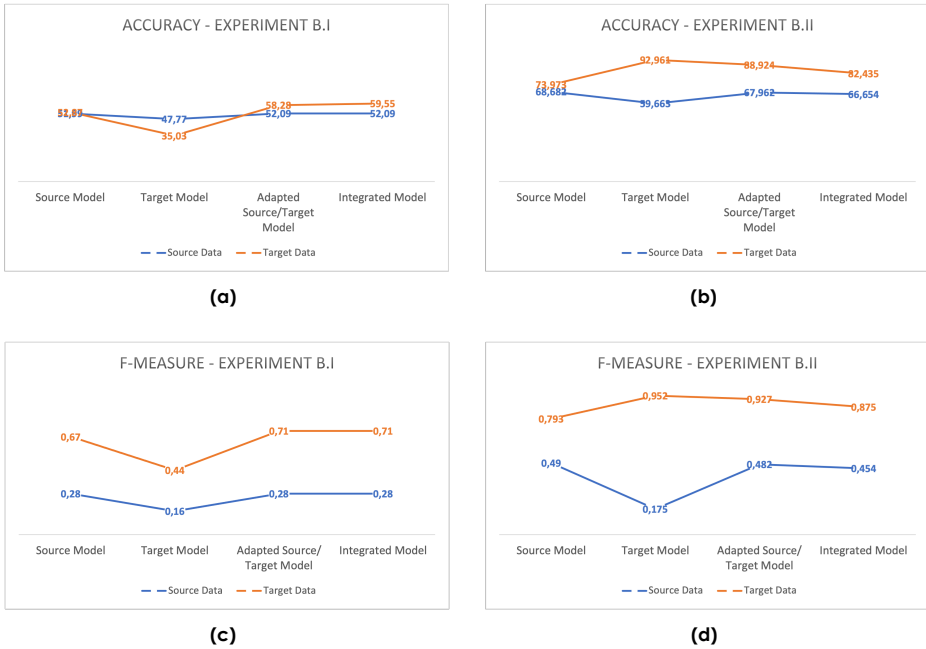


Figure 2: Experimental results of smart logistics use-case: (a) Experimental Scenario B.I: accuracy scores produced by the different models on source and target data.; (b) Experimental Scenario B.II: average accuracy produced by the different models on source and target data.; (c) Experimental Scenario B.I: F-measure scores produced by the different models on source and target data.; and, (d) Experimental Scenario B.II: average F-measure scores produced by the different models on source and target data.

higher performance in their own domains. The high performance of the target model on its own data could be due to over-fitting, as the data set is relatively small.

We believe that the general increase in performance for B.II, in comparison with B.I, is due to the fact that in the case of B.II, similar cellular network scenarios to ones met in the test data are already seen while building the model in most cases. In experiment B.I, as the test data contains the last chunk of data (except for data points without GPS coordinates), they might have slightly different characteristics (new and unseen scenarios) than the data the model is trained on. A larger data set where the training data covers a big variety of cellular network scenarios may solve this problem.

7 Conclusions and Future Work

This study proposes a novel domain integration bi-correlation clustering algorithm (DIBCA). The proposed algorithm is designed by taking the computational as well as memory requirements into consideration. As the algorithm uses cluster representatives to perform its operations, the number of resources used is reduced significantly.

This also preserves the privacy of the data set, as the algorithm does not use actual data points. Different properties of the algorithm are evaluated on two data sets, PAMAP2, a publicly available HAR data set from the UCI machine learning repository, and a real-world data set related to the smart logistics domain supplied by one of our industrial partners. The algorithm's performance on the smart logistics use-case data is promising. The performance of integrated, adapted source/target models generated using DIBCA is better or comparable to the original models of the considered domain with respect to accuracy and F-measure. The experimental results on the HAR data set have shown that the proposed algorithm is able to correctly label up to 91.1% of the total available clusters in the target domain. In addition to this, none of the clusters are mislabeled. Based on the results, DIBCA has the potential to be applied for labeling tasks as a reliable automatic data annotation solution.

As a part of our future work, the labeling property of the algorithm will be further explored by studying new experimental scenarios on new data sets. In addition, our proposed algorithm will be tested in the area of domain adaptation for new use cases like personalized user recommendations to evaluate its potential. The plan also includes evaluating the algorithm on richer data sets from smart logistics use cases that our industrial partner provides.

Acknowledgements

This work is a part of Sony RAP 2020 Project, "Distributed and Adaptive Edge-based AI Models for Sensor Networks".

References

[1] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[2] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76. DOI: 10.1109/JPROC.2020.3004555.

[3] M. AlShehhi, E. Damiani, and D. Wang. "Toward Domain Adaptation for small data sets". In: *Internet of Things* 16 (2021), p. 100458. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2021.100458.

[4] Y. Madadi, V. Seydi, K. Nasrollahi, R. Hossieni, and T. Moeslund. "Deep Visual Unsupervised Domain Adaptation for Classification Tasks: A Survey". In: *IET Image Processing* 14.14 (2020), pp. 3283–3299. ISSN: 1751-9659. DOI: 10.1049/iet-ipr.2020.0087.

[5] H. Tang, Y. Wang, and K. Jia. “Unsupervised domain adaptation via distilled discriminative clustering”. In: *Pattern Recognition* 127 (2022), p. 108638. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2022.108638.

[6] H. Wang, J. Tian, S. Li, H. Zhao, F. Wu, and X. Li. “Structure-conditioned adversarial learning for unsupervised domain adaptation”. In: *Neurocomputing* 497 (2022), pp. 216–226. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.04.094.

[7] J. Li, G. Li, Y. Shi, and Y. Yu. *Cross-Domain Adaptive Clustering for Semi-Supervised Domain Adaptation*. 2021. DOI: 10.48550/ARXIV.2104.09415. URL: https://arxiv.org/abs/2104.09415.

[8] Z. Deng, Y. Luo, and J. Zhu. “Cluster Alignment With a Teacher for Unsupervised Domain Adaptation”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9943–9952. DOI: 10.1109/ICCV.2019.01004.

[9] G. Li, G. Kang, Y. Zhu, Y. Wei, and Y. Yang. “Domain Consensus Clustering for Universal Domain Adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 9757–9766.

[10] W. Lu, Y. Chen, J. Wang, and X. Qin. “Cross-domain activity recognition via substructural optimal transport”. In: *Neurocomputing* 454 (2021), pp. 65–75. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.04.124. URL: https://www.sciencedirect.com/science/article/pii/S0925231221007025.

[11] J. Xu, J. Song, Y. Sang, and L. Yin. “CDAML: a cluster-based domain adaptive meta-learning model for cross domain recommendation”. In: *World Wide Web* (2022), pp. 1573–1413. DOI: 10.1007/s11280-022-01068-5.

[12] N. Gunasekara, H. Gomes, A. Bifet, and B. Pfahringer. “Adaptive Online Domain Incremental Continual Learning”. In: *Artificial Neural Networks and Machine Learning – ICANN 2022*. Ed. by E. Pimenidis, P. Angelov, C. Jayne, A. Papaleonidas, and M. Aydin. Cham: Springer International Publishing, 2022, pp. 491–502.

[13] S. Tang, Y. Zou, Z. Song, J. Lyu, L. Chen, M. Ye, S. Zhong, and J. Zhang. “Semantic consistency learning on manifold for source data-free unsupervised domain adaptation”. In: *Neural Networks* 152 (2022), pp. 467–478. ISSN: 0893-6080.

[14] M. Zhu. “Source Free Domain Adaptation by Deep Embedding Clustering”. In: *2021 18th Int. Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. 2021, pp. 309–312. DOI: 10.1109/ICCWAMTIP53232.2021.9674068.

[15] D. L. Iverson. “Inductive System Health Monitoring.” In: *IC-AI*. 2004, pp. 605–611.

[16] V. Boeva and B. De Baets. “A new approach to admissible alternatives in interval decision making”. In: *2004 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No.04EX791)*. Vol. 1. 2004, 110–115 Vol.1. DOI: 10.1109/IS.2004.1344647.

[17] A. Reiss and D. Stricker. *Introducing a New Benchmarked Dataset for Activity Monitoring*. 2012. URL: https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring.

[18] J. Wang, Y. Chen, L. Hu, X. Peng, and P. S. Yu. “Stratified Transfer Learning for Cross-domain Activity Recognition”. In: *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 2018, pp. 1–10. DOI: 10.1109/PERCOM.2018.8444572.

[19] S. Abghari, V. Boeva, E. Casalicchio, and P. Exner. “An Inductive System Monitoring Approach for GNSS Activation”. In: *Artificial Intelligence Applications and Innovations*. Ed. by I. Maglogiannis, L. Iliadis, J. Macintyre, and P. Cortez. Cham: Springer International Publishing, 2022, pp. 437–449.

Paper VII

A Domain Adaptation Technique through Cluster Boundary Integration

Vishnu Manasa Devagiri, Veselka Boeva, and Shahrooz Abghari

Submitted for journal publication (under review)

Abstract

Many machine learning models deployed on smart or edge devices experience a phase where there is a drop in their performance due to the arrival of data from new domains. This paper proposes a novel unsupervised domain adaptation algorithm called DIBCA++ to deal with such situations. The algorithm uses only the clusters' mean, standard deviation, and size, which makes the proposed algorithm modest in terms of the required storage and computation. The study also presents the explainability aspect of the algorithm. DIBCA++ is compared with its predecessor, DIBCA, and its applicability and performance are studied and evaluated in two real-world scenarios. One is coping with the Global Navigation Satellite System activation problem from the smart logistics domain, while the other identifies different activities a person performs and deals with a human activity recognition task. Both scenarios involve time series data phenomena, i.e., DIBCA++ also contributes towards addressing the current gap regarding domain adaptation solutions for time series data. Based on the experimental results, DIBCA++ has improved performance compared to DIBCA. The DIBCA++ has performed better in all human activity recognition task experiments and 82.5% of experimental scenarios on the smart logistics use case.

The results also showcase the need and benefit of personalizing the models using DIBCA++, along with the ability to transfer new knowledge between domains, leading to improved performance. The adapted source and target models have performed better in 70% and 80% of cases in an experimental scenario conducted on smart logistics.

Keywords: Clustering techniques, Domain adaptation, Big data, Cluster integration

1 Introduction

Many smart applications are used in different types of environments, which causes a drop in models' performance due to changes in the data characteristics of these environments. In such circumstances, the model needs to be updated to accommodate the new data characteristics. Domain adaptation, a subbranch of transfer learning, can be used to achieve this. In transfer learning [1], the source and target domains can be completely different, whereas, in domain adaptation [2], the source and target domains are related, addressing the same problem but can have different data distributions.

This work is an extension of [3], in which the Domain Integration Bi-correlation Clustering Algorithm (DIBCA) is introduced. The algorithm initially identifies the correlations between the source and target domain models and then integrates similar clusters from the two domains. DIBCA produces an integrated clustering model that can be used for both domains. The produced model is composable and can be split into two lighter adapted models, one per domain. This facilitates knowledge transfer across the domains and enriches the models with knowledge from other domains. DIBCA has used a learning algorithm proposed by [4] that presents each cluster using the low and high attribute value vectors as representatives (also referred to as minimum bounding box); hence DIBCA is also designed to handle clusters using these low and high attribute value vectors. The algorithm's performance has been studied and evaluated in the areas of domain adaptation on a smart logistics use case using data obtained from our industrial partner and automatic data labeling on a publicly available Human Activity Recognition (HAR) data set [5]. The preliminary experimental results show the algorithm's potential in these two areas. On average, with respect to the domain adaptation task, the integrated and adapted models perform comparably or better than the source and target models, and for the labeling task, DIBCA labels up to 91.1% of classes correctly.

The cluster representatives (low and high attribute value vectors) used in the DIBCA and the learning algorithm are sensitive to outliers. For example, if one data point is far away from the other population of the cluster, it impacts how the cluster is represented. In order to mitigate this effect, the DIBCA and the learning algorithm are optimized in the current study to use the mean, standard deviation, and size of each cluster as representatives. This makes the optimized algorithm, DIBCA++, more robust to handling outliers than the previous version, as such information better reflects the clusters' actual data characteristics. In the current optimized version, instead of bounding boxes based on low and high attribute value vectors, confidence intervals are used to determine cluster boundaries and predict new data points of a cluster. This idea is inspired by the work done by Davidsson in [6], where the distribution of data points of each attribute or feature is assumed to be normally distributed. Since this is not generally the case in all situations, we use Chebyshev's inequality to determine the interval describing cluster boundaries. Information about the im-

provements or modifications made to DIBCA++ and the learning algorithm when compared to its predecessors is elaborated further in Sections 3.3 and 3.4, respectively.

In this work, DIBCA++ (the optimized DIBCA) is introduced, and its modules are formally described. DIBCA++ is evaluated and compared with the DIBCA algorithm on richer data sets and in a variety of experimental scenarios to study its robustness and applicability. The summary of contributions of the work is elaborated below:

1. An improved learning algorithm is devised and used for the initial building of the domain cluster model.

2. An optimized version of the DIBCA algorithm (DIBCA++), robust to outliers, is proposed.

3. The potential of DIBCA++ in domain adaptation tasks is evaluated in new specially designed experimental scenarios in order to study and gain a complete and in-depth picture of the algorithm characteristics, e.g., we explore

    a. how knowledge obtained from one device or user can be transferred to a different device or user. For example, this can also be used to help avoid a cold start.

    b. the usefulness of the proposed algorithm in use-cases that require personalization, e.g., personalized predictions and recommendations.

    c. how the similarity between the data of the source and target domains affects the algorithm performance.

    d. the performance of the clustering models generated by the DIBCA++ compared to that of the clustering model built on the integrated data from both domains.

    e. how the algorithm explainability features can be used to support the understanding and interpretation of the produced results.

This section is followed by related work in Section 2, where an overview of relevant existing studies is presented. Next, we provide a detailed description of the proposed algorithm along with a discussion about its explainability and applicability in Section 3. Details about the data sets and evaluation measures are presented in Sections 4 and 5, respectively. The experiments conducted in our study are explained in Section 6. The results from the latter are analyzed and discussed in Section 7. Finally, the conclusions and future work of the study are presented in Section 8.

2 Related Work

This section gives an overview of the domain adaptation state-of-the-art studies related to our work.

Transfer learning refers to transferring the knowledge acquired in one usually large (source) model, with rich knowledge, to another (target) model that is typically smaller. It aims to improve learners' performance on target domains by exploiting knowledge acquired from another task or domain (the source) [1]. In comparison with continual learning [7], it does not involve continuous adaptation after learning the target task. Moreover, performance on the source task(s) is not considered during transfer learning. Pan and Yang [8] and Zhuang *et al.* [1] have published studies that overview recent works in the area of transfer learning. Pan and Yang [8] in their work review and categorize transfer learning works in a wide range of fields including classification, regression, and clustering. They categorize transfer learning into three categories, inductive, transductive, and unsupervised, and highlight that more attention might be towards unsupervised transfer learning in the future. In the work done by Zhuang *et al.* [1] the focus of review is on one type of transfer learning where the source and target have the same set of attributes, named homogeneous transfer learning. The work done by [9] is also a survey, but it focuses on deep unsupervised domain adaptation dealing with classification problems.

Domain adaptation is a subbranch of transfer learning, where the knowledge is transferred between two domains addressing the same problem, having different data distributions [2]. In other words, it relaxes the classical machine learning assumption of having training and testing data drawn from the same distribution [10]. Domain adaptation is widely used in various applications where training data distribution is different from the data distribution that is used in real-time. Some examples include personalized recommender systems and autonomous vehicles. Adversarial clustering is one of the popular methods used to reduce the shift between the source and target domains [11, 12]. Authors of [11] propose *Cross-domain Adaptive Clustering*, a semi-supervised domain adaptation approach. In [12] an unsupervised domain adaptation algorithm capable of preserving the inter-class compactness is proposed for image classification. Hundschell *et al.* [13] in their work evaluate two state-of-the-art adversarial clustering techniques one based on Recurrent Neural Networks, called VRADA, and the other based on Convolutional Neural Networks, called CoDATS on four publicly available time-series data sets. Li *et al.* [14] study the universal domain adaptation problem, where the source and target domains have different label spaces. A novel domain consensus clustering algorithm is proposed, which separates the common and private clusters of both domains. For the common clusters, the class alignment technique is used to minimize the distribution shift. Xu *et al.* [15] use domain adaptive meta-learning to avoid cold start in recommender systems.

One of the concerns in the field of domain adaptation is the privacy of source data. When the model from the source domain is being adapted to the target domain,

the privacy of the source data is not protected in many of the studies. Studies such as [16, 17] highlight the importance and need for source data-free domain adaptation algorithms to preserve the privacy of source data. In [16], the authors propose a source data-free self-supervised learning method entitled *semantic consistency learning on manifold*. In the work of [17], which is also a source data-free domain adaptation algorithm, initially pseudo labels and prediction confidence for the target data are obtained using the source model. This is followed by using high-confidence prediction samples to cluster target data toward the centers of these samples.

Most of the existing domain adaptation approaches are based on deep learning [9, 18] or semi-supervised learning [11, 19] models that require significant computational resources, which make them inappropriate to be used on the edge devices. In the work [18], unsupervised domain adaptation is treated as the discriminative clustering task on target data based on information from closely related labeled source data. The authors propose a novel algorithm entitled DisClusterDA. Orbes-Arteainst et al. [19] use knowledge distillation to generalize deep neural networks so that they can be used in semi-supervised domain adaptation problems. Some recent developments in the online domain of incremental continual learning [20] are worth to be studied in this research context. Gunasekara *et al.* [20] in their work propose a novel domain adaptation algorithm using an Online Domain Incremental Pool (ODIP) of learners and dynamic task predictor. The task predictor selects the appropriate neural network for the current task from the ODIP.

In summary, the majority of the research work done in the field of domain adaptation is in the areas intersecting deep learning and computer vision. Domain adaptation in the temporal dimension is also less presented in the literature. Novel domain adaptation techniques that are resource-efficient and able to support robust model adaptation to new contexts are evidently needed. The proposed DIBCA++ is based on clustering techniques, which is its distinguishing characteristic and also makes the algorithm applicable in resource-constrained computational setups. Our work contributes towards domain adaptation for time series data, which, as mentioned above, is a less explored area compared to computer vision and other deep learning algorithms applications. In addition, the algorithm uses a data-free approach, i.e., only the clustering models of source and target are required in the adaptation process. This preserves the privacy of both source and target domains.

3 Proposed Algorithm

In this paper, we propose DIBCA++ (an optimized version of DIBCA), and an optimized version of the learning algorithm used in [21] to do initial clustering, such that both these algorithms are robust to outliers. The DIBCA algorithm was designed to be used in combination with clustering algorithms like Inductive System health Monitoring (ISM), proposed initially in [4] and later adapted in [21], using low and high

attribute value vectors of a cluster as cluster representatives (also called boundaries). The low and high attribute vectors are derived using the least and highest values for each attribute of a cluster, respectively. This makes them sensitive to the outliers, implying that even if one data point (or one of its attributes) is located away from the densely populated area of the cluster, it impacts the cluster representation.

In DIBCA++ and the learning algorithm proposed, this is eliminated by estimating the cluster boundaries using the cluster's mean and standard deviation. Mean, standard deviation, and additionally cluster size are considered representatives in the current version. In [6], the author uses the probability density function of the normally distributed data (assuming data is normally distributed) to obtain the cluster boundaries. However, not all data is normally distributed, hence, in the current work, we use Chebyshev's inequality [22] to determine the cluster boundaries with the help of the clusters' mean, standard deviation, and a variable k. The value of k is determined based on the confidence level at which the cluster boundaries are defined. The advantage of using Chebyshev's inequality is that it holds true for a wide range of distributions. The proposed algorithm is intended to transfer existing historical knowledge to a model that will be used in a new domain. The algorithm can update or adapt the existing clustering solution, considering the data characteristics of newly added domains. This enriches the clustering solutions of both source and target domains.

To facilitate the reader in the forthcoming explanation of different parts of our DIBCA++ algorithm, we summarize the used notations alphabetically in Table 1.

Table 1: Summary of notations used

C^1:	Source model
C^2:	Target model
C_i^1:	Cluster i of Source domain
C_i^2:	Cluster i of Target domain
d:	Data point
$d(.,.)$:	Euclidean distance between two data vectors
D:	Labeled data set
h_A:	Higher boundary value vector of Cluster A
k:	Real number (1), for Chebyshev's inequality
l_A:	Lower boundary value vector of cluster A
m_A:	Mean value vector of cluster A
n:	Number of clusters in source domain
n_{corr}:	Identified correlations between source and target domains
n_c:	Number of clusters
n_d:	Number of data points in the test set
o:	Number of clusters in target domain
$R(.,.)$:	Range based distance measure between two clusters
s_A:	Size of cluster A
$\mathcal{T}$:	Threshold
X_A:	Random variable of Cluster A
σ_A:	Standard deviation value vector of cluster A

3.1 Model Generalization and Cluster Representation

The initial clustering model of the domain (historical/source and the new domain) can be built using any partitioning or a hierarchical clustering algorithm. The important part is that for each cluster, A, in the built model, three entities, mean value vector (m_A), standard deviation value vector (σ_A), and the size of the cluster (s_A) are stored which are used as cluster representatives. m_A and σ_A are obtained by calculating the mean and standard deviation of each attribute of a cluster. The cluster size is needed to recalculate the mean and standard deviation of the integrated clusters.

Cluster representatives σ_A, m_A together with a real variable k ($k > 1$) are used to define a confidence area or boundaries for each cluster based on Chebyshev's inequality (Eq. VII.1) [22] as $[m - k\sigma, m + k\sigma]$. For a chosen k, value $1/k^2$ gives the fraction of the values that can lie outside of the interval $[m_A - k\sigma_A, m_A + k\sigma_A]$. Hence, $1 - (1/k^2)$ gives the fraction of values that lie within the stated interval, and $(1 - (1/k^2)) * 100$ gives the percentage.

$$P(\mid X_A - m_A \mid > k\sigma_A) < \frac{1}{k^2} \tag{VII.1}$$

Based on the evaluation of different k values, the experiments in the current study use $k = 6$ or $k = 10$, which implies that a minimum of 97.2% or 99% of the data points lie within the interval $[m_A - k\sigma_A, m_A + k\sigma_A]$, respectively. The value of k is flexible and can be determined based on the requirements of the use case or application where the algorithm is used. As the value of k increases, the model becomes more generalized and categorizes more data points into the cluster. Therefore, for sensitive applications where it is desired to have fewer false positives, a lower k value is preferred. To assign a data point to a cluster, initially, the closest cluster to the data point is identified, then if the data point is within the cluster boundary, it is categorized into that cluster. If the aforementioned condition is not satisfied, the data point is categorized as being non-located (given label -1) in the considered clustering solution.

3.2 Range-based Distance Measure

The range-based distance measure originally introduced in [3] is inspired by the work that studies how the overlap of the interval alternatives' evaluations can be used to express valued preferences among the alternatives [23]. The work [23] proposes a metric for the comparison of information available as a form of intervals similar to interval decision-making. The range-based distance measure is used to determine the closeness between two clusters. It is defined by Eq. VII.2, where A and B are two clusters. In this equation, each cluster, e.g., A, is represented by three entities, denoted by l_A, h_A, and m_A, of its lower boundary, higher boundary, and mean value vectors, respectively. Notice that $d(.,.)$ is the Euclidean distance between two data vectors.

$$R(A,B) = \frac{d(m_A, m_B)}{d(l_A, m_A) + d(m_B, h_B)} \tag{VII.2}$$

Figure 1 illustrates the degree of cluster overlap in a 2D space for different possible ranges of range-based distance measure. Values closer to zero imply that the clusters are similar to each other. The further away from zero, the more distinct they are. For example, one can notice in Figure 1a that the distance between their mean vectors is smaller than the sum of the distance between the mean vector and the lower boundary vector of the first cluster and the distance between the mean vector and the higher boundary vector of the second cluster (i.e., $d(m_A, m_B) < d(l_A, m_A) + d(m_B, h_B)$). The latter implies to merge the two clusters since they are significantly overlapping, i.e., $R(A,B) < \mathcal{T}$, while in Figure 1c $d(m_A, m_B) \geq d(l_A, m_A) + d(m_B, h_B)$, i.e., evidently the overlap between the two clusters is not significant to merge them.

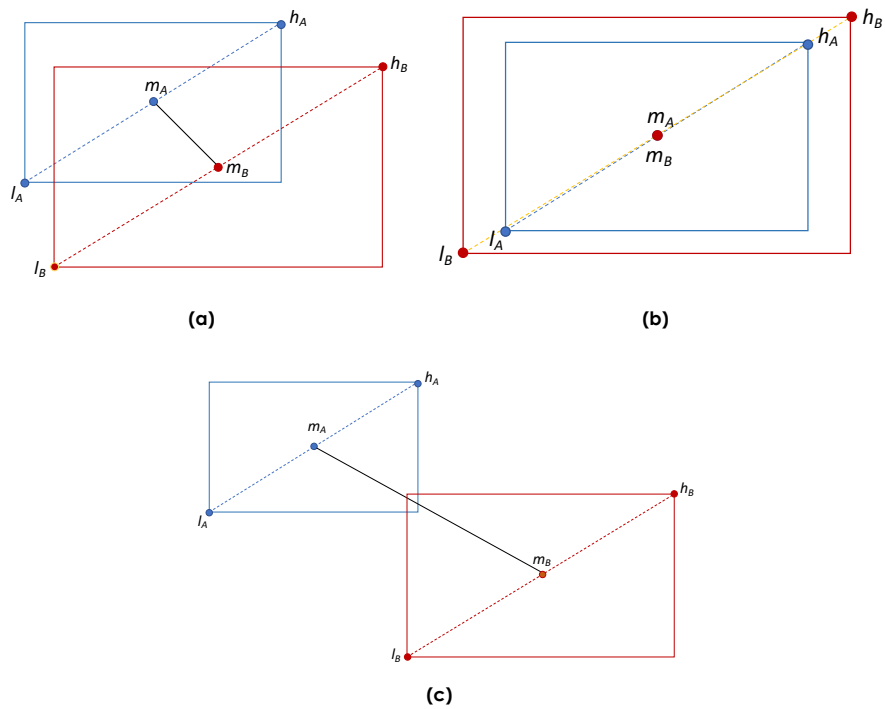


Figure 1: Visualisation of three different scenarios illustrating the calculation of range-based distance metric in 2D space with respect to a specific threshold, $\mathcal{T}$, expressing clusters closeness: (a) $R(A,B) < \mathcal{T}$, i.e., the clusters significantly overlap with each other, and therefore, they are merged; (b) $d(m_A, m_B) = 0$, i.e., one cluster is a subset of the other cluster, and thus they are merged to form a new cluster; (c) $R(A,B) \geq \mathcal{T}$, i.e., the clusters do not significantly overlap with each other, and therefore they are not merged.

3.3 DIBCA++

DIBCA algorithm is initially proposed in [3]. In this work, the algorithm is optimized to be robust to outliers and is evaluated in more experimental scenarios. The core algorithm of DIBCA++ consists of three main modules: *Domain Correlation*, *Domain Integration*, and *Domain Adaptation*, which are elaborated on in this section. The module structure of DIBCA++ is a novelty that is not presented in the original work. The clear discrimination among the main phases of the algorithm and modularization of them can be considered an improvement in comparison with DIBCA. In the latter, those phases are not explicitly outlined and defined in separate algorithms, except for the final integration of the two domain models. In addition to this, the three main modules of DIBCA++ are designed to be robust to outliers.

The main improvements between DIBCA and DIBCA++ can be highlighted as follows:

1. DIBCA uses the low and high attribute value vectors as cluster representatives, i.e., the vectors composing the least and highest values of each attribute of the cluster. These value vectors are also sensitive to outliers. Whereas DIBCA++ uses the mean, standard deviation, and size of the clusters as representatives, thus also making it robust to outliers.

2. DIBCA initially calculates the mean of each of the clusters of source and target domain using its two representatives, the low and high attribute value vectors. DIBCA++ on the other hand, uses one of its cluster representatives (mean) directly. It can be noted that the mean calculated in DIBCA is just an average of the low and high attribute vectors, whereas DIBCA++ uses the mean obtained based on all the data points of the cluster.

3. As the cluster representatives of DIBCA++ are different from those of DIBCA, the integration process done to obtain them for the new integrated clusters varies in both algorithms. The Domain Integration module of DIBCA++ uses the mean, standard deviation, and size of clusters to be integrated to obtain the new mean, standard deviation, and size of the newly integrated cluster. Whereas in DIBCA, the min and max values for each attribute are obtained from the union of all the cluster representatives that need to be integrated to form the new low and high attribute value vectors.

4. DIBCA++ is clearly divided into three main modules, which help in the easy adaptation of the algorithm to new scenarios by replacing some modules with new updated/optimized implementations. In DIBCA, these are not clearly distinguished; only the cross-labeling (part of the Domain Correlation module used in DIBCA++) and the integration phases (Domain Integration module in DIBCA++) are discussed separately.

Domain Correlation

The domain correlation module is used to identify the correlations between the clusters of the source and target models. This is done in two steps: 1) identifying similar clusters across the domains using cross-labeling, and 2) using the range-based distance measure, Eq. VII.2, to find the correlation between similar clusters. In the cross-labeling phase, the source cluster representatives (mean value vector of each cluster) are labeled using the target clustering model, and vice versa, i.e., the target cluster representatives are labeled by the source clustering model. As the general use of the labeling function of a clustering algorithm is to determine the appropriate cluster to which the considered data point belongs, this step helps identify similar clusters in the two domains.

After the cross-labeling step, the mapping (similarity) between the source and target clusters can be categorized into four types: (i) mapping is in both directions, (ii) mapping from source to target (source representatives are labeled using target model), (iii) mapping from target to source (target representatives are labeled using source model), and (iv) no mapping. Clusters in case (i) are identified to be strongly correlated to each other as the similarity is identified in both directions. The similarity obtained in case (ii) is ignored as the mapping is done based on the target model, which is assumed to have less knowledge than the source model; these clusters remain as they are if no other mapping exists. The range-based correlation metric (see Section 3.2) is used to determine the correlation for clusters falling into the case (iii). Clusters in case (iv) are left as they are as these are not identified as similar to any other cluster in the other domain.

Domain Integration

The domain integration module is applied to produce the overall integrated model capturing the specific characteristics of the two domains. In this module, the cluster pairs, among which the correlations are identified from the previous step (the correlation module), are integrated (new mean, standard deviation, and cluster size are obtained for the integrated cluster). These are correlations in case (i), that is, when similarity is identified in both directions and filtered out pairs from case (iii) where range-based distance is less than the chosen threshold ($\mathcal{T} = 0.45$). The threshold of 0.45 is chosen to ensure that the integrated clusters are not too far away from each other. It can be noted that the lower the threshold, the more strict the cluster correlation requirements. If the threshold is set to 0, only clusters with overlapping mean vectors are merged in the integration phase. As a result of the integration of the clusters based on the identified correlations, a clustering model is obtained, referred to as the integrated model, capturing the knowledge from both domains. The integrated clusters presenting knowledge of both domains are referred to as *common* clusters, and the rest containing knowledge of only a single domain are referred to as *private* clusters of the respective domain. The clustering solution composed of common and private clusters from both domains is referred to as an *integrated* clustering solution.

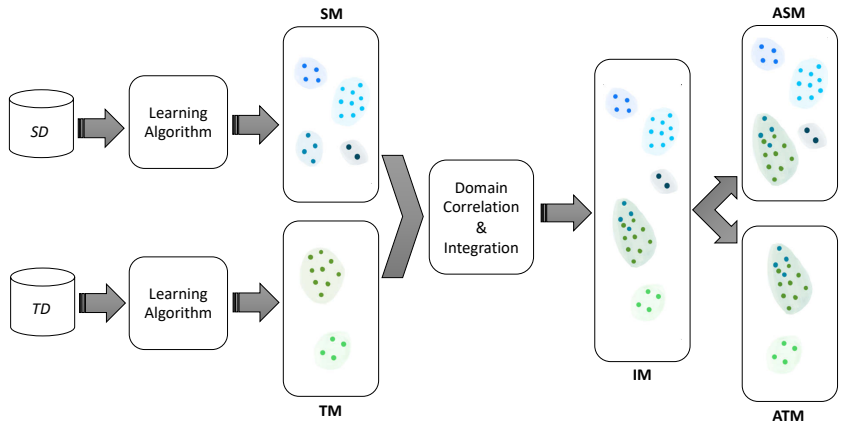


Figure 2: High-level overview of building integrated and adapted models when data from two different domains are available. SD, TD, SM, TM, IM, ASM, and ATM stand for Source Data, Target Data, Source Model, Target Model, Integrated Model, Adapted Source Model, and Adapted Target Model, respectively.

Domain Adaptation

The domain adaptation module uses the composable nature of the overall integrated clustering model to generate two adapted private domain models, one per domain. Each model adapts to the characteristics of its own domain but, at the same time, reflects the identified commonality between the two domains. These models are referred to as adapted models (e.g., source-adapted model and target-adapted model). The adapted models contain the domain-specific private clusters and the common clusters.

A high-level overview of the complete process of integrating knowledge from two different domains and generating both integrated and adapted models is presented in Figure 2. Pseudo-code presenting the overview of DIBCA++ is presented in Algorithm VII.3.

3.4 Learning Algorithm

As stated before, a clustering algorithm is required to be used in combination with DIBCA++. The algorithm introduced in [21] is improved to be robust to outliers and is proposed to be used in the current study.

The major improvements that are worth to be mentioned are as follows:

1. Instead of the low and high attribute value vectors (composed of the least and highest values of each attribute of the cluster), the modified version of the

algorithm uses the mean, standard deviation, and size of the cluster as representatives.

2. Previously, the cluster boundaries were defined by the low and high value vectors, and the cluster's center (mean or representative) is obtained by calculating the mean of these two vectors. This is modified such that the mean vector obtained using all the data points is used as the cluster representative, and the cluster boundaries are obtained by Chebyshev's inequality using the mean and standard deviation of the clusters.

3. The mean vector obtained by using all the cluster data points is used to identify the closest cluster to the data point that needs to be assigned to a cluster. Previously, the mean of low and high value vectors is used for the same purpose.

The pseudo-code of the clustering-based prediction algorithm (learning algorithm) used in DIBCA++ and addressing the above-listed challenges are presented in Algorithms VII.1 and VII.2. Note that this algorithm can be replaced with any other learning algorithm generating clusters represented by the mean, standard deviation, and size of the cluster.

Algorithm VII.1 Learning Algorithm - Fitting

Input: Labeled data set D
1: **for** each class (cluster) in D **do**
2: Calculate m, σ, s.
3: Obtain cluster boundaries ($[m - k\sigma, m + k\sigma]$) using Chebyshev's inequality.
4: **end for**
Output: Representatives (m, σ, s) and Boundaries $[m - k\sigma, m + k\sigma]$ =0

Algorithm VII.2 Learning Algorithm - Predict

Input: List of all cluster boundaries $[m - k\sigma, m + k\sigma]$ and mean vectors (m), data point (d)
1: Identify the closest mean vector to d.
2: **if** all the feature values of d are within the cluster boundary of the selected cluster **then**
3: Assign to the cluster.
4: **else**
5: Assign -1.
6: **end if**
Output: Return the label for the data point.

3.5 Computational Complexity

The proposed algorithm is designed to be resource-efficient and operates using the cluster representatives of mean, standard deviation, and cluster size. In this sec-

Algorithm VII.3 DIBCA++ Algorithm

Input: Clustering models C^1 (source model) and C^2 (target model) with cluster representatives obtained using Alg. VII.1.

Label $m_{C_i^1}, (i = 1, \ldots, n)$ using C^2 (Alg. VII.2)
Label $m_{C_i^2}, (i = 1, \ldots, o)$ using C^1 (Alg. VII.2)
if $(m_{C_i^1} \in C_i^2) \wedge (m_{C_i^2} \in C_i^1)$ **then**
    Correlation$(C_i^2, C_i^1) = 0$
end if
for each $C_i^1 \in C^1$ **do**
    for each $C_i^2 \in C^2$ **do**
        if $(m_{C_i^2} \in C_i^1) \wedge$ (Correlation$(C_i^2, C_i^1) < \mathcal{T} = 0.45$) (Using Eq. VII.2) **then**
            ClusterList.add(C_i^2)
        end if
    end for
    if ClusterList $\neq \emptyset$ **then**
        Integration(C_i^1, ClusterList), calculate new m, σ, s
        Common cluster of C^1 and C^2 is obtained.
    end if
end for
Clusters not integrated are private in their respective domains.

tion, the computational complexity of the main parts of the DIBCA++ algorithm is discussed. The complexity of the Fitting (Algorithm VII.1) and Predict (Algorithm VII.2) parts of the learning algorithm can be approximated to $O(n_c)$ and $O(n_d)$, respectively, where n_c is the number of clusters or groups in the clustering solution and n_d is the number of data points in the test set. As stated before, it can be noted that the used learning algorithm, can be replaced with other suitable learning algorithms, and the computational complexity of this part varies based on the algorithm chosen.

The computational complexity of the correlation module of the algorithm can be approximated to $O(no)$, where n and o are the numbers of clusters in the source and target domains, respectively. The computational complexity of the integration phase can be approximated to $O(n_{corr})$, where n_{corr} is the number of the identified correlations between the source and target domain. In conclusion, the computational complexity of DIBCA++ is similar to that of the initial DIBCA [3].

3.6 Algorithm Explainability

In this section, we discuss the DIBCA++ algorithm with respect to its explainability characteristics. Explainable Artificial Intelligence, also referred to as XAI in the literature, is a field stating the importance and necessity of Artificial Intelligence (AI) and Machine Learning (ML) models to be explainable and transparent. Explaining the

results of AI algorithms is important for their trustworthiness and acceptance, especially if these algorithms are used to make essential decisions in fields like medicine, law, etc. Explainability increases AI acceptance even in areas such as recommender systems where incorrect predictions have less impact on users [24].

On a broader level, explainable AI can be grouped into two categories, post-hoc and ante-hoc [25]. In post-hoc methods, the algorithm itself is not explainable. The algorithm's results are analyzed using various techniques like plots, test cases, etc., to make them explainable. Many new algorithms are also being proposed to be used as an add-on to help explain the results of the main algorithm. On the contrary, ante-hoc algorithms are inherently explainable and inbuilt with the explainability factors from the design stage.

The proposed DIBCA++ algorithm is an ante-hoc algorithm, as different steps of the algorithm can be analyzed and interpreted. Following are a few examples. In the cross-labeling phase, we are able to initially identify the similarity of clusters from the two different domains. Let us assume that the clustering solutions in the source and target domains are called C_1 and C_2, respectively. When a cluster representative in C_1 is labeled as one of the clusters in C_2, it implies the clusters have a resemblance, as the representative of the cluster in C_1 is within the range of the boundaries of the cluster in C_2, and vice-versa which is logical. The correlation between two clusters is further investigated by the various checks done in the algorithm. These checks are transparent, enabling an easy understanding of the final result, i.e., if the clusters should be merged (common clusters) or retained as they are (private clusters in respective domains).

In addition to the above, the output from the different steps of the algorithm can be easily visualized to support further the interpretation and understanding of the obtained final results. Overall, the final results of the algorithm execution are understandable even by humans without any expertise in the field. These results are unique and can also be traced back for reasoning if required.

3.7 Algorithm Applicability

The DIBCA++ proposes a generic integration procedure of two clustering solutions based on cluster or class data models, in which each cluster or class is presented by its boundaries based on Chebyshev's inequality. Only the cluster's mean, standard deviation, and size are used as representatives. The algorithm is applicable to domain adaptation problems in a variety of ML scenarios, namely supervised, unsupervised, and semi-supervised learning scenarios. For example, in an unsupervised scenario, both domain models are presented by clustering solutions in which each cluster models a specific behavioral scenario of the monitored phenomenon. In that way, the integrated model built by the DIBCA++ will be enriched with knowledge about the phenomenon behavior from both domains by identifying and refining scenarios com-

mon for both domains and preserving those unique for each separate domain. Such a domain adaptation model can be suitable for outlier detection tasks, where the monitored phenomenon (e.g., a process or a system) can be described by a set of clusters presenting its ordinary behavioral patterns. Everything that does not fit into any cluster will be interpreted as deviating behavior. Hence, the DIBCA++ can be applied to adapt the outlier detection model to new environmental or contextual conditions.

In the case of a supervised learning setup, both domain data models consist of labeled classes, i.e., each model presents the concepts specific to each domain, respectively. As a result of the integration procedure, the overall model will learn which concepts are common for both domains and also identify the private ones for each domain. This newly gained information will contribute to a better understanding of the monitored phenomenon and can eventually be used for further improvement of domain models. For example, domain private concepts can be extracted from the integrated data model. This property of the DIBCA++ makes it applicable, for example, for the customization of patient/user models in smart monitoring applications.

Finally, the DIBCA++ can be used in partially (semi-)supervised scenarios, where only the source model is based on labeled data, while the target model contains just non-labeled clusters. One use case of this scenario is the automatic data labeling task. The domain correlation procedure of the algorithm can be used to label those target concepts that have been identified to be common with the source model. The potential of the proposed algorithm in labeling scenarios has already been studied and demonstrated in the initial work on DIBCA [3].

In addition to the above-discussed ML scenarios where DIBCA++ can be used, there are some limitations that are worth mentioning. More specifically, the learning algorithm used in the current study applies Chebyshev's inequality to define the cluster boundaries. Even though this inequality holds true for most probability distributions, there might be cases when this is invalid. For example, if we consider normal distribution, which is one of the most common probability distributions, the majority (approx. 99.7%) of the data points lie within three times the standard deviation from the mean. Whereas for Chebyshev's inequality, six and ten times the standard deviation from the mean cover approximately 92.7% and 99% of the data points, respectively. If the data follows a normal distribution, Chebyshev's inequality might produce clusters with higher boundaries than required; therefore, more attention should be paid to such cases. This can lead to mislabeling of data, especially when the clusters are closer to each other. Moreover, the learning algorithm used in conjunction with DIBCA++ should be able to represent the clusters by the mean, standard deviation, and size of the cluster.

4 Data Sets

Two different data sets, one from the smart logistics use case and the other from the HAR data set, are used in the study.

Data from the smart logistics use case is provided by our industry partner. The Global Navigation Satellite System (GNSS) is used by the trackers to report their location. These tracking devices have limited memory and computational capacity, which needs to be considered during their operation. Trying to update the location continuously requires a significant amount of energy, which is wasted if the GNSS signal is not strong or unavailable when accessed. This motivates the need to detect and optimize when the tracker should try to access the GNSS location. The used data set contains information about five tracking devices. For each tracking device, signals from the Long Term Evolution (LTE) base stations, the number of available cells with radio signals, geographical location, etc., are collected.

DaLiAc (Daily Life Activities), a HAR data set generated during the study [26], is used to evaluate the algorithm's potential in the domain adaptation task of recognizing and mapping similar activities from different users. HAR of different users can be categorized as a domain adaptation task as each user performs an activity in different ways, leading to having data with different distributions. Building new clustering solutions covering all the activities considered for each user requires a large amount of data, which is difficult to obtain. Domain adaptation is useful in such situations where knowledge from one user model is used for building HAR models for others. The data set contains information about 19 participants (11 male, 8 female) aged between 18 and 55 performing 13 daily activities. Data consists of the accelerometer and gyroscope readings collected using four sensors placed on the right hip, chest, right wrist, and left ankle of each participant.

5 Evaluation Measures

For the smart logistics use case, Adjusted Mutual Information is used, whereas, for the DaLiAc data set, two different extrinsic cluster evaluation metrics, namely Adjusted Rand Index and Adjusted Mutual Information are considered. The implemented versions of the evaluation metrics by scikit-learn [27] are used.

5.1 Adjusted Rand Index

Adjusted Rand Index (ARI) [28] is the adjusted for chance version of the Rand Index (RI) [29]. RI calculates the similarity between two clustering solutions by counting the number of pairs of samples that are categorized as the same or different in the predicted clustering as opposed to the ground truth. ARI is corrected/adjusted for a chance so that random results produce a score close to 0. ARI is then obtained using

the formula stated in Eq. VII.3.

$$ARI = \frac{RI - ExpectedRI}{max(RI) - ExpectedRI}, \tag{VII.3}$$

where RI for samples i and j is $\sum_{ij} \binom{n_{ij}}{2}$,

$$\max(RI) = \frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right], \text{ and}$$

$ExpectedRI = \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}$, where n is the number of elements, a_i and b_j are the sum of the pair of samples i and j, respectively.

5.2 Adjusted Mutual Information

Mutual Information (MI) is used to compare two clustering solutions. The obtained value is, however, higher if the number of clusters is high, whether more information is actually shared between the clusters or not. Adjusted Mutual Information (AMI) [30, 31] is a corrected for chance version of MI. Eq. VII.4 is used to calculate AMI [27].

$$AMI(U, V) = \frac{[MI - E(MI)]}{[avg(H(U), H(V)) - E(MI)]} \tag{VII.4}$$

In Eq. VII.4, MI represents $MI(U, V)$, where U and V are the two clustering solutions, $H(.)$ is the entropy of the considered clustering solution (either U or V in this case), $E(MI(U, V))$ is the expected mutual information between two random clustering solutions U and V.

6 Experiments

The section initially presents an overview of the preprocessing steps done on each of the considered data sets. This is followed by explaining the different experiments conducted in the study [1].

6.1 Data Pre-processing

Before conducting the experiments, a few steps of data processing are done on the data sets to make them ready to be used in the experiments.

[1] Access to the repository with experiments conducted on the public data set (DaLiAc, HAR) can be provided upon request.

For the first set of experiments, we used the data set from a smart logistics use case provided by one of our industrial partners. Data used in this study is collected from five different tracker devices. Statistical values of the cell signals, i.e., mean, standard deviation, min, and max, are obtained for each instance, which along with the signal strengths, are used as a feature set.

The second set of experiments is conducted on the DaLiAc data set initially containing 24 attributes, with six attributes (3D accelerometer and gyroscope readings) obtained from each sensor placed on different body locations. Each of the 3D values is aggregated using $a = \sqrt{x^2 + y^2 + z^2}$ to obtain one value like in [32, 33], thus giving in a total of eight attributes.

To obtain stable results, each experiment is conducted three and five times on each data set for smart logistics and HAR use cases, respectively. Since the smart logistics and DaLiAc data sets contain time series data, TimeSeriesSplit from the scikit-learn library [27] is used to split the data into five different folds of train and test data sets. Only three folds are used for the smart logistics use case, as low performance was observed in the initial folds where less data is used. Note that training data is incrementally added to each fold in TimeSeriesSplit. In the next step, data from each fold is standardized using z-score. This is done using the StandardScaler module from scikit-learn. Data standardization of both source and target data is done using the scale obtained from the data of the source domain. Mathematically, it can be represented as $z = (x - \mu)/\sigma$, where x is the data point that needs to be standardized, μ is the mean of the source training data, and σ is the standard deviation of the source training data. Unless explicitly stated, all the results presented are aggregated evaluations obtained from experiments conducted on different folds (three for smart logistics and five for HAR data sets) of the time series cross-validation performed.

6.2 Experiments on Smart Logistics Use Case

We study and analyze the effect of similarity between the two domains to that of the performance of DIBCA++ with the experiments discussed in this section. The similarity between different domains (data from different devices) is evaluated using the Optimal Transportation Dataset Distance (OTDD) [34]. The OTDD is based on optimal transport and also incorporates label information in the calculation. The OTDD is designed to evaluate the transferability of ML models between data sets, which is useful information to have for tasks like transfer learning and domain adaptation. Figure 3 presents a heat map showcasing the data set distances between different pairs of devices. When calculating the OTDD, a sampled data set of $7,500$ instanced with random seed '0' is used. Values closer to zero represent that the data sets are similar to each other. It is also interesting to observe that the results of similarity based on OTDD are comparable to the results obtained in [21], where the similarity between the same data sets is calculated based on the RI scores.

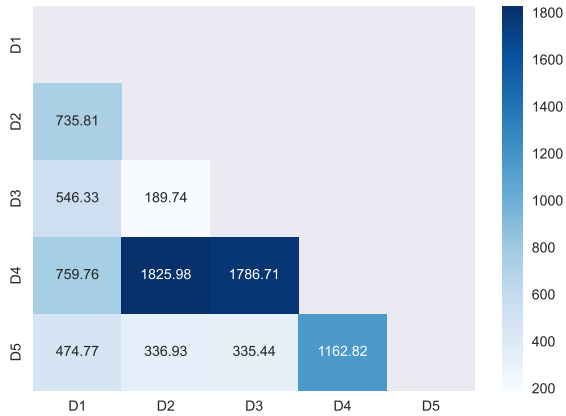


Figure 3: Heat-map presenting the similarity between data obtained from different devices (D) based on the Optimal Transport Dataset Distance (OTDD). Low values represent that the data sets are similar to each other.

Two experiments, A1 and A2, are conducted to understand and analyze how the similarity of the data sets influences the performance of the different clustering models produced by DIBCA++. Evaluation measure AMI (see Section 5) is used to showcase the performance of the clustering solutions obtained for both A1 and A2. Note that we have also done an evaluation using ARI, but since there is no significant difference in the results generated by both evaluation metrics on the various models produced by DIBCA++, only AMI results are reported in the paper. In addition, the current work treats the GNSS activation problem differently from the setting applied in [3]. Namely, in the previous study, we have considered the prediction of the availability of GPS signal as a one-class classification problem, i.e., a version of the ISM algorithm published in [21] is applied to the GPS annotated data to learn the normal system behavior and then recognizes everything different from it, such as when there is no GPS signal coverage. In the current study, we use the data from both classes and model each class by seven categories. Each category is presented by a unique cluster containing all the data points associated with the availability of the number of cells with signal strengths (ranging from 1 to 7 available cells based on the LTE coverage). As a result, the data will be presented in 14 categories in total. In both experiments A1 and A2, these 14 categories are used as true labels, based on which the initial clustering solutions are obtained using the learning algorithm (Section 3.4, Algorithms VII.1 and VII.2). Only 70% of the training data of the target is used for training in both experiments (A1 and A2). This is done to generate a scenario where sufficient target data is unavailable.

In experiment A1, all ten possible combinations of two-pair devices are considered to perform the experiments. For each pair of devices, the device with a higher number of instances is considered as the source (as more data implies more knowledge) domain, and the one with less data is the target domain. Figure 4 presents the

results obtained.

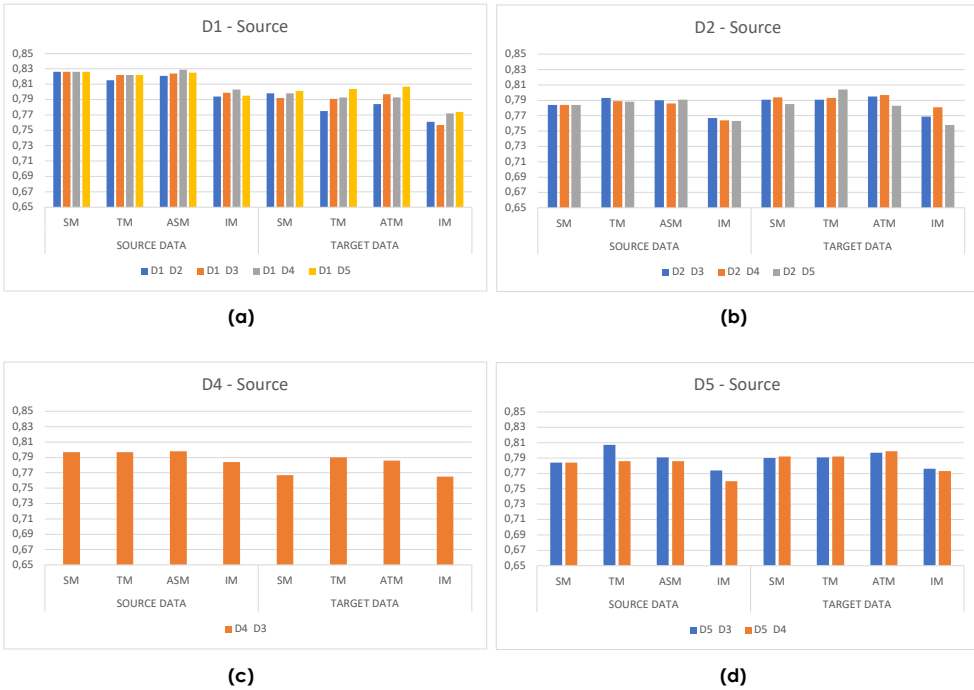
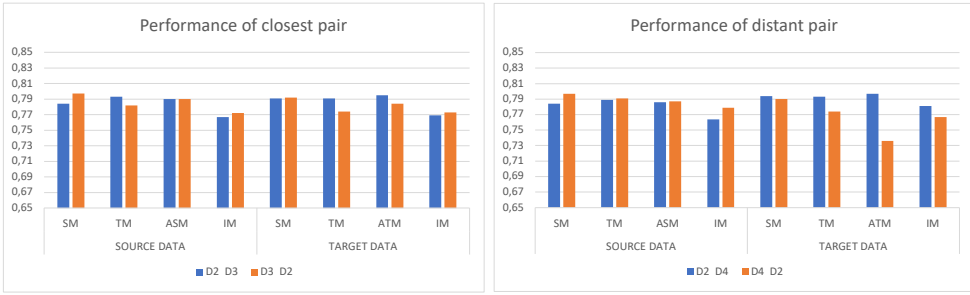


Figure 4: Experiment A1-DIBCA++: Visualisation and comparison of the performance (all the values are average of 3 folds) of the DIBCA++ models for different combinations of source and target data sets with respect to AMI. Each plot depicts the performance of the models when a different device is in the role of the source.

In experiment A2, the closest and the distant pair of devices based on the similarity calculated are considered for comparison. Unlike A1, in this experiment, for each pair of devices, both are alternatively considered as the source and the other as the target, thus giving 4 different combinations of source and target (2 for each closest and distant pair). The results of this experiment are presented in Figure 5.

6.3 Experiments on HAR Use Case

The potential of the DIBCA++ algorithm in the adaptation and personalization of users' human activity clustering models is showcased in this experiment. Since the number of ways of selecting source and target from 19 participants would result in too many combinations, 4 male (users 8, 11, 12, and 15) and 4 female (users 10, 13, 14, 18), users are randomly chosen from the available participants on which the experiments are conducted. Since men and women have different physical capabilities which can impact the way they perform the activities, we chose to perform the experiments on men and women separately. Further information about the users used in the experiments of this study is stated in Table 2.



(a) (b)

Figure 5: Experiment A2-DIBCA++: Performance comparison of the closet and distinct pairs of devices based on the similarity of the data sets. The notation Dx-Dy represents that Dx is the source and Dy is the target. Each plot presents the performance of different models produced by DIBCA++ on (a) D2-D3 and D3-D2, closest pair based on similarity; (b) D2-D4 and D4-D2, distinct pair based on similarity.

Table 2: Information about users used in the experiments B1 and B2

User	Sex	age	Instances	Height	Weight	Handedness
U8	m	29	230,381	180	90	right
U10	f	27	240,494	169	59	right
U11	m	27	239,020	178	72	right
U12	m	18	253,317	175	70	right
U13	f	21	245,041	177	86	right
U14	f	22	243,036	180	55	left
U15	m	27	241,024	196	95	right
U18	f	24	228,697	158	54	right

The considered users can be grouped into 12 pairs such that one user's activities are considered as a source domain and the other as a target domain (6 male pairs and 6 female pairs). Two different types of experiments (denoted as B1 and B2) are conducted.

Experiment B1 is conducted to showcase the potential of the algorithm in domain adaptation tasks. In this experiment, for each pair of users, the user with a higher number of instances is considered as the source domain, and the other user is the target. Initial clustering in each of the domains is done using the true labels, and the learning algorithm stated in Section 3.4 (see Algorithm VII.1). This, along with Algorithm VII.2, is used to assign new data points to one of the clusters. The results of this experiment are presented in Tables 3 and 4 with regard to ARI and AMI, respectively.

In experiment B2, for each pair of users, data from both source and target domains are combined to obtain one dataset. A clustering model on the whole combined data is built and evaluated using the learning algorithm (Algorithms VII.1 and VII.2). Experimental results of B2 are presented in Table 5. Results of Experiment B2 when compared with B1 (SD-ASM, SD-IM, TD-ATM, and TD-IM columns of Tables 3 and 4), can be used to understand and demonstrate the importance of personalized user models.

Table 3: Experiment B1-DIBCA++: Results obtained on DaLiAc data set with respect to Adjusted Rand Index using DIBCA++; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM-Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model. Bold font in black represents the best model in the row, cyan color represents the better performance of IM in comparison to the corresponding SM or TM. Blue and red represent the better performance of ASM over SM and ATM over TM, respectively.

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.248	0.222	**0.271**	**0.271**	0.201	0.261	0.248	0.233
U11	U8	0.280	0.230	**0.296**	**0.292**	0.230	0.272	**0.302**	0.256
U12	U8	**0.299**	0.236	0.288	0.288	0.236	0.260	**0.261**	0.257
U12	U11	**0.299**	0.210	0.284	0.267	0.220	0.276	**0.292**	**0.280**
U12	U15	0.299	0.220	**0.303**	**0.301**	0.250	0.264	**0.292**	**0.282**
U13	U10	**0.289**	0.196	0.274	0.274	0.235	0.246	0.237	**0.251**
U13	U14	0.289	0.255	**0.297**	**0.296**	0.250	0.283	0.262	0.246
U13	U18	**0.289**	0.232	0.278	0.278	0.266	0.265	**0.272**	0.264
U14	U10	**0.300**	0.186	0.287	0.279	0.195	0.262	**0.273**	**0.282**
U14	U18	**0.300**	0.227	0.278	0.271	0.252	0.277	**0.293**	0.268
U15	U8	0.274	0.227	**0.277**	**0.277**	0.215	**0.291**	0.290	0.256
U15	U11	0.274	0.164	0.270	0.267	0.245	0.275	**0.285**	0.259

Table 4: Experiment B1-DIBCA++: Results obtained on DaLiAc data set with respect to Adjusted Mutual Information using DIBCA++; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM-Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model. Bold font represents the best model in the row, cyan color represents the better performance of IM in comparison to the corresponding SM or TM. Blue and red represent the better performance of ASM over SM and ATM over TM, respectively.

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.441	0.421	**0.451**	**0.451**	0.400	0.433	0.434	0.417
U11	U8	0.463	0.406	**0.474**	**0.472**	0.403	0.445	0.452	0.439
U12	U8	**0.459**	0.426	0.447	0.447	0.414	0.432	**0.433**	0.430
U12	U11	0.459	0.391	0.450	0.445	0.386	0.460	**0.471**	0.459
U12	U15	0.459	0.407	**0.461**	**0.463**	0.444	**0.475**	0.473	0.467
U13	U10	0.441	0.387	**0.443**	**0.443**	0.435	**0.444**	0.428	0.440
U13	U14	0.441	0.405	**0.441**	0.440	0.446	**0.457**	0.443	0.448
U13	U18	0.441	0.413	**0.441**	**0.441**	0.435	0.435	**0.438**	**0.436**
U14	U10	**0.475**	0.392	0.464	0.467	0.387	0.459	**0.459**	**0.459**
U14	U18	**0.475**	0.443	0.467	**0.475**	0.431	0.449	**0.458**	**0.452**
U15	U8	**0.480**	0.425	0.467	0.466	0.417	0.463	0.456	0.446
U15	U11	**0.480**	0.377	0.477	0.476	0.432	0.466	0.454	0.444

Table 5: Experiment B2: Results obtained on DaLiAc data set when two data sets are combined. Red and blue represent the best and worst performance values, respectively.

Users	ARI	AMI
U10 + U18	**0.223**	0.404
U11 + U8	0.246	0.410
U12 + U8	0.276	0.430
U12 + U11	0.278	0.436
U12 + U15	**0.286**	**0.440**
U13 + U10	0.237	**0.401**
U13 + U14	0.277	0.428
U13 + U18	0.250	0.405
U14 + U10	0.262	0.434
U14 + U18	0.255	0.417
U15 + U8	0.231	0.404
U15 + U11	0.262	0.434

6.4 Experiments with DIBCA

The performance of DIBCA++ is also compared with its predecessor, i.e., DIBCA. For this purpose, two experimental scenarios, A1 and B1, are exactly replicated on the smart logistics and HAR use cases using DIBCA. Tables 6 and 7 present the experimental results of A1 on smart logistics data set with regard to AMI using both

Table 6: Experiment A1-DIBCA: Results obtained on smart logistics data set with respect to Adjusted Mutual Information using DIBCA; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM-Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model.

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
D1	D2	0.792	0.810	0.787	0.787	0.755	0.764	0.760	0.757
D1	D3	0.792	0.806	0.791	0.791	0.757	0.760	0.762	0.760
D1	D4	0.792	0.789	0.788	0.788	0.778	0.766	0.811	0.787
D1	D5	0.792	0.791	0.787	0.787	0.764	0.772	0.764	0.763
D2	D3	0.768	0.756	0.768	0.768	0.781	0.763	0.780	0.780
D2	D4	0.768	0.746	0.771	0.771	0.786	0.766	0.791	0.789
D2	D5	0.768	0.742	0.763	0.763	0.785	0.772	0.779	0.779
D4	D3	0.784	0.784	0.792	0.792	0.738	0.762	0.775	0.775
D5	D3	0.763	0.766	0.775	0.775	0.760	0.759	0.766	0.766
D5	D4	0.763	0.780	0.753	0.753	0.787	0.767	0.794	0.780

Table 7: Experiment A1-DIBCA++: Results obtained on smart logistics data set with respect to Adjusted Mutual Information using DIBCA++; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM-Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model.

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
D1	D2	0.826	0.815	0.821	0.794	0.798	0.775	0.784	0.761
D1	D3	0.826	0.822	0.824	0.799	0.792	0.791	0.797	0.757
D1	D4	0.826	0.822	0.829	0.803	0.798	0.793	0.793	0.772
D1	D5	0.826	0.822	0.825	0.795	0.801	0.804	0.807	0.774
D2	D3	0.784	0.793	0.790	0.767	0.791	0.791	0.795	0.769
D2	D4	0.784	0.789	0.786	0.764	0.794	0.793	0.797	0.781
D2	D5	0.784	0.788	0.791	0.763	0.785	0.804	0.783	0.758
D4	D3	0.797	0.797	0.798	0.784	0.767	0.790	0.786	0.765
D5	D3	0.784	0.807	0.791	0.774	0.790	0.791	0.797	0.776
D5	D4	0.784	0.786	0.786	0.760	0.792	0.792	0.799	0.773

Table 8: Experiment B1-DIBCA: Results obtained on DaLiAc data set with respect to Adjusted Rand Index using DIBCA; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM- Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model.

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.107	0.064	0.109	0.090	-0.002	0.070	0.080	-0.002
U11	U8	0.030	0.103	0.027	0.026	-0.000	0.134	0.131	-0.000
U12	U8	0.047	0.079	0.046	0.048	0.021	0.135	0.129	0.028
U12	U11	0.047	0.001	0.050	0.027	-0.007	0.030	0.036	0.013
U12	U15	0.047	0.049	0.047	0.051	-0.006	0.101	0.124	0.046
U13	U10	0.103	0.003	0.083	0.001	0.072	0.118	0.101	0.097
U13	U14	0.103	0.039	0.104	0.033	0.037	0.089	0.091	0.091
U13	U18	0.103	0.070	0.118	0.105	0.099	0.072	0.109	0.111
U14	U10	0.091	0.004	0.087	0.093	0.041	0.115	0.178	0.072
U14	U18	0.091	0.079	0.091	0.092	0.038	0.074	0.072	0.041
U15	U8	0.097	0.130	0.097	0.099	0.074	0.136	0.145	0.080
U15	U11	0.097	-0.015	0.122	0.122	0.051	0.030	0.099	0.101

DIBCA and DIBCA++, respectively. It can be noted that the contents of Table 7 have already been presented as a figure (Figure 4), but in order to facilitate easy comparison between DIBCA and DIBCA++, we have decided to present the results of both the algorithms using tables. Tables 8 and 9 present the experimental results of B1 using DIBCA on the HAR data set based on ARI and AMI, respectively.

Table 9: Experiment B1-DIBCA: Results obtained on DaLiAc data set with respect to Adjusted Mutual Information using DIBCA; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM- Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model.

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.353	0.263	0.351	0.330	0.111	0.292	0.286	0.122
U11	U8	0.250	0.224	0.245	0.242	0.119	0.282	0.290	0.121
U12	U8	0.262	0.208	0.251	0.258	0.175	0.282	0.285	0.195
U12	U11	0.262	0.107	0.260	0.214	0.116	0.249	0.251	0.183
U12	U15	0.262	0.233	0.256	0.273	0.110	0.322	0.333	0.221
U13	U10	0.270	0.075	0.250	0.123	0.235	0.352	0.318	0.326
U13	U14	0.270	0.168	0.270	0.206	0.156	0.304	0.295	0.310
U13	U18	0.270	0.233	0.277	0.271	0.272	0.297	0.297	0.289
U14	U10	0.321	0.065	0.300	0.292	0.201	0.351	0.395	0.244
U14	U18	0.321	0.254	0.315	0.320	0.227	0.304	0.296	0.252
U15	U8	0.317	0.282	0.318	0.319	0.291	0.285	0.300	0.300
U15	U11	0.317	0.113	0.314	0.320	0.246	0.254	0.315	0.324

7 Result Analysis and Discussion

Discussion and analysis of the experimental results are presented in this section. In addition, the algorithm's explainability aspect is also showcased on a few selected instances of different experiments in Section 7.4. The abbreviations Source Data (SD), Target Data (TD), Source Model (SM), Target Model (TM), Adapted Source Model (ASM), Adapted Target Model (ATM), and Integrated Model (IM) are used in this section.

7.1 Smart Logistics

A heat map with similarities calculated for each pair of different data sets is displayed in Figure 3. In experiment A1, the performance of the models produced by the DIBCA++ on the different pairs of devices is analyzed in light of these calculated similarities. Figure 4 presents the built models' performance on different source and target device pairs. From all the sub-figures, it can be observed that the performance of ASM when D1 is considered as the source is higher compared to all the other scenarios. It can be backed by the known fact that D1 contains richer data than its counterparts. Based on the calculated similarities between pairs of devices, when D1 is selected as the source, pairs D1-D3 and D1-D5 can be considered similar, D1-D2 and D1-D4 to be distinct from each other. From Figure 4a, it can be observed that the performance of the ASM and ATM is directly proportional to the similarity of the data sets (i.e., higher the similarity between the two data sets (lower OTDD), higher the performance), except for D1-D4 pair concerning ASM. Whereas for scenarios when devices D2 and D5 act as a source (D2-D3, D2-D5, and D5-D3 are considered similar; D2-D4 and D5-D4 are distinct), as shown in Figures 4b and 4d, the performance of ASM is directly proportional to similarity, and the performance of ATM is inversely proportional. Since there is just one pair when D4 is the source (Figure 4c), it is difficult to interpret how the similarity and the performance of ASM and ATM are correlated. Overall, it is observed that the performance of the ASM

and ATM models produced by DIBCA++ performed better 70% and 80% of the time when compared to that of SM and TM, respectively (see Figure 4 or Table 7). These results on the studied use case show the potential of DIBCA++ to transfer knowledge between domains for improving performance. Note that all the 3 cases (30%) when ASM performance did not exceed the performance of SM are when D1 is considered as the source. It could be because of the fact that the SM build using D1 is already rich and could not gain much from the other domain.

In experiment A2, we specially inspect two pairs of devices, the closest (similar) and the distant (dissimilar) ones; see Figures 5a and 5b, respectively. D2 and D3 are the two closest devices, and the experiments on both D2-D3 and D3-D2 pairs show better performance of ATM than that of TM, on TD. In the case of D2-D3, the ATM even outperforms SM on TD. So, when two devices are very similar, it does not matter which device is in the role of the source. However, this is not observed in the case of the two most distant devices, which are D2 and D4. The data collected by these devices presumably cover very different GNSS coverage scenarios. The experiments demonstrate strange behavior for D4-D2 on TD; namely, although the performance of SM and TM, ASM on SD are comparable, this does not repeat on TD, while in the case of D2-D4, the ATM outperforms all other models on both source and target data. The above might be due not only to the fact that D2 and D4 have very different data sets (the highest OTDD value) but also because D2 has much richer data than D4. It is interesting to note that D2 has more instances in the data set than D4.

Evidently, it can be stated that the target model acquires knowledge from the source and improves its performance. When the knowledge in the source domain is very similar to that in the target, that is, there is no additional knowledge to be transferred, and the benefit of creating IM or ATM is not significant. In such a scenario, using the SM directly on the target domain data would be better. In addition to the above, we have noticed that the performance of the adapted (source or target) model is better than that of the integrated model in all the studied combinations, i.e., it is always recommendable to use the customized model.

7.2 Human Activity Recognition

Tables 3 and 4 summarized the results of DIBCA++ on the DaLiAc data set in experimental scenario B1. They showcase the results of the comparison of the performance of the three models, namely IM, ASM, and ATM, produced by the proposed algorithm to that of the initial SM and TM models used on the SD and TD in terms of ARI and AMI measures, respectively. In that way, for each of the 12 combinations of users studied, we compare eight different scenarios for each metric, i.e., 96 different scenarios per metric in total.

With respect to the ARI evaluation (see Table 3), one can notice that the IM

produces better results in comparison to the two initial SM and TM in 5 and 4 (colored in cyan) of all the combinations, respectively. It is interesting, however, to observe that the same number is achieved by the ASM in comparison with the SM used on the source data (the scores in blue) while the ATM is better than the TM applied on the target data (the scores in red) in twice more combinations, i.e., 8. Evidently, the performance of the integrated model is further improved by building a personalized adapted model for each domain. In addition, we can see that using the SM on the target data (see column TD-SM) produces worse results than the initial TM as well as than the ATM and IM. This clearly shows that our domain integration procedure leads to richer adapted models, with benefits for both parties involved. It can also be observed that the SD-SM scenario performs better than the rest studied in 6 different combinations highlighted in bold black font.

The same 96 scenarios discussed above are also evaluated in terms of AMI measure, and the obtained results are presented in Table 4. These confirm once more the findings extracted by analyzing Table 3. It can be observed that the IM has performed better in 6 and 3 cases when compared against SM and TM on source and target data, respectively. Similarly to the results corresponding to the ARI, we can see that ASM performs better than SM the same number of times as IM, and ATM performs better than TM in 5 combinations, which is almost twice the times IM performs better than TM.

Table 5 reports the evaluation results of the clustering models built directly on the combined data of a pair of users (domains) with respect to the two evaluation measures, ARI and AMI (Experiment B2). This experiment showcases the need for personalized user models for higher performance. When the results of combined data sets of two users are compared to the scenarios where source and target domain models are built separately (Tables 3 and 4, columns SD-SM, SD-ASM, SD-IM, TD-TM, TD-ATM and TD-IM) drop in performance of the combined models can be observed in most cases. The evaluation results of the scenarios where TM is used on SD or SM on TD (columns SD-TM and TD-SM of Tables 3 and 4) are even worse, but it is expected to have the least performance in this case as the model's knowledge and the data used are from different domains.

The clustering model produced on the combined data of users U12 and U15 (see row U12+U15) outperforms all other pairs with respect to all the used measures, while the models of U10+U18 and U13+U10 are the worst-performing ones since they have generated the lowest results at least for a single evaluation measure. It is interesting to notice that the best-performing pair of models is obtained from males, as users in this pair may have performed most of the activities similarly, while the two low-performing models are based on data from female users, which might have occurred due to differences in how they perform the activities. It can also be observed that the U12-U15 pair is among one of the best-performing pairs in Tables 3 and 4.

7.3 Comparison with DIBCA

In this sub-section, the performance of DIBCA++ is compared to DIBCA with respect to the experimental scenarios A1 and B1. Tables 6 and 7 present the performance of DIBCA and DIBCA++ in experimental setup A1 on smart logistics use case based on AMI. It can be observed that DIBCA++ has a better performance in 82.5% of cases, and in 2.5% occasions, both algorithms performed similarly. In only 15% of the cases, DIBCA has shown better performance, and it can be noted that the margin of difference is also very small in some of these cases.

Tables 8 and 9 present the summarised results of DIBCA on the HAR data set with respect to ARI and AMI. On comparing these results with those obtained by DIBCA++ (Tables 8 and 3; Tables 9 and 4), it can be clearly seen that DIBCA++ has outperformed DIBCA in all the cases to a great extent. This might be because of more outliers recorded when the sensors capture human activities, as different people perform the activities differently.

7.4 Explainability

In this section, we specifically study the explainability characteristics of DIBCA++. We initially select a pair of devices from the smart logistics use case to inspect how the correlations identified by the domain correlation module of DIBCA++ can be used to explain the algorithm performance on the selected pair. The performance of ATM on the D2-D5 pair is the worst one (see Figure 4 or Table 7), which has provoked our curiosity to check out how the clusters of the D2 and D5 models are correlated (last fold of experiment using all the data is used for illustration), see Figure 6. Interestingly, there is only one mismatch, namely the connection between cluster 8 from the D2 model and cluster 1 from the D5 model. The other connections are correct, but they correlate only with half of the clusters; the other half will exist as private clusters in the integrated model (IM) and their respective adapted models (ASM or ATM). In addition, a common cluster that unites cluster 8 from D2 and clusters 1 and 8 from D5 will be created in IM, ASM, and ATM. This cluster will contain data points from both classes, namely with and without GPS signals, which certainly will affect the ATM performance. However, the identified connection between cluster 8 from the D2 model and cluster 1 from the D5 model can also provide us with new information about the scenarios presented in the D2 and D5 models. This might be a hint that the scenarios modeled by these two clusters, in fact, present a transition between the two main classes, i.e., from GPS coverage to no GPS coverage and vice versa.

Next, two pairs of users from the HAR use case are chosen to further analyze and understand the results with respect to the explainability aspect of the DIBCA++. One best-performing and one least-performing pairs are picked up from Table 5 (Experiment B2) to understand how the source and target models of these pairs are inte-

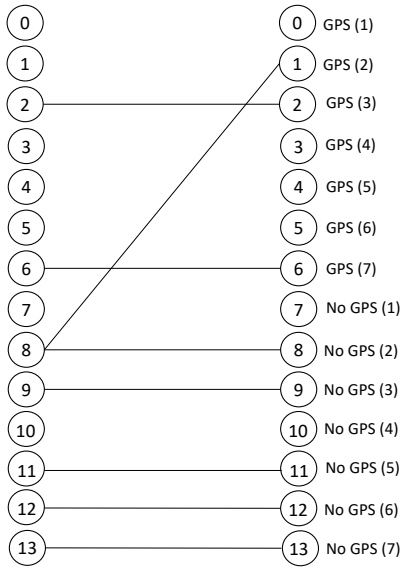


Figure 6: Visualisation of the correlations identified by the domain correlation module of the DIBCA++ on a selected pair of devices, D2-D5, one of the pairs on which ATM has the worst performance. In the cluster label '$A(x)$', A indicates whether the cluster corresponds to one of the two main classes, i.e. available GPS signal or missing, while x ranges from 1 to 7 representing the available cells based on the LTE coverage, i.e. it denotes a specific category (scenario) in each one of the two main classes.

grated (Experiment B1). User pairs U12-U15 and U10-U18 are chosen to be further analyzed. Note that U10-U18 is chosen over U13-U10 as for the first pair, the value of AMI is the second worst performing, whereas, for the second pair, the ARI value is the third worst performing. The performance of the selected pairs U12-U15 and U10-U18 is similar (they are one of the best and worst performing pairs) in experimental scenario B1 when the users' individual models are integrated using DIBCA++ (see Tables 3 and 4). Hence, they can be used to visualize the internal steps of the algorithm. For both the pairs, the fourth fold, i.e., the fold that uses all the data of the data set, is used to illustrate the results.

Figure 7 presents the correlations between these pairs identified by the domain correlation module of DIBCA++. It can be observed that the correlations extracted for U12-U15 are among similar clusters or activities. While in the case of the U10-U18, there are more mismatches when compared to U12-U15. This explains the good and bad performance of U12-U15 and U10-U18 respectively in experiment B1 (Tables 3 and 4). It can also be observed that in the case of U10-U18, the ASM produces higher improvement in comparison with SM than in the case of U12-U15 (see columns SD-SM and SD-ASM). In general, the identified correlations indirectly state that U10 and U18 have differences in the way they perform the same activities, and hence, this negatively affects the model performance when their data are combined together (Table 5) as personalization is lost. The case is the opposite when the data of U12 and U15 are integrated.

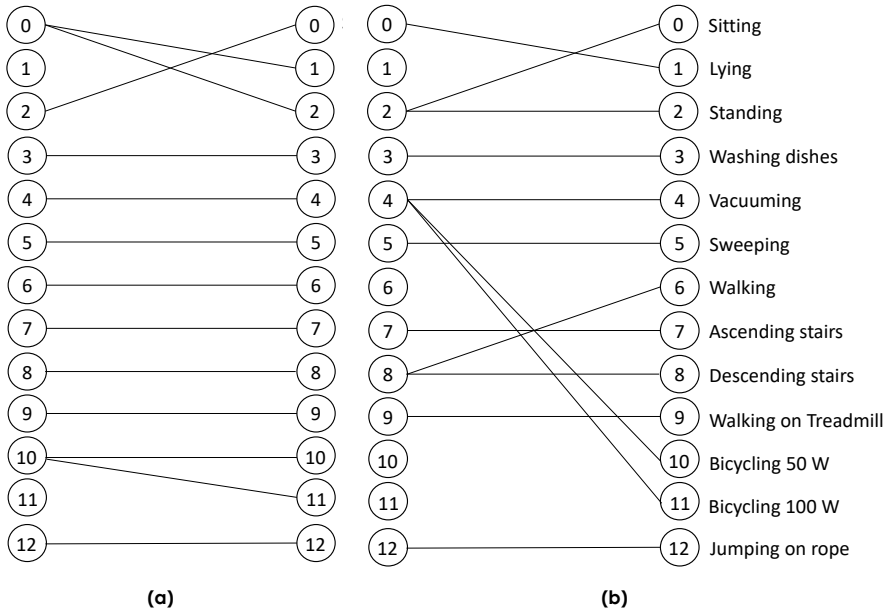


Figure 7: Visualisation of the correlations identified by the domain correlation module of the DIBCA++ on the selected two pairs of users. The correlated clusters are merged by the domain integration module of DIBCA++. (a) Correlations identified for user pair U12-U15, the best-performing pair; (b) Correlations identified for user pair U10-U18, one of the worst performing pairs.

In order to further understand the reason behind the mismatches in the correlations, we visualize one more layer of how DIBCA++ works, and a random pair of mismatched correlations is selected, i.e., cluster 0 of U12 (left column), U15 (right column), and cluster 1 of U15 (see Figure 7a). Figure 8 presents the mean vector pairs of both clusters. Note that mean vectors are used to find the closest cluster (see Algorithm VII.2). Figure 8a depicts the mean vectors of cluster 0 from both U12 and U15, whereas Figure 8b plots the mean vectors of cluster 0 of U12 and cluster 1 of U15. From both figures, it can be clearly observed that cluster 0 of U12 is closer to cluster 1 of U15, explaining the reasoning behind the decision of the algorithm. Through this, we would like to showcase the explainability aspect of the algorithm, which can be useful in building trust and acceptance to use the designed algorithm in a wider range of applications.

8 Conclusion and Future Work

In this paper, we have proposed a domain adaptation algorithm entitled DIBCA++, which is an optimized and robust version of its predecessor DIBCA. DIBCA++ has been designed to be resource-efficient and robust to outliers. In addition, we have

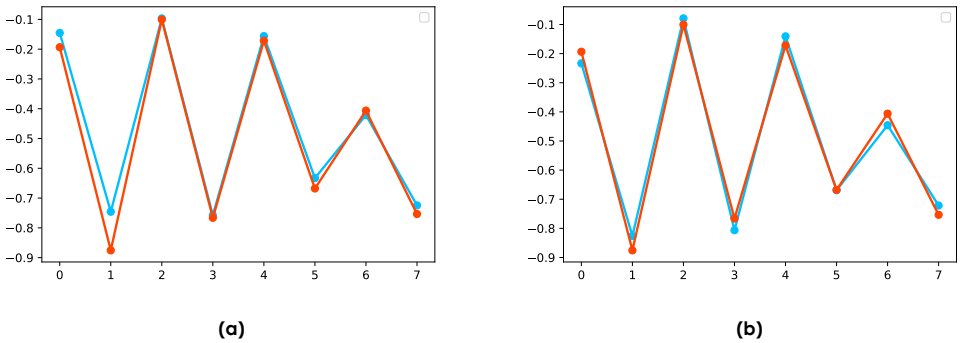


Figure 8: Visualization of the internal step, which can be used to explain the obtained correlation between cluster 0 of U12 and cluster 1 (instead of cluster 0) of U15. (a) Mean vectors of clusters 0 of U12 and U15; (b) Mean vectors of clusters 0 and 1 of U12 and U15 respectively.

developed an improved learning algorithm used for building the initial clustering models of the domains. This work also presents a detailed description of the main modules of DIBCA++, which has been extensively studied and evaluated with respect to well-designed experimental scenarios in two different real-world use cases, namely Human Activity Recognition (HAR) and smart logistics. The experimental results have shown that the DIBCA++ is capable of transferring knowledge between domains that, indeed, leads to improved performance results. In one of the experiments on the smart logistics use case, it has been observed that the adapted source and target models performed better than the original source and target models 70% and 80% of the time, respectively. The results on HAR also showcase improved performance when data from different domains are handled individually compared to when the data is combined, thus highlighting the importance of building personalized models.

In addition to the above-mentioned experiments, DIBCA++'s performance has also been compared to that of DIBCA. The experimental results showcase the better performance of DIBCA++ compared to the original DIBCA. It is convincingly the best performer on the HAR use case and outperforms DIBCA in 82.5% of the conducted experimental scenarios for the smart logistics use case.

Another advantage of the proposed algorithm is its explainability feature. It facilitates the analysis and interpretation of the obtained results, which supports a better understanding and acceptance of the algorithm and eventually can lead to the extraction of new knowledge about the studied data phenomenon.

Our future plans aim to evaluate the usability of the DIBCA++ in new applied domain adaptation scenarios. A sequential version of the algorithm will also be tested and further evaluated. The sequential version of DIBCA++ would allow continuous updating of the current integrated model and generating of new adapted domain models when data from a new domain is available.

Acknowledgments

This research was funded partly by the Knowledge Foundation, Sweden, through the Human-Centered Intelligent Realities (HINTS) Profile Project (contract 20220068). In addition, the data used in one of the experiments were provided due to the involvement in the Sony RAP 2020 Project "*Distributed and Adaptive Edge-based AI Models for Sensor Networks*".

Abbreviations

The following abbreviations are used in this manuscript:

AI:	Artificial Intelligence
ASM:	Adapted Source Model
ATM:	Adapted Target Model
AMI:	Adjusted Mutual Information
ARI:	Adjusted Rand Index
DaLiAc:	Daily Life Activities
DIBCA:	Domain Integration Bi-correlation Clustering Algorithm
DIBCA++:	Optimized Domain Integration Bi-correlation Clustering Algorithm
GNSS:	Global Navigation Satellite System
HAR:	Human Activity Recognition
ISM:	Inductive System Health Monitoring
IM:	Integrated Model
LTE:	Long Term Evolution
ML:	Machine Learning
MI:	Mutual Information
ODIP:	Online Domain Incremental Pool
OTDD:	Optimal Transportation Dataset Distance
RI:	Rand Index
SD:	Source Data
SM:	Source Model
TD:	Target Data
TM:	Target Model
XAI:	Explainable AI

References

[1] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76. DOI: 10.1109/JPROC.2020.3004555.

[2] M. AlShehhi, E. Damiani, and D. Wang. "Toward Domain Adaptation for small data sets". In: *Internet of Things* 16 (2021), p. 100458. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2021.100458.

[3] V. M. Devagiri, V. Boeva, and S. Abghari. “Domain Adaptation Through Cluster Integration and Correlation”. In: *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. 2022, pp. 1–8. DOI: 10.1109/ICDMW58026.2022.00025.

[4] D. L. Iverson. “Inductive System Health Monitoring.” In: *IC-AI*. 2004, pp. 605–611.

[5] A. Reiss and D. Stricker. *Introducing a New Benchmarked Dataset for Activity Monitoring*. 2012. URL: https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring.

[6] P. Davidsson. “Coin Classification Using a Novel Technique for Learning Characteristic Decision Trees by Controlling the Degree of Generalization”. In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. 1996.

[7] M. De Lange and T. Tuytelaars. “Continual Prototype Evolution: Learning Online from Non-Stationary Data Streams”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 8230–8239. DOI: 10.1109/ICCV48922.2021.00814.

[8] S. J. Pan and Q. Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[9] Y. Madadi, V. Seydi, K. Nasrollahi, R. Hossieni, and T. Moeslund. “Deep Visual Unsupervised Domain Adaptation for Classification Tasks: A Survey”. In: *IET Image Processing* 14.14 (2020), pp. 3283–3299. ISSN: 1751-9659. DOI: 10.1049/iet-ipr.2020.0087.

[10] G. Csurka. *Domain adaptation in computer vision applications*. Cham: Springer International Publishing, 2017.

[11] J. Li, G. Li, Y. Shi, and Y. Yu. “Cross-domain adaptive clustering for semi-supervised domain adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2505–2514. DOI: 10.1109/CVPR46437.2021.00253.

[12] H. Wang, J. Tian, S. Li, H. Zhao, F. Wu, and X. Li. “Structure-conditioned adversarial learning for unsupervised domain adaptation”. In: *Neurocomputing* 497 (2022), pp. 216–226. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.04.094.

[13] S. Hundschell, M. Weber, and P. Mandl. “An Empirical Study of Adversarial Domain Adaptation on Time Series Data”. In: *Artificial Intelligence and Soft Computing*. Ed. by L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada. Cham: Springer International Publishing, 2023, pp. 39–50. ISBN: 978-3-031-23492-7.

[14] G. Li, G. Kang, Y. Zhu, Y. Wei, and Y. Yang. “Domain Consensus Clustering for Universal Domain Adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 9757–9766.

[15] J. Xu, J. Song, Y. Sang, and L. Yin. “CDAML: a cluster-based domain adaptive meta-learning model for cross domain recommendation”. In: *World Wide Web* (2022), pp. 1573–1413. DOI: 10.1007/s11280-022-01068-5.

[16] S. Tang, Y. Zou, Z. Song, J. Lyu, L. Chen, M. Ye, S. Zhong, and J. Zhang. “Semantic consistency learning on manifold for source data-free unsupervised domain adaptation”. In: *Neural Networks* 152 (2022), pp. 467–478. ISSN: 0893-6080.

[17] M. Zhu. “Source Free Domain Adaptation by Deep Embedding Clustering”. In: *2021 18th Int. Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. 2021, pp. 309–312. DOI: 10.1109/ICCWAMTIP53232.2021.9674068.

[18] H. Tang, Y. Wang, and K. Jia. “Unsupervised domain adaptation via distilled discriminative clustering”. In: *Pattern Recognition* 127 (2022), p. 108638. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2022.108638.

[19] M. Orbes-Arteainst, J. Cardoso, L. Sørensen, C. Igel, S. Ourselin, M. Modat, M. Nielsen, and A. Pai. “Knowledge Distillation for Semi-supervised Domain Adaptation”. In: *OR 2.0 Context-Aware Operating Theaters and Machine Learning in Clinical Neuroimaging*. Ed. by L. Zhou, D. Sarikaya, S. M. Kia, S. Speidel, A. Malpani, D. Hashimoto, M. Habes, T. Löfstedt, K. Ritter, and H. Wang. Cham: Springer International Publishing, 2019, pp. 68–76.

[20] N. Gunasekara, H. Gomes, A. Bifet, and B. Pfahringer. “Adaptive Online Domain Incremental Continual Learning”. In: *Artificial Neural Networks and Machine Learning – ICANN 2022*. Ed. by E. Pimenidis, P. Angelov, C. Jayne, A. Papaleonidas, and M. Aydin. Cham: Springer International Publishing, 2022, pp. 491–502.

[21] S. Abghari, V. Boeva, E. Casalicchio, and P. Exner. “An Inductive System Monitoring Approach for GNSS Activation”. In: *Artificial Intelligence Applications and Innovations*. Ed. by I. Maglogiannis, L. Iliadis, J. Macintyre, and P. Cortez. Cham: Springer International Publishing, 2022, pp. 437–449.

[22] J. G. Saw, M. C. K. Yang, and T. C. Mo. “Chebyshev Inequality With Estimated Mean and Variance”. In: *The American Statistician* 38 (1984), pp. 130–132.

[23] V. Boeva and B. De Baets. “A new approach to admissible alternatives in interval decision making”. In: *2004 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No.04EX791)*. Vol. 1. 2004, 110–115 Vol.1. DOI: 10.1109/IS.2004.1344647.

[24] M. T. Ribeiro, S. Singh, and C. Guestrin. “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778.

[25] S. Srihari. “Explainable Artificial Intelligence: An Overview”. In: *Journal of the Washington Academy of Sciences* (2020).

[26] H. Leutheuser, D. Schuldhaus, and B. M. Eskofier. “Hierarchical, multi-sensor based classification of daily life activities: comparison with state-of-the-art algorithms using a benchmark dataset”. In: *PloS one* 8.10 (2013), e75196. DOI: 10.1371/journal.pone.0075196.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[28] L. Hubert and P. Arabie. “Comparing partitions”. In: *Journal of Classification* 2.1 (Dec. 1985), pp. 193–218.

[29] W. M. Rand. “Objective Criteria for the Evaluation of Clustering Methods”. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. ISSN: 01621459.

[30] N. X. Vinh, J. Epps, and J. Bailey. “Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?” In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML'09. Montreal, Quebec, Canada, 2009, pp. 1073–1080.

[31] N. X. Vinh, J. Epps, and J. Bailey. “Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance”. In: *Journal of Machine Learning Research* 11.95 (2010), pp. 2837–2854. URL: http://jmlr.org/papers/v11/vinh10a.html.

[32] W. Lu, Y. Chen, J. Wang, and X. Qin. “Cross-domain activity recognition via substructural optimal transport”. In: *Neurocomputing* 454 (2021), pp. 65–75. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.04.124. URL: https://www.sciencedirect.com/science/article/pii/S0925231221007025.

[33] J. Wang, Y. Chen, L. Hu, X. Peng, and P. S. Yu. “Stratified Transfer Learning for Cross-domain Activity Recognition”. In: *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 2018, pp. 1–10. DOI: 10.1109/PERCOM.2018.8444572.

[34] D. Alvarez-Melis and N. Fusi. “Geometric Dataset Distances via Optimal Transport”. In: *NeurIPS 2020*. ACM. Feb. 2020.

Paper VIII

Putting Sense into Incomplete Heterogeneous Data with Hypergraph Clustering Analysis

Vishnu Manasa Devagiri, Pierre Dagnely, Veselka Boeva, and Elena Tsiporkova
In: Symposium on Intelligent Data Analysis (IDA 2024), Stockholm, Sweden, April 2024 (In press).

Abstract

Many industrial scenarios are concerned with the exploration of high-dimensional heterogeneous data sets originating from diverse sources and often incomplete, i.e., containing a substantial amount of missing values. This paper proposes a novel unsupervised method that efficiently facilitates the exploration and analysis of such data sets. The methodology combines in an exploratory workflow multi-layer data analysis with shared nearest neighbor similarity and hypergraph clustering. It produces overlapping homogeneous clusters, i.e., assuming that the assets within each cluster exhibit comparable behavior. The latter can be used for computing relevant KPIs per cluster for the purpose of performance analysis and comparison. More concretely, such KPIs have the potential to aid domain experts in monitoring and understanding asset performance and, subsequently, enable the identification of outliers and the timely detection of performance degradation.
Keywords: Clustering, Heterogeneous data, Missing values, Hypergraph, Shared nearest neighbor similarity

1 Introduction

The majority of real-world data related to monitoring the performance of industrial equipment or of production and engineering processes is typically collected from multiple diverse sources and, thus, is very heterogeneous. It is a challenging task [1] to analyze and derive meaningful insights, e.g., evaluate and compare operational

performance across sources, from such data. Moreover, industrial data usually contains a lot of missing entities due to different reasons such as incomplete metadata records, lack of standardization, equipment malfunctioning, registration errors, communication issues, etc. This missing data usually affects small but varying sets of parameters/measurements, while the rest of the data records are available for analysis. However, mining information from complex multi-source data with missing values can be challenging since many state-of-the-art algorithms are not designed to handle missing values. In order to enable the use of such algorithms, common practices are to either impute missing values or to completely remove that particular instance or feature (the ones with a high degree of missing values). Both of these approaches can negatively affect the data quality though [2]. In addition, high-dimensional data is often composed of entries of a very diverse nature, and it might occur that some interesting, specific properties are associated only with a certain subset/type of features. There exists a risk of missing these when all the available features are explored together. Studies like [3–5], highlight the importance of viewing data from different perspectives or views (i.e., considering relevant feature subsets).

To address these challenges, we propose here a hybrid clustering methodology, realizing a multi-layer data analysis workflow, which is employing shared nearest neighbor similarity (SNNS) and hypergraph clustering. The approach is capable of extracting meaningful insights from high-dimensional data and, at the same time, is efficient at handling missing values without losing valuable information. More concretely, the method allows to organize the assets into separate (overlapping) groups such that the assets in each group share similar properties and, thus, are expected to exhibit comparable performance. In this way, it facilitates the complex task of monitoring and making sense of the performance of a large portfolio of heterogeneous in nature industrial assets. The potential of the proposed methodology is validated on a real-world data set with a substantial amount of missing values originating from the condition monitoring of a portfolio of industrial assets. These assets are very diverse in terms of technical specifications and functionalities, are used in different application contexts, and are produced by different manufacturers.

2 Related Work

Mining of data constructed across multiple domains or modalities (categorical, numerical, transactional, etc.) has been receiving high attention in the last decades due to the continuous increase of the variety in data sources [6]. This kind of data analysis is required within the context of grouping and understanding multi-view, heterogeneous, or multi-modal data in many real-world scenarios. The traditional clustering techniques usually fail to identify the cluster of objects with different characteristics due to difficulties in finding suitable similarity measures, or they are not capable of capturing the intrinsic structure of clusters.

To reduce these limitations, multi-view [3], multi-layered [4] or multi-type [5] clustering techniques are introduced. For example, multi-view clustering uses more than one set of attributes to improve the quality of generated clustering solutions [3]. In [4], a multi-layer clustering technique is introduced, originally designed for analysis of network data available in more than one layer [7]; where in contrast to multi-view clustering, conditional independence of layers is not assumed. In real-life scenarios, heterogeneous information networks (HINs) could be formed by the existence of multiple types of objects that are connected to each other through different kinds of links. In [8], the authors have studied the multi-type co-clustering problem in general HINs. They have proposed a clustering framework that can model general HINs and simultaneously generate clusters for all types of objects. In [5], a novel method that performs both clustering and classification tasks on HINs is proposed. The proposed technique is able to group heterogeneous objects in a network together and assign labels to unlabeled objects.

A common industrial challenge that impacts these methods is the presence of missing data. It is usually addressed by removing entries with missing data or imputing the missing data, e.g., replacing missing values of a feature with the average of known values for that feature or predicting them based on other known features of that entry. For instance, in [9], Yang et al. have proposed a multi-view clustering methodology that tackles missing data through imputation but also addresses inconsistencies between views.

However, data imputation always carries the risk of imputing noisy data, especially for industrial assets that are often highly heterogeneous. The creation of multi-view clustering approaches that deal with missing data in their input is still in its infancy. In [10], the authors use an indicator matrix whose elements indicate which data entries are observed and assess cluster validity only on observed entries. However, this approach cannot easily be generalized to all clustering approaches. Moreover, the proposed methodologies deal with incomplete views or missing values with some constraints, but they struggle when all views have missing values and even when the samples just miss a few features in a view [11].

In this work, we propose a novel multi-view clustering approach, which exploits the power of multi-layer clustering data analysis to transform the highly-dimensional heterogeneous data set into a hypergraph. Subsequently, the final clustering solution is obtained by applying a creative hypergraph clustering methodology. Our method goes beyond the existing state-of-the-art in its ability to deal efficiently with missing values without losing any information and to produce overlapping multi-view partitions without imposing any constraints on the underlying multi-source data.

3 Hypergraph-based Clustering Analysis Method

In this work, we study a real-world industrial use case considering the exploration and analysis of multi-source data originating from a large portfolio of heterogeneous assets (compressors). The available data set is composed of both metadata and sensor measurements (in the form of time series) and contains substantial quantities of missing data. Analyzing and interpreting data from different types of assets is challenging as they cannot be directly compared since they may substantially differ in technical specifications and other essential characteristics. Asset comparison can be facilitated by grouping the assets into more homogeneous subsets (clusters), i.e., composed of assets with similar characteristics and settings. The high-dimensional metadata, describing assets' technical specifications and various other properties, is used for this purpose. The available metadata is very suitable for having a realistic evaluation of the application potential of our clustering approach.

In addition, the validity of the generated clustering solution is further evaluated on the time series data originating from the continuous monitoring of the assets during their operation in the field. The assets' performance is analyzed by estimating and comparing the evolution of diverse, performance-related, Key Performance Indicators (KPIs) (see Section 4.4 for more details).

The proposed clustering workflow for analyzing multi-source heterogeneous data is divided into four main steps as illustrated in Figure 1 and outlined in detail in the subsections below.

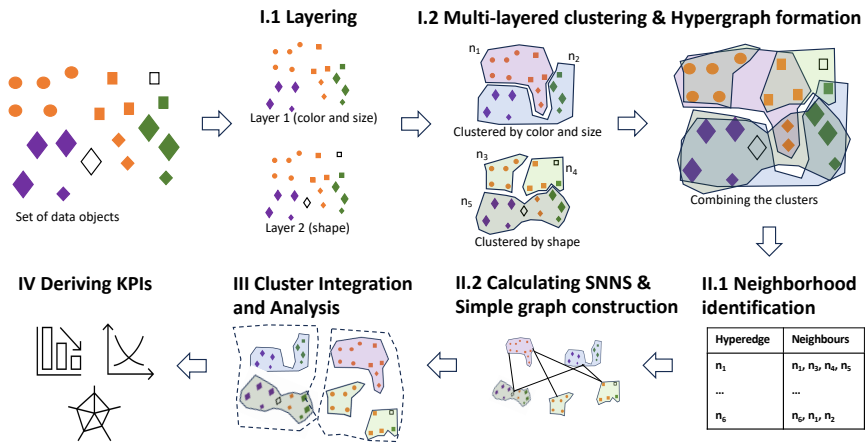


Figure 1: Illustrative summary of the proposed approach using two layers. SNNS stands for shared nearest neighbor similarity.

3.1 Step I: Hypergraph Construction

Let us assume that multi-source information about a set D of data objects with missing entities, which needs to be grouped into a number of similar categories, is avail-

able. The data objects are described in terms of a set F of relevant features. The workflow of the different steps used to construct a hypergraph with the data objects acting as vertices is outlined below:

3.1.1 I.1: Layering

Initially, the features in F are categorized into L different thematic layers, each representing a particular aspect describing the data objects. Domain knowledge is used in this step to identify the different layers. Each layer $i \in L$ is represented by a feature set F_i, $F_i \subseteq F$. Generalizing the layer construction step to different use cases is not straightforward and requires investing time in engaging with domain experts and acquiring a good understanding of the phenomenon under study. Section 4.1 provides more details on how this step is performed for the use case studied here.

3.1.2 I.2: Multi-layered clustering and hypergraph formation

Once the layers are identified, a hypergraph is constructed as it is explained below. A hypergraph is a generalized graph where edges, also referred to as hyperedges or nets can connect more than two nodes [12].

- In each layer i, for $i = 1, 2, \ldots, l$, data objects having missing values in feature set F_i are removed thus a subset of data objects $D_i \subseteq D$ is produced. It must be noted that these removed data objects are still considered in other layers where their features are completely captured. This allows not to exclude data objects with missing values from the analysis.

- Data objects D_i, for $i = 1, 2, \ldots, l$, in each layer are clustered to obtain a disjoint clustering solution represented by C_i.

- The produced clustering solutions from all the different layers are united to build an unweighted hypergraph $H = (V, N)$. The set of vertices V of this graph are the data objects, i.e., $V \equiv D$, and the set of hyperedges or nets N contains all the clusters identified by clustering the different layers, i.e., $N = C_1 \cup C_2 \cup \ldots \cup C_l$.

3.2 Step II: Transformation to Simple Graph

Once the hypergraph is built, it is transformed into an undirected weighted graph also known as a simple graph. This allows us to benefit further from the produced lower data granularity and facilitates grouping the data objects into the final clustering solution. In addition, this helps to handle missing entities, since the raw data are not used in the rest of the computations.

The new simple graph can be represented as $G = (N, E, s)$, where N, set of graph vertices, also the set of edges of the hypergraph H; E is the set of the edges

of the simple graph, and s is a real value function assigning a weight to each graph edge presented by the SNNS between vertices connected by this edge.

3.2.1 II.1: Neighborhood identification

For each hyperedge n_i, (for $i = 1, 2, \ldots, m$ and m the total number of hyperedges) its set of neighbors $\Gamma(n_i)$ is identified. A hyperedge is considered a neighbor of another hyperedge when there is a non-empty intersection between the two, i.e. if they have at least one common data object or pin (vertices in each hyperedge) of the hypergraph in both. Note that the hyperedge itself is also added to the list of neighbors (used while calculating the similarity). This can be formalized as:

$$\Gamma(n_i) = \{n_j \mid n_j \in N \wedge n_i \cap n_j \neq \emptyset\}. \tag{VIII.1}$$

Once the neighborhoods have been identified, the data objects will not be used, since the rest of the computation is entirely conducted using this information.

3.2.2 II.2: Calculating shared nearest neighbor similarity

We use the shared nearest neighbor similarity (SNNS) to measure the resemblance between two hyperedges of a hypergraph by considering their neighborhoods. Once the neighbors are identified, SNNS (s) between each pair of hyperedges n_i and n_j, for $i, j \in \{1, 2, \ldots, m\}$ is calculated as follows (inspired from [13] and adapted):

$$s(n_i, n_j) = \begin{cases} \frac{|\Gamma(n_i) \cap \Gamma(n_j)|}{|\Gamma(n_i) \cup \Gamma(n_j)|}, \text{ if } n_i, n_j \in \Gamma(n_i) \cap \Gamma(n_j). \\ 0, \text{otherwise} \end{cases} \tag{VIII.2}$$

3.3 Step III: Cluster Integration and Analysis

3.3.1 III.1: Partitioning into overlapping clusters

Once the graph $G = (N, E, s)$ is constructed, any clustering technique can be used to divide its vertices into different clusters, e.g., k-medoids or some graph-based clustering algorithm. In the obtained clustering solution, the vertices N of the simple graph (which are also the hyperedges of the hypergraph) are replaced with the vertices V of the hypergraph to generate the final clustering output. Note that we obtain an overlapping final clustering solution of the data objects $D \equiv V$, i.e., each data object can be assigned to more than one cluster. This is a desired outcome as, in many application scenarios, it is difficult to categorize an object into just a single category.

3.3.2 III.2: Deriving peak density hyperedges

Furthermore, we can identify the peak density vertices (hyperedges) in G similarly to the idea introduced in [13]. Namely, the SNNS between different vertices of $G = (N, E, s)$, can be used to calculate local density ρ of each vertex n_i, for

$i = 1, 2, \ldots, m$, as follows: $\rho(n_i) = \sum_{n_j \in \Gamma(n_i)} s(n_i, n_j)$. Subsequently, a threshold t can be used such that vertices having a local density greater than or equal to t, i.e., $\rho(n_i) \geq t$, are considered as the peak density points (hyperedges). These hyperedges can be interpreted as the most representative (typical) groups of objects.

3.4 Step IV: Deriving KPIs to analyze performance

The obtained clustering solution defines several different profiles, each characterized by the specific feature values defining each cluster. Operational data collected from continuous condition monitoring can then be used to enrich these profiles, e.g., to characterize them. In most real-world scenarios hardly any labeled data on the actual performance of the assets is collected. Instead, general indicators like the mean time between failures, the percentage of unplanned maintenance, or the overall maintenance cost are taken into account. These KPIs could be used to link the profiles to asset health. However, often, as in our use case, such data is not available and moreover, these indicators are not always directly linked to operational efficiency and overall performance in general but just to failures. We suggest instead using KPIs that allow us to compare operational behavior and performance across assets. In Section 4.4, different KPIs are described, being identified together with the domain experts, related to concrete performance indicators concerning the use case.

4 Evaluation in Industrial Use-case

This section presents a detailed overview of how the proposed methodology has been evaluated on a real-world industrial use case concerned with the condition monitoring of a fleet of compressors. Due to the heterogeneity of the available data, it is very challenging to derive useful insights about the operational performance of the different compressors. The research methodology proposed in this paper allows to overcome these challenges by facilitating incremental data exploration, resulting in partitioning the heterogeneous compressor population into relatively homogeneous overlapping groups sharing similar characteristics. Thanks to this partitioning, sensor data can be efficiently used to study and compare the operational performance within and across homogeneous compressor groups.

The data set used in our validation study has been offered to us by our industrial partners in the context of a research project exploring how to augment conventional analytics with log data. The company manages data from compressors installed worldwide and used in a wide variety of conditions, e.g., in factories and subjected to harsh conditions, or in hospitals and required to comply with very tight tolerance levels. The data set provided contains information about 265 compressors, characterized by 393 different parameters, e.g., brand, age, multitude of technical specifications. In addition, each of the compressors is monitored in the field by a wide range

of sensors. This study focuses on four commonly used sensors: ambient temperature, compressor outlet temperature and pressure, and internal pressure. The sensor data is reported at a granularity of one second, but the amount of data varies considerably from one compressor to another, from 6 months to 7 years.

4.1 Step I: Hypergraph Construction

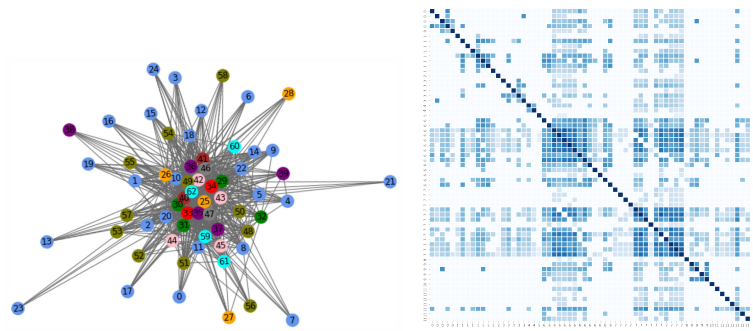
Initially, the data set has been cleaned and analyzed to identify relevant features, as using redundant features can negatively impact the final result. Over the 393 different parameters, only 24 relevant features have been retained using various techniques such as domain knowledge, dropping columns with a high percentage of NaNs ($> 60\%$), uniqueness of the feature values among different compressors, and correlation between the features. Subsequently, the selected features have been grouped into different conceptual layers using expert knowledge. For instance, all features related to pressure tolerance are grouped together. In addition, features of similar types (categorical, binary, and numerical) are kept together. In this use case, a total of ten layers have been created, consisting of 1 to 4 features per layer. The layers are finalized after being validated by domain experts, detailed information is given in Table 1. Based on the type of data available in each layer, different clustering techniques have been used (see Table 1). In layers where the k-means clustering technique is used, the optimal number of clusters is identified by applying four different cluster validation measures, namely Silhouette [14], Calinski Harabasz [14], Davies Bouldin [14], and Connectivity [15]. Once the clustering in each layer is finalized, an unweighted hypergraph is obtained by combining the clustering solutions of different layers. The obtained hypergraph has 63 edges (the sum of the number of clusters in each layer).

Table 1: An overview of the multi-layered clustering phase of hypergraph construction.

Layer	Description	Instances with NaNs	Instances Used	Type of Features	Clustering based on	Clusters
1	Supplier	42	223	Categorical	Categories	25
3	Cooling type	47	218	Binary		4
10	Activated temp. sensors	19	246			4
9	Act. pressure sensors	19	246		K-means	11
2	Pressure tolerance	115	150	Numerical		4
5	Outlet temp. tolerance	99	166			5
6	Output settings	87	178			2
8	Ambient temp. tolerance	118	147			2
4	Installation type	11	254		Domain knowledge	2
7	Age	79	186		Binning	4

4.2 Step II: Transformation to Simple Graph

The obtained hypergraph in the previous step is transformed into a weighted simple graph by considering the hyperedges as the vertices of the new graph. The weight of each edge of the simple graph is obtained by using the SNNS between the two vertices (hyperedges) the edge connects, which can also be represented using a 63×63 SNNS adjacency matrix. The obtained simple graph is presented in Figure 2a. It can be observed that the central region of the graph is very dense. Interesting to note that the four vertices far away from the center $(7, 13, 21, 23)$ are the singletons obtained in the final clustering solution.



(a) (b)

Figure 2: (a) Visual representation of the simple graph, different layers are distinguished by the color of vertices. (b) Dissimilarity matrix based on SNNS between different vertices ordered according to their cluster belonging. White cells represent values closer to one.

4.3 Step III: Cluster Integration and Analysis

The 63×63 SNNS matrix calculated above, converted into a dissimilarity matrix, is the input for k-medoids clustering algorithm. The optimal number of clusters, twelve, has been identified by Silhouette (0.08) and Connectivity (92.3) validation indices. In addition to this, we have also used the dissimilarity matrix, with edges sorted based on the cluster they belong to (see Figure 2b), to visually validate the obtained clustering output. Well-formed similarity patterns can be observed along the diagonal confirming to some extent the validity of the obtained partition. Once the clustering is obtained, the vertices of the simple graph are replaced with those of the hypergraph, thus resulting in an overlapping clustering solution, as different hyperedges have common vertices.

Figure 3 depicts the number of compressors per cluster and their uniqueness in regard to how many clusters they have been assigned to. The clusters can be grouped into three categories: 1) two large, heavily overlapping, clusters (clusters 6 and 7) with 237 and 245 compressors, respectively; 2) six medium-sized clusters with between 6 and 63 compressors (clusters $0, 1, 3, 4, 9$ and 11); 3) four singleton clusters

not visualized in Figure 3. These four singletons are also part of the two big clusters, 6 and 7. It is interesting to note that these singletons capture four unique compressor brands, only registered for these 4 compressors, thus confirming the potential of the approach to identify even small unique cluster groups. These singletons are not considered in the further discussion.

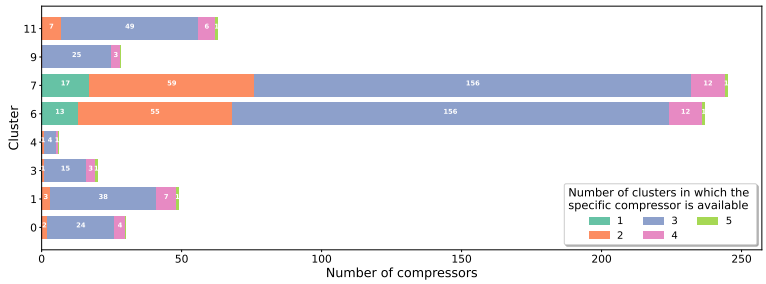


Figure 3: Number of compressors (and their uniqueness) per cluster. The colors represent the compressor uniqueness within the clusters, e.g., one compressor represented in lime-green is present in 5 clusters $(1, 3, 6, 7, 11)$.

The peak-density vertices (defined in Section 3.3, III.2) have been also identified in the resulting simple graph. It is interesting to notice that all the twelve peak density points lie within the two big clusters (6 and 7) of the k-medoids clustering solution. This confirms that all the dense regions of the graph are situated together, thus making it difficult to identify robust clusters if a density-based clustering algorithm is used. This is also visualized in Figure 2a, where one can see that the center of the graph is very densely populated.

It is interesting to investigate how the derived clusters differ in terms of the importance of the different features used in the construction of the initial hypergraph. The kernel density estimation of the different features in each cluster has been calculated for this purpose and visualized for three features in Figure 4. It can be observed that there is quite some variability across the clusters per feature, e.g., cluster 9 appears to have newer compressors, while the compressors in cluster 0 have been in use longer; cluster 11 is characterized by a higher *motor casing temperature high* in comparison to cluster 1; clusters 1 and 9 are characterized with higher *ambient temperatures high*, than clusters 3 and 11.

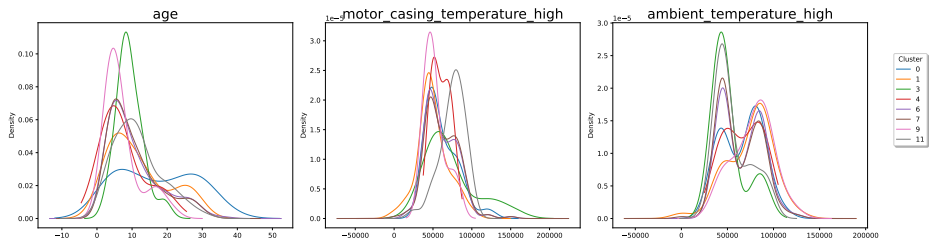


Figure 4: Kernel density distributions of few selected features per cluster.

4.4 Step IV: Deriving KPIs to analyze performance

We also have a large sensor dataset at our disposal, which captures the compressors' performance in the field. Combining both data types (metadata and sensor time series) allows to derive performance-related KPIs, e.g., assess the percentage of time that the compressors are operating within the expected range as defined originally in the metadata. Four sensors are considered: *outlet pressure*, *internal pressure*, *outlet temperature*, and *ambient temperature*, and the results are shown in Figure 5. The first three capture the compressor usage/internal behavior, while *ambient temperature* reflects the operational context.

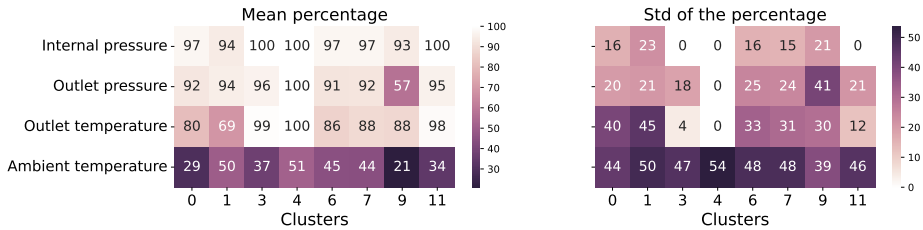


Figure 5: The percentage of time, mean, and standard deviation (std), the compressors have been operating within the expected range for the different sensors.

Considering clusters 6 and 7 are the largest, they can be regarded as representing the typical (baseline) situation. It can also be observed that very few compressors operate at the desired/recommended ranges for *ambient temperature*. The KPIs based on other sensors show fewer periods outside the expected ranges, except for cluster 9 which appears to meet specifications for only around 60% of operating time for *outlet pressure*, and with the highest standard deviation for that feature. It is interesting to observe this is also the cluster that has the lowest compliance, 21% of the time, with respect to *ambient temperature* limits. These two deviations might be linked to one another, which is already an interesting insight to be subjected to further investigation by our industrial partners.

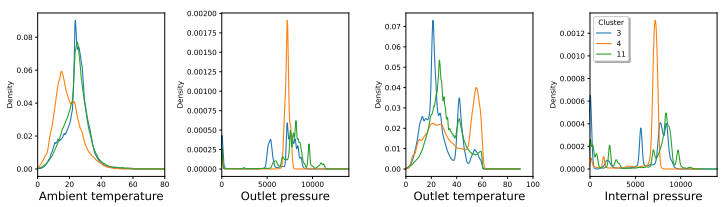


Figure 6: Kernel density distribution of the sensor data of selected clusters.

In general, three different groups of clusters can be distinguished: 1) clusters 6 and 7 representing regular/baseline behavior; 2) clusters 3, 4, and 11 are almost always within range, indicating that these compressors are probably used in an environment where it is of high importance to work within the expected ranges; 3) clusters

$0, 1$ and 9 exhibit many more periods outside the expected ranges, probably containing compressors operating in less constrained contexts. Further characterization of these groups can be performed by examining the kernel density estimation of the different sensor values. For readability purposes, only group 2 (clusters $3, 4,$ and 11) is showcased in Figure 6. It can be observed that cluster 4 is characterized by lower *ambient temperature* but higher *outlet temperature*, while these are the opposite for clusters 3 and 11. Cluster 4 is also characterized by very stable *outlet* and *internal pressures* (around 7000) supporting the hypothesis that some of these compressors are probably being used in some strictly regulated environments.

5 Conclusion

This work has presented a hybrid clustering methodology for analyzing and making sense of complex, multi-source heterogeneous data sets emerging nowadays from real-world industrial applications. Such data sets are typically, if not always, characterized by a high rate of missing values, which makes their analysis by traditional machine learning methods challenging. We have conceived a novel data exploration workflow incorporating concepts such as multi-layered clustering, hypergraph, shared nearest neighbor similarity, and k-medoids, allowing us to arrive at a multi-dimensional data partition without compromising data size due to missing values. The ability of the proposed methodology to derive meaningful insights has been demonstrated in a real-world industrial use case, which also convincingly confirmed the validity of the produced clustering solution.

Acknowledgements

V. M. Devagiri's and V. Boeva's research is partly funded by the Knowledge Foundation, Sweden, through the Human-Centered Intelligent Realities (HINTS) Profile Project (contract 20220068). P. Dagnely's and E. Tsiporkova's research received funding from the Flemish Government (AI Research Program).

References

[1] V. Wenz, A. Kesper, and G. Taentzer. "Clustering Heterogeneous Data Values for Data Quality Analysis". In: *J. Data and Information Quality* 15.3 (2023).

[2] M. C. de Goeij, M. van Diepen, K. J. Jager, G. Tripepi, C. Zoccali, and F. W. Dekker. "Multiple imputation: dealing with missing data". In: *Nephrology Dialysis Transplantation* 28.10 (2013), pp. 2415–2420.

[3] L. Fu, P. Lin, A. V. Vasilakos, and S. Wang. “An overview of recent multi-view clustering”. In: *Neurocomputing* 402 (2020), pp. 148–161. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.02.104.

[4] D. Gamberger et al. “Multilayer Clustering: A Discovery Experiment on Country Level Trading Data”. In: *Discovery Science*. Springer Int. Publ., 2014, pp. 87–98.

[5] G. Pio, F. Serafino, D. Malerba, and M. Ceci. “Multi-type clustering and classification from heterogeneous networks”. In: *Information Sciences* 425 (2018), pp. 107–126.

[6] A. Abdullin and O. Nasraoui. “Clustering Heterogeneous Data Sets”. In: *2012 Eighth Latin American Web Congress*. 2012, pp. 1–8.

[7] G. Caldarelli. *Scale-Free Networks: Complex Webs in Nature and Technology*. Oxford University Press, Incorporated, 2007.

[8] Zhang et al. “Multi-type Co-clustering of General Heterogeneous Information Networks via Nonnegative Matrix Tri-Factorization”. In: *IEEE ICDM 2016*.

[9] M. Yang et al. “Robust multi-view clustering with incomplete information”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 45.1 (2022), pp. 1055–1069.

[10] G. Chao et al. “Multi-view cluster analysis with incomplete data to understand treatment effects”. In: *Information sciences* 494 (2019), pp. 278–293.

[11] G. Chao, S. Sun, J. Bi, et al. “A survey on multi-view clustering”. In: *arXiv preprint arXiv:1712.06246* 2 (2017).

[12] S. Schlag, T. Heuer, L. Gottesbüren, Y. Akhremtsev, C. Schulz, and P. Sanders. “High-Quality Hypergraph Partitioning”. In: *ACM J. Exp. Algorithmics* 27 (Feb. 2023).

[13] R. Liu, H. Wang, and X. Yu. “Shared-Nearest-Neighbor-Based Clustering by Fast Search and Find of Density Peaks”. In: *Inf. Sci.* 450.C (June 2018), pp. 200–226.

[14] I. F. Ashari et al. “Analysis of Elbow, Silhouette, Davies-Bouldin, Calinski-Harabasz, and Rand-Index Evaluation on K-Means Algorithm for Classifying Flood-Affected Areas in Jakarta”. In: *Journal of Applied Informatics and Computing* 7.1 (2023), pp. 95–103.

[15] J. Handl, J. Knowles, and D. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.