Worcester Polytechnic Institute
Digital WPI

Doctoral Dissertations (All Dissertations, All Years) | Electronic Theses and Dissertations

2018-12-07

Pattern Mining and Sense-Making Support for Enhancing the User Experience

Abhishek Mukherji
Worcester Polytechnic Institute, mukherab@wpi.edu

Follow this and additional works at: https://digitalcommons.wpi.edu/etd-dissertations

Repository Citation
Mukherji, A. (2018). *Pattern Mining and Sense-Making Support for Enhancing the User Experience*. Retrieved from https://digitalcommons.wpi.edu/etd-dissertations/497

This dissertation is brought to you for free and open access by Digital WPI. It has been accepted for inclusion in Doctoral Dissertations (All Dissertations, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.

Pattern Mining and Sense-Making Support for Enhancing the User Experience

by

Abhishek Mukherji

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Computer Science

by

December 7, 2018

APPROVED:

Professor Elke A. Rundensteiner, WPI Advisor	Professor Matthew O. Ward, WPI Co-Advisor (Posthumously)
Professor Mohamed Y. Eltabakh, WPI	Professor Emmanuel O. Agu, WPI
Dr. Geetika T. Lakshmanan, Amazon, US. External Committee Member	Professor Craig E. Wills, WPI Head of Department

Abstract

This dissertation research aims to provide support for efficient data mining to enhance the user experience. While data mining techniques such as frequent itemset and sequence mining are well established as powerful pattern discovery tools in domains from science, medicine to business, effective extraction of new knowledge from data often requires interactive exploration and input by domain experts. When applying data mining over data sets of interest, the analysts must iteratively sift through a long list of patterns generated for varying parameters, such as interestingness thresholds (e.g., support and confidence) and the data subset, to discover insightful patterns. During each iteration of such exploratory sessions, the mining may require several hours to process even a medium sized (10s of GB) dataset. To maintain user interest and focus, real-time query turnaround times are essential. Clearly, this demands novel solutions to enhance the interactivity of mining tools.

Scalable data mining solutions for mobile usage data from a variety of sensors and apps have recently gained the attention of researchers. Patterns mined over mobile data may enable a variety of context-aware applications ranging from automating frequently repeated tasks (e.g., turning phone to mute during meetings) to providing personalized recommendations (e.g., restaurant recommendations when a user is likely to eat out). Research on efficient on-device data mining techniques is yet another interesting research topic with the overarching goal of achieving scalability and enhanced user experience.

In this dissertation, we first develop a novel interactive exploration framework that supports domain experts in their exploration of association rules mined over their data sets and make sense of those mined results. Our solution adopts the core principle of *preprocess-once-query-many* (POQM) paradigm and consists of contributions in two aspects, namely, (a.) the back-end PARAS framework and (b.) the front end FIRE rule exploration framework. PARAS framework addresses the problem of efficient data mining techniques. We gain key insights on rule relationships in a parameter space view; resulting in a compact storage of rules that enable query-time reconstruction of complete rulesets. Further, our proposed Framework for Interactive Rule Exploration (FIRE) features innovative visual displays and interactions to enable interactive rule exploration. We propose two linked interactive displays, namely the parameter space view (PSpace) and the rule space view (RSpace), that together enable enhanced sense-making of rule relationships. The usability and effectiveness of the proposed PARAS and FIRE frameworks are evaluated via performance experiments, case studies and user studies using several benchmark datasets.

Analysts also need to be able to dynamically adapt the subset of the data they work on as they explore in which partitions of the data set the association rules may be most significant. We propose a novel multidimensional itemset-based data partitioning (MIP-index) that extends the POQM paradigm to *mine localized rules*. MIP-index offers efficient mining performance by utilizing pre-computed results, while still allowing the user the flexibility of selecting any data subset of interest at run-time. Our proposed COLARM framework consists of a suite of alternative execution strategies using optimization principles such as selection push-up, supported R-tree fil-

ter and differential treatment of contained versus partially overlapped MIPs. Through performance evaluation over benchmark datasets, we establish the effectiveness of our COLARM framework in a rich variety of tested cases.

Towards the overall goal of developing scalable data mining solutions, we then designed OLAPH, an on-device context-aware service that learns phone usage patterns. Such patterns can be used to provide device intelligence to enhance the user experience. This part of my dissertation focuses on learning several key insights regarding alternative models to train over mobile usage data, adequacy of data to achieve high prediction accuracy under constrained resources of the mobile devices, and feasibly of scheduling periodic data mining tasks on a phone. OLAPH is a generic service to which applications can subscribe to receive personalized predictions in real-time.

Overall, this dissertation research encompasses contributions at the intersection of data mining, knowledge management and visual analytics for a rich variety of data sources ranging from scientific datasets to mobile usage logs.

To my mother and first teacher, Rita Mukherji

who believed in my potential, even when I did not...

Acknowledgements

I consider myself truly privileged to have Professor Elke A. Rundensteiner as my advisor during my graduate studies. Dr. Rundensteiner's deep knowledge, expertise, high energy and drive for excellence has always been a source of inspiration to me. She has been a great mentor, always available to listen patiently, support, advise, and encourage throughout the ups and downs of my graduate studies and personal life. I have grown from a naive, inexperienced student into a researcher, thinker and professional I am today, and I owe it all to her. I cherish every day of my association with Dr. Rundensteiner and I hope to continue to seek her guidance throughout my professional career.

My heartfelt gratitude towards my co-advisor late Prof. Matthew O. Ward. He always inspired me to develop visual thinking, which continues to help me in my career. I also miss his friendly attitude and his walkie-talkie meetings even during the winters. I sincerely thank the members of my Ph.D. committee – Dr. Geetika T. Lakshmanan, Prof. Mohamed Y. Eltabakh, Prof. Emmanuel O. Agu, and Prof. Craig E. Wills. Their insightful and critical feedback has helped me throughout the various milestones in my Ph.D. I specially thank Dr. Lakshmanan for her patience and encouragement through these years and mentoring me whenever I seek career advice.

I am thankful for the financial support I have received from my advisors Prof. Elke A. Rundensteiner, and late Prof. Matthew O. Ward as part of CAPE and

XMDV projects and Teaching Assistantship during my pre-dissertation research from the CS department. In absence of this financial support, I would not have been able to pursue my graduate studies. My thanks also goes to the National Science Foundation (NSF) for providing research grants IIS-0414567, SGER-0633930 and CRI-0551584 for my pre-dissertation research and research grants IIS-0812027, CCF-0811510 and IIS-1117139 that supported my dissertation research.

I would like to thank DSRG team members – in particular Dr. Song Wang and Dr. Venkatesh (Venky) Raghavan for being my mentors in the early years of my graduate studentship. Venky and I co-authored several of my pre-dissertation research papers. I feel honored to be part of two key projects − CAPE and XMDV. I extend my gratitude to my collaborators Xika Lin, Ermal Toto, Christopher Botaish, Jason Whitehouse and Mingrui Wei. Xika has been a key contributor to the PARAS framework; I thank her for her fresh ideas, out-of-the-box thinking and hard work. The FIRE journal paper greatly benefited from Ermal's help with the case study and Xika's help with conducting the user study. WPI undergraduate students Jason and Christopher helped implement the FIRE and the SPHINX visual engines. Mingrui collaborated with me in my pre-dissertation research. I thank the present and past students working on the XMDV project − Dr. Zaixian Xie, Dr. Di Yang, Dr. Zhenyu Guo, Dr. Kaiyu Zhao, Charudatta Wad and others for working together as a team. In addition, I would also like to thank the CAPE team members for their help in my early Ph.D. study. I very much appreciate the discussions with as well as friendship of now Dr. Mo Liu, Dr. Mingzhu Wei, Dr. Di Wang, Dr. Karen Works, Dr. Olga Poppe, Medhabi Ray and all previous and current DSRG members.

Beyond my dissertation research, I have learned a broad range of subjects that have helped shape my career. I am indebted to the knowledgeable professors in the Department of Computer Science, namely, Prof. Stanley Selkow, Prof. Carolina Ruiz, Prof. Daniel Dougherty, Prof. David C. Brown, Prof. Hugh Lauer, Prof. Craig Wills, Prof. Mark Claypool, Prof. Michael Gennert, Prof. George Heineman, Prof. Micha Hofri and many others. I also express my sincere thanks to my first graduate advisor during my MS – Professor Murali Mani, who helped build my foundation and interest in data management systems and data mining. A special thanks to the systems and support staff – Mark Taylor, Mike Voorhis, Jesse Banning, Diane Baxter, Refie Cane and Christine Caron, in our department and from the school for providing a state-of-the-art computing infrastructure and a friendly environment.

I am what I am due to the hard work, love and sacrifices of my parents – Amitava Mukherji and Rita Mukherji. Unfortunately, I cannot share my happiness in person with my mother, as she is no more with us, but I am sure in spirit she continues to bless me. My spouse Archana Goyal is a constant source of inspiration. Archana never gave up on me and kept reminding me of the unfinished task of defending my dissertation. My brothers Arindam and Anirban and sisters-in-law Joyeeta and Sushma have always inspired me and have been my role models. I am grateful to my wife's family – Dr. Vinod Prakash, Dr. Sharda Goyal, Ankur and Anuradha for their patience, support and love during my graduate studies. This acknowledgement will be incomplete without mention of my loving nephews Neelanjan, Shikhar, Pranjal and Avish, as well as the newest member of Mukherji family, our dog Marble. They fill my life with happiness and purpose.

My Publications

Publications Contributing to this Dissertation

Part I: Support for Interactive Association Rule Mining

Part I of this dissertation addresses the problem of adding interactive support for fast real-time mining of association rules.

1. Xika Lin, Abhishek Mukherji, Elke A. Rundensteiner, Carolina Ruiz and Matthew O. Ward, *PARAS: A Parameter Space Framework for Online Association Mining.* , **Proceedings of Very Large Data Bases**, Volume 6(3), 2013, pages 193-204.

   Relationship to this dissertation: This is the first paper introducing the parameter space paradigm for exploration of mined patterns. I defined the foundation of the Parameter space model such as stable regions, supported queries, the overall offline and online framework and the algorithm design. This work contains major contributions by fellow PhD Student Xika Lin towards insights gained on rule relationships, compact representation of rule relationships and efficient indexes in the context of the parameter space. Further, Xika also contributed significantly to the experimental evaluation to establish the efficiency of PARAS. I discuss the solution provided in this paper in Sections 2.1 through 2.4 of part I of my dissertation.

2. Abhishek Mukherji, Xika Lin, Christopher R. Botaish, Jason Whitehouse, Elke A.

Rundensteiner, Matthew O. Ward and Carolina Ruiz, *PARAS: Interactive Parameter Space Exploration for Association Rule Mining*, **ACM Special Interest Group on Management Of Data (SIGMOD)**, Demonstration, 2013, pages 1017-1020.

Relationship to this dissertation: This is a demonstration paper that first introduces a front-end visualization over the parameter space (PARAS) framework.

3. Abhishek Mukherji, Xika Lin, Jason Whitehouse, Christopher R. Botaish, Elke A. Rundensteiner and Matthew O. Ward, *FIRE: Interactive Visual Support for Parameter Space-driven Rule Mining*, **ACM Conference on Information and Knowledge Management (CIKM)** 2013, pages 2447-2452.

   Relationship to this dissertation: This is a short paper introducing the FIRE framework. This work specifically focuses on the PSpace view interactions. Section 2.6 in part I of the dissertation includes the content described in this work.

4. Abhishek Mukherji, Xika Lin, Ermal Toto, Christopher R. Botaish, Jason Whitehouse, Elke A. Rundensteiner and Matthew O. Ward, *FIRE: A Two-level Interactive Visualization for Deep Exploration of Association Rules*, **International Journal of Data Science and Analytics (JDSA)**, 2018, pages 1-26.

   Relationship to this dissertation: This is the full journal paper on Part I of the dissertation. We present the overall visual framework for interactive rule exploration (FIRE), that consists of the parameter space view (PSpace) and the rule space view (RSpace) to enhance sense-making of mined rules. This was a collaborative work. I designed the key visualizations and interactions and also drafted all of the user study with the two datasets. Christopher and Jason implemented the FIRE visual engine. Xika conducted the user study with 22 subjects and summarized the results to establish the effectiveness of FIRE. Further, Ermal conducted the case study us-

ing FIRE, which I extended by adding the comparison with rule visualizations in ARulesViz package. Sections 2.5 through 2.8 in Part I of this dissertation describe this work.

Part II: Facilitating Localized Association Rule Mining

Part II of this dissertation addresses the problem of efficiently mining association rules that are hidden in a global context yet are locally significant.

5. Abhishek Mukherji, Elke A. Rundensteiner and Matthew O. Ward, *COLARM: Cost-based Optimization for Localized Association Rule Mining*, **International Conference on Extending Database Technology (EDBT)**, 2014, pages 181-192.

   Relationship to this dissertation: In this work, we present the COLARM framework for efficiently mining localized association rules in real-time. Part II of this dissertation is based on this work.

6. Abhishek Mukherji, Elke A. Rundensteiner and Matthew O. Ward, *Facilitating Interactive Mining of Global and Local Association Rules*, **PhD Workshop @ Conference on Information and Knowledge Management (PIKM)**, 2014, pages 35-44.

   Relationship to this dissertation: This is a PhD workshop paper where I discussed the overall theme of my PhD dissertation and received valuable feedback from expert researchers.

Part III: Sequential pattern based predictions to learn user behavior from mobile context data

With the advent of smartphones, there are new opportunities to utilize data mining over mobile data such as app usage, location (GPS), battery usage, calendar, call and SMS logs. Part III of my dissertation extends the applicability of data mining, in particular, to explore sequence prediction modeling, which is well-known in domains such as science, medicine and business, to learn and predict user behavioral patterns from mobile users' data.

7. Abhishek Mukherji and Elke A. Rundensteiner, *OLAPH: On-device Life-logging And Predicting Habitual Behavior using Context Sequence Predictor,* **In submission**.

   Relationship to this dissertation: In this work, we present the design and implementation of On-device Life-logging And Predicting Habitual Behavior (OLAPH) service. OLAPH adapts sequence prediction modeling to fit the paradigm of context-awareness and mobile sensing. Sections 4.1 to 4.5 of this dissertation is based on this work.

Other Publications (Not Included in This Dissertation)

Below I list three of our research papers that are not included in this dissertation, yet demonstrate how the core principles of PARAS can be extended. Xika Lin is the lead researcher on papers 8 and 9.

8. Xika Lin, Abhishek Mukherji, Elke A. Rundensteiner and Matthew O. Ward, *SPIRE: Supporting Parameter-Driven Interactive Rule Mining and Exploration.* **Proceedings of Very Large Databases (VLDB)**, demonstration, volume 7(13), 2014, pages

1653-1656.

Relationship to this dissertation: This paper extends the concept of parameter space-based exploration to negative rules.

9. Xika Lin, Abhishek Mukherji, Elke A. Rundensteiner and Matthew O. Ward, *Interactive Mining of Association Rules using a Parameter-Space Driven Approach* **in submission to a journal**

   Relationship to this dissertation: This is our joint comprehensive work on the PARAS framework which is at this time in submission to a journal for review.

10. Abhishek Mukherji, Jason Whitehouse, Christopher R Botaish, Elke A Rundensteiner and Matthew O Ward, *SPHINX: Rich Insights into Evidence-Hypotheses Relationships via Parameter Space-based Exploration*, **ACM Conference on Information and Knowledge Management (CIKM)**, demonstration, 2013, pages 2521-2524.

    Relationship to this dissertation: In this work we extended the parameter space paradigm to explore hypothesis and evidence space. Jason and Christopher helped with implementation of the SPHINX visual engine.

Pre-dissertation Publications

The below listed publications are outcomes of my *pre-dissertation research* and my *Masters' thesis* in data stream processing (CAPE project) at WPI. Paper 11 below is my research qualifier work. Venky and I collaborated with the Fire Protection Engineering department of WPI for papers 12 and 13. Paper 12 is my MS Thesis work.

11. Abhishek Mukherji, Elke A. Rundensteiner, David C. Brown and Venkatesh Raghavan, *Achieving high freshness and optimal throughput in CPU-limited execution of*

multi-join continuous queries, **British (Inter-)National Conference on Databases (BNCOD)** 2011, pages 48-65.

12. Abhishek Mukherji, Elke A. Rundensteiner, David Brown, and Venkatesh Raghavan, *SNIF TOOL: Sniffing for patterns in continuous streams*, **ACM Conference on Information and Knowledge Management (CIKM)**, 2008, pages 369-378.

13. Venkatesh Raghavan, Elke A. Rundensteiner, John Woycheese, Abhishek Mukherji, *FireStream: Sensor Stream Processing for Monitoring Fire Spread,* demonstration, **IEEE International Conference on Data Engineering (ICDE)** 2007, pages 1507-1508.

Contents

List of Figures **xx**

List of Tables **xxiv**

1 Introduction **1**

1.1 Motivation 1

1.1.1 Need for Interactive Mining 1

1.1.2 Importance of Localized Patterns 2

1.1.3 Mining Patterns from Mobile Sensor and App Data 4

1.2 Research Challenges Addressed in This Dissertation 6

1.2.1 Interactive Rule Exploration 6

1.2.2 Mining of Localized Rules 9

1.2.3 Mobile Context Sequence Prediction 10

1.3 Proposed Solutions 11

1.3.1 PARAS and FIRE Frameworks for Interactive Rule Exploration . 11

1.3.2 COLARM Framework for Localized Rule Mining 14

1.3.3 OLAPH Sequence Prediction over Mobile Context Data. 15

1.4 Dissertation Organization 16

2 Interactive Rule Exploration with PARAS and FIRE **17**

2.1 Foundation of Parameter-driven Rule Mining 17
2.1.1 Parameter Space Model 18
2.1.2 Stable Region Abstractions. 18
2.1.3 Rule Redundancy in Association Rules 21
2.2 Offline PSpace Construction 22
2.3 Redundancy Relationships 24
2.3.1 Abstracting Redundancy Relationships 24
2.3.2 Optimized Location Computation 29
2.3.3 The Compact PSpace Index 30
2.4 Online Query Processing 32
2.5 Overview of FIRE Visual Paradigm 36
2.6 Interactive Visual Parameter Space Design 38
2.6.1 The PSpace Visualization 38
2.6.2 The PSpace Interactions 39
2.6.3 Visualizing the PSpace-RSpace Relationship 41
2.7 Interactive Visual Rule Space Design 42
2.7.1 The RSpace Tabular View 42
2.7.2 The RSpace Glyph View 43
2.7.3 The RSpace Interactions 46
2.7.4 RSpace Glyph Placement 47
2.8 Experimental Evaluation 49
2.8.1 Evaluation of PARAS 49
2.8.1.1 Evaluation of Preprocessing Times 51
2.8.1.2 Evaluation of Online Processing Time 52
2.8.1.3 Evaluation of Index Sizes 55
2.8.1.4 Conclusions from the Experimental Evaluation 56

2.8.2 Case Study of FIRE Visual Rule Explorer 57
2.8.2.1 Exploring The Dataset Using FIRE 58
2.8.2.2 Exploring the Dataset Using ARulesViz 65
2.8.2.3 Conclusions from the Case Study 68
2.8.3 User Study of FIRE Visual Rule Explorer 69
2.8.3.1 Evaluation Methodology 69
2.8.3.2 Design of Our User Study 71
2.8.3.3 Hypotheses 73
2.8.3.4 Conclusions from the User Study 74
2.9 Related Work 80
2.10 Conclusion 83

3 Localized Rule Mining with COLARM **84**
3.1 Preliminaries 84
3.1.1 Itemsets in Multidimensional Space 84
3.1.2 Localized Association Mining Problem 85
3.2 The COLARM Design 87
3.2.1 The COLARM Framework 87
3.2.2 Offline Preprocessing Phase 88
3.2.3 The Two Level MIP-index 89
3.2.4 Online Query Processing 91
3.3 Strategies for Online Mining 92
3.3.1 The Basic S-E-V Plan 94
3.3.2 Pushing Selection Up The Plan 97
3.3.3 The Supported R-tree Filter 98
3.3.4 Pipelining both Supported Operators 99

3.3.5 Treating Contained and Overlapped MIPs Differently 100
3.3.6 Employing the Traditional Mining Plan 101
3.4 Experimental Validation of the COLARM Framework 102
3.4.1 Accuracy of the COLARM Optimizer 105
3.4.2 Measuring the Optimization Benefits 108
3.4.3 Comparing Local vs. Global Rules 109
3.5 Related Work 111
3.6 Conclusion 111

4 Mobile Context Sequence Modeling with OLAPH 113
4.1 Background 113
4.1.1 Definitions 114
4.1.2 Sequence Prediction Problem Statement 116
4.1.3 Frequent Sequences vs. Frequent Itemsets 117
4.2 Approach 118
4.2.1 OLAPH Architecture 118
4.2.2 Context Event Pre-processor 120
4.2.2.1 Intervaled Event Generation 121
4.2.2.2 Time Features 122
4.2.3 Sequence Database Generator 122
4.2.4 Sequence Prediction Model Trainer 124
4.3 Experimental Evaluation 128
4.3.1 Evaluation Methodology 128
4.3.2 Context Data Collection 129
4.3.3 System Performance 132
4.4 Related Work 139

4.5 Conclusion 142

5 Conclusions of This Dissertation **143**

6 Future Work **147**

6.1 PSpace Paradigm and Online Mining Support 147

6.2 On-device Mining and Prediction Modeling 148

References **151**

List of Figures

1.1 CFIs Distributed by Datasets. (ZH02) . . . 3
1.2 The Example Salary Dataset. (MRW14) . . . 3
1.3 Google Timeline. . . . 5
1.4 Example User Behavioral Sequential Patterns. . . . 5

2.1 Adjacency Lattice. . . . 19
2.2 Parameter Space. . . . 19
2.3 Stable Regions. . . . 23
2.4 Enriched Stable Regions. . . . 23
2.5 Dominating Rules. . . . 23
2.6 Dominating Locations. . . . 23
2.7 The PSpace Index. . . . 31
2.8 The FIRE Architecture . . . 36
2.9 The FIRE Visual Interface . . . 36
2.10 PSpace (All Rules) for the Mushroom dataset . . . 37
2.11 PSpace (Unique Rules) for the Mushroom dataset . . . 37
2.12 PSpace (Unique + Non-red.) for the Mushroom dataset . . . 38
2.13 Rule Cardinality Skyline (>100 Rules) . . . 38
2.14 PSpace-RSpace Linkage . . . 40

2.15 Comparing Two Regions 40
2.16 Lined Glyph 1 43
2.17 Lined Glyph 2 43
2.18 Con'ted Glyph 43
2.19 Filled Glyph 43
2.20 Lined Glyphs 45
2.21 Connected Glyphs 45
2.22 Filled Glyphs 45
2.23 Tabular RSpace. 46
2.24 PCA Placement of Rules. 47
2.25 MDS Placement of Rules. 47
2.26 Preprocessing Times for All Three Datasets. 51
2.27 Avg. Online Query Processing Times (T100k) [Rules w Redundancy Resolution in (a),(b) and w/o in (c),(d)]. 52
2.28 Avg. Online Query Processing Times (T5000k) [Rules w Redundancy Resolution in (a),(b) and w/o in (c),(d)]. 52
2.29 Avg. Online Query Processing Times (Webdocs) [Rules w Redundancy Resolution in (a),(b) and w/o in (c),(d)]. 52
2.30 Multi-Query Costs. 54
2.31 Size of the PSpace Index. 54
2.32 Finding the Highest Rules: the Highest No Common Knowledge Rule . . 57
2.33 Original Casual User Count. 58
2.34 Adjusted Casual User Count. 58
2.35 Orig. Registered User Count. 59
2.36 Adj. Registered User Count. 59
2.37 Filtering for Weekends 62

2.38 Filtering for Registered Users 62
2.39 Skyline Cardinality: Distinguishing Regions with >20 Rules and ≤ 20 Rules. 63
2.40 Comparing Rule Glyphs. 65
2.41 Clustered Rule Glyphs. 65
2.42 ARulesViz Scatterplot UI. 66
2.43 ARulesViz Grouped-Matrix UI. 66
2.44 Time Spent on T1, T2 and T3. 74
2.45 Time Spent on T4 and T5. 74
2.46 Accuracy of T1, T2 and T3. 75
2.47 Accuracy of T4 and T5. 75
2.48 Time Spent on T6 and T7. 76
2.49 Accuracy of T6 and T7. 76
2.50 Time Spent on T8. 77
2.51 Time Spent on T9. 77
2.52 Accuracy of T8. 77
2.53 Accuracy of T9. 77
2.54 Time Spent on T10. 78
2.55 Accuracy of T10. 78
2.56 Votes on the Preference of CRM vs. FIRE Per Task. 79
2.57 Survey Question on T8. 79
2.58 Survey Question on T9. 79
2.59 Overall Accuracy using CRM and FIRE. 79

3.1 Itemsets in n-Dimensional Space. 84
3.2 The COLARM framework. 87
3.3 The two-level MIP-index. 88

3.4 Itemsets Covering the Cells. 91
3.5 The POQM Mining Plans. 93
3.6 Supported R-tree. 96
3.7 More Mining Plans. 97
3.8 # Frequent Itemsets by Primary Thresholds. 105
3.9 Avg. CPU Costs of Mining Plans for Chess Dataset. 106
3.10 Avg. CPU costs of mining plans for mushroom dataset. 107
3.11 Avg. CPU Costs of Mining Plans for PUMSB Dataset. 108
3.12 Optimization Gains. 109
3.13 Average Local vs. Global CFIs. 109

4.1 The OLAPH Architecture with a Running Example Using CPT (GFVT13). 119
4.2 Generation of intervaled app usage events. 120
4.3 Sample clusters of frequently visited locations. 120
4.4 Meaningful and Non-meaningful Sequences. 123
4.5 Number of Users with Days of Data Collected. 130
4.6 Distribution of Data Sizes Collected Across 71 Mobile Users. 130
4.7 Characteristics of Mobile Context Events and Generated Sequences. . . . 130
4.8 Distribution of Sequence Lengths Per Log Type: App Usage and Location Logs. 131
4.9 Distribution of Sequence Lengths Per Log Type: Call and SMS Logs. . . 131
4.10 Accuracy of Models with Order k=5. 133
4.11 Training Execution Times and Model Sizes for the Six Sequence Models. 134
4.12 Impact of Varying Order k for AKOM: Accuracy. 135
4.13 Impact of Varying Order k for AKOM: Training Time and Model Size. . . 135
4.14 Accuracy vs. Training Times by Number of Weeks. 137

List of Tables

2.1 Example Rules to Illustrate Rule Redundancy. 21

2.2 **Thresholds** 48

2.3 **Online Query Settings** 48

2.4 Discretized Attributes: Bike Sharing Dataset. 58

2.5 Comparison of Association Rule Visualization Techniques. 67

2.6 User Study Schedule. 69

3.1 List of Notation. 84

3.2 Notation Used in Cost Estimates. 93

3.3 Summary of the Six Mining Plans. 100

4.1 Notation Used in OLAPH. 114

4.2 Examples of Mobile Context Data. 115

4.3 Example Sequence DB. 116

4.4 Sequence DB. 125

4.5 1^{st} Order Transition Probability Matrix. 125

4.6 2^{nd} Order Probability Matrix. 125

4.7 On-device Performance of OLAPH. 139

1

Introduction

The focus of this dissertation is pattern mining and sense-making support for enhancing user experience. In particular, this dissertation addresses three problems that limit the utility of data mining, namely, (a.) lack of interactive exploration tools for mined patterns, (b.) insufficient support for mining localized patterns, and (c.) high compute requirements of data mining prohibiting mining of patterns on smaller compute units such as a smartphone. Our solutions to these problems promise to be impactful in improving the adoption of data mining by analysts in different fields and over disparate data sources ranging from scientific datasets to mobile sensor and app data.

1.1 Motivation

1.1.1 Need for Interactive Mining

Mining of associations and correlations from huge data sets is critical for applications ranging from market basket analysis (AS94a), bioinformatics (WCWL12) to intrusion detection and web usage mining (LOPS04). However, even the most advanced mining approaches (HPY00, ZPOL97, Bor15) are faced with two key challenges, namely, (a)

unacceptably high response times that are not suitable for interactive analysis (performance); and (b) lack of support for sense-making of rule relationships (usability). Existing rule mining algorithms (AS94a, HPY00, ZPOL97) tend to be compute-intensive, rendering even their fast implementations, such as (Bor15), inadequate for interactive analysis. Mining systems with delayed response time risk losing a user's attention and, more importantly, are often unacceptable in mission critical applications.

While significant focus has been placed on improving the performance of mining algorithms (AY01a, ZPOL97, Bor15), usability of mining systems suffers from the fact that a large number of patterns is typically generated (ARu15, JB02). Figure 1.1 depicts how the number of frequent itemsets range from thousands to millions for different benchmark datasets at different support % values. A detriment is the lack of support for interactive exploration of these high numbers of rules generated with diverse parameter settings and the relationships among these rules.

Recent works on rule relationships (CLWY13, Cao13) and actionable high-utility rules (SYLC15) have begun to make significant advances in defining functions for measuring the utility of rules and complex rule relationships other than the traditional frequency-based measures. Qin et al. (QKW$^+$17) study rule relationships in the specific domain of multi-drug adverse reactions. However, analysts using these advanced techniques would still need to sift through the generated list of rules manually. That is, the focus of these state-of-the-art approaches is not the visual support of the rule discovery and exploration process. Part I of this dissertation focuses on developing interactive frameworks for the guided exploration of mined patterns and their relationships.

1.1.2 Importance of Localized Patterns

Different aspects of rule mining have received attention in the existing literature. In particular, (a) improving the efficiency of mining algorithms (AS94a, BMUT97); (b) gener-

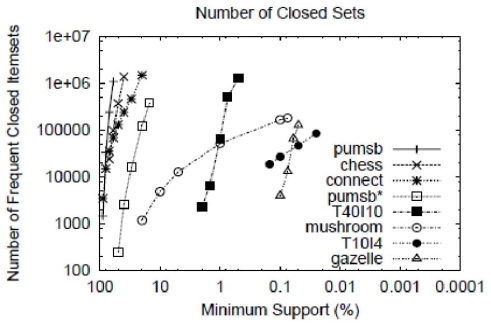


Figure 1.1: CFIs Distributed by Datasets. (ZH02)

Company	Title	Location	Gender	Age	Salary
IBM	QA Lead	Boston	M	30-40	60K-90K
IBM	Sw Engg	Boston	F	20-30	90K-120K
IBM	Engg Mgr	SFO	M	20-30	90K-120K
Google	Sw Engg	SFO	F	20-30	90K-120K
Google	Sw Engg	Boston	F	20-30	90K-120K
Google	Sw Engg	Boston	M	20-30	90K-120K
Google	Tech Arch	Boston	M	40-50	120K-150K
Microsoft	Engg Mgr	Seattle	F	30-40	90K-120K
Microsoft	Sw Engg	Seattle	F	30-40	90K-120K
Facebook	QA Mgr	Seattle	F	30-40	90K-120K
Facebook	QA Engg	Seattle	F	20-30	30K-60K

Figure 1.2: The Example Salary Dataset. (MRW14)

ating rules at different data granularities (SA95, HF95); (c) mining rules over heterogeneous data types (SA96); (d) defining rule interestingness measures (WCH07); and, (e) parameter space based interactive rule exploration (LMR$^+$13, MLB$^+$13, MLT$^+$18). All these efforts, including part I of this dissertation work on the PARAS / FIRE framework (LMR$^+$13, MLT$^+$18), focus on discovering *global* rules valid in the entire dataset. Yet *local* rules valid for subsets of the dataset, while potentially different from global rules, are often also of tremendous importance to analysts. The problem of *online discovery of localized rules* from subsets of data has received little attention, and is topic of the part II of this dissertation.

Simpson's paradox (Sim51) establishes that *local* patterns can be very different from *global* patterns. We illustrate this phenomenon in the context of rule mining. The *salary* dataset (Table 1.2) shows salary information of anonymized IT employees in different regions of the United States. Based on the complete dataset, a global salary trend given by rule $R^G = (A0 \rightarrow S2)$ [1] can be mined. Rule R^G means that the employees with *age* between 20 and 30 usually have *salaries* ranging between \$90K and \$120K. R^G has 45% support, i.e., it holds true for five out of the total of eleven records. It also has a confidence

[1]For rule mining over quantitative data (SA96), the attribute-value pairs are discretized into disjoint intervals (e.g., Age 20-30,30-40, and so on.). We denote the intervals for each dimension as Age = {A0,A1,A2,...}, Salary = {S0,S1,S2,...},and so on. Here, A0= (Age in $20-30$ years) and S2 = (Salary in $90K-120K$).

of 83% (5/6). Next, if an analyst wants to learn the local trends for the *female employees in Seattle* (here, the last four records), a *localized* rule, namely, R^L = (A1 $\rightarrow$ S2) will be discovered with a 75% support and a 100% confidence. Interestingly, the *global* rule R^G does not hold true in this subset.

The above example highlights that *local rule* R^L will be hidden in the *global* context unless the analyst lowers the minsupport to $< 27\%$ (possibly outputting overwhelmingly large set of rules). Even if, in the *global* context, R^L is discovered as a low support rule, the analyst may not discover that this low ranked rule R^L is prominent in a local context. In general, as trends may vary greatly from region to region, from job to job and across different age groups, *global* patterns may not represent the dataset adequately. Part II of this dissertation addresses the challenges of mining local rules from data sets of interest in near real-time.

1.1.3 Mining Patterns from Mobile Sensor and App Data

While data mining techniques such as rule and sequence mining are well established as powerful pattern discovery tools in domains such as science, finance and retail (AS94b, AY01b, PHMA$^+$04); these techniques are considered too compute-intensive to execute on mobile devices (RPA$^+$12, LBM$^+$17). Modern day mobile devices are capable of logging their sensor and apps data such as app usage, location, call/SMS logs, battery usage, and calendar events. Such rich variety of mobile context data can be leveraged to learn mobile usage patterns. For example, a mobile user's contexts such as "which apps the user typically uses at home versus at work", "how much time the user spends at home versus at work","who does she communicate with regularly", "which places does she typically visit", and many more such patterns can be learnt. Such discovered patterns enable a variety of context-aware applications ranging from automating frequently repeated tasks (e.g., turning phone to mute during meetings, turning Wi-Fi on when returning home and off

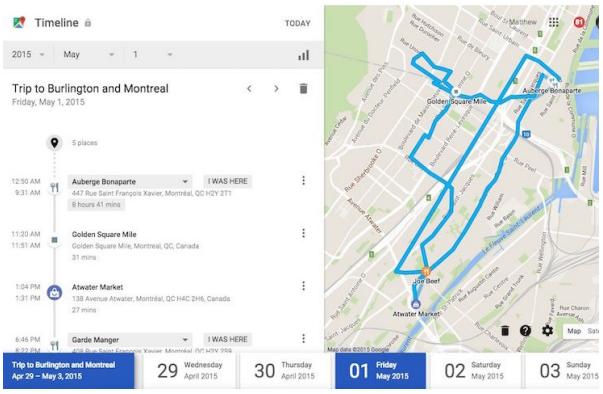


Figure 1.3: Google Timeline.

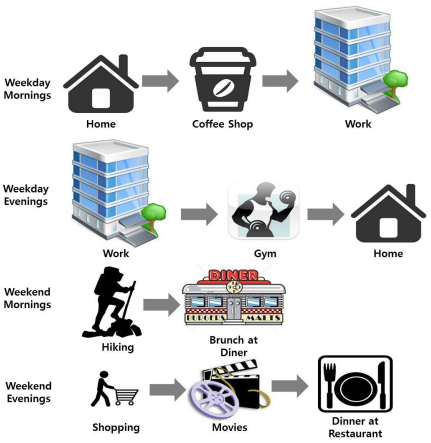


Figure 1.4: Example User Behavioral Sequential Patterns.

when away to conserve battery) to providing personalized recommendations (e.g., restaurant recommendations when a user is likely to eat out). Google Timeline 1.3 is one such service where the smartphone collects user's location history and asks for feedback to improve Google maps location services and possibly also improve the user's experience.

Initial research efforts towards learning frequent mobile user patterns have been primarily limited to offloading computation to the cloud as done in (Nat12, XLZ+18). However, cloud-based applications are heavily dependent on availability of the internet. One example application that may fail due to its dependence on the internet is a map application. Not too long ago, our map applications failed in bad coverage areas such as tunnels. Map applications now have solved the problem by downloading directions to the phone and using an offline mode. If context-aware services on mobile devices are cloud-based, they may also be unavailable during poor network coverage. Context-aware services often also require user's mobile context data for training purposes. Thus, in a cloud-based approach, such as (Nat12, XLZ+18), user's data such as visited locations and call logs, must be uploaded to the cloud. This is both a privacy concern and utilizes network bandwidth. Few weeks of such mobile context data can be about 10s of MB, which not only is fairly large but also may not be important to upload to the cloud. Developing on-device

intelligence via machine learning on mobile context data has recently gained attention of researchers (LBM+17). For example, Srinivasan et al. (SMM+14) learn co-occurrence patterns over user's mobile context data using a variant of Apriori (AS94a) that runs on the phone. Rawassizadeh et al. (RMD+16) systematically study the feasibility of discovering frequent behavior patterns focusing on temporal granularity of frequent itemsets. Similarly, in (MSW14) preliminary results related to sequential behavior mining over mobile device data are shared. The focus of these works (RMD+16, MSW14) is user profiling of mobile usage behavior. More recently, Srinivasan et al. (SKJ18) further present RuleSelector, an interactive rule selection tool to allow smartphone users to browse, modify, and select action rules from a small set of summarized rules presented to them. Part III of this dissertation, continuing on the theme of enhancing data mining for practical applications, explores how data mining approaches such as sequence prediction modeling may be adopted to fit this paradigm of mobile sensing and context-awareness.

1.2 Research Challenges Addressed in This Dissertation

Below research challenges are addressed in this dissertation. First, the challenges hindering the usability of rule mining systems as an interactive tool are described. Next, we discuss the challenges of employing a preprocess-once-query-many (POQM) paradigm for mining localized rules. Finally, the challenges of applying data mining, in particular sequence prediction modeling, to mobile sensor and app data.

1.2.1 Interactive Rule Exploration

With respect to interactive rule exploration, below we discuss the challenges related to both the management of generated patterns (rules) and their visual exploration.

1. **Managing Large Number of Rules.** In general, for k items in a dataset $\mathcal{D}$, a

total of 2^k-1 itemsets can be composed (GKP94). Clearly, for a dataset with several hundreds or thousands of items, it is prohibitively expensive to store all rules individually. This raises two critical requirements for rendering the precomputation of final rulesets practical. First, we must design a data structure to compactly prestore rulesets. Next, we must develop an efficient strategy that uses this compact storage at query-time for processing online mining requests.

2. Managing Rules for All Parameter Settings. To facilitate a direct lookup of rulesets given specific input parameter settings, the rules must be indexed on the two-dimensional space of *support* and *confidence* parameters. Yet *minsupport* and *minconfidence* values input by users are from a continuous domain [0,1]. For a large dataset, prestoring rules for all possible parameter settings will be extremely challenging. On the other hand, some *regions* of the parameter space may be *stable*, such that despite changes in the input parameter settings, the output ruleset remains unchanged. This raises the need for effective indexing of rules within the parameter space with an understanding of *stable regions*.

3. Numerous Ways of Sense-making. One of the biggest challenges of mining interesting association rules is that there are numerous "interestingness" parameters and that interestingness tends to be domain-dependent (QKW$^+$17). Sense-making in the context of association rules means not only discovering co-occurring itemsets with high support and confidence, but also understanding and distinguishing interesting rules from obvious ones. For example, in banking data, if the restriction is those maintaining below the minimum limit will be charged a monthly maintenance fee, then a low account balance and fine paid will have high support and/or confidence yet be an obvious rule. Instead, an interesting rule to learn may be "rent check in the beginning of every month causes the account to go low balance". However, its support and/or confidence may not be as strong as those for the obvious rule, because other reasons for the low balance may exist.

Overall, there is no fixed rule of thumb for sense-making of association rules. Interestingness depends on several aspects such as the domain, the particular dataset and the user's perspective. Thus, a tool that allows users to systematically explore the mined rules is needed and not just one that simply presents only the rules with high scores.

4. Lack of Parameter Space-driven Exploration. Several rule visualization techniques have been proposed (HFH$^+$09, ARu15, JB02), yet none provide a broad parameter space view for rule mining. Neither do they provide support for parameter selections or refinements. In the absence of parameter space insights, analysts may not be aware of the appropriate *minsupport* and *minconfidence* parameter settings required to obtain the rulesets of interest from a particular dataset. While high parameter settings may discover too few rules, too low parameters may present the analysts with an overwhelmingly large number of rules. Figure 1.1 (taken from (ZH02)) depicts how the distribution of closed frequent itemsets (CFIs) differs significantly from data set to data set. While gazelle and T10I4 benchmark data sets (UCI15) have most CFIs concentrated around only 0.1% support, chess and mushroom data sets instead feature $\geq$ 2000 CFIs at 94% and 50% support, respectively.

5. Lack of Parameter Space-based Recommendations. Finding the top-k rules from a dataset is a sought after feature. Existing systems (HFH$^+$09, ARu15, JB02) can only extract top-k rules based on *one* parameter (either support or confidence) at a time. However, certain rules may have high support yet low confidence, and vice-versa. Such a two-dimensional combination of support and confidence (or, any combination of rule interestingness measures, including those proposed by (SYLC15, QKW$^+$17)) for top-k rule extraction is not yet available. This feature has the potential to improve the usability of interactive mining systems by providing parameter space-based recommendations.

6. Limited Insights into Rule Relationships. For a set of rules that consists of identical itemsets, yet items may be distributed differently in antecedent and consequent, few

dominating rules from the set may implicitly imply the others, defined as redundancy relationship among rules (AY01a). These relationships could be leveraged to represent the complete ruleset with just a subset of rules, thus reducing the clutter. However, in (LMR$^+$13) we discover that redundancy is a query-time phenomenon, i.e., redundancy among rules must be resolved based on the user selected parameters. Unfortunately, existing association rule tools (HFH$^+$09, ARu15, JB02) lack a mechanism to manage these dynamic rule relationships. Additionally, graphical representation of other rule relationships (CLWY13, QKW$^+$17) are also required.

7. Lack of Support for Ruleset Comparison. When existing systems are used for discovering interesting rules in a given dataset, analysts must go about a tedious and time-consuming trial-and-error process of parameter selection interleaved with rule generation and sifting through the extracted rules to discover interesting ones. Ability to compare rulesets across different parameter values with minimal clicks in real-time will truly enable interactive rule exploration.

1.2.2 Mining of Localized Rules

Towards mining of localized rules, the below challenges are addressed by the COLARM framework described in Part II of this dissertation.

1. Offline Data Pre-processing. Pre-computing *locally* frequent itemsets is complex. For a dataset containing m records, the total number of possible subsets that the analyst can select at query-time is given by the Bell Number (GKP94) as $\sum_{i=0}^{m}\{{m \atop i}\}$, where $\{{m \atop i}\}$ is the Stirling number. Clearly, pre-storing all possible partitions and their corresponding frequent itemsets is infeasible. An effective indexing strategy for compactly pre-storing itemsets is needed to render POQM feasible in a local context.

2. Online Focal Subset Selection. We denote the subset of interest to the analyst as the *focal subset*. The focal subset is specified as a query-time parameter and may

range from as few as a single record to as many as all records of the dataset. Clearly, a versatile online query processing strategy is required that can utilize the pre-computed index structure to efficiently identify the focal subset and the candidate frequent itemsets.

3. Online Localized Rule Verification. Most existing online mining solutions (AY01a, DPDK11) only verify the *minsupport* (as it is relatively inexpensive to compute), while avoiding the costlier *minconfidence* checks. Due to the importance of null-invariant measures (WCH07), we propose to verify both *minsupport* and *minconfidence*. However, as these measures can only be verified at query-time, efficient mechanisms are needed for the same.

1.2.3 Mobile Context Sequence Prediction

The key challenges of on-device sequence prediction modeling over mobile context data are twofold, namely, (a.) on-device computation of resource-intensive sequence prediction modeling (training), and (b.) achieving reasonable prediction accuracy. Below we discuss these challenges in detail.

1. On-device Sequence Training and Prediction. Overall, both storage and computation resources are limited on smart phones. The mobile context data when collected in raw form can amount to approximately 40+ MB of mobile space in a matter of few weeks. Unless the context logs are processed in a structured manner, it will be difficult to draw any insights from the logs. Therefore, plain life-logging is unfeasible on mobile devices. Further, sequence prediction modeling consists of two steps, namely, (i.) sequence data generation, and (ii.) training the sequence-based context model. It is well known that the sequence data generation dominates the cost of sequence prediction modeling (FVLG$^+$16). For few weeks of data, the sequence DB generation alone may take 10s of minutes when run on a mobile device. Therefore, the goal of this work is to identify sequence models that can feasibly be deployed on mobile devices without impacting the

user experience.

2. Achieving High Accuracy. In CPT+ (GFVRT15) several data sets have been tested with a variety of sequence prediction models. As shown by Gueniche et al. (GFVRT15), the accuracy achieved by the best of the models over these benchmark datasets is low (maximum accuracy $< 74\%$), and mostly in the range of 30-60%. These results are discouraging at first glance as sequence prediction models appear to have low accuracy in general. However, on closer observation we find that the tested datasets are only loosely sequential. While the majority of those datasets are webpage visitations, the rest are sentence databases treated as sequences of words or characters. In contrast, we find that mobile context data is inherently sequential as they correspond to a time-series of events. Thus, in this work, we want to evaluate and identify which of the sequence prediction models would work for mobile context data. Further, the order k of a sequence model is an important factor. Order k means up to k previous events can be used to predict the current event. A higher value of k helps improve accuracy but induces very high state complexity and makes the models large (GFVRT15). Thus, one challenge we target in this paper is to explore "what minimal value of k of model would be reasonable to store on the smartphone that could achieve high accuracy while having reasonably small model state complexity and model size". Further, we explore how much longitudinal data must the model contain that will not be huge, yet will achieve reasonable accuracy suitable for real-time predictions.

1.3 Proposed Solutions

1.3.1 PARAS and FIRE Frameworks for Interactive Rule Exploration

Our PARAS framework (LMR$^+$13) proposes the back end innovations including mining performance optimizations, data modeling and indexing. Then we introduced the notion

of rule distribution visualization on the parameter space in our short paper (MLW$^+$13), and extended these preliminary ideas by including the dual-space interactive rule visualization paradigm. Finally, in the proposed ***F**ramework for **I**nteractive **R**ule **E**xploration* **(FIRE)** (MLT$^+$18) we introduced the full visual paradigm that features innovative visual displays and interactions to enable analysts to conduct rule exploration in real-time. Therefore, in part I of this dissertation we developed innovative visualizations for rule exploration that are described across several publications (LMR$^+$13, MLB$^+$13, MLW$^+$13, MLT$^+$18). Below we list the key contributions of the PARAS and FIRE frameworks.

• We first propose the parameter space model, called **PARAS**, that organizes association rules in a space of query parameters. Instead of maintaining the huge number of individual rules in an infinite parameter space, our compact space representation, also called *PSpace*, abstracts rule sets at a coarse granularity of stable regions.

• The **PARAS** framework offers efficient algorithms for offline **PSpace** index construction and provides *stable region*-aware indexing where each rule is stored exactly once and stable regions are compactly stored in a region neighborship graph.

• **PARAS** supports a rich set of novel *exploratory mining queries* beyond traditional rule mining. Effective strategies for online processing of these novel query types using the **PSpace** index are also developed.

• We observe that redundancy relationship among rules is a query-time phenomenon. Thus, we establish a theoretical foundation of redundancy relationships among associations that allows us to pre-compute compact abstractions of redundancy. This abstracted redundancy meta-knowledge enables effective *redundancy resolution* at query-time. For N rules in the output ruleset, our approach reduces the complexity of online redundancy resolution (AY01a) from $\mathcal{O}(2^N)$ to linear time ($\mathcal{O}$(N)) by performing a $\mathcal{O}(N^2)$ time offline redundancy abstraction step.

• Our extensive experiments using IBM Quest (AS94a), webdocs (LOPS04) and other

benchmark datasets demonstrate that **PARAS** achieves 2 to 5 orders of magnitude improvement over commonly used techniques in online rule mining.

• We further propose a novel visual rule exploration framework, called **FIRE**. **FIRE** supports rule exploration at two layers of abstractions, namely, the overall parameter space view (**PSpace**) and the detailed rule space view (**RSpace**). Both layers are supplemented with innovative features and interactions.

• The **PSpace** view displays the overall distribution of rules within the space of interestingness parameters (such as support and confidence). Salient features of the PSpace view include (a) parameter recommendations via stable region abstractions; (b) rich insights into region-wise rule cardinality; (c) capture of rule redundancy relationships; and (d) the rule cardinality skyline to explore alternative results.

• The PSpace-RSpace hierarchical relationship enables real-time exploration using the two views. The stable regions are selected via the **PSpace** view and the **RSpace** view shows detailed information about particular rule sets within the selected region. To enable interactive filtering of rules, the RSpace view includes antecedent/consequent auto-fill filters and rule sorting on parameters.

• While the tabular RSpace view introduced in the short paper (MLW$^+$13) lists rules with detailed information, to facilitate visual sense-making of rule sets, we now introduce an effective visualization technique called **rule glyphs** for *graphically* representing association rules. To visually capture key properties of rules and their interestingness measures, we design three variants of the RSpace glyph views, namely, *lined*, *connected* and *filled* glyphs. We also explore various glyph placement strategies that enable analysts to gain insights into clusters of similar rules as well as to detect outliers of rules that deviate significantly from the norm.

• In this dissertation, we present a case study comparing **FIRE** with the state-of-the-art rule visualization techniques in ARulesViz R package (ARu15). The case study

is qualitative in nature where a researcher learns to use FIRE for the first time, and is tasked with documenting his interactions while exploring a new dataset of interest. The researcher also explored the same dataset using a combination of 10 rule visualization techniques in the ARulesViz R package. He concluded that FIRE enabled him to discover patterns in the dataset that were either undiscovered or cumbersome to derive using state-of-the-art techniques.

• We also conducted a user study to evaluate the diverse capabilities of our FIRE visual paradigm. 22 participants were used to evaluate the usability and effectiveness of various features of the FIRE framework over several benchmark datasets (UCI15). The user study is comprised of two evaluations, namely, of the PSpace view and the new RSpace glyph view, respectively. This extensive user study provides evidence that our proposed FIRE visualizations are efficient and effective in helping analysts to understand the rule distribution over the parameter space and to gain rich insights into the rule relationships via the RSpace glyph view and the glyph placement strategies.

1.3.2 COLARM Framework for Localized Rule Mining

In Part II of this dissertation we extended the POQM paradigm towards online mining of localized rules. For the localized rule mining scenario, due to the uncertain nature of subset selection, one cannot guarantee that a POQM solution always outperforms a naive solution of running the rule mining algorithm from scratch on the chosen subset. Thus, there is scope for comparison of alternative execution plans and plan selection using cost-based optimization. The key contributions in part II of this dissertation research are summarized as follows:

• We formulate the *online localized rule mining query*. In addition to the traditional rule interestingness thresholds (*minsupport* and *minconfidence*), a localized rule mining query introduces two new parameters, namely, *range* and *item* to enable *focal* subset

selection and item attribute specification, respectively.

• We introduce the ***M**ultidimensional **I**temset **P**artitioning* index (MIP-index). MIP-index offers efficient mining performance by utilizing pre-computed results, while still allowing the flexible query-time selection of data subsets. This is achieved by compactly pre-storing two features of the itemsets, namely, (a) the multidimensional bounding box of the itemsets; and, (b) the items composing the itemsets.

• We isolate different online mining steps and optimize each of these steps by employing novel optimization principles in the context of online localized rule mining. We construct several alternative *mining plans* by pipelining the different optimized steps. We develop a cost model to estimate benefits and overheads of these plans. We further develop a **C**ost-based **O**ptimizer for **L**ocalized **A**ssociation **R**ule **M**ining, in short **COLARM**, for online selection of the most cost-effective plan.

• Our experimental study over benchmarks such as chess, mushroom and PUMSB datasets from UCI ML repository (UCI15) establishes that our COLARM optimizer is highly accurate ($\sim$93%) in selecting the most efficient mining plan for a rich diversity of online mining requests.

1.3.3 OLAPH Sequence Prediction over Mobile Context Data.

In the part III of this dissertation, the design and implementation of **O**n-device **L**ife-logging **A**nd **P**redicting **H**abitual Behavior **OLAPH** service for mobile context prediction is presented. The key contributions are as follows:

• OLAPH is a service that runs *entirely on the phone* to transform logs of longitudinal mobile context data into compact sequence models to predict a mobile user's context in real-time; thus making life-logging on mobile devices feasible. For example, 40+ MB of raw data can be transformed into a compact sequence model of approximately 0.2 MB.
• One key contribution of OLAPH is that the sequence prediction modeling is adapted

to fit the paradigm of context-awareness and mobile sensing. Towards that end, OLAPH adapts several key concepts for modeling mobile data as sequences such as compressing context logs to intervaled context events, adding generalized time features, and identifying meaningful sequences via filter expressions.

• We conducted a thorough evaluation of OLAPH using four types of mobile context logs (app usage, location, call and SMS) from 85 users. We demonstrate several key observations such as models trained on 4 consecutive weeks of data provide good prediction accuracy with reasonable training times. Based on our analysis, we develop guidelines for feasibly scheduling OLAPH training on phones.

• We present a comparative evaluation of six sequence prediction algorithms that train and test over mobile context data. We present interesting, and, in some cases, contrasting observations of applying sequence models over mobile context data. The best model AKOM achieves a median of 90+% sequence prediction accuracy. As AKOM has higher state complexity compared to other first order markov approaches, we identify that AKOM with order k = 5 is the best suited to achieve high accuracy yet have reasonable state complexity and model size for our mobile context dataset.

1.4 Dissertation Organization

Chapter 2 describes PARAS and FIRE frameworks including their extensive performance evaluations as well as user and case study. The COLARM work for localized rule mining together with its experimental results are explained in Chapter 3. Further, Chapter 4 provides details of the OLAPH sequence prediction modeling of mobile context data and its evaluation. The conclusion of this dissertation is presented in Chapter 5 and future works are discussed in Chapter 6.

2

Interactive Rule Exploration with PARAS and FIRE

2.1 Foundation of Parameter-driven Rule Mining

In this chapter of my dissertation I first review the foundation of parameter space model from (LMR$^+$13). I then present the backend innovations of PARAS framework followed by the FIRE visual engine.

The core principle we adopt for our interactive rule exploration framework corresponds to the *preprocess-once-query-many* (POQM) paradigm (AY01a). In an offline step, we extract all rules from a dataset that satisfy a minimum *primary support*. Then we compactly index the large number of extracted rules for subsequent interactive rule exploration by analysts. In particular, we adopt the *parameter space-driven approach* (LMR$^+$13) which, in the context of rule mining, consists of a two-dimensional space of *support* and *confidence*, as described below.

2.1.1 Parameter Space Model

We use the *parameter space* as a model for managing, retrieving, and exploring the *association rules* of a dataset. For a dataset $\mathcal{D}$, this space contains the rules of $\mathcal{D}$, denoted by $\{\mathcal{R}\}^{\mathcal{D}}$. Each dimension p_k of the space represents an interestingness measure, such as support, confidence and lift (WCH07).

Definition 2.1 ***Parameter Space:*** *Given a dataset $\mathcal{D}$ and d user-chosen interestingness parameters each denoted as p_i, we use a d-dimensional parameter space, denoted by $\mathcal{P} = \{p_1, \ldots, p_k, \ldots, p_d\}$ for organizing the rules $\{\mathcal{R}\}^{\mathcal{D}}$ for $\mathcal{D}$. Each rule $\mathcal{R}_j$ is represented by its* ***parametric location*** *($\mathcal{R}_j$.value(p_1), ..., $\mathcal{R}_j$.value(p_k), ..., $\mathcal{R}_j$.value(p_d)) where $\mathcal{R}_j$.value(p_i) denotes the value of the i^{th} parameter for rule $\mathcal{R}_j$.*

For simplicity, we henceforth work with a two-dimensional parameter space using *support* and *confidence* as dimensions. A *parametric location* ℓ_1 is defined by a combination of support and confidence values, denoted by (ℓ_1.supp, ℓ_1.conf) (Fig. 2.2). Many association rules may map to the same *parametric location*, e.g., (XZ $\Rightarrow$Y) and (YZ $\Rightarrow$X) both map to (0.1,0.5). Therefore, all rules mapping to the same parametric location can be compactly indexed in our parameter space model.

The user specified *minsupp* and *minconf* values may range between 0 and 1, making the number of locations infinitely large. Yet, the number of distinct *parametric locations* is typically much smaller than the actual number of association rules. Thus, if the association rules are grouped by their *parametric locations*, there is a modest number of such locations compared to the number of actual rules.

2.1.2 Stable Region Abstractions.

In Figure 2.2, we observe that certain regions of the parameter space often either contain no rules or contain the same set across a large range of parameter settings (e.g., the shaded

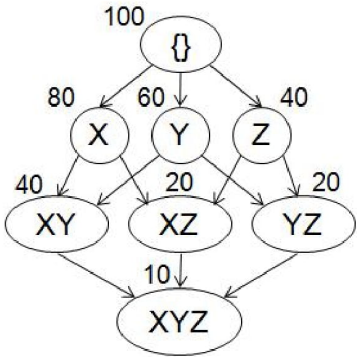


Figure 2.1: Adjacency Lattice.

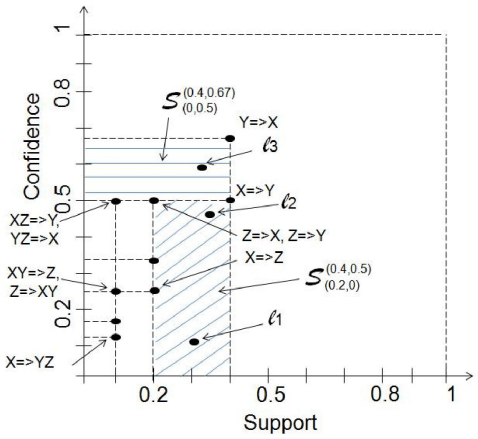


Figure 2.2: Parameter Space.

regions marked $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$ and $\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$). This leads us to the notion of stable regions, which forms our *coarse granularity abstractions* for rules.

Definition 2.2 ***Stable Region:*** *Given a parameter space $\mathcal{P}$ of d interestingness dimensions $\{p_1, \ldots, p_d\}$ for dataset $\mathcal{D}$, a* ***stable region*** $\mathcal{S}^{(upper(p_1),\ldots,upper(p_d))}_{(lower(p_1),\ldots,lower(p_d))}$ *is a d-dimensional rectangular hyperbox with extreme points ($\mathcal{S}$.lower(p_1),. . ., $\mathcal{S}$.lower(p_d)) and ($\mathcal{S}$.upper(p_1), . . ., $\mathcal{S}$.upper(p_d)) within which no matter how the parameter values are adjusted, the set of rules generated from $\mathcal{D}$ remains unchanged.*

In Figure 2.2, the shaded region $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$ is bounded by the parametric locations (0.2,0) and (0.4,0.5) as its extreme points. Let us suppose that a user inputs two separate queries Q_1 and Q_2 with the parametric locations ℓ_1 and ℓ_2, respectively. Then both parametric locations lie within $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$. Thus, we can infer that the outputs for both Q_1 and Q_2 will be the same, i.e., $\{R\}^{Q_1} = \{R\}^{Q_2} = \{(X \Rightarrow Y), (Y \Rightarrow X)\}$. However, if the user inputs another query Q_3 with parametric location ℓ_3, that lies within region $\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$), the output would only contain rule ($Y \Rightarrow X$). Thus, the output remains unchanged as long as parameter values are chosen within the bounds of a stable region, whereas the output changes when crossing between stable regions. Next, Lemma 2.1.2 establishes a neighborhood relationship among adjacent stable regions.

Consider two stable regions $S^{(usupp1,uconf1)}_{(lsup1,lconf1)}$ *and* $S^{(usupp2,uconf2)}_{(lsup2,lconf2)}$ *such that usupp2* $\geq$ *usupp1 and uconf2* $\geq$ *uconf1, then all association rules valid in stable region* $S^{(usupp2,uconf2)}_{(lsup2,lconf2)}$ *are also valid in region* $S^{(usupp1,uconf1)}_{(lsup1,lconf1)}$*. The reverse is not true.*

Proof: A rule $\mathcal{R}_i$ at a parametric location ($\mathcal{R}_i$.*supp*,$\mathcal{R}_i$.*conf*) qualifies to be output for any mining request with *minsupp* $\leq \mathcal{R}$.*supp* and *minconf* $\leq \mathcal{R}_i$.*conf*. Assuming rule $\mathcal{R}_i$ belongs to stable region $\mathrm{S}^{(usupp2,uconf2)}_{(lsup2,lconf2)}$, $\mathcal{R}_i$.*supp* = *usupp2* and confidence $\mathcal{R}_i$.*conf* = *uconf2*. Therefore, $\mathcal{R}_i$ will also be valid in region $\mathrm{S}^{(usupp1,uconf1)}_{(lsup1,lconf1)}$ as *usupp1* $\leq \mathcal{R}_i$.*supp* and *uconf1* $\leq \mathcal{R}_i$.*conf*. ■

Definition 2.3 ***Lending Neighbor Stable Region:*** *Consider two neighbor stable regions* $S^{(usupp1,uconf1)}_{(lsup1,lconf1)}$ *and* $S^{(usupp2,uconf2)}_{(lsup2,lconf2)}$*, related by Lemma 2.1.2. We then say that* $S^{(usupp2,uconf2)}_{(lsup2,lconf2)}$ *is the* ***lending neighbor stable region*** *for* $S^{(usupp1,uconf1)}_{(lsup1,lconf1)}$*, in short, l-neighbor.*

By Def. 2.3, in Figure 2.2 $\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$ is the *lending neighbor stable region* for $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$. $\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$ lends rule (Y $\Rightarrow$ X) to $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$, as (Y $\Rightarrow$ X) first appears in $\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$ and is also valid for $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$. We refer to the *lending neighbor stable region(s)* as *l-neighbors* in short.

We partition the *parameter space* $\mathcal{P}$ into a finite number of *non-overlapping stable regions*, denoted by $\{\mathcal{S}\}$. The non-overlapping property among stable regions can be guaranteed due to our approach of modeling the regions (Section 2.3.3). Utilizing the concept of *neighbor stable regions* (Def. 2.3), for each such *stable region*, we maintain (a.) the rules that are valid within that region and (b.) the links to its *l-neighbors*. This partitioning of the parameter space into stable regions enables us to process novel exploratory online queries as well as to recommend query parameter settings based on user interest.

Rule	Support	Confidence
X⇒YZ	S(X ∪ Y ∪ Z) = 0.1	S(X ∪ Y ∪ Z) / S(X) = 0.125
XY⇒Z	S(X ∪ Y ∪ Z) = 0.1	S(X ∪ Y ∪ Z) / S(X ∪ Y) = 0.25
XZ⇒Y	S(X ∪ Y ∪ Z) = 0.1	S(X ∪ Y ∪ Z) / S(X ∪ Z) = 0.5
X⇒Y	S(X ∪ Y) = 0.4	S(X ∪ Y) / S(X) = 0.5
X⇒Z	S(X ∪ Z) = 0.2	S(X ∪ Z) / S(X) = 0.25

Table 2.1: Example Rules to Illustrate Rule Redundancy.

2.1.3 Rule Redundancy in Association Rules

Aggarwal et al. (AY01a) defined rule redundancy relationships, such that redundant rules may be filtered out to present succinct results to the user. The redundant rules could always be derived on demand, if so desired. We examine how these redundancy relationships can be identified in the parameter space model. In particular, redundancy can be of two types (AY01a), as defined below.

Definition 2.4 ***Simple Redundancy:*** *Let $A \Rightarrow B$ and $C \Rightarrow D$ be two rules such that the itemsets A, B, C and D satisfy the condition $A \cup B = C \cup D$. The rule $C \Rightarrow D$ is* ***simply redundant*** *with respect to the rule $A \Rightarrow B$, if $C \supset A$.*

Definition 2.5 ***Strict Redundancy:*** *We consider two rules generated from itemsets X_i and X_j respectively such that $X_i \supset X_j$. Let $A \Rightarrow B$ and $C \Rightarrow D$ be rules satisfying $A \cup B = X_i$, $C \cup D = X_j$, and $C \supseteq A$. Then the rule $C \Rightarrow D$ is* ***strictly redundant*** *with respect to the rule $A \Rightarrow B$.*

The concept of redundancy can be illustrated using the rules generated from the lattice (Figure 2.1) as listed in Table 2.1. Based on Definitions 2.4 and 2.5, if a rule $\mathcal{R}_1$ is *simple* or *strict redundant* with respect to another rule $\mathcal{R}_2$, then $\mathcal{R}_2$ is said to *simple* or *strict dominate* $\mathcal{R}_1$, respectively. In Table 2.1, the rule (X ⇒ YZ) *simple dominates* the rules (XY ⇒ Z) and (XZ ⇒ Y) (Def. 2.4). In Table 2.1, the rule (X ⇒ YZ) *strict dominates* rules (X ⇒ Y) and (X ⇒ Z) (Def. 2.5). In general, a rule may be dominated by several *dominating rules* and may in turn dominate several other *dominated rules*.

2.2 Offline PSpace Construction

Our proposed **offline PSpace construction** (Algorithm 1) is composed of three tasks, task 1: generate all associations; task 2: compute stable regions; and task 3: determine neighborhoods among regions. To perform the above tasks we adapt the algorithms *ConstructLattice* and *GenerateRules* from (AY01a). For a dataset $\mathcal{D}$, an adjacency lattice $\mathcal{L}$ (Figure 2.1) is constructed using the *ConstructLattice* algorithm (explained in (AY01a)). This lattice $\mathcal{L}$ is then utilized to perform the first two tasks below:

Task 1: Generate All Associations. The original *GenerateRules* algorithm (AY01a) utilizes a lattice $\mathcal{L}$ to generate non-redundant rules. This is achieved by a subroutine *FindBoundary* that, for a given itemset node $\mathcal{N}_i$ in lattice $\mathcal{L}$, returns only the *boundaryList*$^{\mathcal{N}_i}$(AY01a) of parent nodes. However, **PARAS** needs to pre-store all associations for dataset $\mathcal{D}$. By replacing *FindBoundary* with *FindFullParentList*, the list of all parent itemsets is produced, denoted by *parentList*$^{\mathcal{N}_i}$. This modified *GenerateRules* generates all rules.

Task 2: Compute Stable Regions. We further modify the *GenerateRules* algorithm such that the stable regions are constructed in parallel with rule generation, as described in subroutine *Rules* & *Regions Generator* (Algo. 1.A). To partition the parameter space into stable regions, we first compute the *cut locations*. A *cut location* is identified by the *upper* location of a stable region. For $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$ the *cut location* is (0.4,0.5). Therefore, while generating rules using lattice $\mathcal{L}$, for itemset nodes $\{\mathcal{N}_i\}$ with identical support count (denoted by S($\mathcal{N}_i$)) in $\mathcal{L}$, the *support* for the *cut location* for $\mathcal{N}_i$ = $\frac{S(\mathcal{N}_i)}{|\mathcal{D}|}$, where $|\mathcal{D}|$ (of $\mathcal{L}$ in Figure 2.1 = 100) is the total number of records. Similarly, for $\mathcal{N}_i$ and a parent node $\mathcal{N}^P_j$ ϵ *parentList*$^{\mathcal{N}_i}$, the *confidence* value for the corresponding *cut location* is given by $\frac{S(\mathcal{N}_i)}{S(\mathcal{N}^P_j)}$. In other words, the *cut location* for $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$ is the location of association (X $\Rightarrow$ Y) in the parameter space. Therefore, each stable region is constructed in parallel during

rule generation process as described in Algo. 1.A.

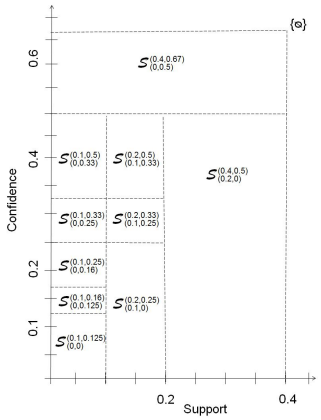


Figure 2.3: Stable Regions.

S. Regions	Neighbors	Associations
$\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$	$\varnothing$	$\{(Y \Rightarrow X)\}$
$\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$	$\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$	$\{(X \Rightarrow Y)\}$
$\mathcal{S}^{(0.2,0.5)}_{(0.1,0.33)}$	$\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$	$\{(Z \Rightarrow X),(Z \Rightarrow Y)\}$
$\mathcal{S}^{(0.2,0.33)}_{(0.1,0.25)}$	$\mathcal{S}^{(0.2,0.5)}_{(0.1,0.33)}$	$\{(Y \Rightarrow Z)\}$
$\mathcal{S}^{(0.2,0.25)}_{(0.1,0)}$	$\mathcal{S}^{(0.2,0.33)}_{(0.1,0.25)}$	$\{(X \Rightarrow Z)\}$
$\mathcal{S}^{(0.1,0.5)}_{(0,0.33)}$	$\mathcal{S}^{(0.2,0.5)}_{(0.1,0.33)}$	$\{(XZ \Rightarrow Y),(YZ \Rightarrow X)\}$
$\mathcal{S}^{(0.1,0.33)}_{(0,0.25)}$	$\mathcal{S}^{(0.1,0.5)}_{(0,0.33)} + \mathcal{S}^{(0.2,0.33)}_{(0.1,0.25)}$	$\{\varnothing\}$
$\mathcal{S}^{(0.1,0.25)}_{(0,0.16)}$	$\mathcal{S}^{(0.1,0.33)}_{(0,0.25)} + \mathcal{S}^{(0.2,0.25)}_{(0.1,0)}$	$\{(XY \Rightarrow Z),(Z \Rightarrow XY)\}$
$\mathcal{S}^{(0.1,0.16)}_{(0,0.125)}$	$\mathcal{S}^{(0.1,0.25)}_{(0,0.16)}$	$\{(Y \Rightarrow XZ)\}$
$\mathcal{S}^{(0.1,0.125)}_{(0,0)}$	$\mathcal{S}^{(0.1,0.16)}_{(0,0.125)}$	$\{(X \Rightarrow YZ)\}$

Figure 2.4: Enriched Stable Regions.

Task 3: Determine Neighborhood Relationships Among Regions. For each stable region $\mathcal{S}$, the *Neighborhood Miner* subroutine (Algorithm 1.B) adds the minimum parameter values. It also determines the list of *lending* neighbor stable regions. The final parameter space partitioned into its stable regions is depicted in Figure 2.3. Figure 2.4 lists all stable regions along with their associations and their *l-neighbor* stable regions.

In our example space, region $\mathcal{S}^{(0.4,0.5)}_{(0.2,0)}$ contains rule $\{(X \Rightarrow Y)\}$ and has the region $\mathcal{S}^{(0.4,0.67)}_{(0,0.5)}$ as its *lending* neighbor. The complete list of associations for a particular stable

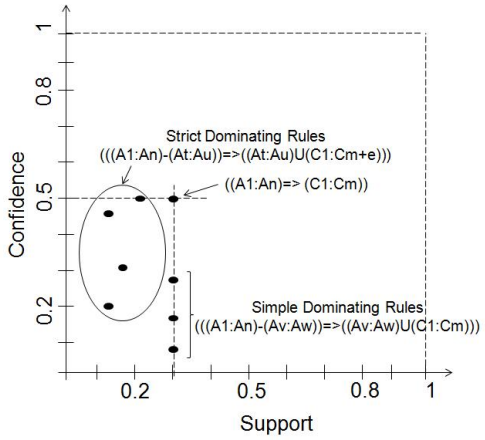


Figure 2.5: Dominating Rules.

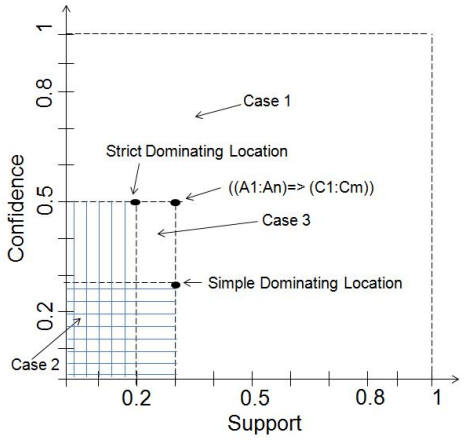


Figure 2.6: Dominating Locations.

region is given by the associations within the region plus the associations *recursively* collected from its *l-neighbors*. This way, a compact representation of stable regions along with the associations valid within them is achieved such that no association rule is stored repeatedly. The collection of regions enriched with neighbors is denoted by $\{\mathcal{S}^{+}\}$.

2.3 Redundancy Relationships

The rules produced using the stable regions constructed above may contain redundancies. In the parameter space model, our analysis derives certain properties of redundancy relationships that enable us to abstract redundancy information compactly as an offline step. When the user desires to retrieve non-redundant associations, **PARAS** can thus generate them efficiently at query-time.

2.3.1 Abstracting Redundancy Relationships

As a key observation we note that the **redundancy relationship is a query-time phenomenon**. In other words, the redundancy among rules depends on query-time input parameters rendering it impossible to eliminate a rule as redundant at an offline step. Instead, the redundancy relationships can be established only at query-time. For the parameter space in Figure 2.2, suppose that the user inputs (0.1,0.1) as query parameters. Here, the *dominating rule* $\mathcal{R}^{\gg}$ = (X $\Rightarrow$ YZ) located at (0.1,0.125) qualifies as output. Then, to produce only non-redundant rules, the *dominated rules* $\{\mathcal{R}^{\ll}\}$ = { (XY $\Rightarrow$ Z), (XZ $\Rightarrow$ Y), (X $\Rightarrow$ Y) and (X $\Rightarrow$ Z) } must be eliminated. However, if the user inputs (0.1,0.2) instead, then (X $\Rightarrow$ YZ) would not qualify for output and the rules previously deemed redundant are no longer redundant for this query. Thus, as the query parameters are supplied by the user at query-time, the decision about rule elimination can only be determined at query-time.

While elimination of the *dominated rules* can only be performed at query-time, our goal is to isolate as much as possible the redundancy relationships among rules inside the parameter space model in the preprocessing phase. This leads to the challenge that we must design a corresponding query-time strategy to produce non-redundant associations by utilizing this predetermined redundancy model. A straightforward yet expensive approach may proceed as follows. In the offline phase, for each rule $\mathcal{R}_j$, store the set of rules that dominate $\mathcal{R}_j$, denoted by $\{\mathcal{R}^{\gg}\}_j$. At query-time, if the rule $\mathcal{R}_j$ is included in the set of rules for the stable region containing query parameters (*minsupp*,*minconf*), then test if any of the rules dominating $\mathcal{R}_j$, denoted by $\mathcal{R}_i^{\gg} \epsilon \{\mathcal{R}^{\gg}\}_j$, qualify the query parameters. If yes, then rule $\mathcal{R}_j$ is eliminated, else $\mathcal{R}_j$ is output.

Simple Dominating Rules and Location. Unfortunately, a rule $\mathcal{R}_j^{\ll_{sim}}$ may be *simple dominated* by multiple rules (Def. 2.4). For example, (XY $\Rightarrow$ Z) is *simple dominated* by two rules, namely, (X $\Rightarrow$ YZ) and (Y $\Rightarrow$ XZ). Therefore, (XY $\Rightarrow$ Z) can only qualify for output if neither of its *simple dominating rules* qualify. We call them the *set of simple dominating rules of rule* $\mathcal{R}_j^{\ll_{sim}}$ as we described in Def. 2.6. Here, the *simple dominating ruleset* is denoted by $\{\mathcal{R}^{\gg_{sim}}\}_j$ = {(X $\Rightarrow$ YZ),(Y $\Rightarrow$ XZ)}.

Definition 2.6 ***Simple Dominating Rule Set:*** *For a rule $\mathcal{R}_j^{\ll_{sim}}$, its simple dominating rule set, denoted by $\{\mathcal{R}^{\gg_{sim}}\}_j$, is the set of all the rules that* ***simple dominate*** *$\mathcal{R}_j^{\ll_{sim}}$.*

1em

Algorithm 1 Offline PSpace Construction

Dataset $\mathcal{D}$ PSpace Index $\mathcal{P}$

$\mathcal{L} \longleftarrow$ ConstructLattice($\mathcal{D}$);

$\{\mathcal{S}\}, \{\mathcal{R}\} \longleftarrow$ Rules&RegionsGenerator($\mathcal{L}$);

$\{\mathcal{S}^+\} \longleftarrow$ NeighborhoodMiner($\{\mathcal{S}\}$);

$\{\mathcal{R}^+\} \longleftarrow$ RedundancyAbstractor($\{\mathcal{R}\}$);

$\mathcal{P} \longleftarrow$ PSpaceIndexConstructor($\{\mathcal{S}^+\}$,$\{\mathcal{R}^+\}$);

return $\mathcal{P}$;

1.A: Rules&RegionsGenerator($\mathcal{L}$)

$\{\mathcal{S}\} \longleftarrow \emptyset$;

*[f]Get Parent List all k-itemsets in $\mathcal{L}$, k>1.

each $\mathcal{N}_i \in \mathcal{L}$ parentList$^{\mathcal{N}_i} \longleftarrow$ FindFullParentList($\mathcal{N}_i$);

*[f]Generate rules and regions.

each $\mathcal{N}_i \in \mathcal{L}$ $\{\mathcal{R}\} \longleftarrow \emptyset$;

each $\mathcal{N}_j^P \in parentList^{\mathcal{N}_i}$ $\mathcal{R}^{i,j} \longleftarrow \{I(\mathcal{N}_j^P) \Rightarrow I(\mathcal{N}_i) - I(\mathcal{N}_j^P)\}$;

*[f]I($\mathcal{N}_i$) = itemset for $\mathcal{N}_i$

$\{\mathcal{R}\} \longleftarrow \{\mathcal{R}\} \cup \mathcal{R}^{i,j}$;

$supp \longleftarrow \frac{S(\mathcal{N}_i)}{|\mathcal{D}|}$; $conf \longleftarrow \frac{S(\mathcal{N}_i)}{S(\mathcal{N}_j^P)}$;

($\mathcal{S}^{old} \longleftarrow$ getRegion($\{\mathcal{S}\}$,$(supp, conf)$) $\neq \emptyset$) $\mathcal{S}^{old}$.addToRuleList($\mathcal{R}^{i,j}$);

Create New Region $\mathcal{S}^{new}$;

$\mathcal{S}^{new}$.addUpperParameters($supp, conf$);

$\mathcal{S}^{new}$.addToRuleList($\mathcal{R}^{i,j}$);

$\{\mathcal{S}\} \longleftarrow \{\mathcal{S}\} \cup \mathcal{S}^{new}$;

return $\{\mathcal{S}\}$, $\{\mathcal{R}\}$;

1.B: NeighborhoodMiner($\{\mathcal{S}\}$)

each $\mathcal{S}_i \in \{\mathcal{S}\}$ $\mathcal{S}_i$.findLowerParameters();

$\mathcal{S}_i$.findClosestHigherNeighbors();

return $\{\mathcal{S}^+\}$;

1.C: RedundancyAbstractor($\{\mathcal{R}\}$)

each $\mathcal{R}_j \in \{\mathcal{R}\}$ $\{\mathcal{R}^{\gg_{sim}}\}_j \longleftarrow$ CollectTopSimpleDomRules($\mathcal{R}_j$);

$\ell_j^{\gg_{sim}} \longleftarrow$ FindMaxDomLocation($\{\mathcal{R}^{\gg_{sim}}\}_j$);

$\{\mathcal{R}^{\gg_{str}}\}_j \longleftarrow$ CollectTopStrictDomRules($\mathcal{R}_j$);

$\ell_j^{\gg_{str}} \longleftarrow$ FindMaxDomLocation($\{\mathcal{R}^{\gg_{str}}\}_j$);

$\mathcal{R}_j$.AddDominatingLocations($\ell_j^{\gg_{sim}}$,$\ell_j^{\gg_{str}}$);

return $\{\mathcal{R}^+\}$;

For each *simple dominated rule* $\mathcal{R}_j^{\ll sim} \equiv ((A_1{:}A_n) \Rightarrow (C_1{:}C_m))$, all rules that potentially *simple dominate* it conform to the template $(((A_1{:}A_n)\text{-}(A_v{:}A_w)) \Rightarrow ((A_v{:}A_w) \cup (C_1{:}C_m)))$. For the rule (XY ⇒ Z) having two items X and Y in the antecedent, there are two *simple dominating rules*, considering all subsets of XY in the antecedent, namely, X and Y. Our observation is further generalized as in Lemma 2.3.1 below. For a *simple dominated rule* $\mathcal{R}_j^{\ll sim} \equiv ((A_1{:}A_n) \Rightarrow (C_1{:}C_m))$ with n antecedent items, the number of **simple dominating rules**, denoted by $|\{\mathcal{R}^{\gg sim}\}_j|$ is 2^n-2. [1]

We further observe in Figure 2.5, that all rules in the set of simple dominating rules $\{\mathcal{R}^{\gg sim}\}_j$ have the same support value as rule $\mathcal{R}_j^{\ll sim}$. Thus it is possible to uniquely identify one single location containing one or more rules that is closest to $\mathcal{R}_j^{\ll sim}$ as described in Lemma 2.3.1.

Simple Dominating Location: For each *simple dominated rule* $\mathcal{R}_j^{\ll sim}$, the set of simple dominating rules $\{\mathcal{R}^{\gg sim}\}_j$ contains a rule $\mathcal{R}_i^{\gg sim}$ closest to the dominated rule $\mathcal{R}_j^{\ll sim}$, such that $\forall \mathcal{R}_k^{\gg sim} \epsilon \{\mathcal{R}^{\gg sim}\}_j$ and (k≠i), $\mathcal{R}_i^{\gg sim}$.conf $\geq \mathcal{R}_k^{\gg sim}$.conf. The location of rule $\mathcal{R}_i^{\gg sim}$ is called the **simple dominating location**[2] of $\mathcal{R}_j^{\ll sim}$, denoted by $\ell_j^{\gg sim}$.

Strict Dominating Rules and Location. Similar to the above case of *simple dominating rules*, a *strict dominated rule* $\mathcal{R}_j^{\ll str}$ can be dominated by several *strict dominating rules* $\{\mathcal{R}^{\gg str}\}_j$. We call them the *set of strict dominating rules of rule* $\mathcal{R}_j^{\ll str}$ as defined below. In the example in Figure 2.1, for a rule $\mathcal{R}_j^{\ll str}$ = (X ⇒ Y), the *strict dominating rule set* $\{\mathcal{R}^{\gg str}\}_j$ = {(X ⇒ YZ)}.

Definition 2.7 *For a rule* $\mathcal{R}_j^{\ll str}$*, its* ***strict dominating rule set****, denoted by* $\{\mathcal{R}^{\gg str}\}_j$*, is the set of all the rules that* ***strict dominate*** $\mathcal{R}_j^{\ll str}$.

For a rule $\mathcal{R}_j^{\ll str}$, the number of such strict dominating rules can also be estimated as in Lemma 2.3.1.

[1] Proofs of the lemmas and the theorems are omitted due to space restriction and can be found in (LMR+13).

[2] Multiple rules may map to the simple / strict dominating location that collectively represents them.

For a *strict dominated rule* $\mathcal{R}_j^{\ll str}$ having n antecedent items ((A_1:A_n) $\Rightarrow$ (C_1:C_m)), the strict dominating rules $\{\mathcal{R}^{\gg str}\}_j$ conform to the template (((A_1:A_n)-(A_t:A_u)) $\Rightarrow$ ((A_t:A_u) $\cup$ (C_1: C_{m+e}))). The cardinality of **strict dominating rules**, denoted by $|\{\mathcal{R}^{\gg str}\}_j|$, is 2^{n+e}, where e is the number of additional consequents in the dominating rules within the set $\{\mathcal{R}^{\gg str}\}_j$.

Using Lemma 2.3.1, for a dominated rule $\mathcal{R}_j^{\ll str}$, the number of strict dominating rules can be determined. We further observe in Figure 2.5, that all rules in the set of strict dominating rules $\{\mathcal{R}^{\gg str}\}_j$ must have both their support and confidence values less than or equal to those of the rule $\mathcal{R}_j^{\ll str}$. Thus it is possible to uniquely identify one single rule location closest to $\mathcal{R}_j^{\ll str}$ as described in Lemma 2.3.1.

Strict Dominating Location: For each *strict dominated rule* $\mathcal{R}_j^{\ll str}$, the set of strict dominating rules $\{\mathcal{R}^{\gg str}\}_j$ contains one or more rules $\mathcal{R}_i^{\gg str}$ (in the same location) closest to the dominated rule $\mathcal{R}_j^{\ll str}$, such that $\forall \mathcal{R}_k^{\gg str} \epsilon \{\mathcal{R}^{\gg str}\}_j$ where (i$\neq$k), $\mathcal{R}_i^{\gg str}$.supp $\geq$ $\mathcal{R}_k^{\gg str}$.supp AND $\mathcal{R}_i^{\gg str}$.conf $\geq$ $\mathcal{R}_k^{\gg str}$.conf. The location of rule $\mathcal{R}_i^{\gg str}$ is called the **strict dominating location**[4], denoted by $\ell_j^{\gg str}$.

Lemmas 2.3.1 and 2.3.1 now lead us to a **key insight**, namely, they allow us to compactly store the association rules along with respective redundancy relationships. For each rule $\mathcal{R}_j$, only its two locations, namely, the *simple dominating location* $\ell_j^{\gg sim}$ and the *strict dominating location* $\ell_j^{\gg str}$ must be captured by the offline step. At the online query processing phase, these two locations are sufficient to determine the redundancy relationship for rule $\mathcal{R}_j$, as described in Theorem 2.3.1.

Constant Time Criteria for Online Redundancy Determination: Given an online mining query with parameter values (*minsup*, *minconf*), for each rule $\mathcal{R}_j$ in the result set $\{\mathcal{R}_j\}$, only **two parameter space locations**, namely, the **simple dominating location** $\ell_j^{\gg sim}$ and the **strict dominating location** $\ell_j^{\gg str}$ are sufficient for determining whether the rule $\mathcal{R}_j$ is redundant. In the case of (minsupp$\leq$ $\ell_j^{\gg sim}$.supp AND minconf$\leq$ $\ell_j^{\gg sim}$.conf)

OR (minsupp$\leq \ell_j^{\gg str}$.supp AND minconf$\leq \ell_j^{\gg str}$.conf) is true, the rule $\mathcal{R}_j$ is redundant. Otherwise, the rule $\mathcal{R}_j$ is not redundant.

As illustrated in Figure 2.6, the online algorithm for *redundancy resolution* requires checking the three cases of where the user input (*minsupp*,*minconf*) lies with respect to each rule in the result set and their dual dominating locations.

2.3.2 Optimized Location Computation

The *Redundancy Abstractor* subroutine (Algorithm 1.C) captures redundancies for each rule $\mathcal{R}_j$. Suppose $\mathcal{R}_j$ is of the form ((A_1:A_n) $\Rightarrow$ (C_1:C_m)) as in Figure 2.5. One straightforward approach for computing the simple dominating location for rule $\mathcal{R}_j$ is as follows: (a.) collect all simple dominating associations and (b.) find the simple dominating rule with the maximum (*supp*,*conf*) value pairs. The location of that rule is the simple dominating location $\ell_j^{\gg sim}$ for rule $\mathcal{R}_j$. Clearly, the same process could be employed to find the strict dominating location $\ell_j^{\gg str}$.

However, this process of finding the dominating location by searching through the set $\{\mathcal{R}^{\gg sim}\}_j$ of k rules is equivalent to *finding the largest of k numbers*. It requires $\mathcal{O}$(k) time. From Lemma 2.3.1, the total number of simple dominating rules for a rule ((A_1:A_n) $\Rightarrow$ (C_1:C_m)) having n antecedents is 2^n-2. Therefore, finding the simple and strict dominating locations using the complete set of dominating rules (here k = 2^n-2) is very computationally intensive. Below, we instead demonstrate that a much smaller subset of dominating association rules is sufficient for computing the dominating locations, as stated in Theorem 2.3.2.

Top Simple Dominating Rules: For a rule ((A_1:A_n) $\Rightarrow$ (C_1:C_m)) having n antecedents, to find the simple dominating location, it is necessary and sufficient to search only the **(n-1)-antecedent** simple dominating rules with format (((A_1:A_n)- A_i) $\Rightarrow$ ($A_i \cup$ (C_1:C_m))), called the **top simple dominating rules**.

The total number of **top simple dominating rules** is given by $\binom{n}{n-1}$ = n.

Top Strict Dominating Rules: For a rule ((A_1:A_n) $\Rightarrow$ (C_1:C_m)) to find the strict dominating location, it is is necessary and sufficient to search only the *e* dominating rules with format (((A_1:A_n)) $\Rightarrow$ ((C_1:C_m) $\cup$ C_h)), where *e* is the number of consequent items in the dominating rules but not in (C_1:C_m) and C_h is a single item out of the set (C_{m+1}:C_{m+e}). We call them the **top strict dominating rules**.

Using Theorem 2.3.2, only the *top n simple dominating rules* must be collected using *CollectTopSimpleDomRules* method, instead of all the 2^n-2 simple dominating rules. Similarly, using Theorem 2.3.2, only the *top e strict dominating rules* must be collected using the *CollectTopStrictDomRules* method, where e is the number of additional consequent items in the dominating rules but not in (C_1:C_m). The overall approach is given in Algorithm 1.C. The optimizations achieved by Theorems 2.3.2 and 2.3.2 result in significant improvements as the offline redundancy abstraction now requires $\mathcal{O}(n^2)$ time opposed to the approach utilizing all dominating rules that would require $\mathcal{O}(2^n)$. The collection of rules enriched with redundancy information is denoted by $\{\mathcal{R}^+\}$.

2.3.3 The Compact PSpace Index

Using the *enriched stable regions* $\{\mathcal{S}^+\}$ and the enriched ruleset $\{\mathcal{R}^+\}$, the **PSpace** is created using the subroutine *ConstructIndex* (Algorithm 1) and indexed by a two-layered index, called the **PSpace Index**. The top level of the PSpace index facilitates the search to locate a particular stable region given input parameters. As such, any spatial indexing method could be utilized. In our implementation, grid-based spatial indexing is utilized to partition the PSpace into equal sized grid cells and allocate the stable regions to their respective positions in the grid. A stable region may span over one or more grid cells, while a grid cell is allocated at least one stable region. The stable regions in each cell point to the coresponding nodes in the next level of the PSpace index. In Figure 2.7,

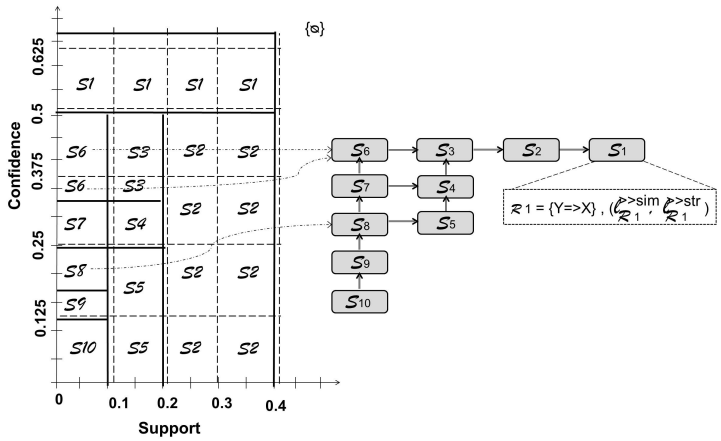


Figure 2.7: The PSpace Index.

we partition PSpace into 0.1 $\times$ 0.125 sized grid cells. *S8* and *S9* are in the same grid cell, while *S6* spans over two grid cells. For optimization, we reduce the number of grid cells by marking the upperbounds for support and confidence and indicating that no stable region exist beyond the upperbound in *PSpace*. For online query processing using the grid see Section 2.4. Using the proposed grid structure, the online search for a stable region can be performed in near constant time as shown in Section 2.4.

The next level of the PSpace index, namely, the *region neighborship graph* (R.H.S. of Figure 2.7), expedites the collection of rules from neighbor regions. Each stable region forms a node in the graph and each node is linked to its lending neighbors. As each region may have at most two such neighbors, these links result in a *sparse directed graph* of stable regions. The *region neighborship graph* enables us to not only locate lending neighbor regions in near constant time, but also to produce complete rule sets in time linear to the number of rules involved. Inside the *region neighborship graph*, each region node consists of the enriched ruleset unique to the region (Figure 2.4) along with their respective simple and strict dominating locations. For example, node $\mathcal{S}_1$ in Figure 2.7 stores rule $\mathcal{R}_1 \equiv (\mathrm{Y} \Rightarrow \mathrm{X})$ as well as its dominating locations $\ell_{\mathcal{R}_1}^{\gg sim}$ and $\ell_{\mathcal{R}_1}^{\gg str}$. The **PSpace index** is used to efficiently process a rich variety of online exploratory queries as shown in Section 2.4.

2.4 Online Query Processing

We now explain how the different classes of online mining queries are processed by **PARAS** (Algorithm 2). The *PSpace Access* module is employed to load the *PSpace index* $\mathcal{P}$ (see Section 2.3.3 for index details). Depending on the *query class*, interpreted by the *Query Parser*, the apropriate subroutine for the query class is invoked. If the *query class* is RM (or SR or RR), then the subroutine *RuleMiningQ* (or *StableRegionQ* or *RedundancyQ*) is invoked with appropriate parameter values.

Algorithm 2 Online Query Processing

Query $\mathcal{Q}$ RuleSet $\{\mathcal{R}\}$ or RegionSet$\{\mathcal{S}\}$
$\mathcal{P} \longleftarrow$ PAccess.GetIndex();

QParser.GetQClass($\mathcal{Q}$) == RM RuleMiningQ(QParser.GetQRE($\mathcal{Q}$),($minsupp, minconf$),$\mathcal{P}$);

QParser.GetQClass($\mathcal{Q}$) == SR StableRegionQ(qType,($minsupp, minconf$),$\mathcal{P}$);

QParser.GetQClass($\mathcal{Q}$) == RR RedundancyQ(qType,($minsupp, minconf$),$\mathcal{P}$);

2.A: RuleMiningQ(REFlag,($minsupp, minconf$),$\mathcal{P}$)

$\{\mathcal{R}\} \longleftarrow \emptyset$; $\mathcal{S} \longleftarrow \mathcal{P}$.LocSearchRegion($minsupp, minconf$);

$\{\mathcal{R}\} \longleftarrow \{\mathcal{R}\} \cup \mathcal{S}$.GetRuleSet();

neighborList $\longleftarrow$ NeighborCollector($\mathcal{S}$); *[f]Get all neighbors.

each $\mathcal{S}_i \in$ neighborList $\{\mathcal{R}\} \longleftarrow \{\mathcal{R}\} \cup \mathcal{S}_i$.GetRuleSet(); *[f]Collect RuleSets.

REFlag.IsTrue() $\{\mathcal{R}\} \longleftarrow$ RedundancyResolver($\{\mathcal{R}\}$,($minsupp, minconf$));

return $\{\mathcal{R}\}$;

2.B: StableRegionQ(qType,($minsupp, minconf$),$\mathcal{P}$)

qType == Q2 return $\mathcal{P}$.LocSearchRegion($minsupp, minconf$);

qType == Q3 $\mathcal{S} \longleftarrow \mathcal{P}$.LocSearchRegion($minsupp, minconf$);

return $\mathcal{S}$.GetRuleSet();

qType == Q4 $\mathcal{S} \longleftarrow \mathcal{P}$.LocSearchRegion($minsupp, minconf$);

return $\mathcal{S}$.GetNeighborList();

return false;

2.C: RedundancyQ(qType,($minsupp, minconf$),$\mathcal{P}$)

$\{\mathcal{L}^{\gg}\} \longleftarrow \emptyset$; $\{\mathcal{R}\} \longleftarrow \emptyset$;

$\{\mathcal{R}\} \longleftarrow StableRegionQ(Q3, (minsupp, minconf), \mathcal{P})$;

*[f]Iterate over rules.

each $\mathcal{R}_j \in \{\mathcal{R}\}$

*[f]Collect Simple Dom Loc.

$\mathcal{L}^{\gg_{sim}} \longleftarrow copyLocation(\mathrm{R}_j.\ell^{\gg_{sim}}.supp, \mathcal{R}_j.\ell^{\gg_{sim}}.conf)$;

$\{\mathcal{L}^{\gg}\} \longleftarrow \{\mathcal{L}^{\gg}\} \cup \mathcal{L}^{\gg_{sim}}$;

*[f]Collect Strict Dom Loc.

$\mathcal{L}^{\gg_{str}} \longleftarrow copyLocation(\mathrm{R}_j.\ell^{\gg_{str}}.supp, \mathcal{R}_j.\ell^{\gg_{str}}.conf)$;

$\{\mathcal{L}^{\gg}\} \longleftarrow \{\mathcal{L}^{\gg}\} \cup \mathcal{L}^{\gg_{str}}$;

return $\{\mathcal{L}^{\gg}\}$;

RuleMiningQ. Algorithm 2.A describes how the rule mining query is processed. The *LocSearchRegion* method performs a location search on the **PSpace index** using (*min-*

supp,*minconf*) as input to retrieve the stable region $\mathcal{S}$ that contains (*minsupp*,*minconf*). $\mathcal{S}$ is *enriched* with its ruleset and neighbors. The *NeighborCollector* module recursively collects all the neighbor stable regions. The output ruleset $\{\mathcal{R}\}$ consists of the ruleset of region $\mathcal{S}$ and the rulesets of the *lending neighbors*. If *REFlag* is set to *TRUE*, the *RedundancyResolver* (Algorithm 3) performs the inexpensive checks, as illustrated in Figure 2.6, to reduce the output ruleset.

Algorithm 3 Redundancy Resolver

RuleSet $\{\mathcal{R}\}$, $(minsupp, minconf)$ Redundancy Eliminated RuleSet $\{\mathcal{R}\}^{RE}$

$\{\mathcal{R}\}^{RE} \longleftarrow emptyset$;

each $\mathcal{R}_i \in \{\mathcal{R}\}$ $simDL \longleftarrow \mathcal{R}_i$.GetSimDomLoc();

$strDL \longleftarrow \mathcal{R}_i$.GetStrDomLoc();

*[f]$\mathcal{R}_i$ qualifies by failing Case 1. Case 2:

(((*minsupp* $\leq$ $simDL.supp$)AND(*minconf* $\leq$ $simDL.conf$))OR (((*minsupp* $\leq$ $strDL.supp$) AND ($minconf \leq strDL.conf$))) PRINT ($\mathcal{R}_i$ is dominated.);

*[f]Case 3

$\{\mathcal{R}\}^{RE} \longleftarrow \{\mathcal{R}\}^{RE} \cup \mathcal{R}_i$; return $\{\mathcal{R}\}^{RE}$;

The **response time** for the rule mining query consists of three components, namely, Cost(LocSearchRegion) + Cost(Neighbor Collection) + Cost(Redundancy Resolution). Our **PSpace index** is an in-memory structure that compactly represents all the stable regions *enriched* with rulesets and neighbors. In case the data exceeds the memory, disk space is utilized for extra storage. The cost for a location search on the grid structure of the **PSpace index** is given as Cost(LocSearchRegion) = $\mathcal{O}(1)$.As illustrated in Figure 2.7, by converting input parameters (0.15,0.2) into offsets, the appropriate cell can be found in constant time and the stable region (here, S_5) is retrieved. For *neighbor collection* on the *sparse directed graph* of stable regions a *depth first search* (DFS) is required starting at the node containing (*minsupp*,*minconf*). The time complexity of DFS is $\mathcal{O}(|\mathrm{V}| + |\mathrm{E}|)$. In our

case, E $\leq$ (2 $\times$ V) as each vertex has a fanout of at most two edges, thus, Cost(Neighbor Collection) = $\mathcal{O}(|V|)$. Further, assuming uniform distribution of regions in the 2-D space, the number of stable regions that lie above (*minsupp*,*minconf*) is denoted by V = $\{N_S \times$ (1 - *minsupp*) $\times$ (1 - *minconf*)$\}$, where N_S is the number of stable regions. For very low (*minsupp*,*minconf*) input, all N_S stable regions must be collected. If the **I**ntermediate set of **R**ules $\{\mathcal{R}\}$ input to the *redundancy resolution* method contains a total of N_{IR} rules, the time required for *redundancy resolution* is $[N_{IR} \times C_{RR}]$, where C_{RR} is the constant cost of redundancy checking over a single candidate rule (Algo. 3). If the **PSpace index** requires secondary storage such that both grid cells and stable regions are stored on disk, additional costs for disk access are added into the costs of location search and neighbor collection. Redundancy resolution can still be performed in-memory over the retrieved stable regions.

StableRegionQ. Algorithm 2.B describes how the three stable region queries are processed. All three stable region query types invoke the *LocSearchRegion* method to retrieve the stable region containing (*minsupp*,*minconf*). *Query type* Q2 simply returns stable region $\mathcal{S}$ as output. As each stable region $\mathcal{S}$ is *enriched* with its ruleset, the *GetRuleSet* method in Q3 returns the ruleset of $\mathcal{S}$. Similarly, for Q4, *GetNeighborList* returns the neighbor list for $\mathcal{S}$.

Similar to query Q1, each of the *stable region queries* incurs the location search cost (Cost(LocSearchRegion)). The ruleset for Q3 and the neighbor list for Q4 can be retrieved in near constant time as each stable region is *enriched* with that information. Thus, Cost(Q2)=Cost(Q3)=Cost(Q4)= Cost(LocSearchRegion) + C_{SRQ}, where C_{SRQ} is the constant cost of accessing the rules and/or neighbors of a stable region.

RedundancyQ. Algorithm 2.C describes how the redundancy query is processed. The *dominating stable regions* are desired. For this query the *StableRegionQ* subroutine is invoked with *query type* Q3, (*minsupp*,*minconf*) and $\mathcal{P}$ as input. For each rule $\mathcal{R}_j$ in the

returned ruleset $\{\mathcal{R}\}$, the *simple* and *strict dominating* locations of $\mathcal{R}_j$ are retrieved to collect the output stable regions $\{\mathcal{S}^{\gg}\}$.

The **response time** of *redundancy query* Q5 is given by Cost(Q5) = [Cost(LocSearchRegion) + N_R] = $\mathcal{O}(N_R)$. Here, N_R denotes the number of rules in the stable region containing (*minsupp*,*minconf*). For each rule $\mathcal{R}_j$, its *simple* and *strict* dominating locations are retrieved in near constant time.

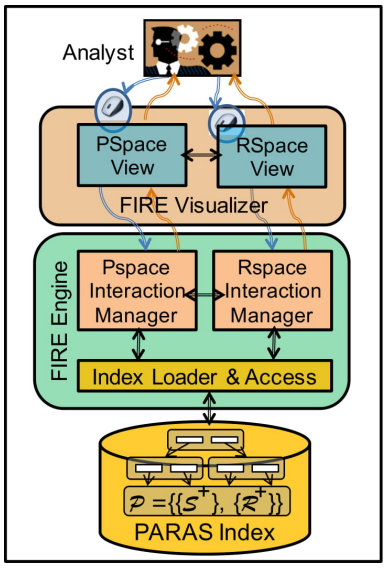


Figure 2.8: The FIRE Architecture

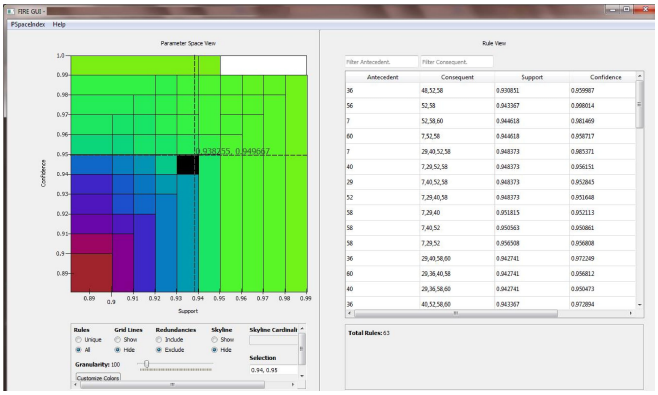


Figure 2.9: The FIRE Visual Interface

2.5 Overview of FIRE Visual Paradigm

Our **FIRE** Visualizer (Figure 2.8) supports a rich variety of analytical interactions over the PARAS index. The FIRE visual interface[1] (Figure 2.9) enables analysts to explore the stable region abstractions of the parameter space model and the corresponding rule-sets with ease - thus supporting effective visual analytics. FIRE is composed of a visual paradigm with two layers of interlinked visual interfaces, namely, the *PSpace* view and

[1]The FIRE tool is available at (FIR15) as a web interface for researchers to upload their own datasets, generate association rules on the datasets and visualize the rules.

the *RSpace* view. The *PSpace* view (Sec. 2.6) displays the overall distribution of rules within the space, facilitating parameter tuning and exploration at a higher level of abstraction. The *RSpace* view (Sec. 2.7) provides alternate tabular and rule glyph visuals. The tabular view displays the rules in text format, including their itemsets in its antecedent and consequent together with the support and confidence values. The RSpace glyph view, which is a novel visualization to show association rules. The glyph view enables analysts to gain rich insights by applying glyph placement strategies to find clusters of similar rules and to detect outliers. The FIRE visualizer is powered by the PARAS backend algorithms (LMR$^+$13). When a dataset is first loaded into FIRE, the PARAS backend generates all rules and organizes the rules into the PARAS index for compact storage. PARAS also includes query processing algorithms to respond to the user visual requests efficiently in real-time. The Index Access module offers an API for accessing the PARAS index that would have been constructed in an offline step using PARAS. When the same dataset is reloaded, the index is directly used as rules are already pre-generated.

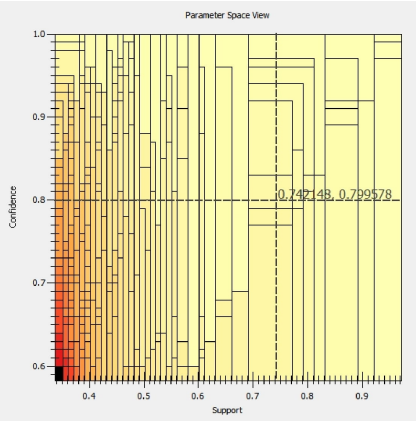


Figure 2.10: PSpace (All Rules) for the Mushroom dataset

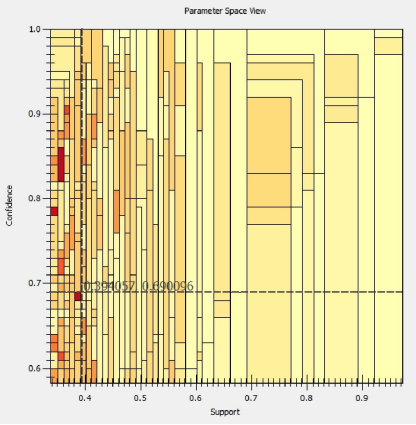


Figure 2.11: PSpace (Unique Rules) for the Mushroom dataset

2.6 Interactive Visual Parameter Space Design

Below we introduce FIRE's parameter-space (PSpace) visual paradigm where rules are distributed over the two-dimensional space of parameters (here, support and confidence) together with all the visual interactions available to analysts.

2.6.1 The PSpace Visualization

In this work, we design a novel abstract view of the distribution of rules on the parameter space called the *PSpace* visualization. As depicted on the left hand side (LHS) of Figure 2.9, the PSpace view displays rules in a two-dimensional plot of the stable regions within a space of support (x-axis) and confidence (y-axis) dimensions. Depending on the distribution of rules within the two-dimensional space, datasets may differ in number, size and density of the stable regions. Two such examples are shown in Figure 2.9 (LHS) depicting the Chess data set and in Figure 2.10 depicting the rule distribution for the Mushroom data set. Both are benchmark data sets taken from the UCI Machine Learning Repository (UCI15). The PSpace view offers a compact rule space driven by a parameter-centric perspective.

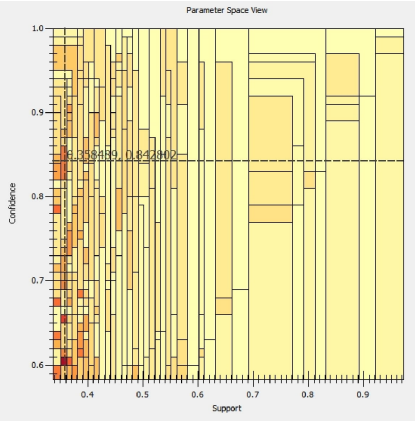


Figure 2.12: PSpace (Unique + Non-red.) for the Mushroom dataset

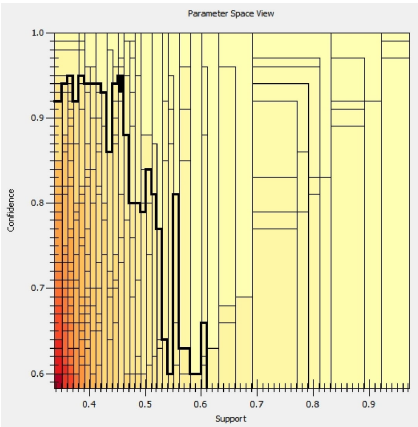


Figure 2.13: Rule Cardinality Skyline (>100 Rules)

2.6.2 The PSpace Interactions

The following user interactions are provided on the PSpace visual engine.

Stable Region Display for Fast Parameter Exploration. For a dataset with a sparse distribution of rules in the parameter space, even when a user submits several successive mining requests with distinct (minsupp,minconf) input parameter values, a rule miner may often repeatedly return the same set of rules. When using an existing rule miner (HFH$^+$09, ARu15), the analyst may have to progress through a frustrating trial-and-error process to finally get a new set of rules. When using the PSpace visualizations, the analyst can instead explore the parameter space by clicking through different regions. Every time she is guaranteed to receive a distinct rule set for investigation. This way, FIRE saves time and effort by laying out the complete distribution of rules in the parameter space. In FIRE, analysts can navigate through regions by either indirectly typing in the support and confidence values in the textbox (Figure 2.9) or by directly clicking on the stable regions displayed on the PSpace view.

Rich Insights into Region-wise Rule Cardinality. To provide rich insights into the density of rules within different regions, a color map is used where different colors denote different cardinality / count of rules. Figure 2.9 (left) and Figure 2.10 show two example datasets. Each shade of color denotes the count of rules within the region. Here, a lighter color depicts low count and a darker color depicts high count. FIRE offers a variety of color palettes to chose from including variants of sequential, diverging and qualitative ramps (XMD15). This tool enables users to select color schemes of interest to customize their displays.

Analysts can use the left bottom panel of the PSpace visualizer for a variety of interactions with the PSpace. For example, users can interactively show either all rules that appear in a region or only the rules unique to each region. For a dense dataset such as *Chess* (UCI15), each parameter setting produces a huge number of rules. Suppose that

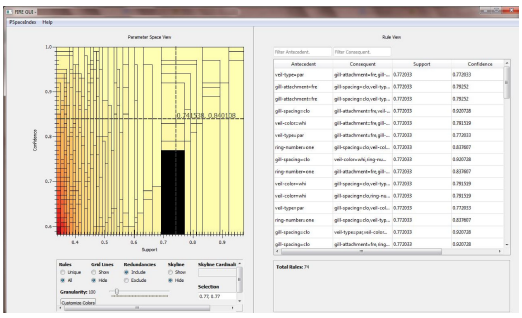


Figure 2.14: PSpace-RSpace Linkage

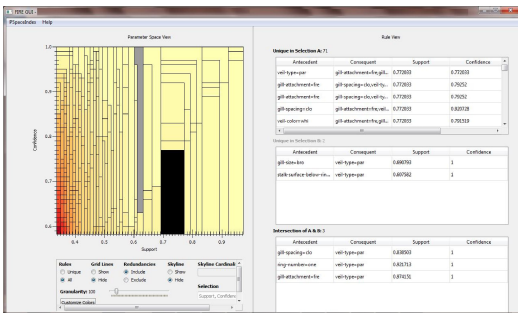


Figure 2.15: Comparing Two Regions

an analyst changes the parameter input by clicking on the PSpace interactive UI from (minsupp^{old},minconf^{old}) to (lsupp,lconf) such that minsupp^{old} $\geq$ lsupp and minconf^{old} $\geq$ lconf. Then the ruleset $\{\mathcal{R}\}^{(lsupp,lconf)}$ would also contain the rules in the original ruleset satisfying (minsupp^{old},minconf^{old}). The change in the ruleset may be difficult to quickly grasp by manual inspection. Here, a delta output of rules is desirable which can be achieved in FIRE simply by selecting the *Unique* option. While Fig. 2.10 depicts an *All* rules view, Fig. 2.11 shows the PSpace view for the same dataset when the user selects the *Unique* option via the radio button.

Rule Redundancy Resolution. To the best of our knowledge, FIRE is the first rule visualization system that allows analysts to optionally select to display only the non-redundant rules for a data set. By excluding redundant rules, a succinct set of fewer rules can be displayed in the PSpace view that covers all rules for ease of analysis. In the context of the stable region abstractions, interesting patterns can be observed when redundancies are excluded (Figure 2.12) compared to when they are included (Figure 2.11). In fact, any combination of unique/all and redundant/ non-redundant rules can be selected via the radio buttons to observe different patterns of rule distributions over the PSpace view. Further the results displayed can be analyzed using interactions.

Rule Cardinality Skyline Interaction. Figure 2.13 depicts the skyline view that provides recommendations beyond a single stable region boundary. Consider the situation

when the analyst wants to find the top-k (say, 100) rules in a dataset. However, at times it is unclear which parameter (support or confidence) to give priority to. By selecting the skyline option on the LHS bottom panel the analyst can input the desired cardinality in the *skyline cardinality* textbox (say, 100). The skyline is then drawn on the PSpace view to mark for each support value (x-axis), the confidence value (y-axis) having ≥ 100 rules. As a lower confidence value will result in a higher number of rules, the regions below the skyline will contain ≥ 100 rules while those above the skyline will contain < 100 rules. Therefore, the analyst can now select from a range of support and confidence settings that will all return up to the top 100 rules based on particular support and confidence combinations. Furthermore, the analyst can now quickly determine various observations about the data set. For instance, using the rule cardinality skyline in Figure 2.13 one can observe that no region contains ≥ 100 rules above support = 0.61.

Assisted Navigation through PSpace Visualization. Additional features such as cursor positions, optional grid line and zooming are provided to assist the analyst in navigating through the PSpace view. Some of these features can be seen in Figure 2.14. In our early user study, we found that while using FIRE, analysts may not be comfortable initially in identifying the support and confidence of desired regions on the PSpace view. Therefore, we have introduced the cursor position feature. Namely, as the analyst moves the cursor over the PSpace, the current cursor position is displayed. In Figure 2.14, the current cursor position is (0.74. . .,0.84. . .).

2.6.3 Visualizing the PSpace-RSpace Relationship

Viewing the rule distribution in the PSpace stable region display is at a level of abstraction higher than the RSpace view of individual rules or rulesets with their respective antecedent and consequent. PSpace-RSpace linkage enables real-time exploration using the two views as described below. By default, the RSpace view loads with all rules mined

from the dataset.

Drill-down via PSpace-RSpace Linkage. As shown in Figure 2.14, when the analyst selects a single region on PSpace view (highlighted in *black*) the rules valid within that region can be viewed in the RSpace view via cross links between the two views. This supports instant drill-down into individual rules while still maintaining the global context via the PSpace view.

Visual Region Ruleset Comparison. Analysts can also select two regions at a time to compare their respective rulesets. In Figure 2.15 comparing two stable regions facilitates the analysis of how the change in parameter settings effects the output. Region A is selected with a left click (highlighted in *black*) and region B is selected with shift+click (highlighted in *grey*). Through cross links, the RSpace view then will present a comparative display of unique rules within each region and also the common rules shared among these two regions A and B, if any. Here, we see that region A 71 unique rules and region B has 2 unique rules, with 3 common rules.

2.7 Interactive Visual Rule Space Design

Here, we describe the two detailed RSpace views designed for FIRE, namely, *tabular* and *glyph* views along with the respective interactions supported on them.

2.7.1 The RSpace Tabular View

Rules are traditionally listed in a *tabular* RSpace view common to most mining tools as depicted on the right hand side (RHS) of Figure 2.14. This tabular view allows the users to learn the detailed information such as antecedents and consequents of each rule together with support and confidence values. The total number of rules within the selected region on the PSpace view is displayed at the bottom of the RSpace table.

2.7.2 The RSpace Glyph View

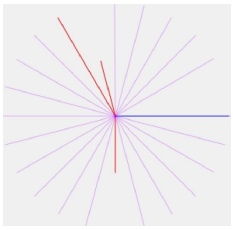


Figure 2.16: Lined Glyph 1

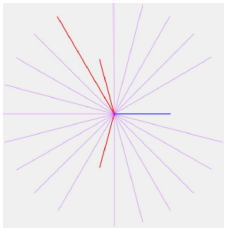


Figure 2.17: Lined Glyph 2

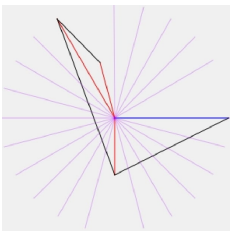


Figure 2.18: Con'ted Glyph

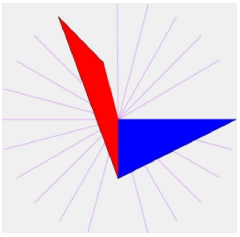


Figure 2.19: Filled Glyph

The purpose of the detailed RSpace view is for the analysts to visually analyze similarities or differences between the rules being displayed. However, as confirmed by initial user testing, this task is difficult to accomplish by using only the tabular view due to the overload of textual information. Beyond the straightforward tabular view described above, we thus designed a novel RSpace glyph view for graphically representing association rules to facilitate efficient visual analysis of rulesets. A glyph is known to be an effective visualization technique for displaying multi-variate data (War02). Glyphs are effective for visual shape comparisons as well as finding clusters or outliers by applying glyph placement strategies. However, to the best of our knowledge, glyphs have never been used to visualize association rules before. Below we describe three variants of our proposed RSpace glyph views.

Lined Glyph Design. A lined glyph (Figure 2.16) resembles a 360 degree clock dial with multiple hands. Given a data set with n attributes, we represent each attribute with a hand on the dial. Attribute hands are placed at equal angles to each other within the total of 360 degree dial. In Figure 2.16, the Mushroom data set (UCI15) containing 22 attributes is represented with 22 hands. The lined glyph represents the rule $\{$*poisonous? = edible*$\} \longrightarrow \{$*gill-attachment = free, veil-type = partial, veil-color = white*$\}$ from the Mushroom data set. The attributes that participate in a rule are highlighted while the rest of the attributes are displayed in a faded manner.

For each attribute the distinct values are displayed using different hand lengths. For example, the attribute *poisonous?* has 2 distinct values, namely, {*edible, poisonous*}. Thus, the hand lengths are encoded such that *poisonous? = edible* is represented by a full length hand (*blue* hand in Figure 2.16) whereas a half length hand would represent *poisonous? = poisonous* (*blue* hand in Figure 2.17).

Further, in order to distinguish the antecedents from the consequents, we propose to draw them using two different colors. In Figure 2.16, the single antecedent *poisonous? = edible* is represented with a *blue* hand whereas the three consequents {*gill-attachment = free, veil-type = partial, veil-color = white*} are each represented with a *red* hand.

Connected Glyph Design. The intuition for the connected glyph design is that it is easier to visually comprehend the similarities and differences between different shapes rather than those of the combination of hand positions. For the same example rule discussed above using the lined glyph shown in Figure 2.16, we now depict the connected glyph in Figure 2.18. The connected glyph is a simple modification of the line glyph with the outside ends of the highlighted attribute hands connected to each other so as to give it a shape. Two adjoining hands are connected only if they are within less than 180 degrees of each other in the clockwise direction. Otherwise this will introduce ambiguity. Further, we use a distinct color for the connection lines (here, *black*) to distinguish them from the antecedents and the consequents.

Filled Glyph Design. Initial user trials revealed that the connected glyphs were not effective for certain tasks such as distinguishing between antecedents and consequents. Thus, we propose a third glyph design called the filled glyph. The filled glyph display further fills colors inside the shapes created by connecting adjoining hands. The space between two adjoining highlighted attribute hands is filled with the color of the first attribute hand in a clockwise manner. In Figure 2.19, the space between hands representing attribute *stalk-color-above-ring = white* and *stalk-color-below-ring = white* is filled

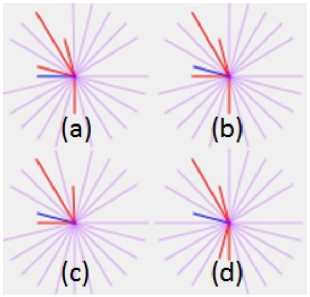


Figure 2.20: Lined Glyphs

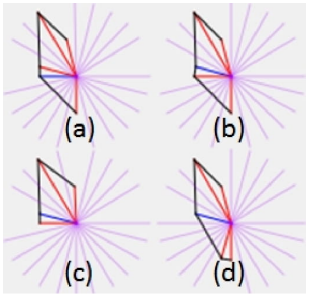


Figure 2.21: Connected Glyphs

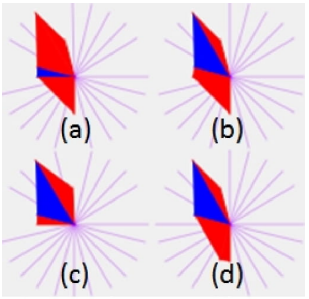


Figure 2.22: Filled Glyphs

with *blue*, i.e., the color of the antecedent. Namely, in this case the antecedent is *stalk-color-above-ring = white*. The space between the attribute hands *stalk-color-below-ring = white* and *veil-type = partial* is filled with *red*, i.e., the color of the consequent *stalk-color-below-ring = white*. Again, the space between two adjoining hands is filled only if the hands are within less than 180 degrees of each other in the clockwise direction.

Comparison of Glyph Designs. The purpose of these three glyph representations is to enable the analysts to visually comprehend the similarities and differences between the rules displayed in the glyph view. Our intuition is that these graphical representations are easier to comprehend and work with than the tabular display. Further, the purpose of providing multiple glyph options is that for different tasks, different glyph displays may be more effective, as confirmed by our evaluation. In Figures 2.20, 2.21 and 2.22 a set of 4 rule glyphs are shown using lined, connected and filled glyph designs, respectively. Our hypothesis is the following, based on initial user trials. If a task involves counting of hands such as *"to find the rule with the minimum number of consequents (**red** hands)"*, the lined glyphs are most effective. On the other hand, if a task involves similarity detection such as *"to find the rules containing the same antecedents (**blue** hands)"*, then the filled glyphs can effectively reveal the most prominent pattern. Connected glyphs, however, will be efficient for tasks that may involve both counting hands and requiring some shape information. A user study to examine the glyph designs is presented in Sec. 2.8.3.

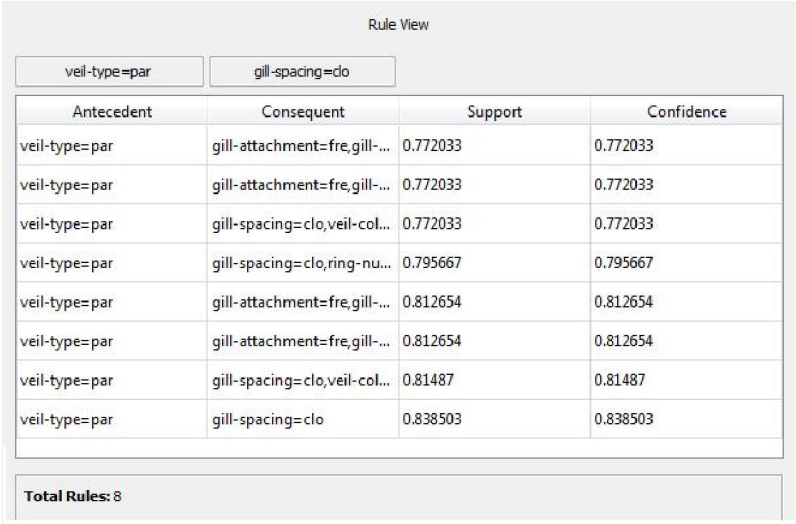


Figure 2.23: Tabular RSpace.

2.7.3 The RSpace Interactions

Using different interactions designed for the RSpace view, analysts can drill-down to gain rich insights into rule subsets as described below.

Filtering and Sorting of Rulesets. In case of an overwhelmingly large number of rules being displayed in the RSpace view, the analyst can filter the rules based on antecedent and/or consequent values using an auto-fill control. In general this allows the analyst to determine which rules are prominent for a given item/itemset. For example, in Figure 2.23, the antecedent is filtered on *veil-type* = *partial* and the consequent is filtered on *gill-spacing* = *close*. We note that only 8 rules out of the original 74 rules (Figure 2.14) satisfy the filter. This is a more manageable number for human analysis. The antecedent and consequent filters are available for both the RSpace views, namely tabular and glyph. As shown in Figure 2.23, the rules can also be sorted by *descending/ ascending* support or confidence. This is achieved by clicking on the *support* or *confidence* column header, respectively. This is particularly useful if a set contains some rules that have high support

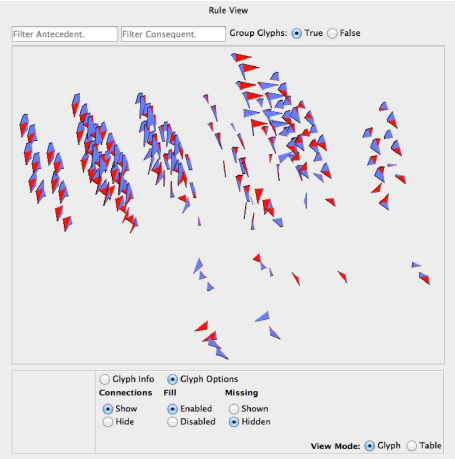


Figure 2.24: PCA Placement of Rules.

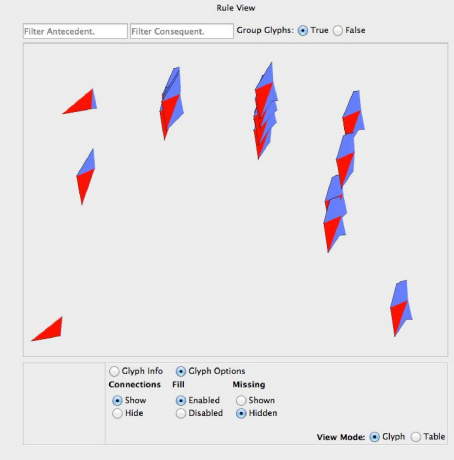


Figure 2.25: MDS Placement of Rules.

yet low confidence and others have high confidence yet low support.

Customizable Glyphs. Lastly, the ability to customize colors for distinguishing between antecedent and consequent provides a powerful visualization as certain patterns can be visualized with contrasting color schemes. Further, analysts can choose among any of the three glyph displays; each facilitating easy discovery of different pattern types.

2.7.4 RSpace Glyph Placement

Yet another important capability in information visualization is the placement or layout of glyphs on a display to communicate significant information regarding the values of individual glyphs themselves as well as relationships between the objects represented by the glyphs (War02). Here, we explore various placement strategies in the context of our proposed RSpace glyph view. The explored methods range from *data-driven* strategies that use data dimensions as positional attributes to *structure-driven* strategies that base the placement on implicit or explicit structure inherent within the data set. A comprehensive taxonomy of placement strategies has been developed in (War02) to assist the visualization designer in selecting the technique most suitable to his or her data and task. In our context, this feature enables analysts not only to gain insights about clusters of similar

Dataset	AdjLatticeRR *(supp)*	PARAS *(supp, conf)*
T100k	(0.0001)	(0.0001, 0.1)
T5000k	(0.0003)	(0.0003, 0.15)
Webdocs	(0.08)	(0.08, 0.30)

Table 2.2: Thresholds

Dataset	Varying *minsupp* ({*minsupp*},*minconf*)	Varying *minconf* (*minsupp*,{*minconf*})
T100k	({0.0004, 0.0006, 0.0008, 0.0010}, 0.60)	(0.0004. {0.20, 0.40, 0.60, 0.80})
T5000k	({0.0005, 0.0010, 0.0015, 0.0020}, 0.60)	(0.0010, {0.20, 0.40, 0.60, 0.80})
Webdocs	({0.10, 0.15, 0.20, 0.25}, 0.60)	(0.15, {0.45, 0.60, 0.75, 0.90})

Table 2.3: Online Query Settings

rules (e.g., rules with identical antecedents) but also to detect outliers that are separated from the rest of the rules.

In this work, we employ derived data-driven placement techniques that generate glyph positions using analytics applied to the data values as a whole input. Thus, instead of a location reflecting only one, two, or three of the data dimensions, it reflects a combination of all the dimensions in an attempt to convey N-dimensional relational information in the smaller number of dimensions. Common dimensionality reduction techniques (War02) include Principal Component Analysis (PCA), Multidimensional Scaling (MDS), Self-Organizing Maps (SOMs), spring-based models and so on.

We have adapted two of these layout techniques, namely, *PCA-based* placement (Figure 2.24) and *MDS-based* (Figure 2.25) placements in our FIRE visualization. PCA finds linear combinations of the dimensions that best explains the largest variation in the multivariate data set. The first two principal components are then used to determine the position of a glyph in a 2-D space as they capture the most prominant combinations of the original attributes that distinguish the data. In contrast, MDS is an iterative refinement process that attempts to adjust weights or positions until a certain criteria is met. In the context of rule glyphs the criteria would be common antecedents and / or consequents. In our case, the distances (or similarities) between glyphs in 2-D is a good approximation of the similarity of the rules based on the participating itemsets as antecedents and consequents.

2.8 Experimental Evaluation

Here we describe our experimental evaluation of PARAS backend, followed by details of case study and user study of the FIRE visual framework.

2.8.1 Evaluation of PARAS

Experimental Setup. We conducted experiments on a Windows 7 machine with Intel(R) Xeon(R) CPU X3440@2.53 GHz processor and 8 GB of RAM. All algorithms were coded in C++ using Visual Studio 2010.

Experimental Datasets. We evaluated the performance of the **PARAS** system and its competitors using synthetic and real dataset benchmarks. We used two synthetic datasets generated by the *IBM Quest data generator* (AS94a) modeling transactions in a retail store, *T10I4D100k* (T100k) and *T10I4D5000k*(T5000k). T5000k has 5 million transactions with 1000 items. On average, each transaction has 10 items. The data file size is about 200 MB. We also tested the *Webdocs* dataset from FIMI Repository (LOPS04). The *webdocs* dataset captures real data of spidered web html documents. Webdocs has 1.7 million transactions with 5,267,656 distinct items. The maximal length of a transaction is 71472. The data file size is about 1.5 GB. The results for two additional real datasets from the UC Irvine Machine Learning Repository (UCI15), namely, *chess* and *mushroom* are available in the technical report (LMR$^+$13) due to space constraints in this paper. Thus, these diverse datasets are suitable for evaluating the scalability of **PARAS** and its competitors.

Alternate State-of-the-art Techniques. While we had difficulty in executing the original rule mining algorithms in Apriori (AS94a), Eclat (ZPOL97) and *FP-growth* (HPY00) on the large data sets, improved C++ implementations of these algorithms available in (Bor15) run successfully. We evaluated the performance of online mining queries

with and without redundancy resolution. For mining requests without redundancy resolution, the performance of **PARAS** is compared against the original Apriori, Eclat, FP-growth from (Bor15). For requests with *redundancy resolution*, we compared **PARAS** (which produces non-redundant rules) against the above three online mining algorithms by adding online redundancy resolution code such that the results produced by all algorithms are comparable (identical). The algorithms enhanced with redundancy resolution (RR) are henceforth referred to as AprioriRR, EclatRR and FPgrowthRR. Next, we also compared **PARAS** against the POQM solution (AY01a). As it involves an offline step to generate and pre-store frequent itemsets within an *adjacency lattice* (Sec. **??**), we call it AdjLatticeRR. The online step generates the rules with redundancy resolution (pseudocode in (AY01a)). As **PARAS** and the other mining techniques adopt the redundancy definitions in AdjLatticeRR (AY01a), for each mining request, all five approaches produce identical results.

Experimental Methodologies. Performance measures are:

• **Offline Preprocessing Times.** We measure the total offline preprocessing times for AdjLatticeRR and **PARAS**. As AprioriRR, EclatRR and FPgrowthRR do not involve any preprocessing, they are excluded.

• **Mean Online Processing Times.** We measure the online processing time for a query averaged over several runs, for all five methods. We varied the *minsupp* and *minconf* query input parameters in the range [0,1].

• **Index Sizes.** We compare the sizes of the preprocessed information. AprioriRR, EclatRR, and FPgrowthRR are fully online techniques without any preprocessing involved. Thus, we compared the size of the adjacency lattice in AdjLatticeRR (i.e., # of frequent itemsets) and the *PSpace index* size in **PARAS** (i.e., # of stable regions) against the # of associations. We studied the impact of varying the primary support threshold on these index sizes.

2.8.1.1 Evaluation of Preprocessing Times

We first compare the preprocessing times for **PARAS** and AdjLatticeRR. AdjLatticeRR generates the frequent itemsets offline, whereas offline preprocessing in **PARAS** involves the four steps of *frequent itemset generation*, *rule&region generation*, *redundancy abstraction* and *PSpace index creation*. Among these, for each dataset, frequent itemset generation takes the longest preprocessing time for both **PARAS** and AdjLatticeRR (Fig. 2.26). This confirms prior works (AY01a, HPY00, NDD99) that rule generation is more efficient compared to frequent itemset generation. However, we now show that if redundancy resolution is required, the overall online processing time becomes significantly higher. In **PARAS**, while redundancy abstraction at the offline step adds offline overhead, it significantly reduces the online redundancy resolution costs (as we will see in Sec. 2.8.1.2). In Fig. 2.26, *T100k* and *T5000k* datasets (left y axis), redundancy abstraction has higher overhead than rule&region generation and PSpace index creation. However, for *Webdocs* (right y axis), compared with the cost of frequent itemset generation (60k+ seconds), the costs of the other steps, namely, rule&region generation, redundancy abstraction and PSpace index creation are negligible. Overall the three additional preprocessing steps in **PARAS** require no more than 10% extra time than *AdjLatticeRR*. Since they are done only once offline, acceptable in practice.

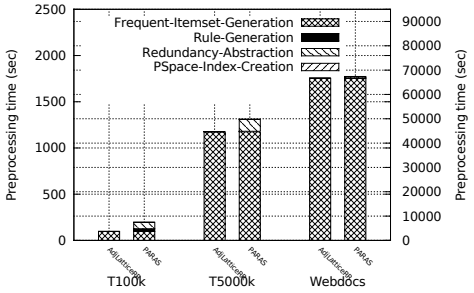
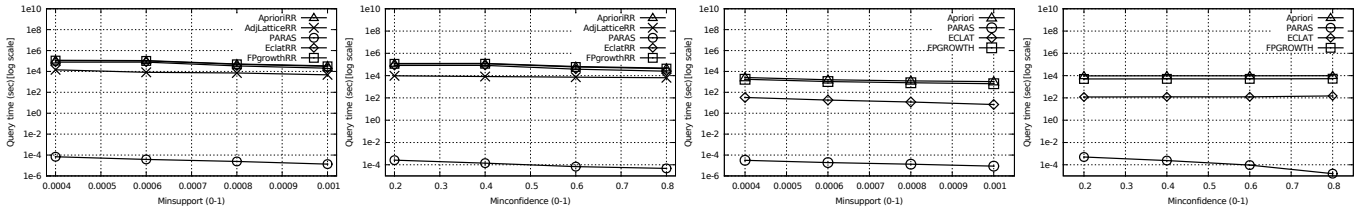
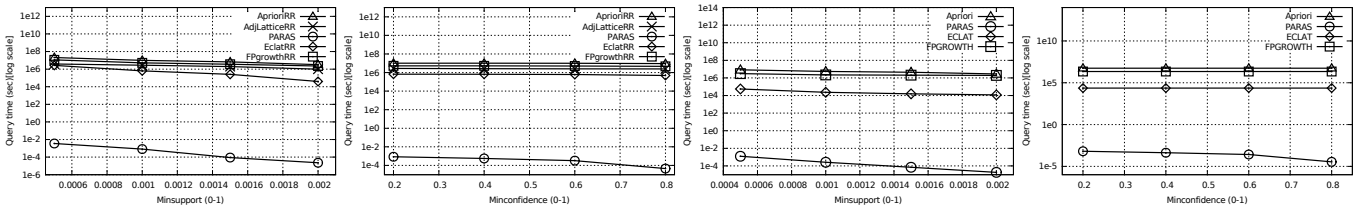


Figure 2.26: Preprocessing Times for All Three Datasets.



(a) minconf = 0.60 (b) minsupp = 0.0004 (c) minconf = 0.60 (d) minsupp = 0.000

Figure 2.27: Avg. Online Query Processing Times (T100k) [Rules w Redundancy Resolution in (a),(b) and w/o in (c),(d)].



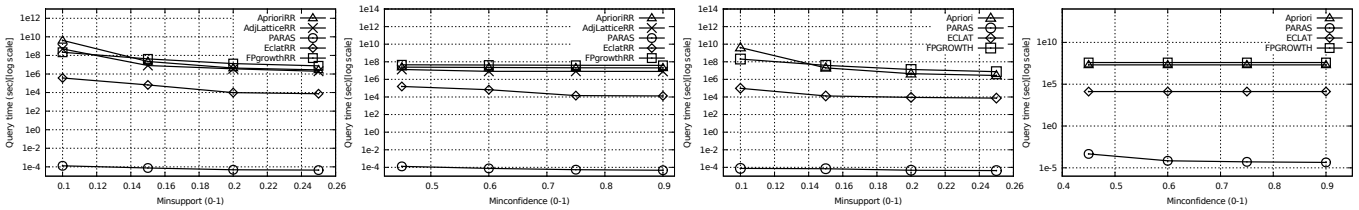
(a) minconf = 0.60 (b) minsupp = 0.0010 (c) minconf = 0.60 (d) minsupp = 0.001

Figure 2.28: Avg. Online Query Processing Times (T5000k) [Rules w Redundancy Resolution in (a),(b) and w/o in (c),(d)].

2.8.1.2 Evaluation of Online Processing Time

Next, we varied parameters *minsupp* or *minconf* (x-axis) and compared the online processing times (y-axis in log scale) of the alternative techniques. Table 2.2 (column two) lists for each tested dataset, the primary support threshold used to prestore frequent itemsets in the *adjacency lattice*. Column three lists the primary support and confidence thresholds used for populating the **PSpace index** of **PARAS**. We performed two sets of experiments.

Evaluation Involving Redundancy Resolution. First, we compare AprioriRR, Ad-



(a) minconf = 0.60 (b) minsupp = 0.0004 (c) minconf = 0.60 (d) minsupp = 0.000

Figure 2.29: Avg. Online Query Processing Times (Webdocs) [Rules w Redundancy Resolution in (a),(b) and w/o in (c),(d)].

jLatticeRR, EclatRR, FPgrowthRR and **PARAS** for user queries involving *redundancy resolution*. For query Q1 in **PARAS**, we set *REFlag* = TRUE. The query processing times are averaged over several runs of each query. To determine the effect of varying *minsupp*, we conducted several experiments by fixing *minconf* to a constant value and varying just the *minsupp* value.

Impact of Varying *minsupp*. Table 2.3 (column one) lists the fixed *minconf* and different *minsupp* values used for the three datasets. Figs. 2.27(a), 2.28(a) and 2.29(a) illustrate the query processing times for T100k, T5000k and Webdocs datasets, respectively. For all five techniques, the query processing time decreased with increase in the *minsupp*. As *minsupp* increases more rules get filtered - producing fewer rules as output. For AprioriRR, EclatRR, FPgrowthRR and AdjLatticeRR, a smaller number of frequent itemsets are processed for rule generation. For **PARAS** fewer stable regions are considered for composing the output ruleset and fewer rules require redundancy resolution.

Overall, **PARAS** consistently performed *several orders of magnitude* better than the four competitors. In particular, **PARAS** outperformed AprioriRR by 4, 5 and 5 orders, AdjLatticeRR by 4, 4 and 5 orders, EclatRR by 4, 4 and 4 orders and FPgrowthRR by 4, 5 and 5 orders for *T100k*, *T5000k* and *Webdocs* datasets, respectively.

Impact of Varying *minconf*. Next, we fixed the *minsupp* to a constant value and measured query processing times by varying *minconf* values (Table 2.3, column two). Figs. 2.27(b), 2.28(b) and 2.29(b) depict the processing times for T100k, T5000k and Webdocs datasets, respectively. The trend of the five alternate algorithms is similar as before. **PARAS** outperformes the four competitor approaches by several orders of magnitude.

Overall, **PARAS** consistently outperformed AprioriRR by 3, 4 and 5 orders, AdjLatticeRR by 3, 4 and 5 orders, EclatRR by 4, 3 and 4 orders and FPgrowthRR by 4, 4 and 5 orders for *T100k*, *T5000k* and *Webdocs* datasets, respectively. We note that the rate of decrease (slope) of the query processing times with the increase in *minconf* is not as steep

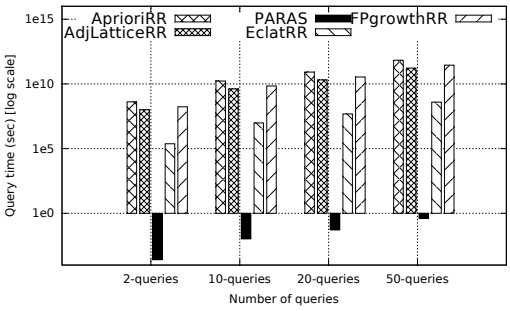
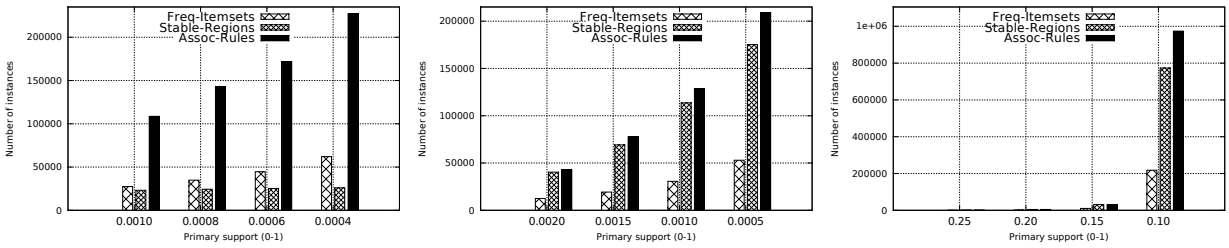


Figure 2.30: Multi-Query Costs.



(a) T100k (b) T5000k (c) Webdocs

Figure 2.31: Size of the PSpace Index.

as the slope with increase in *minsupp*.

Evaluation of Handling Multiple Queries. We now compare the processing times for multiple successive queries. Fig. 2.30 depicts the chart for the *Webdocs* dataset with a number of successive 2, 10, 20 and 50 queries (x-axis). The total processing time is show on y-axis. A diversity of queries are generated by randomly selecting *minsupp* values between 0.10 and 0.30 and *minconf* values between 0.10 and 0.90, respectively. AprioriRR performs all steps at query-time. AdjLatticeRR only performs rule generation with redundancy resolution at query-time. Thus it outperforms AprioriRR. **PARAS** only requires a look-up in the **PSpace index** and performs the inexpensive *redundancy resolution* using the dominating locations prestored within the **PSpace index**. Thus, **PARAS** delivers instantaneous responses even for large workloads with 50 queries or more. In Figure 2.30, for the case of *two* successive queries, all five approaches performed reasonably well. As the number of queries increases, the gains of using **PARAS** became more apparent. For 50

successive queries, **PARAS** took less than 1 second whereas AprioriRR, AdjLatticeRR, EclatRR and FPgrowthRR used approximately 38, 21, 2 and 26 hours, respectively.

Evaluation of Queries without Redundancy Resolution. Next, we considered user requests without redundancy resolution. We compared Apriori, Eclat and FPgrowth against **PARAS** by setting the Boolean *REFlag* to FALSE. AdjLatticeRR cannot be compared as it only produces non-redundant association rules. Similar as above, we conducted separate experiments by fixing one of the query parameters and varying the other as discussed below.

Impact of Varying *minsupp*. Figures 2.27(c), 2.28(c) and 2.29(c) depict charts for the three tested datasets. **PARAS** outperformed Apriori by 4, 5 and 5 orders, Eclat by 3, 4 and 3 orders, FPgrowth by 4, 4 and 5 orders for *T100k*, *T5000k* and *Webdocs* datasets, respectively.

Impact of Varying *minconf*. Figures 2.27(d), 2.28(d) and 2.29(d) depict charts for the three tested datasets. **PARAS** outperformed Apriori by 3, 4 and 5 orders, Eclat by 2, 3 and 3 orders, FPgrowth by 3, 4 and 5 orders for *T100k*, *T5000k* and *Webdocs* datasets, respectively.

2.8.1.3 Evaluation of Index Sizes

We compare the sizes of the prestored index structures used in AdjLatticeRR and **PARAS**. AprioriRR, EclatRR and FPgrowthRR are skipped as they are entirely online. For AdjLatticeRR, the adjacency lattice size is determined by the number of frequent itemsets, while *PSpace index* size by the number of stable regions. The actual index sizes (in say, MB) can be estimated by multiplying the number of instances (itemsets or stable regions) with the average space required per instance. The lower the primary support threshold, the larger the number of frequent itemsets stored in the *adjacency lattice*. Similarly, the choices of primary support and confidence thresholds determine the number of stable

regions and rules stored within the *PSpace index*.

In Figs. 2.31.(a),(b),(c), for the three datasets, we examine how the numbers (y-axis) of frequent itemsets, stable regions and association rules change with respect to changes in the *primary support* (AY01a) threshold values (x-axis). The primary support thresholds are in reverse order to show how the index sizes increase as the primary support threshold is relaxed to lower values.

For *T100k* (Fig. 2.31(a)), as the primary support changes from 0.0010 to 0.0004, the numbers of stable regions remain unchanged or at best increase slightly whereas the numbers of frequent itemsets and rules increase gradually. For *T5000k* (Figure 2.31(b)), the numbers of frequent itemsets, stable regions and rules increase when primary support changes from 0.0020 to 0.0005. For *Webdocs* (Figure 2.31(c)), the primary confidence is fixed at 0.30. the numbers of frequent itemsets, stable regions and rules increase gradually with the relaxation in primary support from 0.25 to 0.15, whereas the change is rapid for primary support 0.15 and 0.10. Overall, our PSpace index is slightly larger than the lattice of AdjLatticeRR.

2.8.1.4 Conclusions from the Experimental Evaluation

The main findings in evaluation of PARAS are:

• **PARAS** requires about 10% extra offline preprocessing time compared with AdjLatticeRR, which is acceptable.

• For a large diversity of online queries, **PARAS** consistently outperforms the state-of-the-art competitors from the literature by 2 to 5 orders of magnitude over the tested datasets.

• The benefits of **PARAS** are more apparent when multiple successive queries are processed. As **PARAS** processes several queries within a second, thus staying within the needs of human attention span for interactive exploration. On the other hand, the com-

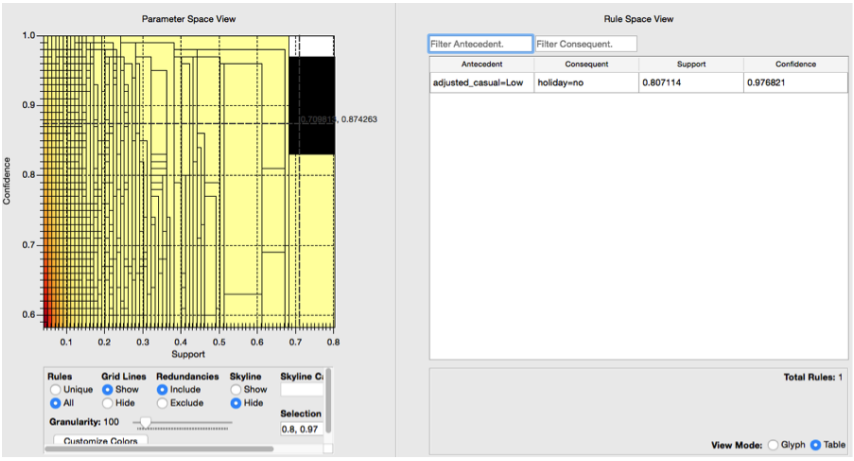


Figure 2.32: Finding the Highest Rules: the Highest No Common Knowledge Rule

petitors take several hours for the same.

• The *PSpace index* size of **PARAS** is on average 3.3× the *adjacency lattice* of AdjLatticeRR. The modern costs of memory makes this tradeoff practical given the huge CPU savings.

• Overall, the gains of several orders of magnitude when using **PARAS** for online processing outweigh the one-time minimal offline preprocessing time and storage requirements.

2.8.2 Case Study of FIRE Visual Rule Explorer

We evaluated the usability and effectiveness of our FIRE framework in two stages. In this first stage, we introduce a case study[1] during which a researcher explored a dataset of interest. The case study is qualitative in nature. The researcher independently explored the bike sharing dataset (Bik15) from the UCI machine learning repository using (a.) FIRE and (b.) ARulesViz as described below.

[1]This case study was performed by an avid bike user with an interest in data mining.

2.8.2.1 Exploring The Dataset Using FIRE

Dataset Description. The bike sharing dataset contains two years of bike usage data. Each data instance contains the counts of casual (walk-ins) and registered users in a given day and information about weather conditions (temperature, humidity) and holiday status (weekday, weekend, holiday). The contributors of the dataset claim that "most of the important events in the city could be detected by monitoring these data" (Bik15). The aim is to mine rules that link bike usage to holidays, workday status and weather conditions.

Pre-processing Bike Sharing Dataset. The Bike Sharing dataset (Bik15) was pre-processed before loading into FIRE (FIR15) and R ARulesViz (ARu15), as described below.

Attribute	Discretization	Category
adjusted_casual	[0,986.006) [986.006,1972.0118) [1972.0118,∞)	LOW MEDIUM HIGH
adjusted_registered	[0,1334.0613) [1334.0613,2668.1227) [2668.1227,∞)	LOW MEDIUM HIGH
air_temperature	[0,0.333012) [0.333012,0.586954) [0.586954,∞)	LOW MEDIUM HIGH
humidity	[0,0.324167) [0.324167,0.648333) [0.648333,∞)	LOW MEDIUM HIGH
windspeed	[0,0.184082324167) [0.184082,0.345773) [0.345773,∞)	LOW MEDIUM HIGH

Table 2.4: Discretized Attributes: Bike Sharing Dataset.

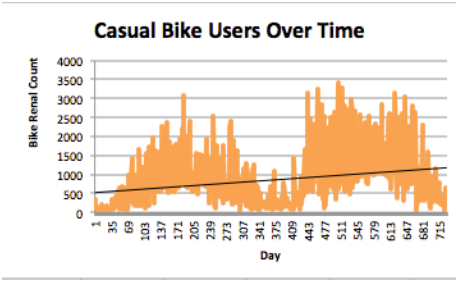


Figure 2.33: Original Casual User Count.

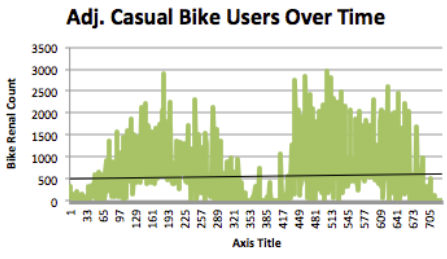


Figure 2.34: Adjusted Casual User Count.

1. Data corresponding to three of the attributes was eliminated. The attributes are

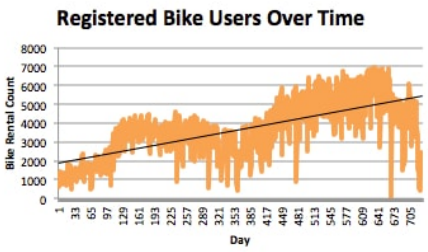


Figure 2.35: Orig. Registered User Count.

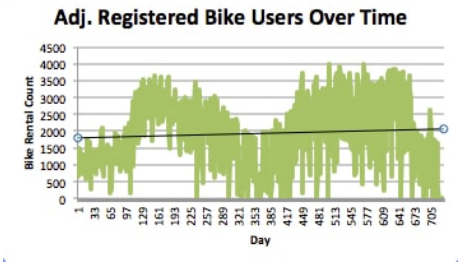


Figure 2.36: Adj. Registered User Count.

instant (unique identifier), *dteday* (date) and *yr* (contains 2 values: year 1 and year 2).

2. The casual and registered users increased over time. In particular, the casual users increase at 0.895 users per day whereas the registered users increase at a rate of 4.874 users per day. To cancel the effect of the overall growth, the data was rotated to negate the slope of the trend lines. In Figures 2.33 and 2.35 we show the original user counts and in Figures 2.34 and 2.36 we show the adjusted user counts for the casual and registered users categories. This processing is similar in flavor to season trend decomposition in (CCMT90).

3. Further, the attributes were discretized as shown in Table 2.4.

Generating Rules and Loading Them into FIRE. The data was loaded into FIRE with a minimum support of 5% and minimum confidence of 60%. As the dataset contains 732 instances, each representing a day, 5% support means a rule would be mined only if it is present in at least 36 days, or in more than 1 month out of 24 months. Therefore, the researcher believes this value constitutes a good primary support. Given these parameters FIRE generated the rules and loaded them within a few seconds. The total number of rules generated was 9673. In the ALL rules setting (Section 2.6.1), the set of 9673 rules can be listed in the RSpace view by clicking on the lowest stable region (0.05, 0.6) on the PSpace view. The highest rules are located in the upper right corner (Figure 2.32).

However, from the PSpace view, we find that the stable region with maximum support and confidence is empty. The two neighboring stable regions are as follows. The region (0.683,1) contains a rule with confidence = 1 and support <1 and the region (0.807,0.976) contains a rule with maximum possible support (80.7%) and confidence <1. These rules are listed below:

1. $\{$workingday=yes $\longrightarrow$ holiday=no$\}$ (support = 0.683, confidence=1): a common knowledge rule, correctly derived by the data, yet uninteresting.

2. $\{$ adjusted_casual=low $\longrightarrow$ holiday=no$\}$ (support = 0.807, confidence=0.976): this rule can be interpreted as the number of casual users being low during non-holidays.

As holidays are rare ($\sim$ 11 days per year, or less than 5% of the data), the primary support of 5% does not cover rules with "holiday=yes". The PSpace view for the bike sharing dataset in Figure 2.32 clearly lets us learn with just one glance that the rules with support $>$50% are rare in this dataset. We were able to quickly explore all such regions. One interesting rule we found in this space is: $\{$workingday = yes $\longrightarrow$ adjusted_casual=low$\}$ (support: 0.671, confidence= 0.982). This means that overall walk-ins are low on working days. It is common knowledge that 5 out of 7 days are working days, which gives an expected maximum support level of 71%. However, 22 of the working days are holidays for the duration the bike sharing dataset was collected, or approximately 3% of data. Thus, working days make up approximately 68% of the data. Dividing support by confidence serves as a sanity check, and arrives at the same number without the need to have prior knowledge about the data: 0.671/0.98 $\simeq$ 0.68 (68%). In other words, this rule implies that working days are $\sim$ 68% of the instances, and for $\sim$98% of those working days, the casual user count is low.

Using Rule Filtering and Sorting Features. Having explored all regions with support $>$50%, the next step was to explore rules with support $\leq$ 50%. The rule mentioned

in the previous section is: {workingday=yes $\longrightarrow$ adjusted_casual=low} (support: 0.671, confidence=0.982). This rule has a strong support value due to the large number of instances that contain working days. What about rules when "workingday=no"? In the bike sharing dataset "workingday=no" includes all weekends as well as holidays. To find these rules The researcher took the following steps:

1. In the PSpace view, he clicked on the stable region with the lowest coordinates (0.05,0.6). This then resulted in 9673 rules being listed,

2. Then in the RSpace view, he filtered for rules with "workingday=no" in the antecedent. This resulted in 550 rules being listed,

3. Lastly in the RSpace view, he sorted rules by descending support values.

The same three steps can be repeated by filtering for "workingday=no" in the consequent. These features of FIRE are described in Section 2.6.1.

The highest support value for a rule containing "workingday=no" was 28.7%. However, this rule represents common knowledge (Figure 2.37) {workingday= no $\longrightarrow$ holiday=no} (support=0.287, confidence=0.909). Thus, in other words, 90.9% of the non-working days are weekends and the rest are holidays. However, a non-common knowledge rule found in this space was: {adjusted _casual=High $\longrightarrow$ workingday=no} (support =0.0519, confidence=0.974). This rule indicates that bike rentals by walk-in users were high 6% of the days (0.0519/0.974) and that in 97% of these instances it was not a working day. This rule makes sense as walk-in users have other means of transport for their daily lives and they are instead much more likely to rent bikes on weekends and holidays.

Thus far the researcher found rules mostly related to casual users. He further explored rules related to the registered users. For this purpose he followed the same three steps as in the case of filtering for "workingday=no". Instead, in step 2, he then filtered for rules with "adjusted_registered=High" in the antecedent as shown in Figure 2.38. There

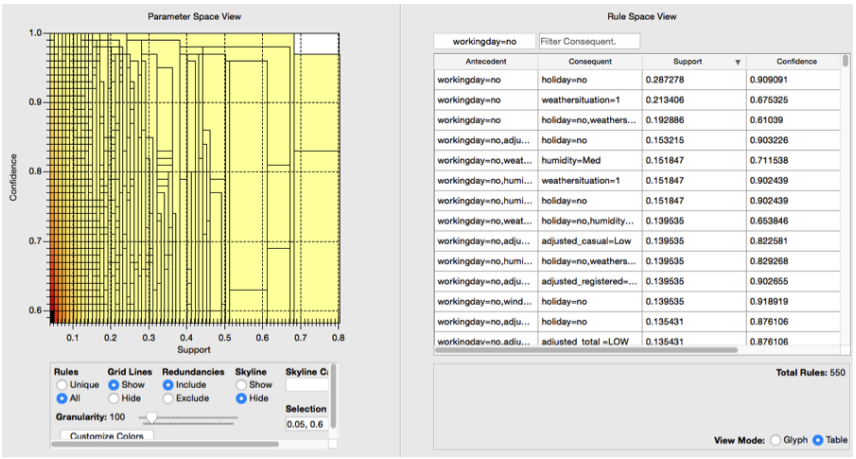


Figure 2.37: Filtering for Weekends

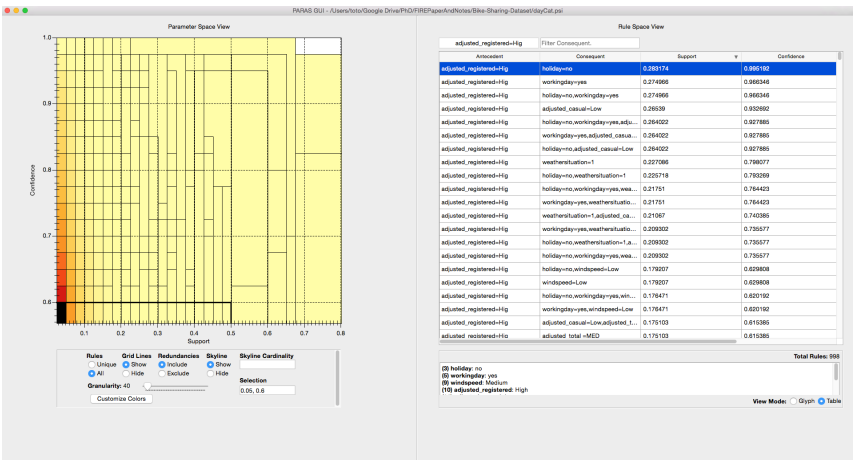


Figure 2.38: Filtering for Registered Users

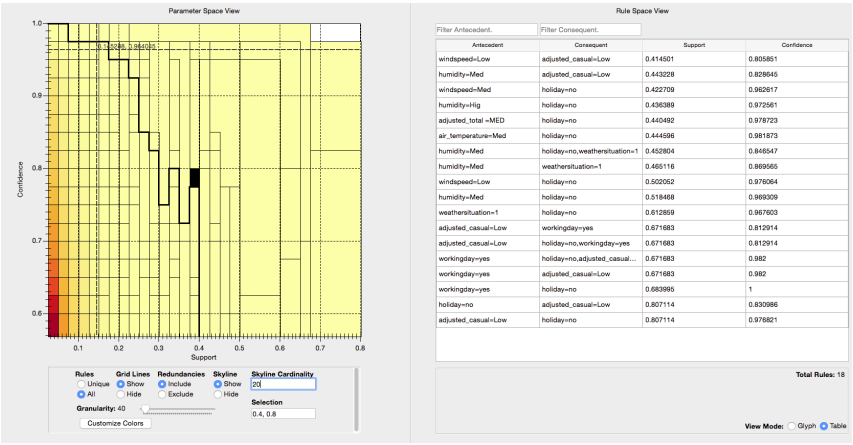


Figure 2.39: Skyline Cardinality: Distinguishing Regions with >20 Rules and $\leq$ 20 Rules.

are 998 such rules. He then sorted them by their descending support. The highest support possible for a rule with "adjusted_registered=High" is 28.3% with a 99.5% confidence (highlighted in blue color). The top 3 rules indicate that registered bike users are high in numbers during working days. However, the fourth rule: {adjusted_ registered = High $\longrightarrow$ adjusted_casual=Low} (support= 0.264, confidence=0.932) shows an interesting inverse relationship between the count of registered and casual users. Specifically, whenever "adjusted_registered=High", 93% of those days "adjusted_casual=Low".

Utilizing the Skyline Feature over the PSpace View for Retrieving the Regions with a Certain Cardinality. The case study thus far involved exploring the different stable regions and going through the list of rules in each region. In the *ALL* rules view, the count of rules cumulatively increases as the researcher moves towards lower support or confidence settings. Further, he wanted to list the top k (say, 20) rules. However, as there are two rule ranking criteria, namely, support and confidence, he employed the skyline cardinality feature that allowed him to separate stable regions with more than k rules from those with less than k rules (Figure 2.39).

The regions adjacent to the skyline were the most interesting, because they had a high number of rules with good support and confidence value. For example, the highlighted

region (0.4, 0.8), which is above the skyline, contains 18 rules. In addition to the rules explored thus far, several new rules involving temperature, humidity, weather situation can be found in this region. One of these rules is {adjusted_total=MEDIUM $\longrightarrow$ holiday=no} with a support of 44% and a high confidence of 97.8%. Here, the total rentals are discretized into three values {LOW,MEDIUM,HIGH}.

Comparing Rules Using the Glyph View. For the stable region (0.4,0.8) that the researcher explored above, he noticed from the tabular view that several of the 18 rules had common attributes in the antecedent and/or the consequent. Thus, he next wanted to compare the rules and see which ones are similar, i.e., have common attribute values. In order to compare all rules, he needed to manually compare C(n,r) = n!/(r!(n - r)!) possible combinations of rules. In our concrete example n = 18 and the r = 2, we have 153 possible comparisons to make. This problem becomes increasingly complex with a large number of rules and is difficult to do in the tabular list of rules. While the glyph view does not reduce the number of comparisons he had to make, he found it easier to look for the similarity among shapes using the rule glyph view (rule glyphs are defined in Section 2.7.2). Figure 2.40 shows several such examples. The two rules depicted using the blue boxes are opposite to each other, i.e., one with "holiday=no" in the antecedent and "adjusted_casual=Low" in the consequent, and the other vice versa. Similarly the two rules within the red box contain three attributes each and their antecedent/consequent sequence ("adjusted_casual=Low"/"workingday=yes") is swapped. Further, we see that the rules depicted in the red box can be obtained by combining the attributes of the rules in the blue box and the rule highlighted within the green box. Overall, the researcher found the glyph representation convenient for visual shape comparisons among rules and rulesets.

Glyph Clustering Functionality. The next step in the exploration was to enable clustering for the group of rules as above. The goal was to look for outliers that might

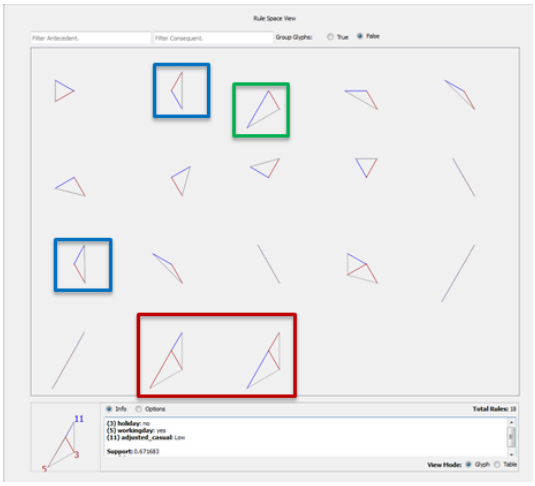


Figure 2.40: Comparing Rule Glyphs.

Figure 2.41: Clustered Rule Glyphs.

contain interesting rules. When enabling clustering, as expected, the rules described in the previous section grouped together based on the commonality of attributes (see Figure 2.41). Further, several rules that have common set of attribute-value pairs are grouped together such that the common attribute-value pairs are depicted by a shared line or lines close to each other.

2.8.2.2 Exploring the Dataset Using ARulesViz

ARulesViz is a popular R package that contains a total of 10 state-of-the-art association rule visualization techniques (ARu15). The visualizations include: (a.) scatterplot (2 variants), (b.) matrix-based (4 variants), (c.) graph (2 variants), (d.) parallel co-ordinates, and, (e.) double decker. Details of each visualization technique can be found in (ARu15). Inside the R environment, the researcher typed in R commands to load the Bike Sharing dataset (Bik15). Then using the ARules package association rules were generated. Finally, using the ARulesViz package the rules were visualized using the different rule visualization techniques available in the ARulesViz package. The overall comparison of these visualization techniques is shown in Table 2.5. This comparison extends the origi-

nal comparison given in (ARu15) by adding the two primary visualization techniques of FIRE, namely, (a.) FIRE PSpace stable regions view, and (b.) FIRE RSpace rule glyph view.

Figure 2.42: ARulesViz Scatterplot UI.

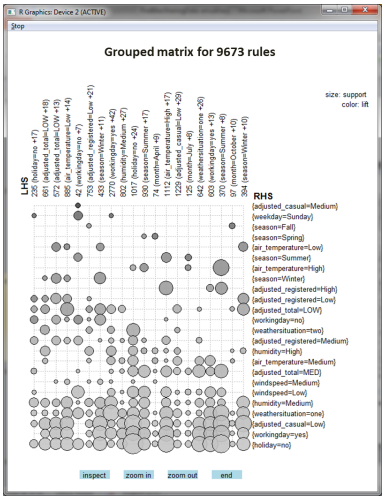


Figure 2.43: ARulesViz Grouped-Matrix UI.

Three of the matrix-based visualizations, graph-based, parallel co-ordinates and double decker visualizations support a medium to a small number of rules at a time. On the other hand, scatterplot variants, grouped matrix, and graph-based (external) as well as FIRE PSpace and Glyph views can support a large rule set. In the interactive scatterplot view (Figure 2.42), one can select an arbitrary region (shown as a red shaded box) and show the list of rules that qualify for the selected support and confidence in the region in the console output (here, a total of 43 rules). This interaction in effect is equivalent to the *unique* rules view in the FIRE PSpace visualization. The limitation of the scatterplot view is that for an arbitrarily chosen region that includes several rules, a high number of rules will be listed in the console view. Thus several rules may be hidden unless the rule list is explored exhaustively. Moreover, no reordering support is available in the scatterplot visualizations.

The FIRE PSpace view can be considered as a layer of the stable region abstraction

Technique	Rule set	Measures	Interactive	Reordering	Ease of use
Scatterplot	large	3	✓		++
Two-Key plot	large	2 + order	✓		++
Matrix-based	medium	1		✓	0
Matrix-based (2 measures)	medium	2		✓	–
Matrix-based (3D bar)	small	1		✓	+
Grouped matrix	large	1	✓	✓	0
Graph-based	small	2			++
Graph-based (external)	large	2	✓	✓	+
Parallel co-ordinates	small	1		✓	-
Double decker	single rule	(2)			-
FIRE PSpace	large	2	✓	✓	+++
FIRE Rule Glyph	large	2	✓	✓	+++

Table 2.5: Comparison of Association Rule Visualization Techniques.

over the scatterplot view. Additional features of the FIRE PSpace view such as unique rules, redundancy exclusion, and skyline provide semantic filters based on support and confidence measurements, rule redundancy definitions as well as the cardinality of rules, respectively. Further, all these techniques (Table 2.5) can be categorized by the number of measures (e.g., support, confidence and lift) that can be simultaneously visualized. While the scatterplot allows three measures (two on the axes, one using color/shade), most other approaches allow two measures at a time. The FIRE PSpace view utilizes color mapping schemes to denote the density of rules in the stable regions.

As described in (ARu15), to explore large sets of rules with graph-based visualization, advanced interactive features like zooming, filtering, grouping and coloring nodes are needed. Such features are available in interactive visualization and exploration platforms for networks and graphs like Gephi. From the ARulesViz package (ARu15), graphs for sets of association rules can be *exported* in the GraphML format or as a Graphviz dot-file to be explored in tools like *Gephi*. This process of exporting the rule graphs is cumbersome for interactive exploration. On the other hand, the FIRE RSpace tabular view is enabled with antecedent and consequent auto-fill filters as well as support and confidence ordering for enhanced exploration through list of rules. The FIRE rule glyph view utilizes color schemes to differentiate antecedents from consequents. The details of each rule rep-

resented by a glyph, such as the antecedent and consequent values of the rule can be seen at the bottom of the RSpace view by hovering over or selecting the glyph.

The grouped matrix view (Figure 2.43) is a variant of matrix-based visualization technique such that the rules are grouped based on common antecedents and consequents (see (ARu15) for details). The view utilizes a K-mean clustering algorithm for the same, where the user needs to provide the value of K (default value of K=20). In Figure 2.43, all 9673 rules are shown with K=20. The appropriate value of K for any dataset needs to be learnt using trial-and-error. Moreover, the LHS, shown in the top x-axis, consists of clusters of multiple antecedent values grouped together. This made it difficult for the researcher to comprehend the items other than the single one listed in each column. Similar in flavor to the grouped matrix view is the FIRE rule glyph clustering approach, where the researcher utilized PCA and MDS layout (see Section 2.7.4 for details). However, the full details of the attributes in the antecedent and the consequent can be viewed in the RSpace view by hovering over the group.

2.8.2.3 Conclusions from the Case Study

Overall, the FIRE PSpace view together with its rich diversity of features effectively supports interactive exploration for a high number ($\sim$9673) of rules for the bike sharing dataset (Bik15). In addition the RSpace view, in particular, the rule glyph visualizations enables effective comparison of rules. Having graphical displays and interactions on otherwise static sets of rule enable novel interactions with the data and a rapid exploration of the rule space. Moreover, compared to the state-of-the-art association rule visualization techniques in (ARu15), that required the researcher to understand and type in syntactically correct R command line inputs or scripts, FIRE is a completely graphical visualization tool as every feature is available through intuitive clicks through labeled interactions.

2.8.3 User Study of FIRE Visual Rule Explorer

2.8.3.1 Evaluation Methodology

Here, we further present the second stage of our evaluation of the FIRE system. We conducted a controlled user study to compare the features of FIRE to that of the state-of-the-art systems such as Weka (HFH+09) and measured the effectiveness of different visual representations compared to the list of rules provided by Weka.

User Study Procedure The overall process was as follows: The subjects perform a series of 5 studies listed in Table 2.6. As the studies progressed, the study administrator explained the purpose and process for each task with examples. Lastly, the subject fills out an exit questionnaire. On average the study took between 26 and 47 minutes per subject.

Tested Feature	Tasks	Duration
S1. PSpace stable region and interaction	T1 T2 T3	2-4 min 2-5 min 3-6 min
S2. PSpace-redundancy / RSpace-tabular-filter	T4	3-6 min
S3. PSpace skyline exploration	T5	4-6 min
S4. RSpace glyph rule analysis	T6 T7	2-3 min 2-3 min
S5. RSpace glyph placement analysis	T8 T9 T10	2-3 min 4-8 min 2-3 min
Total		26-47 min

Table 2.6: User Study Schedule.

Compared Tools. Our user study compares our FIRE visualizer to the cached association rule miner (CRM). CRM is a association rule miner based on the APRIORI algorithm (AS94a) but with instant response time due to the cached rules. CRM provides users with a tabular view of rules and all functions offered by existing rule mining systems (e.g., WEKA (HFH+09)).

Metrics of Evaluation. We measured both efficiency and accuracy of the subjects in accomplishing the tasks. For efficiency, we measured the time consumed by each subject

for each task. For accuracy, we measured the percentage of correctly answered tasks by the subjects.

Datasets. We chose two datasets from the UC Irvine Machine Learning Repository (UCI15), namely, chess and mushroom. The chess dataset is derived from the game step. The mushroom dataset contains characteristics of various species of mushrooms. Chess and mushroom data sets have ≥ 2000 closed frequent itemsets at 94% and 50% support, respectively. (see Fig. 1.1.)

General Method. Each subject was asked to perform all of the five studies (S1 - S5) described in Section 2.8.3.2. To avoid carryover effects and learned knowledge about a dataset, we counter-balanced the order of tasks, datasets and tools. For S1, we switched datasets and tools. For example, half of the subjects performed the task T1 on the chess dataset using CRM, and T1 on the mushroom dataset using FIRE. On the other hand, the other half of the subjects performed the task T1 instead on the mushroom dataset using CRM and T1 on the chess dataset using FIRE. For S2 and S3, we switched both the questions and the tools. Particularly, we asked subjects to find characteristics of edible mushrooms using CRM and characteristics of poisonous mushrooms with FIRE. This way addressed the "pre-knowledge" problem. For S4, we randomized the order of showing different glyph displays for each subject. For S5, we randomized the order of applying different glyph placement strategies for each subject. In general, we avoided practice and fatigue effects by randomizing the order of tools and tasks. In these task assignments, no carryover problems arose, as each subject was asked to only finish a particular task on a given dataset using the tools in a random order.

Environment Setup. We conducted our experiments on a Windows 7 PC with Intel(R) Core(TM)i5-2410M CPU@2.3 GHz processor and 4 GB of RAM, with a display resolution of 1600 by 900. Our visualizations displayed in a 1000 by 600 window.

Study Population. We performed the user study with a population of 22 subjects

(10 undergraduate students and 12 graduate students). They were either from computer science, computer engineering or mathematical sciences programs. The user study was conducted on a one-to-one basis, i.e., a tester to subject test.

2.8.3.2 Design of Our User Study

S1. Stable Region Usage Study. In our stable region usage tests we asked the subjects to perform three different tasks (T1-T3) by varying tools and data sets, such that each dataset was tested for each visualization in a random order. The three tasks were designed to verify the ability of the subjects to explore the parameter space, to utilize the stable region abstractions and to compare rulesets. The questions were as follows:

T1 What are the most prominent rules by support and/or confidence?

T2 Which setting (out of 4 choices) gives a different set of rules than the given setting?

T3 Find the common and unique rules for two different parameter settings.

S2. Filter/Redundancy Study. In this study we used only the mushroom dataset. We asked our subjects to first filter the antecedents of the rules and then to remove redundant rules. Some users used FIRE first and CRM next, and vice-versa. The goal was to test the ability of our subjects to use filter and redundancy removal features by asking them to perform the following task.

T4 Find the most frequent characteristics of edible/ poisonous mushrooms.

S3. Skyline View Study. In the skyline view study we asked the subjects to find the top-k rules from the mushroom dataset by varying the tools (FIRE and CRM). The goal was to test if our subjects can make use of the rule skyline cardinality. For this, we presented our subjects with the following task.

T5 Find the parameter settings that produce top-k rules in the dataset, where k = 20, 50, or 100.

S4. Glyph Display Study. In our glyph view study we showed the subjects a set of 6 glyphs using different glyph designs, namely, lined, connected and filled. We told the subjects that the antecedent(s) is/are represented by the *blue* color and the consequent(s) is/are represented by the *red* color. We verified the hypothesis that different glyph designs may be more effective for different tasks. We presented our subjects with the following tasks.

T6 Given a set of 6 glyphs, find the rules with the same antecedents. Three questions were asked, each using a different glyph design.

T7 Given a set of 6 glyphs, find the rule(s) with the greatest number of consequents. Three questions were asked, each using a different glyph design.

S5. Glyph Placement Study. In this study three glyph placement strategies were presented using the glyphs generated from the mushroom dataset. The goal was to test if the subjects are able to leverage glyph placement strategies to identify cluster or outlier among a set of glyphs. In addition, we verified the hypothesis that different glyph placement strategy may be more effective for different tasks. In these tests, the connected glyph design was chosen to present the questions due to the fact that the connected glyph gives a visual shape to the glyph together with serving the purpose of showing each hand (attribute) clearly.

T8 Identify outliers within a given set of glyphs using two different glyph placement strategies (i.e., the unclustered layout versus the clustered layout). Two questions were asked- each using a different glyph placement strategy.

T9 Given a set of glyphs, identify glyph(s) with a certain attribute-value pair using three different glyph layout strategies, i.e., unsorted layout, sorted layout and clustered layout. Total of six questions were asked, two questions using each of the placement strategies.

T10 Using the clustered layout, identify groups of similar glyphs and count the groups containing a given attribute-value pair. Two different sets of glyphs were tested.

Exit Questionnaire. A survey questionnaire was presented to the subjects at the end of the studies. We asked them to rate the two alternative tools, namely, FIRE and CRM in terms of their ease of use on a scale of 1-5 (where 5 = very easy, 1 = very difficult). We also asked them which tool they preferred for each of the 3 studies (S1-S3). We also asked the subjects to rank the alternate glyph designs (S4) and glyph layouts (S5) by their ease of use. Overall, they were asked the following questions about each task.

Q1 Which task(s) is/are easier with FIRE than CRM? (list tasks)

Q2 Which task(s) is/are easier with CRM than FIRE? (list tasks)

2.8.3.3 Hypotheses

As FIRE provides several features for interactive rule exploration, we anticipated that conducting certain tasks using FIRE would be faster and more accurate than using CRM. Also, we expected that the glyph designs and glyph layout strategies may vary in their effectiveness for different tasks. This led to the following hypotheses.

H1 For T1, T2, T3, T4 and T5, subjects perform better using FIRE than CRM in term of both time spent and accuracy.

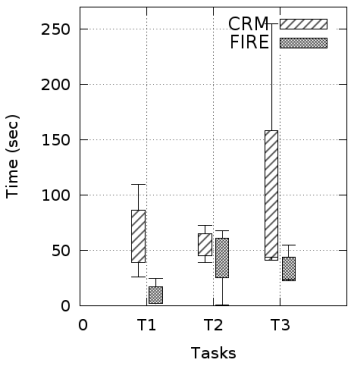
H2 For T6, the filled glyph is more effective than other glyph designs, whereas for T7, the lined glyph is more effective than others.

H3 For T8, the clustered layout is more effective than other glyph placement strategies in detecting outlier glyphs.

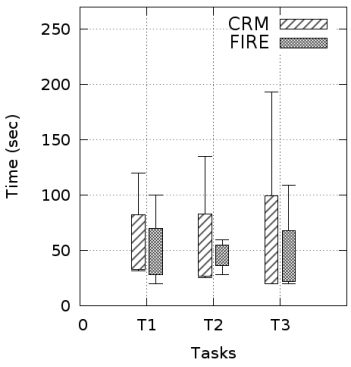
H4 For T9, the sorted layout is more effective than other layouts in aiding the analyst in finding glyph(s) with a certain attribute-value pair.

H5 For T10, the subjects can easily identify group of similar glyphs using the clustered layout.

2.8.3.4 Conclusions from the User Study



(a) Mushroom



(b) Chess

Figure 2.44: Time Spent on T1, T2 and T3.

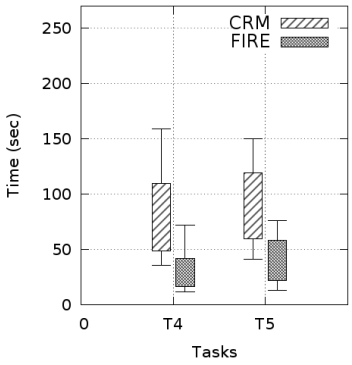
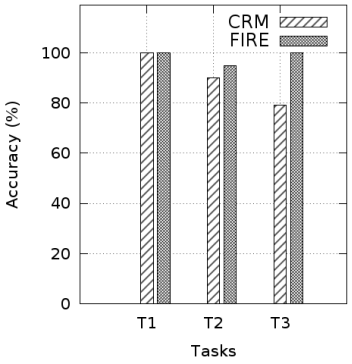


Figure 2.45: Time Spent on T4 and T5.

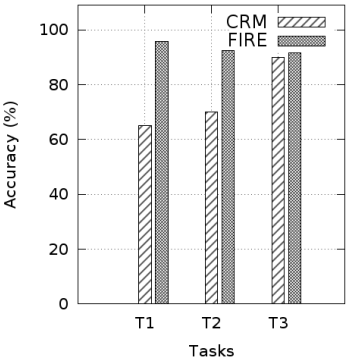
Stable Region Usage Study. As confirmed in Figure 2.44, subjects took less time when working with FIRE compared to that while using CRM.

This is because the tabular view in CRM does not provide any aid or intuition for subjects to accomplish the tasks.

As shown in Figures 2.44(a.) and 2.46(a.), for task T1, subjects spent 9 seconds on average using FIRE to get 100% accuracy while subjects used 62 seconds on average with CRM to achieve the same accuracy. For T2, the minimum time spent was 2 seconds using FIRE while that using CRM required was at least 26 seconds. Thus for T2, FIRE outperformed CRM in measures of accuracy by 5%.



(a) Mushroom



(b) Chess

Figure 2.46: Accuracy of T1, T2 and T3.

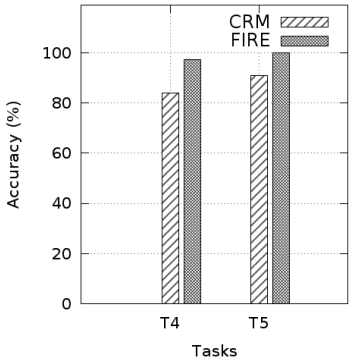


Figure 2.47: Accuracy of T4 and T5.

For T3, the maximum time spent with FIRE was 55 seconds, while in CRM it was 255 seconds. Subjects using FIRE achieved 100% correctness while in CRM this figure was 80% .

Similarly, in Figures 2.44(b.) and 2.46(b.), our subjects took less time using FIRE than CRM to complete all three tasks. At the same time, they made fewer mistakes using FIRE than CRM. In particular, the accuracy of T1 using FIRE was 30% higher than the accuracy of CRM. This is because more than one rule existed that satisfied the question in the chess dataset. Subjects tended to omit some rules that resulted in this low accuracy. In contrast, FIRE is able to reveal the full answer with just 1 or 2 clicks.

Filter/Redundancy + Skyline View Studies. In Figures 2.45 and 2.47 we show the time spent and accuracy for tasks T4 and T5, respectively. Again, subjects using FIRE spent less time to perform the tasks, yet were able to achieve better accuracy than subjects using CRM for the same task. More specifically, subjects used 29 seconds on average with FIRE yet achieved near 100% accuracy for T4. The subjects using CRM, on the other hand, took 80 seconds and reached only 84% accuracy. Overall, the results confirmed our hypothesis H1, i.e., our FIRE technology is a win-win in terms of both efficiency and accuracy.

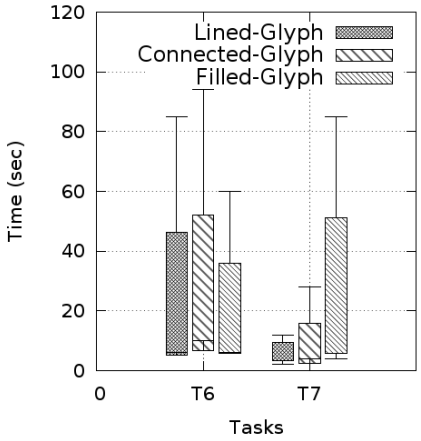


Figure 2.48: Time Spent on T6 and T7.

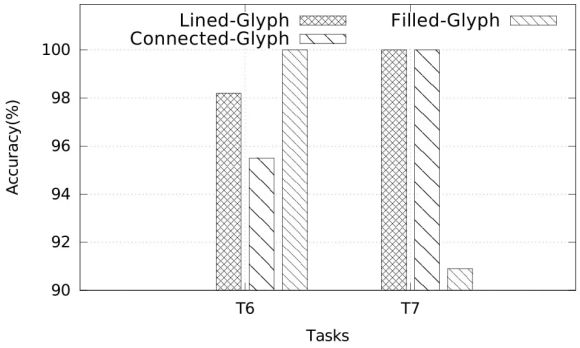


Figure 2.49: Accuracy of T6 and T7.

Glyph View Study. Figures 2.48 and 2.49 show the time spent and the accuracy when using the three glyph designs. The results confirmed our hypothesis H2. For task T6 that asked for antecedent similarity detection, the *filled* glyph indeed is proven to be the most effective among the three glyph designs. In particular, subjects spent 20 secs on average to correctly answer this similarity detection question using filled glyphs. Those using other glyph displays took longer time and yet committed several mistakes. For T7 involving counting of the number of consequents, the lined glyph showed an impressive efficiency (avg. 6 secs) and 100% accuracy. Most subjects rated the lined glyph as the easiest to use in their exit questionnaire. In T7 subjects using the connected glyph design also achieved 100% accuracy with a slightly higher time spent.

Glyph Placement Study. In Figures 2.50 and 2.52 we show the time spent and the accuracy of task T8 when using unclustered and clustered layouts. The subjects used less time when supported by our clustered layout, while they needed significantly more time using the unclustered layout. The fastest subject took only 1 second to complete this task with the help of our clustered layout. Accuracy-wise, subjects achieved 97% correctness using the clustered layout while only 80% accuracy was achieved by subjects with the unclustered layout. This is because our clustered layout essentially groups similar glyphs

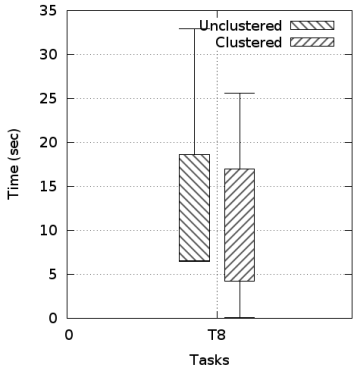


Figure 2.50: Time Spent on T8.

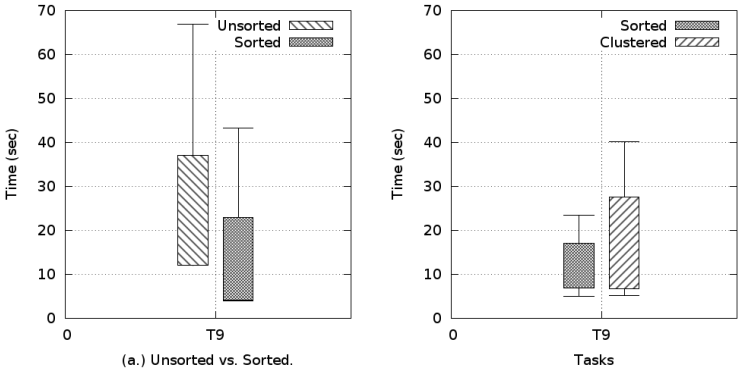


Figure 2.51: Time Spent on T9.

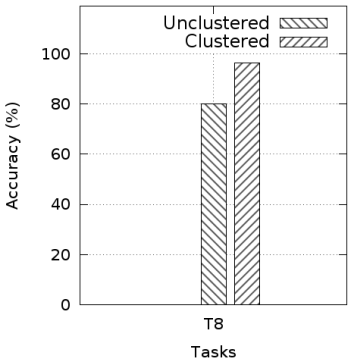


Figure 2.52: Accuracy of T8.

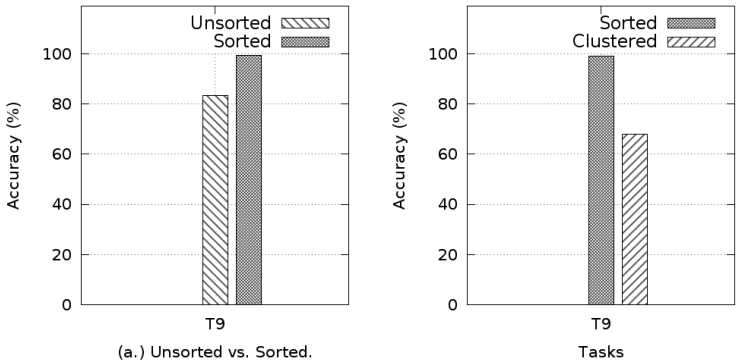


Figure 2.53: Accuracy of T9.

together and simultaneously unveils the outliers to subjects. The results confirmed our hypothesis H3. The subjects are able to leverage our clustered layout to recognize the outlier within a set of glyphs effectively.

For task T9, which asked the user to identify glyphs with a certain attribute-value pair, the sorted view indeed was proven to be most effective among the three placement strategies. As shown in Figures 2.51 and 2.53, the subjects using the sorted layout achieved 99% accuracy and took less time, while the subjects using the unsorted layout achieved 80% correctness and took more time. This is because the sorted view allows subjects to sort glyphs by a single attribute using the ”sort-by” function. The set of glyphs is thus classified by the specified attribute and the glyphs with the same value are naturally

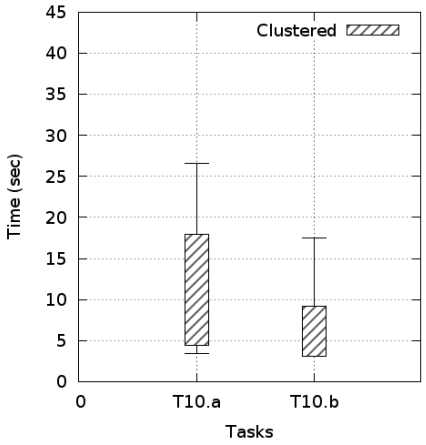


Figure 2.54: Time Spent on T10.

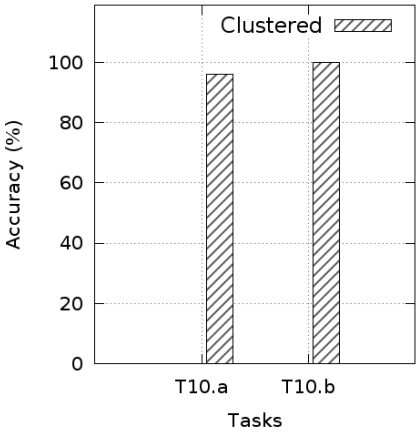


Figure 2.55: Accuracy of T10.

grouped together to facilitate search. Notable among these three layout strategies was the clustered layout, which does not behave well in this task. Clustered layout tends to group glyphs using all of their attributes instead of a designated one, which renders it less suitable for this task.

Figures 2.54 and 2.55 show the effectiveness of identifying groups of glyphs using the clustered layout. In particular, the subjects used on average 11 and 6 seconds, respectively, to achieve near 100% correctness on both questions in task T10. Our hypothesis H5 is thus confirmed. In our initial trial on the subjects, they were unable to perform this task well without the help of the clustered layout. The subjects could not group the glyphs correctly within an acceptable response time. Therefore, this task is best suited for the clustered layout.

Exit Questionnaire. Answers to Q1 and Q2 on the exit questionnaire are shown in Figure 2.56. There is a clear endorsement in favor of FIRE versus CRM, especially in test T5 where none of the subjects chose CRM over FIRE. The most common reason cited for this choice was the facilitated exploration of PSpace. The only exception was T1, as some subjects stated that they are more familiar with the sorted rules in the tabular view. In terms of ease of use, on a scale from 1 to 5, FIRE was rated 4.3 on average and

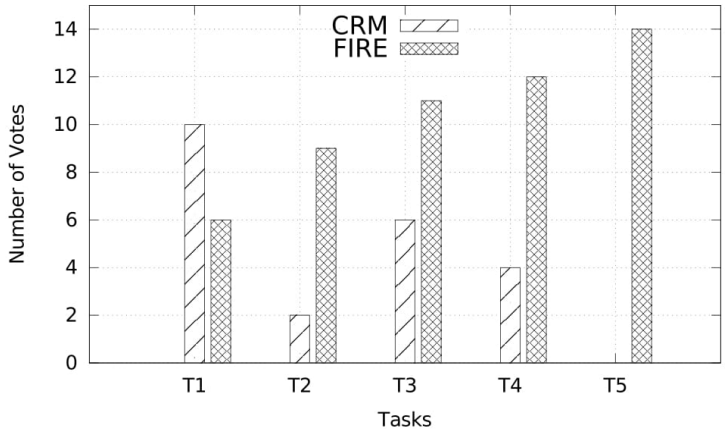


Figure 2.56: Votes on the Preference of CRM vs. FIRE Per Task.

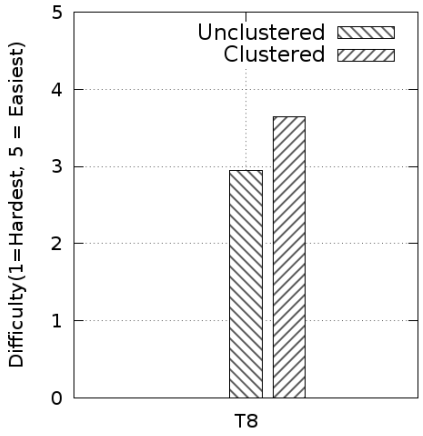


Figure 2.57: Survey Question on T8.

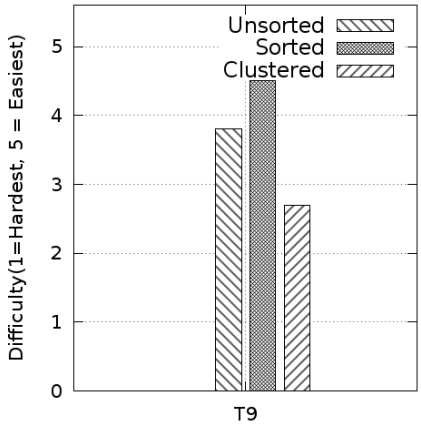


Figure 2.58: Survey Question on T9.

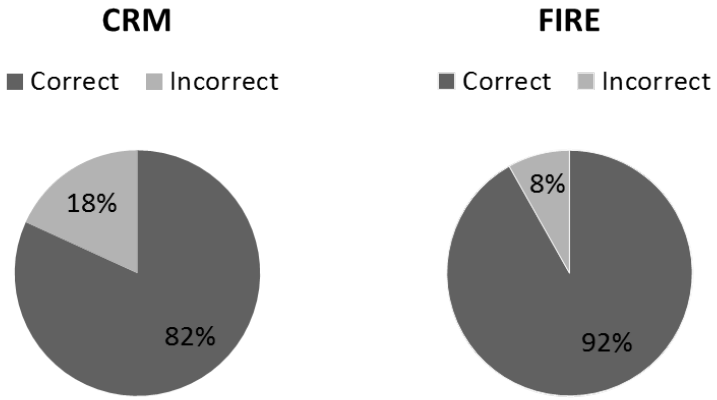


Figure 2.59: Overall Accuracy using CRM and FIRE.

CRM was rated 3. Here, 1 = very difficult and 5 = very easy. Figures 2.57 and 2.58 show the results for the glyph placement study that verified the task of finding glyphs with a given attribute-value pair using different layouts. On a scale from 1 to 5, the clustered layout was rated 2.7, the sorted layout was rated 4.5 and the unsorted glyph layout was rated 3.8. In terms of identifying dissimilar glyphs, the clustered view was rated 3 and the unclustered view was rated 3.7.

Overall, as shown in Figure 2.59, the user study showed that 92% of our subjects could perform the task correctly with FIRE while 82% of them produced correct answers with CRM. In addition, the glyph representation of rules and the glyph layout strategies offered users great benefits in association rule exploration. In conclusion, all hypotheses were confirmed by our user study. Our study shows that FIRE indeed aids human analysts in performing interactive rule exploration tasks efficiently and accurately.

2.9 Related Work

Parameter Space Exploration. Prior research has explored the space of parameters for handling parameterized database queries (CLN10) and tuning database configuration parameters (DTB09). Most data mining queries are parameterized, which, while making the algorithm flexible and tunable to one's own problem, often causes huge difficulty as typically the selection of appropriate parameter values is left to the human analysts. Closest to our work, (YRW09) aims to help analysts understand the relationship among clusters produced with different parameter settings to better understand good results for density-based clusters. We instead explore the parameter space for rule mining. Closest to our proposed parameter space display is the recent demonstration called AssocExplorer (LSZ$^+$12) that proposes a scatterplot of rules on a 2-D space. However, they overlook the visual clutter problem that is common even for a moderate number of rules. We tackle

the clutter problem with our proposed stable region, zoom and granularity features.

Interactive Association Rule Mining. Hahsler et al. (ARu15) presented the R-extension package *arulesViz* which implements several visualization techniques to display individual rules. In that sense, these efforts only focus on subset of our problem, namely, on designing displays for visualizing association rules as in our RSpace view. Analogous to our RSpace view, they work with standard visualizations found in visualization toolkits including variants of scatterplots, histograms and parallel coordinates visualization techniques. In this paper, we instead propose three variants of RSpace glyph views for graphically representing individual rules. We found rule glyphs and associated placement strategies to be well-suited to facilitate exploration and comparison among rules. These core techniques can potentially be integrated into ARulesViz as well.

Couturier et al. (CHYN07) proposed an integrated framework covering both rule extraction and visualization steps of the mining process. They provided a guided exploration based on clustering of rules. Neither of these approaches provide support for understanding the distribution of rules within the space of interestingness parameters (such as support, confidence and lift). Last but not least, unlike other efforts on interactive rule mining, a key contribution of our work is its focus on evaluating the usability of our FIRE framework via a formal user study.

Online Association Rule Mining. Online mining techniques (AY01a, KA08, KHR$^+$03) typically prestore the intermediate frequent itemsets. Here, we instead adopt the approach of rule prestoring from (LMR$^+$13) to achieve the required real-time interactive behavior. (LMR$^+$13, MLB$^+$13) propose to store the final rule results instead. They achieve near real-time responsiveness, laying the foundation for offering speedups sufficient for interactive rule exploration. However, sense-making of rulesets extracted from a data set, which is the topic of our current study, is not the focus of these existing rule mining systems (HFH$^+$09, LMR$^+$13, MLB$^+$13, ARu15, JB02).

Interestingness Measures as Parameters. Han et al. (WCH07) identify the importance of analyzing the interestingness measures of rules. They compare different null-invariant measures, such as confidence, to provide insights into similarities and differences among them. However, they do not tackle interactive rule mining through pre-computation as undertaken by our work. In a more recent work, Cao et al. (CLWY13) proposes a new interestingness parameter *Max Coverage Gain*. They also introduce a *MCGminer* algorithm with a series of built-in mechanisms and pruning strategies to handle complex rule interactions and reduce computational complexity towards identifying the globally optimal rule set in large imbalanced dataset. By extensive evaluation over 13 UCI datasets (UCI15), their metric is proven to be accurate, scalable, stable and effective. Our work is orthogonal to (WCH07, CLWY13) as we provide an overall framework for interactive rule mining. In other words, the parameters and strategies proposed in these works can be added to our framework to provide richer experiences to analysts.

Rule Relationships and Actionable High Utility Rules. Combined mining (Cao13) techniques focus on determining and managing various aspects of patterns such as rules, e.g., relationships among patterns, pattern representation, etc. Further, works on actionable high-utility itemset mining (SYLC15) establish that itemsets that are frequent may not necessarily be of high-utility. These works propose a new paradigm of utility functions to establish how significant a itemset/rule is, and bridge the gap between research outcomes and business needs. While these two relevant works make significant advances in discovering high-utility rules and defining complex rule relationships, our interactive FIRE engine powers the discovery process itself by presenting rules in an easy explorable manner. We use rule redundancy relationships as an example rule relationships, the concepts in these relevant works can be adopted into the backend (PARAS) of our visual FIRE engine to then together provide richer insights to the analysts for sense-making of association rules.

2.10 Conclusion

In this work we designed, implemented and evaluated innovative back-end and visualization technologies for interactive rule exploration called the PARAS and the FIRE frameworks. In our PARAS back end framework for fast online association mining, we propose a novel parameter space model for pre-storing rules such that a near real-time performance is guaranteed for online mining queries. FIRE offers parameter recommendations and enhanced sense-making of rule relationships. Particularly, we propose two linked views, namely, the PSpace and RSpace views. Both views are supplemented with innovative visualizations and interactions that enable analysts to effectively conduct visual rule exploration. While PSpace offers a rule distribution abstraction, RSpace facilitates detailed analysis of rules and their relations. In addition our novel RSpace glyph display enables visual comparison of rule shapes further augmented by glyph placement strategies (War02).

Our case study using the Bike sharing dataset (Bik15) illustrates the capabilities of the FIRE system and compares it with that of the state-of-the-art ARulesViz rule visualization techniques. Further, our user study with 22 subjects demonstrates the usability and effectiveness of the proposed FIRE framework using several benchmark datasets.

3

Localized Rule Mining with COLARM

3.1 Preliminaries

We first describe how we represent itemsets in a multidimensional space and then formulate the *online localized rule mining problem* using the terms defined in Table 3.1.

3.1.1 Itemsets in Multidimensional Space

Consider a relational database $\mathcal{D}$ with *n* attributes $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. Let there be *m* data records in $\mathcal{D}$. Each data record r consists of value $<$v$_1$,. . .,v$_i$,. . .,v$_n$$>$, where v$_i$ corre-

Symbol	Definition
n	Total number of attributes in a relation.
minsupp	User-specified min. support.
minconf	User-specified min. confidence.
A^{range}	Range attribute-value pairs in the WHERE clause of Q.
A^{item}	Item attributes specified in the WHERE clause of Q.
$\mathcal{D}^Q$	User-chosen focal subset.
$Supp_I^G$	Global support of an itemset I in the complete dataset $\mathcal{D}$.
$Supp_I^Q$	Local support of an itemset I w.r.t. the subset $\mathcal{D}^Q$.
$Conf_R^G$	Global confidence of Rule R in the complete dataset $\mathcal{D}$.
$Conf_R^Q$	Local confidence of Rule R w.r.t. the subset $\mathcal{D}^Q$.
$\{I_S^Q\}$	Set of candidate itemsets w.r.t. $\mathcal{D}^Q$ output by SEARCH.
$\{I_E^Q\}$	Set of candidate itemsets in $\mathcal{D}^Q$ output by ELIMINATE.
$\{R^Q\}$	Set of local rules in $\mathcal{D}^Q$ output by VERIFY.

Table 3.1: List of Notation.

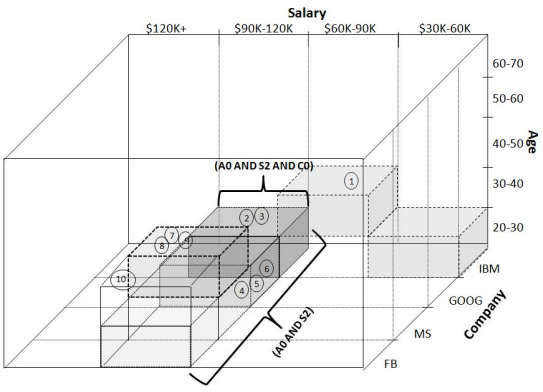


Figure 3.1: Itemsets in n-Dimensional Space.

sponds to attribute $\mathcal{A}_i$ of the *n* attributes. Rule mining (AIS93) works with only nominal attributes, while quantitative attributes are first discretized[1]. For $\mathcal{D}$, a single attribute-value pair ($\mathcal{A}_i$ = v_i) forms an *item* and a collection of such items is called an *itemset*. For example, in the salary dataset (Table 1.2), A0=(Age=20-30) is an *item*. (Age=20-30, Salary=90K-120K), also denoted as (A0,S2), is a *2-itemset* as it consists of two items. Further, (Age=20-30, Salary=90K-120K, Company=IBM), denoted as (A0,S2,C0), is a *3-itemset*.

For ease of depiction, in Figure 3.1 we assume that the salary dataset consists of three dimensions, namely, age, salary and company. Here, the records $\{2,3\}$ belong to the cell (A0, S2, C0). The cells (3-itemsets), namely, (A0, S2, C0), (A0, S2, C1), (A0, S2, C2) and (A0, S2, C3) combine to form the 2-itemset (A0,S2). The bounding box for (A0,S2) is shown in Figure 3.1. This conforms to the *downward closure property*[2] of itemsets that we utilize in our work. While cells (A0, S2, C0) and (A0, S2, C1) contain records $\{2,$ $3\}$ and $\{4, 5, 6\}$ respectively, cells (A0, S2, C2) and (A0, S2, C3) are empty. Thus, in general we divide an n-dimensional space into (n-itemset) *cells* at the lowest granularity. Several i-itemsets can be combined to form each (i-1)-itemset. This bottom up process can be repeated until 2-itemsets are composed.

3.1.2 Localized Association Mining Problem

An online mining query for *localized* rules can be specified by query Q. Given a dataset $\mathcal{D}$, localized rules valid in the *focal subset* $\mathcal{D}^Q$ are requested. The *range parameter* $\mathcal{A}^{range}$ in the WHERE clause specifies $\mathcal{D}^Q$. For each range attribute $\mathcal{A}_i^{range}$, the user selects values, where $v_{i,p}^{range}$ denotes the p^{th} value.

[1]Discretization of quantitative data (e.g., discretization of attribute Age as $\{$20-30, 30-40, ...$\}$ versus $\{$20-40,40-60,...$\}$) is an orthogonal problem and existing works (SA96, HF95) can be applied offline to achieve the best discretization for a dataset.

[2]For a frequent itemset, all its subsets are also frequent and thus for infrequent itemset, all its supersets must also be infrequent.

```
Q: REPORT LOCALIZED ASSOCIATION RULES

FROM Dataset 𝒟

WHERE RANGE 𝒜^range = { 𝒜_1^range = (v_{1,1}^range, ...), ..., 𝒜_k^range = (v_{k,1}^range, ...) }

AND

[ ITEM ATTRIBUTES 𝒜^item = {𝒜_1^item, ..., 𝒜_f^item} ]

HAVING minsupport = minsupp and

minconfidence = minconf;
```

For the salary dataset (Table 1.2), (Age={20-30, 30-40} and Company = {IBM}) forms an example focal subset $\mathcal{D}^Q$ denoting the IBM employees between ages 20 and 40. By default $\mathcal{A}_i^{range}$ spans over the domain of $\mathcal{A}_i^{range}$. The users can optionally use the *item* clause ($\mathcal{A}^{item}$) to specify the attributes for generating rules. Therefore, ($\mathcal{A}^{range} \subseteq \mathcal{A}$) and ($\mathcal{A}^{item} \subseteq \mathcal{A}$).

Let the user-defined focal subset be $\mathcal{D}^Q$. Size of the focal subset ($|\mathcal{D}^Q|$) is the number of tuples in the focal subset. It is computed by counting the tuples overlapping with the *range parameter* $\mathcal{A}^{range}$. For an itemset I, if $\mathcal{D}_I^Q$ denotes the records of $\mathcal{D}^Q$ that support I, the *local support* for I (Supp_I^Q) = $\frac{|\mathcal{D}_I^Q|}{|\mathcal{D}^Q|}$. Itemset I is *frequent* in $\mathcal{D}^Q$ if $\text{Supp}_I^Q \geq$ *minsupp*. Further, for a rule R = (X $\Rightarrow$ Y), where I = (X $\cup$ Y), the *local confidence* Conf_R^Q is given by the fraction $\frac{Supp_I^Q}{Supp_X^Q}$. Only if $\text{Conf}_R^Q \geq$ *minconf*, Rule R is included in output $\{R^Q\}$. As *range* and *item* attributes are known only at query-time, offline precomputation of local support and confidence values is not possible. The overall problem is formulated below.

Definition 3.1 ***Localized Association Mining Problem:*** *Given the range attributes $\mathcal{A}^{range}$, the item attributes $\mathcal{A}^{item}$ and the thresholds, namely, minsupp and minconf, find the set of association rules $\{R^Q\} = \{R_1^Q, R_2^Q, \ldots, R_m^Q\}$ valid for the focal subset $\mathcal{D}^Q$ (defined by $\mathcal{A}^{range}$). such that for every rule $R_k^Q = X_k \Rightarrow Y_k$, $X_k \cup Y_k \subseteq \mathcal{A}^{item}$, the local support $Supp_k^Q \geq$ minsupp and the local confidence $Conf_k^Q \geq$ minconf.*

3.2 The COLARM Design

To process a localized rule mining query Q, our approach adapts a *preprocess-once-query-many* (POQM) paradigm-based solution composed of two phases, namely, *offline preprocessing* and *online query processing* as described below.

3.2.1 The COLARM Framework

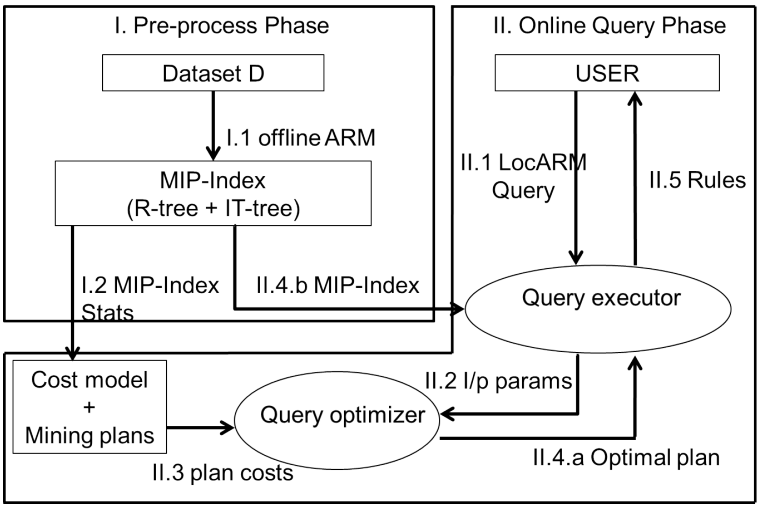
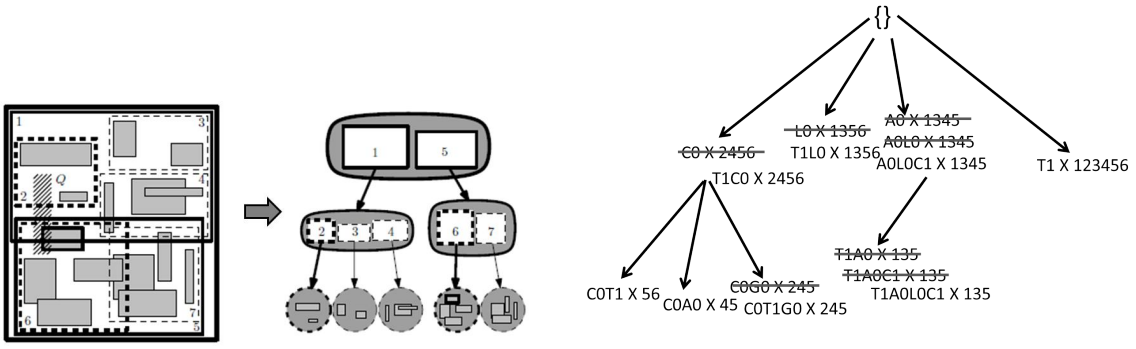


Figure 3.2: The COLARM framework.

We now give an overview of our proposed approach that we call the COLARM framework (Figure 3.2). The framework consists of an *offline preprocessing phase* and an *online query processing phase*. The *offline preprocessing phase* computes and stores the itemset information using efficient index structures called the MIP-index (explained in Section 3.2.3). The index statistics are also pre-computed and stored for analysis of online mining strategies. In the *online query processing phase*, a user submits a mining request that consists of *minsupp*, *minconf* and focal subset $\mathcal{D}^Q$.

In this work, we propose a suite of six alternate mining plans for executing the localized mining request submitted by the user. The cost for each mining plan is derived and discussed in detail in Section 3.2.4. Based on the index statistics provided by the



(a) R-tree (b) Closed IT-tree

Figure 3.3: The two-level MIP-index.

offline preprocessing phase and the online query parameters, namely, *minsupp*, *minconf* and focal subset $\mathcal{D}^Q$, our proposed COLARM query optimizer estimates the costs of the alternate plans. The estimates are a constant time computation of six formulae, each corresponding to one mining plan. For a given mining request, the COLARM optimizer suggests the plan with the lowest estimated cost which is then used by the executor to process the online mining request.

3.2.2 Offline Preprocessing Phase

The preprocessing phase is a one-time offline step. For answering localized mining queries using POQM, we extend ideas from two works. First, as summarized by Boulicaut (**?**), existing works prestore frequent itemsets in a compact hierarchical itemset-based index such as a closed IT-tree (ZH02). At query-time, the index can be utilized for rule generation and to verify if the generated rules qualify the minsupp and *minconf* thresholds. Next, Das et al. (DPDK11) answer windowed frequent itemset queries by prestoring the bounding boxes of the window attributes. We thus propose to utilize a multidimensional index to store bounding boxes of itemsets. But in our relational model as *range* and *item* are specified at query-time, the multidimensional index must be more flexible than required by their transactional model. In contrast to these two works, we find that our

target problem requires both features of an itemset to be prestored. Thus, for each itemset we prestore (a.) the bounding box of the itemset within the multidimensional space and (b.) the items composing the itemset in a compact hierarchical index. While feature (a.) enables us to search overlaps between the focal subset and the pre-stored itemsets, (b.) helps in efficient online rule generation and threshold verification.

Similar to prior online rule mining works (AY01a, DPDK11), we prestore all itemsets that satisfy a domain-specific *primary support threshold* by employing a rule mining algorithm (Charm (ZH02)) to collect all the closed frequent itemsets at an offline step. However, the support and confidence of the rules composed from the prestored itemsets are determined at query-time based on the focal subset. Extending our knowledge from Section 3.1.1, in a multidimensional space of n (=3 in Figure 3.1) attributes, we construct a hierarchy of itemsets upto 2-itemsets, starting with the *n-itemset* cells at the finest granularity. This hierarchical collection of itemset partitions the multidimensional space into bounding boxes denoted by $\{\mathcal{D}^P\}$. We call these partitions *Multidimensional Itemset Partitions* (MIPs). Each MIP, denoted by I_k^P, represents both the bounding box $\mathcal{D}_k^P$ of the itemset in a multidimensional space as well as the actual items (attribute-value pairs) composing the itemset I. In other words, the symbols $\mathcal{D}_k^P$ and I_k^P are henceforth used interchangeably in the rest of this document to denote a MIP for an itemset I. As shown in Figure 3.1, each *i-itemset* MIP at level i is composed of one or more (i+1)-itemset MIPs. This MIP-index enables *downward closure property* to be implicitly applied during online mining such that if an MIP I at level i does not qualify the *minsupp*, all (i+1)-itemset MIPs that contain I can also be eliminated.

3.2.3 The Two Level MIP-index

We adopt a two-layered structure to store the two features of the MIPs, namely, (a.) the bounding boxes and (b.) the items composing the itemsets, as follows.

The Multidimensional Index for MIPs. The bounding boxes of MIPs can be indexed using any multidimensional index. Here, we use an R-tree, as it is proven to be efficient for searches over multidimensional boxes (KF93). An R-tree index can be used to perform a *range* search to retrieve the MIPs that overlap with the focal subset $\mathcal{D}^Q$. An R-tree supports partitions of different granularities as well as overlaps and containments among partitions. An example *R-tree* is shown in Figure 3.3.(a).

The Closed IT-tree for Itemsets. We employ the *itemset-tidset search tree* (ZH02), or in short the *IT-tree*[1]. The *IT-tree* is compact as it stores only the *closed* frequent itemsets. As shown by Zaki et al. (ZH02), there are significant gains both in storage and computation time by utilizing the *closed IT-tree* for rule generation. An example *closed IT-tree* is shown in Figure 3.3.(b).

Offline MIP-index Construction. Construction of MIP-index at the offline step consists of first generating all closed frequent itemsets (CFIs) using the CHARM algorithm (ZH02) using the primary support threshold. The generated CFIs are stored in an IT-tree. The details of time and space complexity of generating CFIs and constructing IT-tree can be found in (ZH02). Further, we construct the R-tree using bounding boxes of the closed frequent itemsets. As this is a one time offline construction, we employ the R-tree packing scheme proposed by (KF93). They proposed a method to build a packed R-tree that achieves (almost) 100% space utilization. A detailed time and space analysis for R-tree construction can be found in (KF93). In this work we focus on online query processing costs that are more important than costs of one-time offline MIP-index construction costs for the problem of online localized rule mining.

[1]Detailed description of the IT-tree can be obtained from (ZH02).

3.2.4 Online Query Processing

In the *online query processing phase*, the user submits a localized mining query Q with four parameters $\mathcal{A}^{range}$, $\mathcal{A}^{item}$, *minsupp* and *minconf*. $\mathcal{A}^{range}$ defines the focal subset $\mathcal{D}^Q$. The candidate frequent itemsets for $\mathcal{D}^Q$ are identified by performing a range search using the focal subset $\mathcal{D}^Q$ over the MIPs $\{\mathcal{D}^P\}$ pre-stored in a R-tree. The *candidate itemsets*, denoted by $\{I_S^Q\}$, only contain the itemsets within the *item attributes* $\mathcal{A}^{item}$. Next, for each *candidate itemset* I ϵ $\{I_S^Q\}$, rules are generated using the IT-tree and the *minsupp* and *minconf* thresholds with respect to the focal subset $\mathcal{D}^Q$ are verified. Here, we make a simplifying assumption that the users are allowed to specify $\mathcal{A}^{range}$ with the prestored n-itemset cells at the lowest granularity to avoid sub-cell computations. For example, if age attribute is specified in the MIP-index with increments of 10 as {20-30, 30-40, ...}, the user selection must align with them and ranges such as Age=25-30 or Age=35-40 cannot be specified. This is a valid assumption as optimal discretization decisions using key ideas from (SA95, SA96, HF95) can be made for a given dataset during preprocessing based on the domain and the data characteristics.

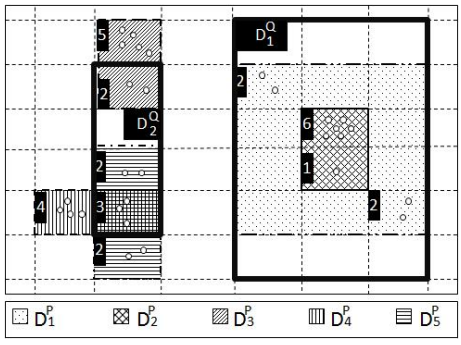


Figure 3.4: Itemsets Covering the Cells.

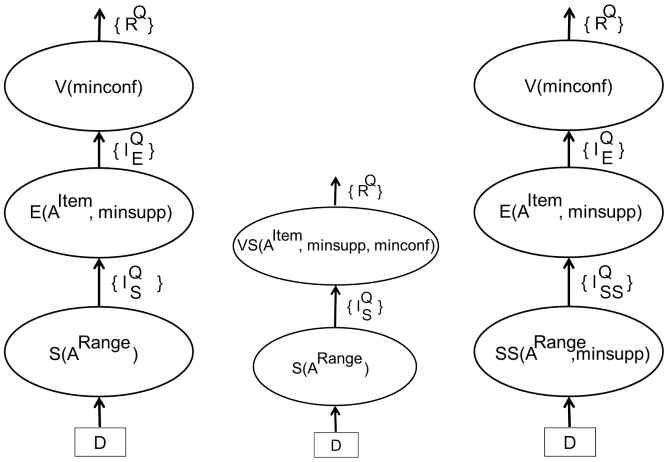
Containment vs. Partial Overlap. Based on the expanse of the focal subset $\mathcal{D}^Q$, the precomputed MIPs can be categorized into three mutually exclusive groups, namely, *contained within* $\mathcal{D}^Q$ ($\{\mathcal{D}^P\}_c$), *partially overlapping with* $\mathcal{D}^Q$ ($\{\mathcal{D}^P\}_p$) and *disjoint with* $\mathcal{D}^Q$ ($\{\mathcal{D}^P\}_d$). Figure 3.4 illustrates five MIPs (marked with different patterns) of a dataset

$\mathcal{D}$, $\{\mathcal{D}_1^P, \ldots, \mathcal{D}_5^P\}$. For an example *focal subset* $\mathcal{D}_1^Q$, the MIPs $\mathcal{D}_1^P$ and $\mathcal{D}_2^P$ are fully *contained* in it, i.e., $\{\mathcal{D}^P\}_c = \{\mathcal{D}_1^P, \mathcal{D}_2^P\}$. There are no *partially overlapped* MIPs, i.e., $\{\mathcal{D}^P\}_p = \emptyset$. The rest are *disjoint* MIPs, i.e., $\{\mathcal{D}^P\}_d = \{\mathcal{D}_3^P, \mathcal{D}_4^P, \mathcal{D}_5^P\}$. Example focal subset $\mathcal{D}_1^Q$ is straightforward to process as only *contained* MIPs exist eliminating the need for record-level processing.

The focal subset $\mathcal{D}_2^Q$ (Figure 3.4) represents a distinct scenario with *partially overlapped* MIPs $\{\mathcal{D}^P\}_p = \{\mathcal{D}_3^P, \mathcal{D}_4^P, \mathcal{D}_5^P\}$ and *disjoint* MIPs $\{\mathcal{D}^P\}_d = \{\mathcal{D}_1^P, \mathcal{D}_2^P\}$. There are no *contained* MIPs, i.e., $\{\mathcal{D}^P\}_c = \emptyset$. In such cases, for each *partially overlapped* MIP $\mathcal{D}_k^P$ ($\epsilon \{\mathcal{D}^P\}_p$), the records lying at the intersection of $\mathcal{D}^Q$ and $\mathcal{D}_k^P$ must be collected using a costly database scan. The collected records must then be used to verify the thresholds for the *candidate itemset* $\{I_k^P\}$. Hence, processing partially overlapped MIPs is much costlier than processing fully contained MIPs. In a query scenario, different $\mathcal{D}^Q$ may vary in their sizes and locations within the multidimensional space. Overall, a variety of query scenarios are possible ranging from all *contained* MIPs, to mix of *contained* and *partially overlapped* MIPs, to all *partially overlapped* MIPs. A single solution may not be suitable to process all query scenarios in the most efficient manner. Therefore, we develop a suite of alternate *mining plans* and an online optimizer for selection of the most efficient plan to execute online localized mining requests.

3.3 Strategies for Online Mining

As opposed to treating the rule mining process as a black box, we now isolate each step in the process as an operator. Each operator has precise inputs, outputs and functionality. The goal is to find scope for optimizing each isolated operator without affecting the other operators. We first present the overall plan for processing online localized rule mining queries using the MIP-index, followed by our proposed optimizations. The plan consists



(a) S-E-V (b) S-VS (c) SS-E-V

Figure 3.5: The POQM Mining Plans.

Symbol	Definition
C^I	Number of singleton items in an Itemset I.
TR_{const}	Constant cost of reading records from an R-tree node.
h	Height of the R-tree.
N_j	Number of nodes at level j of the R-tree.
$\mathcal{D}^Q_{i_{avg}}$	Avg. extent of the i^{th} range attribute in the focal subset.
$\mathcal{D}^P_{j,i_{avg}}$	Avg. extent of the MIPs in R-tree at level j and attribute i.
{R}	Candidate set of rules for confidence check.

Table 3.2: Notation Used in Cost Estimates.

of a pipeline of three basic operators as shown in Figure 3.5.(a).

1. **SEARCH**: Given the *range* attributes $\mathcal{A}^{range}$ in the WHERE clause of Query Q, the R-tree is searched to output overlapping *candidate itemsets* denoted by $\{I^Q_S\}$. The *SEARCH* operator is defined as $\mathbf{S}[\mathcal{A}^{range}, R-tree] \longrightarrow \{I^Q_S\}$. Details of the R-tree search can be found in (TSS00).

2. **ELIMINATE**: Given the list of *candidate itemsets* $\{I^Q_S\}$ output by the *SEARCH* operator and the *item* attributes $\mathcal{A}^{item}$, the support of the candidate itemsets must satisfy the *minsupp* and the itemsets must be composed of only the *item* attributes $\mathcal{A}^{item}$. Thus, a reduced list of candidate itemsets, denoted as $\{I^Q_E\}$, is produced. The *ELIMINATE* operator is defined as $\mathbf{E}[\{I^Q_S\}, \mathcal{A}^{item}, \boldsymbol{minsupp}] \longrightarrow \{I^Q_E\}$.

3. **VERIFY**: Given the reduced list of candidate itemsets $\{I^Q_E\}$, the *closed IT-tree*

(ZH02) and the records in $\mathcal{D}^Q$ are used to first *generate* rules and then *verify* whether the confidence values of the rules satisfy the *minconf* threshold. The rules $\{R^Q\}$ are returned to the user. The *VERIFY* operator is defined as $\mathbf{V}[\{I_E^Q\}$**, *minconf***$] \longrightarrow \{R^Q\}$. Refer to (ZH02) for details on the traditional algorithms for rule generation and minconf verification using the *IT-tree*.

Below, we apply several novel optimization principles in the context of online localized rule mining. Each resulting optimized plan comes, not only with the benefits achieved, but also with some overhead costs. We design a model for estimating the costs of each plan. For each online request, a single most efficient plan is used for execution. We conclude this section with a summary of the six proposed alternative mining plans in Table 3.3.

3.3.1 The Basic S-E-V Plan

The basic mining plan can be composed by pipelining the three operators, namely, *SEARCH*, *ELIMINATE* and *VERIFY*. The resultant *S-E-V plan* is depicted in Figure 3.5.(a). The *SEARCH* operator, performing an R-tree search to output the overlapping MIPs $\{I_S^Q\}$, works as an inexpensive *coarse* granularity filter. For each candidate itemset I ϵ $\{I_S^Q\}$, the *ELIMINATE* operator filters items using $\mathcal{A}^{item}$ and verifies if local support Supp_I^Q exceeds the user-defined *minsupp*. The qualified candidate itemsets $\{I_E^Q\}$ (where $\{I_E^Q\} \subseteq \{I_S^Q\}$) are used in the *VERIFY* operator to generate rules. For each rule r, only if its local confidence (Conf_r^Q) satisfies the user-defined *minconf* will r get included in the final output $\{R^Q\}$. The detailed algorithm for threshold verification and rule generation using the *IT-tree* can be found in (ZH02). Both the *ELIMINATE* and the *VERIFY* operators require *finer* granularity checks at the *record-level*, i.e., the records that lie at the intersection of I and $\mathcal{D}^Q$. The candidate itemsets $\{I_S^Q\}$ and $\{I_E^Q\}$ produced by *SEARCH* and *ELIMINATE*

respectively may contain *false positives* but are guaranteed to not generate any *false negatives* as even the partially overlapping itemsets are pushed up as candidates, but itemsets that must form the answers are never eliminated.

Execution Costs and Analysis. The cost of the *SEARCH* operator (i.e., an R-tree lookup), depends on the expected number of disk accesses (TSS00). Lemma 3.3.1[1] gives an estimated count of the candidate itemsets ($\{I_S^Q\}$) output by *SEARCH*.

An R-tree storing a set of N MIPs $\{\mathcal{D}_1^P, \ldots, \mathcal{D}_N^P\}$ *with average extent for the* i^{th} *attribute as* $\mathcal{D}_{i_{avg}}^P$ *and the focal subset* $\mathcal{D}^Q$ *with average extent of the* i^{th} *attribute as* $\mathcal{D}_{i_{avg}}^Q$*, for n attributes; the average number of candidate itemsets* $\{I_S^Q\}$ *intersected by* $\mathcal{D}^Q$ *is approximately computed as:*

$$|\{I_S^Q\}| = N \times \prod_{i=1}^{n} (\mathcal{D}_{i_{avg}}^P + \mathcal{D}_{i_{avg}}^Q)$$

While the cost for *ELIMINATE* is given in Equation 3.1, Lemma 3.3.1[2] gives the estimated count of candidate itemsets ($\{I_E^Q\}$) output by *ELIMINATE*.

Given a set of candidate itemsets $\{I_S^Q\}$ *with itemset I having local support* $Supp_I^Q$ *and the focal subset* $\mathcal{D}^Q$ *with threshold requirement of minsupp, the average number of candidate itemsets* $\{I_E^Q\}$ *output by ELIMINATE is:*

$$|\{I_E^Q\}| = \sum_{i \epsilon \{I_S^Q\}} (Supp_i^Q + minsupp)$$

In the *VERIFY* operator, the output ruleset $\{R^Q\}$ can be generated using the IT-tree. For an itemset I, one needs to traverse the IT-tree up to the level of I in the IT-tree (Lemma 3.3.1[3]). The overall cost of the *S-E-V plan* as an addition of the individual costs is shown in Equation 3.1.

[1] Derived from the R-tree search algorithm (TSS00).
[2] Derived from the IT-tree minsupp check algorithm (ZH02).
[3] Derived from the IT-tree rule generation algorithm (ZH02).

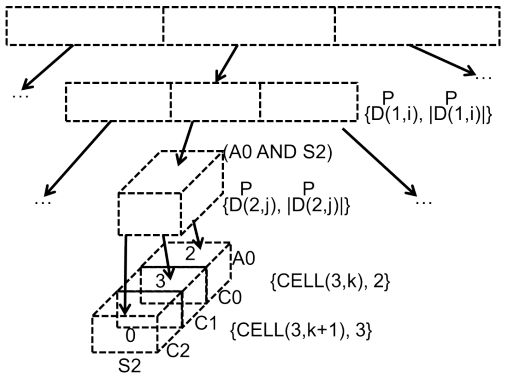
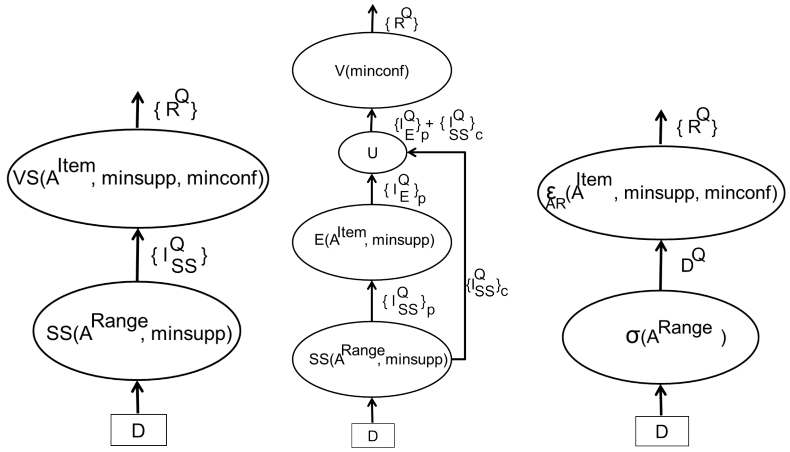


Figure 3.6: Supported R-tree.

The level of the IT-tree at which an itemset I exists equals the number of singleton items composing I denoted by C^I.

$$
\begin{aligned}
&COST(S-E-V) = COST(S)\text{+}COST(E)\text{+}COST(V), where,\\
&COST(S) = [\sum_{j=1}^{h-1}\{N_j \times \prod_{k=1}^{n}(\mathcal{D}^P_{j,i_{avg}} + \mathcal{D}^Q_{k_{avg}})\}],\\
&COST(E) = [|\{I^Q_S\}| \times |D^Q|],\\
&COST(V) = [\sum_{i\epsilon\{I^Q_E\}}(C^i \times |D^Q|)\text{+}\sum_{r\epsilon\{R\}}(Conf^Q_r + minconf)].
\end{aligned}
\tag{3.1}
$$

The SEARCH is inexpensive as it is a coarse granularity check. However, if a larger range is chosen by the user it may require multiple disk accesses and may impact the execution costs. In summary, the S-E-V plan would perform well for small $\mathcal{D}^Q$ expanses when most of the MIPs are filtered out in the R-tree search. The VERIFY operator requires record-level operations and multiple iterations over the IT-tree. Thus it tends to be the bottleneck of this plan. The S-E-V plan is effective if the ELIMINATE achieves high reduction of candidate itemsets from $\{I^Q_S\}$ to $\{I^Q_E\}$, with possibly low overhead. The smaller the set $\{I^Q_E\}$ reaching the costly VERIFY operator, the better the plan's performance. However, a small overhead for performing the record-level ELIMINATE operator arises.



(a) SS-VS (b) SS-E-U-V (c) ARM

Figure 3.7: More Mining Plans.

3.3.2 Pushing Selection Up The Plan

The reduction of the candidate itemsets by ELIMINATE is factor affecting the effectiveness of the S-E-V plan. However, if the expensive record-level ELIMINATE filters no or very few candidate itemsets, i.e., $\{I_E^Q\} \simeq \{I_S^Q\}$, then ELIMINATE no longer remains a beneficial filter. We now introduce the *S-VS strategy* that rewrites the S-E-V plan by merging the ELIMINATE and the VERIFY operations into a single operator called the *SUPPORTED-VERIFY* operator (defined below). The *S-VS* plan is depicted in Figure 3.5.(b).

SUPPORTED-VERIFY Operator. This operator is represented as $VS[\{I_S^Q\}, \mathcal{A}^{item}$, *minsupp*, *minconf*$] \longrightarrow \{R^Q\}$. It takes as input the candidate itemsets from the SEARCH ($\{I_S^Q\}$), the item attributes $\mathcal{A}^{item}$ and the thresholds *minsupp* and *minconf* from the query. It first filters itemsets not in $\mathcal{A}^{item}$. With the remaining qualified itemsets, a set of rules $\{R\}$ is generated but only the rules satisfying both *minsupp* and *minconf* are output as $\{R^Q\}$.

Execution Costs and Analysis. SEARCH cost for S-VS remains the same as S-E-V. The *minsupp*-based elimination of itemsets is interleaved with the *minconf* verification using the IT-tree. For cases $\{I_E^Q\} \simeq \{I_S^Q\}$, S-VS is bound to outperform S-E-V. This plan

is depicted in Fig. 3.5.(b) and Eq. 3.2 shows the overall costs.

$$\begin{aligned} &COST(S-VS) = COST(S)\text{+}COST(VS), where, \\ &COST(VS) = [\sum_{i\epsilon\{I_S^Q\}}(C^i \times |\mathcal{D}^Q|)\text{+}\sum_{r\epsilon\{R\}}(Conf_r^Q + minconf)]. \end{aligned} \tag{3.2}$$

3.3.3 The Supported R-tree Filter

In the above two plans, SEARCH selects all MIPs overlapping with with $\mathcal{D}^Q$ for the expensive record-level support check irrespective of the extent of overlap, i.e., MIPs overlapping by as few as a single record are also chosen. In cases where $\mathcal{D}^Q$ partially overlaps with a large number of MIPs, an excessive number of record-level support checks are needed. With the goal of further reducing $\{I_S^Q\}$, we now design the *SS-E-V* plan (depicted in Fig. 3.5.(c)). The SEARCH operator is modified using the insights from Lemma 3.3.3.

Given a focal subset $\mathcal{D}^Q$ ($\subseteq \mathcal{D}$) and any itemset I whose bounding box overlaps $\mathcal{D}^Q$, the number of records in $\mathcal{D}^Q$ that contain I, denoted by $|\mathcal{D}_I^Q|$, has an upper bound $|\mathcal{D}_I^G|$, where $|\mathcal{D}_I^G|$ denotes the total number of records supporting I in the full dataset $\mathcal{D}$. In short, $|\mathcal{D}_I^Q| \leq |\mathcal{D}_I^G|$.

Let $|\mathcal{D}_I^G|$ (e.g., 2000) denote the global count of prestored itemset I with respect to the entire dataset $\mathcal{D}$ (e.g., 100000). Given a focal subset $\mathcal{D}^Q$ (e.g., 10000) and *minsupp* = 25%, we define $\frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$ (here, 2000/10000 = 20%) as the upper bound of the local support of I, i.e., $\text{Supp}_I^Q \leq \frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$ (using Lemma 3.3.3). $\text{Supp}_I^Q = \frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$ only if all records of $\mathcal{D}$ that contain I are included in $\mathcal{D}^Q$. By prestoring global support counts ($|\mathcal{D}_I^G|$) of an itemset I with its MIP bounding box inside the R-tree, we can eliminate I if *minsupp* $> \frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$. This customized R-tree structure is henceforth called the **Supported R-tree** (Figure 3.6). This *supported R-tree filter* may significantly reduce the list of candidate itemsets without having to perform a record-level support checks. The modified *SEARCH* operator exploiting the supported R-tree is defined below.

SUPPORTED-SEARCH Operator. The SS[$\mathcal{A}^{range}$, *minsupp*] $\longrightarrow \{I_{SS}^Q\}$ operator

takes in the range attributes $\mathcal{A}^{range}$ and the *minsupp* threshold. It outputs the candidate itemsets $\{I^Q_{SS}\}$ that overlap with the $\mathcal{A}^{range}$ only if their global support counts exceed the *minsupp*. Thus the SS operator works as a *coarse granularity supported R-tree filter* together with performing the range search.

Execution Costs and Analysis. The costs incurred by the SS-E-V are shown in Equation 3.3. Unlike the regular SEARCH operator, the selectivity of the SS operator also takes the support of the stored MIPs and the *minsupp* into account. This coarse granularity *minsupp* check can be cheaply interleaved with the range search. The costs of ELIMINATE and VERIFY are identical to Equation 3.1. The key contribution of the SS-E-V plan is that the SUPPORTED-SEARCH potentially generates fewer candidate itemsets ($\{I^Q_{SS}\}$) than produced by the original SEARCH operator, i.e., $\{I^Q_{SS}\} \ll \{I^Q_S\}$. Thus, potentially reducing the inputs to ELIMINATE and VERIFY operators.

$$
\begin{aligned}
&COST(SS-E-V) = COST(SS)\text{+}COST(E)\text{+}COST(V), where,\\
&COST(SS) = [\sum_{j=1}^{h-1}\{N_j \times \prod_{k=1}^{n}(\mathcal{D}^P_{j,k_{avg}}\text{+}\mathcal{D}^Q_{k_{avg}})\times\\
&(Supp_j + minsupp)\}].
\end{aligned}
\tag{3.3}
$$

3.3.4 Pipelining both Supported Operators

The SS-E-V plan can run into the same issues with the ELIMINATE as the S-E-V plan, i.e., if $\{I^Q_E\} \simeq \{I^Q_{SS}\}$, then ELIMINATE may not be an effective filter. We may instead opt to pipeline SUPPORTED-SEARCH and SUPPORTED-VERIFY operators resulting in the *SS-VS* plan (Figure 3.7.(a)).

Execution Costs and Analysis. The cost incurred by the SS-VS (Equation 3.4) is the sum of the costs of SS and VS operators. The VS operator processes candidate itemsets $\{I^Q_{SS}\}$ output by SS. SS-VS is guaranteed to outperform SS-E-V when $\{I^Q_E\} \simeq \{I^Q_{SS}\}$.

$$
COST(SS-VS) = COST(SS)\text{+}COST(VS). \tag{3.4}
$$

Mining Plan	Optimization	Query Cost
S-E-V	Basic SEARCH+ELIMINATE+VERIFY plan	COST(S) + COST(E) + COST(V)
S-VS	Selection push-up	COST(S) + COST(VS)
SS-E-V	Supported R-tree filter	COST(SS) + COST(E) + COST(V)
SS-VS	Supported R-tree filter + selection push-up	COST(SS) + COST(VS)
SS-E-U-V	Supported R-tree filter + differential treatment of containment and overlap	COST(SS) + COST(E) + COST(U) + COST(V)
ARM	Traditional rule mining over focal subset	COST(σ) + COST(ε_{AR})

Table 3.3: Summary of the Six Mining Plans.

3.3.5 Treating Contained and Overlapped MIPs Differently

In the plans described thus far, irrespective of whether a MIP is fully contained within $\mathcal{D}^Q$ or it only overlaps by as few as a single record, it is sent for the record-level threshold check. We now introduce a key property in Lemma 3.3.5 that opens a new opportunity for optimization.

*If the MIP bounding box D_I^P of an itemset I is completely **contained** within the focal subset $\mathcal{D}^Q$, the **local support** of I equals its **global support**. In other words, if ($D_I^P \subseteq \mathcal{D}^Q$), then ($Supp_I^Q = Supp_I^G$). Proof is trivial.*

By Lemma 3.3.5, once a fully contained Itemset I is included in the candidate list $\{\mathrm{I}_{SS}^Q\}$ by the SS operator, any record-level check of I's support is redundant. Therefore, we now propose a mining plan that treats the fully *contained* MIPs differently from the *partially overlapped* MIPs. The MIPs contained within $\mathcal{D}^Q$, denoted by $\{\mathrm{I}_{SS}^Q\}_c$, can be safely sent to VERIFY for rule generation and *minconf* verification. Only the MIPs that *partially overlap* with $\mathcal{D}^Q$, denoted by $\{\mathrm{I}_{SS}^Q\}_p$, are required to undergo the expensive record-level support check. Here, subscripts c and p stand for *contained* and *partially overlapped*, respectively. ELIMINATE outputs the candidate itemsets $\{\mathrm{I}_E^Q\}_p$ ($\subseteq \{\mathrm{I}_{SS}^Q\}_p$). Figure 3.7.(b) depicts the resultant SS-E-U-V plan. Lastly, we introduce an itemset-level UNION operator to merge the two lists of itemsets to form a single list of itemsets for rule generation in VERIFY.

UNION Operator. This operator, denoted by $\mathrm{U}[\{\mathrm{I}_{I1}^Q\}, \{\mathrm{I}_{I2}^Q\}, \ldots] \longrightarrow \{\mathrm{I}^Q\}_{p,c}$, takes

as input two or more lists of itemsets, (here, $\{\mathrm{I}_E^Q\}_p$ and $\{\mathrm{I}_{SS}^Q\}_c$) and produces a single list which is the union of the two inputs.

Execution Costs and Analysis. The SS-E-U-V plan now incorporates all the optimizations designed above. It is expected to be highly effective in processing a large spectrum of query requests. The SS operator's cost remains the same as SS-VS, but SS now splits the candidate itemsets into two mutually exclusive sets, namely, the *contained* MIPs $\{I_{SS}^Q\}_c$ and *partially overlapped* $\{I_{SS}^Q\}_p$ MIPs. The partially overlapped MIPs ($\{I_{SS}^Q\}_p$) must first pass through ELIMINATE to produce a reduced list $\{I_E^Q\}_p$ ($\subseteq \{I_{SS}^Q\}_p$). As the two lists $\{I_{SS}^Q\}_c$ and $\{I_E^Q\}_p$ merged by UNION are mutually exclusive (requiring no duplicate removal), UNION incurs a constant cost. VERIFY processes a total of ($|\{I_E^Q\}_p|$ + $|\{I_{SS}^Q\}_c|$) candidate itemsets for rule generation (Equation 3.5).

$$
\begin{aligned}
&COST(SS-E-U-V) = COST(SS)+COST(E)+COST(U)+\\
&COST(V), where,\\
&COST(U) = [U_{const}],\\
&COST(V) = [\sum_{i\epsilon(\{I_E^Q\}_p+\{I_{SS}^Q\}_c)} (C^i \times |\mathcal{D}^Q|)+\\
&\sum_{r\epsilon\{R\}} (Conf_r^Q+minconf)].
\end{aligned}
\tag{3.5}
$$

3.3.6 Employing the Traditional Mining Plan

For the extreme case when huge numbers of partially overlapped MIPs might require expensive record-level checks, a traditional two-step rule mining algorithm may be more practical than employing the sophisticated MIP-index-based mining plans. This traditional plan may employ a mining algorithm (Charm (ZH02)) directly over the extracted focal subset $\mathcal{D}^Q$. We refer to this baseline plan as the ARM plan (Figure 3.7.(c)) that is composed of the following two operators:

SELECT Operator. Denoted as $\sigma[\mathcal{A}^{range}, \mathcal{D}] \longrightarrow \mathcal{D}^Q$, SELECT takes as input the range attributes $\mathcal{A}^{range}$ and the dataset $\mathcal{D}$ to output the records of the focal subset $\mathcal{D}^Q$.

ARM Operator. This operator, denoted by $\varepsilon_{AR}[\mathcal{D}^Q, \mathcal{A}^{item}, \textit{minsupp}, \textit{minconf}\ ] \longrightarrow \{R^Q\}$, performs the traditional ***A**ssociation **R**ule **M**ining* (ARM) from scratch. It takes the focal subset $\mathcal{D}^Q$, the item attributes $\mathcal{A}^{item}$ and the thresholds *minsupp* and *minconf* as inputs. It outputs the desired ruleset $\{R^Q\}$.

Execution Costs and Analysis. SELECTION using an R-tree requires extracting the records from the overlapped bounding boxes. The costs for the rule mining step depend on the input data size, the length of the maximum frequent itemset and the number of dimensions. The costs of this baseline plan is computed by adding the SELECTION and ARM costs (Equation 3.6).

$$\begin{aligned} &COST(ARM-Plan) = COST(\sigma)+COST(\varepsilon_{AR}), where, \\ &COST(\sigma) = [(\sum_{j=1}^{h-1}\{N_j \times \prod_{k=1}^{n}(\mathcal{D}^P_{j,k_{avg}}+\mathcal{D}^Q_{k_{avg}})\}) \times TR_{const}], \\ &COST(\varepsilon_{AR}) = [|\mathcal{D}^Q| \times (\max_{I \epsilon \{I^Q\}}(C^I)) \times n]. \end{aligned} \tag{3.6}$$

3.4 Experimental Validation of the COLARM Framework

In this section we provide experimental results to explore the validity of our COLARM framework. We focus on three questions:

- Is the COLARM framework capable of providing correct decisions regarding the best choice among the set of six plans?
- Do the proposed optimizations in the five MIP-index based mining plans achieve the expected execution cost benefits for the tested datasets?
- What is the extent of Simpson's Paradox observed in the context of localized rule mining?

We conducted experiments on a Windows 7 machine with Intel(R) Xeon(R) CPU X3440@2.53 GHz processor and 8 GB of RAM. The mining plans and the COLARM optimizer are coded using C++.

Experimental Datasets. We use three real benchmark datasets from the UC Irvine Machine Learning Repository (UCI15), namely, chess, mushroom and PUMSB. Chess contains 3196 records with 76 distinct items. Mushroom contains 8124 records with 120 distinct items. PUMSB contains 49046 records with 7117 distinct items. Figure 3.8 depicts the number of closed frequent itemsets stored in the MIP-index structure for chess, mushroom and PUMSB datasets, respectively as we vary the primary threshold values[1]. For both Chess and PUMSB datasets the number of closed itemsets drastically increases with a decrease in the primary threshold. The change in Mushroom is rather gradual. A detailed analysis of the three chosen datasets including a discussion of the distribution of closed frequent itemsets (CFIs) by their length can be found in (ZH02). The length of an itemset I corresponds to the *number of singleton items in I*, also denoted by C^I (see Table 3.2). In other words, the length also denotes the number of attributes (dimensions) in the R-tree. Overall, chess and PUMSB have a symmetric distribution of CFIs whereas mushroom has a bi-modal distribution of closed frequent itemsets (CFIs). Thus, these three datasets represent diverse characteristics for testing the effectiveness of the COLARM framework in terms of local itemsets discovered and selecting mining plans with quick response times.

Preparing the MIP-index is a one-time offline step irrespective of the number of online requests processed thereafter. Thus, the index construction costs are of less importance. Infact, prior works (ZH02, LMR$^+$13) have studied, in particular, the impact of the number of prestored itemsets on the subsequent online execution. For the online local mining experiments, we focus on a different set of measurements as described below. For our

[1]The primary support ranges are as suggested in (ZH02).

tested scenarios of online mining queries, we use the MIP-index structures created using primary support 60% for chess, 5% for mushroom and 80% for PUMSB datasets, respectively. The corresponding MIP-index structure stores approximately 300000, 10000 and 450000 closed frequent itemsets for chess, mushroom and PUMSB, respectively.

Experimental Methodology. Our experiments test two metrics, namely, (a.) the query response time benefits of using the cost-based optimizer for plan selection, and (b.) the extent of Simpson's paradox in localized mining. The *range* and *item* attributes are part of the user request (see Q in Sec. 3.1.2). This query-time selection from a common pool of attributes means that existing solutions (DPDK11) are not applicable and thus can not be compared against. We study the following measures:

1. **Execution Time Metric**: We compare the average execution times of the six alternate mining plans for different query parameter settings i.e., by changing the *focal subset size*, *minsupp* and *minconf*, respectively. We compare plan selection by the COLARM optimizer based on the estimated costs with the actual execution costs of the plans.

2. **Optimization Benefits**: In addition to the above factors derived from the cost model, we also evaluate the effectiveness of our optimized MIP-index-based mining plans measured by their respective gains in reducing the execution costs compared with the basic S-E-V plan.

3. **Local Rules vs. Global Rules**: For the tested local mining sccenarios, we compare the counts of fresh local rules versus the global rules to determine the extent of Simpson's paradox observed.

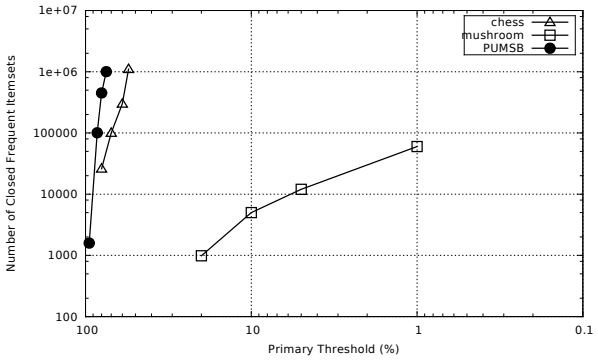
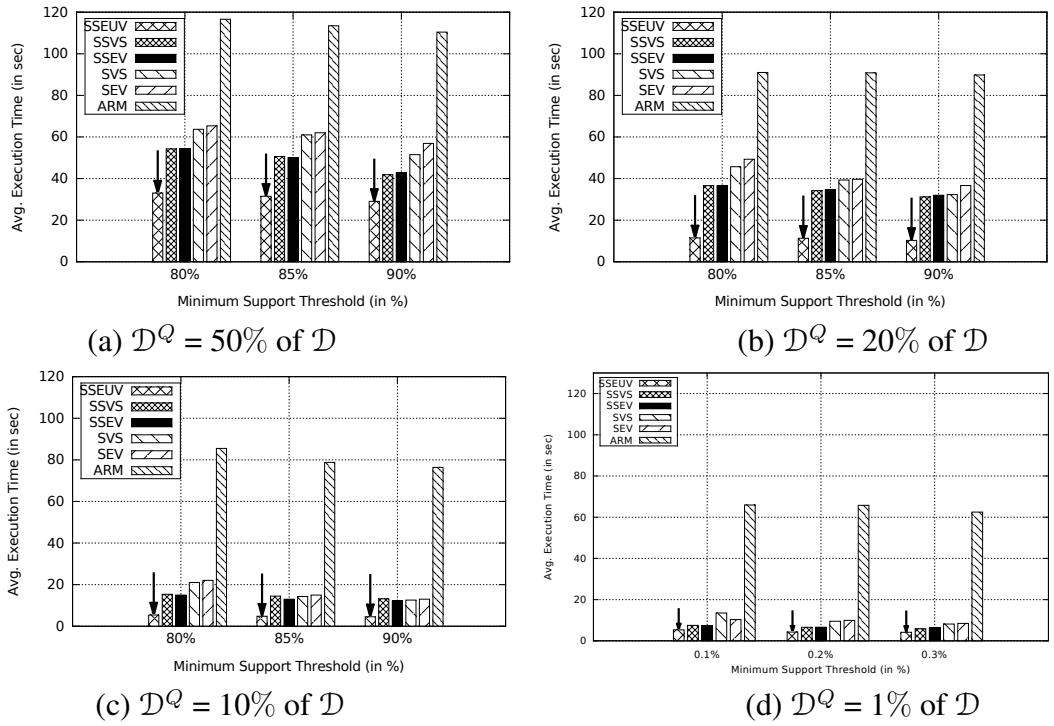


Figure 3.8: # Frequent Itemsets by Primary Thresholds.

3.4.1 Accuracy of the COLARM Optimizer

Here we present our experiments to validate the COLARM optimizer's capability to identify the most cost effective plan for a diversity of tested mining requests. We vary the mining parameters such as $\mathcal{D}^Q$ *size* (4 distinct values), *minsupp* (3 distinct values) and *minconf* (3 distinct values) to create a total of 36 distinct localized mining requests for each of the three datasets. For each tested mining request, the COLARM optimizer computes the estimated costs of the six plans using the cost formulae derived in Section 3.3 and suggests the plan with the minimum estimated cost. To demonstrate the effectiveness of our COLARM optimizer, we plot the average execution times of the six plans for the tested scenarios and indicate with an arrow the plan chosen by COLARM in majority of the cases.

In our experiments we vary the $\mathcal{D}^Q$ sizes as % of the complete dataset $\mathcal{D}$, namely, 50%, 20%, 10% and 1%. Similarly, different *minsupp* (as % of subset $\mathcal{D}^Q$) values are used, namely, $\{80, 85, 90\}$ for the chess dataset, $\{70, 75, 80\}$ for the mushroom dataset and $\{85, 88, 91\}$ for the PUMSB dataset, respectively. Further, we used high minconf values, namely, $\{85, 90, 95\}$, for all datasets. These chosen *minsupp* and *minconf* values are based on the primary support values chosen for creating the MIP-index and the analysis of the distribution of their closed frequent itemsets by their lengths provided in



(a) $\mathcal{D}^Q$ = 50% of $\mathcal{D}$ (b) $\mathcal{D}^Q$ = 20% of $\mathcal{D}$

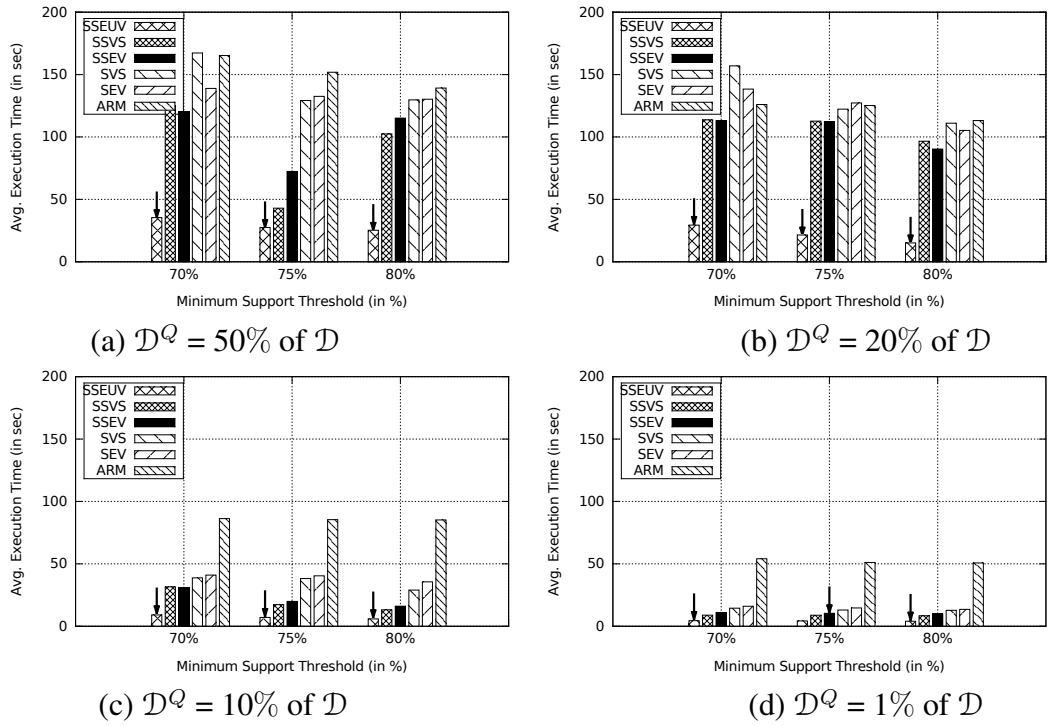
(c) $\mathcal{D}^Q$ = 10% of $\mathcal{D}$ (d) $\mathcal{D}^Q$ = 1% of $\mathcal{D}$

Figure 3.9: Avg. CPU Costs of Mining Plans for Chess Dataset.

(ZH02).

Impact of Varying Focal Subset Sizes. Figures 3.9, 3.10 and 3.11 depict the average execution times of the six mining plans for chess, mushroom and PUMSB, respectively. For each $|\mathcal{D}^Q|$ (say, 50% of $|\mathcal{D}|$), the execution time is averaged over several runs by submitting queries with fixed sized $\mathcal{D}^Q$ over different regions of the dataset. The plan chosen by the COLARM optimizer on majority of runs based on the estimated costs is marked with an arrow. We fix the *minconf* to 85%. For each dataset, $|\mathcal{D}^Q|$ equals (a) 50%, (b) 20%, (c) 10% and (d) 1% of $|\mathcal{D}|$ of the tuples. For all datasets (Figures 3.9, 3.10 and 3.11), with a decrease in the focal subset size from 50% to 1% (charts (a) to (d)), the execution costs of the plans decrease drastically. This conforms to predicted trends of the cost model. The smaller the focal subset $|\mathcal{D}^Q|$, the fewer the overlapping MIPs to be verified. This leads to a faster response times.

Impact of Varying Minimum Support. For the chess dataset (Figures 3.9 (a)-(d)),

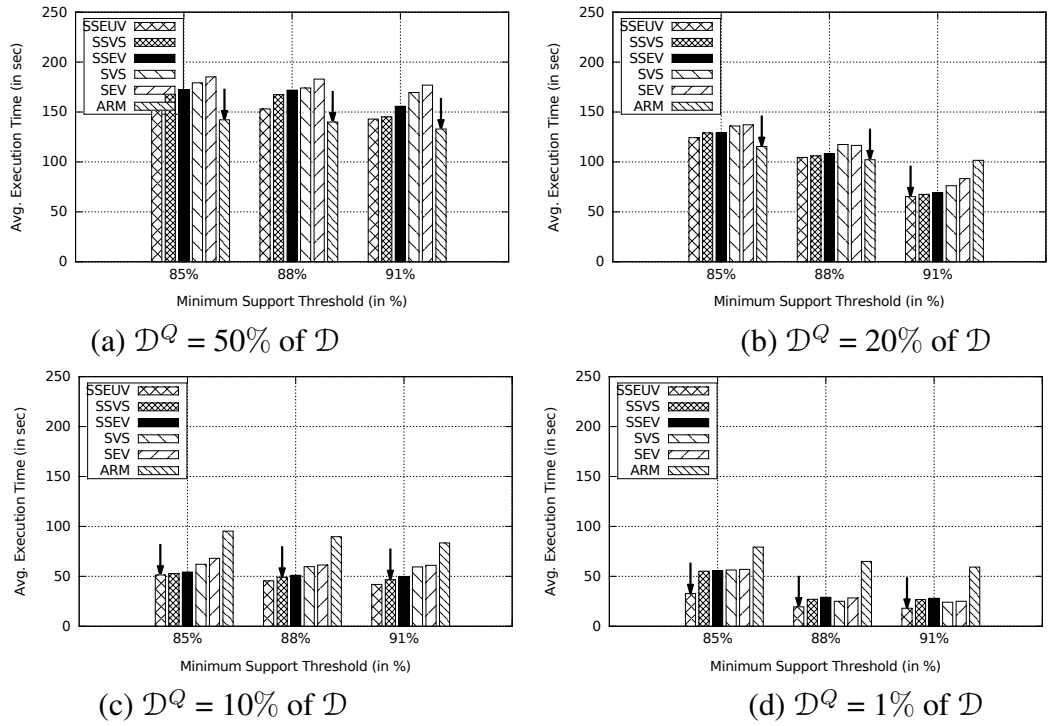


(a) $\mathcal{D}^Q$ = 50% of $\mathcal{D}$ (b) $\mathcal{D}^Q$ = 20% of $\mathcal{D}$

(c) $\mathcal{D}^Q$ = 10% of $\mathcal{D}$ (d) $\mathcal{D}^Q$ = 1% of $\mathcal{D}$

Figure 3.10: Avg. CPU costs of mining plans for mushroom dataset.

our proposed mining plans consistently outperform the traditional *ARM* plan. As expected, the SS-E-U-V plan is optimized to deliver the best execution time. This trend is attributed to the sparse population of itemsets in chess and a symmetric distribution of itemsets by length. In Figures 3.10 (a)-(d) a similar trend is observed for the mushroom dataset. We also observe that among the MIP-index based plans the average execution cost decreases from S-E-V to SS-E-U-V in most cases as each of the optimizations are employed. In Figures 3.11 (a)-(d) for the PUMSB dataset, the MIP-index-based mining plans continue to perform distinctly better at the lower $|\mathcal{D}^Q|$ sizes. However, for the higher $|\mathcal{D}^Q|$ (50% and 20%), we observed no clear winner. In some cases, the baseline ARM plan outperforms the other plans. This distinct trend in PUMSB dataset is attributed to due the symmetric distribution of itemsets by length and high density of the dataset. The *impact of varying* ***minconf*** is similar to, yet less drastic than, that of varying *minsupp*. Thus, those experimental results are omitted due to space constraints.



(a) $\mathcal{D}^Q$ = 50% of $\mathcal{D}$ (b) $\mathcal{D}^Q$ = 20% of $\mathcal{D}$

(c) $\mathcal{D}^Q$ = 10% of $\mathcal{D}$ (d) $\mathcal{D}^Q$ = 1% of $\mathcal{D}$

Figure 3.11: Avg. CPU Costs of Mining Plans for PUMSB Dataset.

Plan Selection Accuracy of COLARM Optimizer. In Figures 3.9, 3.10 and 3.11, we depict with an *arrow* the mining plans chosen by the COLARM optimizer using our cost model. As the results are averages over several runs, the arrow indicates the plan that was chosen in majority. Thus, we find that COLARM indeed almost always selects the most efficient plan. Out of the total 108 distinct tested request scenarios (3 datasets and 36 parameter settings), COLARM makes erroneous decisions for only three cases, namely, in Figure 3.10.(d) for minsupp = 75% and in Figure 3.11.(c) for minsupp 88% and 91%. Thus, COLARM is more than 93% accurate. When failing to select the optimal plan, COLARM selects a plan with at most 5% extra cost compared with the optimal.

3.4.2 Measuring the Optimization Benefits

Here, we zoom into the optimized operators of the MIP-index based mining plans. We use the *S-E-V plan* as the baseline and compare its execution costs against our optimized

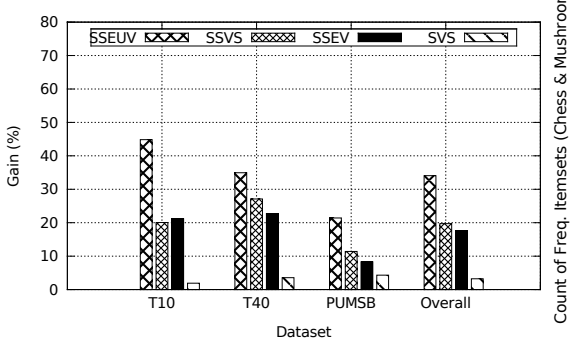


Figure 3.12: Optimization Gains.

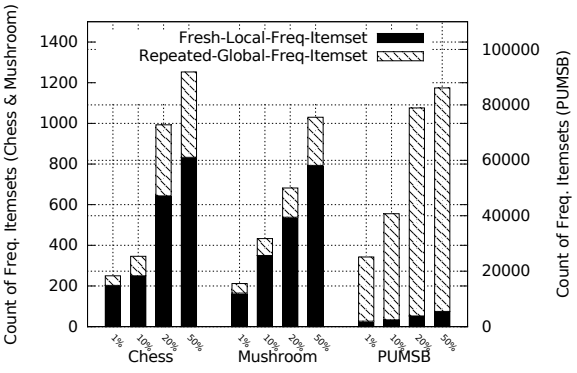


Figure 3.13: Average Local vs. Global CFIs.

plans, namely, *S-VS*, *SS-E-V*, *SS-VS* and *SS-E-U-V*. For a plan P, the gain is computed as $\frac{(CPU_{SEV} - CPU_P)}{CPU_{SEV}}$. Figure 3.12 depicts the % gain for each optimized plan. In Figure 3.12, the benefits of pushing selection up, i.e., the VS operator, are minor. On the other hand, the plans using the SS operator show 8% to 44% gains, indicating the success of the *supported R-tree filter*. Particularly, SS-E-U-V exhibits high gains ($\sim$22% to 44%).

3.4.3 Comparing Local vs. Global Rules

In Figure 3.13 we summarize the average number of closed frequent itemsets (CFIs) mined in our tested cases. Here it is important to understand that these local frequent itemsets were captured in the offline MIP-index construction due to the use of low primary support, namely, 60% for chess, 5% for mushroom and 80% for PUMSB datasets, respectively. These local itemsets qualify at high support ($\geq$80% for chess, $\geq$69% for mushroom, and $\geq$85% for PUMSB) only at the 1%-50% subset granularities. These local CFIs will be either missed when reasonable global minsupp ($>$ primary threshold) values are input, or will be hidden within a large number of itemsets if the user inputs a global minsupp less than the primary threshold.

Compared with the global itemsets mined at reasonable minsupp values 80% for chess and 60% for mushroom, we find that majority of the CFIs mined in the tested query

scenarios are local CFIs, providing strong evidence for Simpson's paradox. For example, for the mushroom dataset, when a subset of mushroom samples with attribute-value pair *stalk-shape=tapering* is chosen, we observe 32 local CFIs emerge that have a low global support of $\simeq$39%. Two example CFIs are {*stalk-surface-below-ring=smooth, veil-color=white*} and {*stalk-surface-above-ring=smooth, veil-type =partial, ring-number=one*}. These 32 CFIs are hidden in the global context as 625 other CFIs exist in the global context with higher support values. For this subset of mushroom samples satisfying attribute-value pair *stalk-shape=tapering*, these CFIs have local support values $\geq$69%. For PUMSB the itemsets mined with global minsupp = 85% are in majority also for the local queries, yet several additional prominent local CFIs are discovered as well.

Experimental Conclusions. Our main experimental findings are summarized as follows:

• The experimental evaluation establishes the *accuracy* of the cost model as the trends observed are coherent with the predictions of the cost model.

• In most of our tested cases, the *SS-E-U-V plan* consistently outperforms the other plans. In some cases for PUMSB, the ARM plan marginally outperforms the others. Overall, no single mining plan is a clear winner.

• The COLARM optimizer successfully identifies the most efficient plan with more than 93% accuracy for all our tested cases including extreme cases where the traditional ARM plan is most efficient.

• Among the MIP-index-based mining plans, the plans using the SS operator are significantly less expensive indicating the success of the supported R-tree filter.

• We observe strong evidence for Simpson's paradox in the context of local rule mining requests.

3.5 Related Work

Aggarwal et al. (APY02) identify the importance of localized rules. Yet, they propose a one-time clustering-based summary of the market basket dataset with a *fixed* minsupport value. Han et al. (NLHM99) propose a 2-phased processing framework for online *constraint-based mining*. However, both these works improve existing query-time rule mining algorithms rather than precomputing itemsets for online mining of localized rules. These improved algorithms are thus orthogonal to our efforts and could in fact be used in conjunction with our proposed strategies.

Two recent works on mining rules on multidimensional data (WTH06, DPDK11) relate to our work. However, a closer analysis reveals that they solve a different problem altogether. First, they assume that in a *transactional* data model (e.g., market basket dataset (APY02)), there is a set of *window* attributes, such as location and time of transaction, on which partitions are created. Thus, in their data model, the transaction data forming the itemsets are required to be distinct from the partitioning attributes. In contrast, our work focuses on a *relational* data model, such as in (SA96), where the subset attributes and the items used for forming rules are from a common pool of attribute-value pairs. The offline assumptions to aid the precomputation as done in (DPDK11) are unrealistic in our model. Thus, these existing approaches (WTH06, DPDK11) are inapplicable to the relational model.

3.6 Conclusion

We address the novel problem of *online localized rule mining*. We design a novel MIP-index that makes POQM feasible for our target problem. Using the efficient two-layered MIP-index, we design several alternate mining plans that employ sophisticated optimization strategies such as *selection pushup*, *supported R-tree filter* and *differential treatment*

of contained versus partially overlapped MIPs. Our **COLARM** optimizer selects the most efficient plan for processing each mining request. Our extensive experiments using benchmark datasets establish the effectiveness of our COLARM framework in a rich variety of tested cases. We also find strong evidence of Simpson's paradox in the context of localized rule mining.

4

Mobile Context Sequence Modeling with OLAPH

4.1 Background

Modern day mobile devices are capable of logging their sensor and app data such as app usage, location, call/SMS logs, battery usage, and calendar events. In the ubiquitous computing literature (KLJ+08, SH11, WLLB05) such mobile usage data is called mobile *context* data, as the data provides context about the user such as her location, application usage, online activity, call and SMS behavior, charging behavior, and battery usage. A large volume of research in the Ubicomp community has been devoted to using the raw location and physical sensor data to infer and log higher level user context such as jogging, driving, in meeting and at work (Nat12). A supervised approach is used with labeled data for each context, i.e., labeled data is collected while jogging, driving, in meeting and at work. Then a model is generated that infers each of these contexts in real-time. Contrary to these works, part III of this dissertation focuses on learning user's mobile context patterns from longitudinal logs of the collected mobile context data in an unsupervised

manner.

Mobile context patterns can be expressed in a number of different ways, such as frequently co-occurring context items (e.g., {Morning,AtHome,ReadNews,ListenToJazz}) (SMM+14) or a frequent sequence (e.g., on *weekend evenings*, user typically exhibits the pattern [{ShoppingMall}, {MovieTheatre}, {Dinner}]) (MSW14). The overarching vision of this research is to use longitudinal smartphone context to infer diverse frequent patterns that capture different aspects of the users context, and explore the utility of each type of pattern in improving overall user experience. A key aspect of this research is to leverage the computing potential of modern smartphones to perform the pattern mining *entirely on the device*. Therefore, below description assumes that all mobile context events and mined patterns are specific to a single user and within her own mobile device.

4.1.1 Definitions

Symbol	Description
e	Mobile context event is a 2-tuple (C,T)
C	Context data {<D:A:V>$_1$,<D:A:V>$_2$,...}
T	Timestamp, duration and derived time features {tf$_1$,tf$_2$,tf$_3$,...}.
D	Data type, e.g., AppUsage, Location, Call, SMS, etc.
A	Attribute of context, e.g., AppUsed, Location Provider, Location PlaceID.
V	Value of attribute, e.g., Facebook app, GPS, at work.
S	Sequence of mobile context events $[e_i, e_{i+1}, \ldots, e_n]$
θ_T	Maximum allowable time gap between two consecutive elements of a sequence (in minutes).

Table 4.1: Notation Used in OLAPH.

Mobile Context Event. Mobile usage data can be modeled as a time-series where each context event has an associated timestamp. In Table 4.1, we define notation used in this work. A mobile context event e is a 2-tuple (C,T), comprising of a context C and a time T. The context C is denoted as a list of data {<D:A:V>$_1$,...,<D:A:V>$_n$}, where

Context	Attribute	Values
AppUsage	AppUsed	Outlook | Facebook | Uber | GMaps | ....
Location	Provider	GPS | Cell | Wi-Fi
	PlaceID	Home | Work | Gym | Mall | ....
	MotionState	Stationary (0) | Moving (1)
Call	Contact	<set of contacts>
	Type	Incoming | Outgoing | Missed | Voicemail
SMS	Contact	<set of contacts>
	Type	Incoming | Outgoing
Time	DaySegment	Morning | Afternoon | Evening | Night
	DayOfWeek	Sunday,Monday,...,Saturday
	DayType	Weekday | Weekend

Table 4.2: Examples of Mobile Context Data.

D is the data type, A is the attribute name and V is the value of the attribute. Examples of context data are provided in Table 4.2. For each of data type there is a *primary* attribute and the rest are secondary. For example, AppUsage:AppUsed, Location:PlaceID, Call:Contact and SMS:Contact are primary attributes. Examples of *secondary* attributes are Location:Provider, Call:Type, and SMS:Type. In this work, we mainly focus on the *primary* attributes. However, in particular, for Call and SMS data types, the attribute *Type* is important as an *incoming* call semantically differs from an *outgoing* call or a *missed* call.

Time Features. The time T for a context event may be a single timestamp t (e.g., {<SMS:Contact:Bob>,<SMS:Type:Incoming>},t) or may have an interval (e.g., {<Call: Contact:Bob>,<Call:Type:Outgoing>}, {t_{start},t_{end}}). To generalize for all events, in this work we denote the time T of a context event as an interval. SMS can thus be a context event with t_{start}=t_{end}. In order to compute frequencies, exact timestamps are of little use. For example, a user may check her email on *Outlook* at 8AM on one day, and at 8:15AM on another day. In both instances, the frequent pattern to be captured is that *the user checks her email on Outlook every morning*. Therefore, to compute frequencies at the time granularity of segments of the day, we add the *DaySegment* time feature. A user

may be going for a nearby hike every Saturday or going to the Gym every Thursday. To capture day specific patterns, we add the *DayOfWeek* time feature. Further, a user may be using certain apps such as a video game only on the weekends, or receiving an incoming call from a work colleague on weekdays only. To capture frequency of mobile context events at such granularity, we add *DayType* time feature. Similarly, in an extension to our data model, monthly patterns can be captured using *Month* as the attribute. In other words, given the timestamp t of the context event, we generate all the time features.

Sequence$_{id}$	**Sequence**
S_{10}	$[e_1, e_2, e_3, e_4, e_6]$
S_{20}	$[e_3, e_5, e_{10}]$
S_{30}	$[e_4, e_1, e_2, e_3, e_2, e_3]$
S_{40}	$[e_5, e_6, e_{10}, e_{12}]$
...	...

Table 4.3: Example Sequence DB.

4.1.2 Sequence Prediction Problem Statement

Sequence of Mobile Context Events. A sequence, denoted by S, is an *ordered* list of mobile context events $[e_i, e_{i+1}, \ldots, e_n]$. The number of context events in a sequence is called the length of the sequence. A sequence with length *l* is called an *l*-sequence. A sequence database $\mathcal{S}_{\mathcal{DB}}$ is a set of tuples $<s_{id}, S>$, where s_{id} is a sequence identifier and S is a sequence. Table 4.3 depicts an example sequence DB with sequence identifiers and sequences. A tuple $<s_{id}, S>$ is said to contain a sequence α as a sub$-$sequence, if α is a subset of S, i.e., $\alpha \subseteq S$. The support of a sequence α in a sequence database $\mathcal{S}_{\mathcal{DB}}$ corresponds to the number of tuples in the database containing α, i.e., $\text{support}_{\mathcal{S}_{\mathcal{DB}}}(\alpha) = |< s_{id}, S > |(< s_{id}, S > \in \mathcal{S}_{\mathcal{DB}}) \wedge (\alpha \subseteq S)|$. For example, the sequence $\alpha = [e_1, e_2, e_3]$ is a 3-sequence. α is a subset of sequences with identifiers s_{10} and s_{30} in Table 4.3, and $\text{support}_{\mathcal{S}_{\mathcal{DB}}}(\alpha) = 2$.

Sequencing Parameters. Given a positive integer $\mathcal{I}$ as the **support threshold**, a sequence α is called a (frequent) sequence in the sequence database $\mathcal{S}_{\mathcal{DB}}$, if at least $\mathcal{I}$ tuples

in the database contain the sequence α, i.e., $\text{support}_{\mathbb{S}_{\mathcal{DB}}}(\alpha) \geq \mathcal{I}$. Further, a **maximum gap** parameter θ_T is defined for all valid sequences [e_i,e_j,. . ., e_k] (say with start timestamps t_i,t_j,. . .,t_k) to be included in sequence DB. The consecutive events must be separated in time by no more than the maximum gap θ_T. For example, if θ_T = 30 minutes and two consecutive events e_i and e_j differ in time by more than 30 minutes ($t_j - t_i > \theta_T$), the later event e_j will be part of a separate sequence. Sequence mining literature also defines a **maximum length** parameter, but we do not use it for simplicity.

Sequence Prediction. Given the sequence models generated from the training sequence DB, the sequence prediction is defined as follows. For a sequence $S = [e_{t_1}, e_{t_2}, \ldots, e_{t_n}]$ of n elements, the suffix of s of size y with $1 \leq y \leq n$ is defined as $P_y(s) = [e_{t_{n-y+1}}, e_{t_{n-y+2}}, \ldots, e_{t_n}]$. Predicting the next items of s is performed by identifying the sequences similar to $P_y(s)$, i.e., the sequences containing all context events in $P_y(s)$ in any order. The purpose of **sequence prediction** is to predict the next context event that best matches certain score defined in the prediction model.

4.1.3 Frequent Sequences vs. Frequent Itemsets

Given the individual mobile context events $[e_i, e_{i+1}, \ldots, e_n]$, in contrast to OLAPH, *MobileMiner* (SMM+14) learns co-occurrence patterns of events. A very small fraction of consecutive events e_i and e_{i+1} will be overlapping with each other. For example, using Facebook while listening to music. Therefore, *MobileMiner* has a *lenient* definition of co-occurrence, i.e., set of events that occur within 30 seconds to each other. In (RMD+16) Rawassizadeh et al. extend the definition of co-occurring events by exploring a variety of temporal granularities ranging from 5 seconds to 120 seconds to find what best fits different mobile context logs. On the generated sets of context events, frequent itemset mining or association rule mining is performed (AIS93, AS94a, AS96).

MobileMiner achieves 60-70% accuracy when predicting the next most likely event

for the mobile context data collected using a crowd-sourcing platform (WWBMT14) (same as OLAPH). To further improve prediction accuracy, *MobileMiner* explores using top 3,5 and 7 predictions. With 3-5 being acceptable number of predictions, and providing 80+% accuracy. Our hypothesis is that sequential patterns better capture the mobile usage patterns as often mobile contexts are used in an order rather than simultaneously. Further, frequent sequence modeling is known to be computationally and spatially more expensive than frequent itemset mining (FVLG$^+$16, PHMA$^+$04). Therefore, we conduct experimental evaluation to validate if sequence prediction modeling can achieve better than 60-70% accuracy of frequent co-occurrence models.

4.2 Approach

Below we first describe the overall architecture of ***O**n-device **L**ifelogging **A**nd **P**redicting **H**abitual Behavior using Sequence-based Context Predictor **(OLAPH)*** followed by a detailed description of each of the components.

4.2.1 OLAPH Architecture

Figure 4.1 depicts how the **(OLAPH)** engine runs on a smartphone with a running example using app usage logs. Mobile context logs are collected using a data collection service (API$^+$11, Sen) running on phone and stored in a mysql database. The choice of this database is not important and logs could alternatively be stored on files. The *Context Event Pre−processor* first extracts the events from the respective context logs and adds the time features that we discussed earlier in Section 4.1. The output of this step are the individual context events e_{t_i} as shown in step I of Figure 4.1. During the *Sequence DB Generator* step, the consecutive context events are combined to form the sequences to be stored in a sequence DB. The sequence DB can be built incrementally over time,

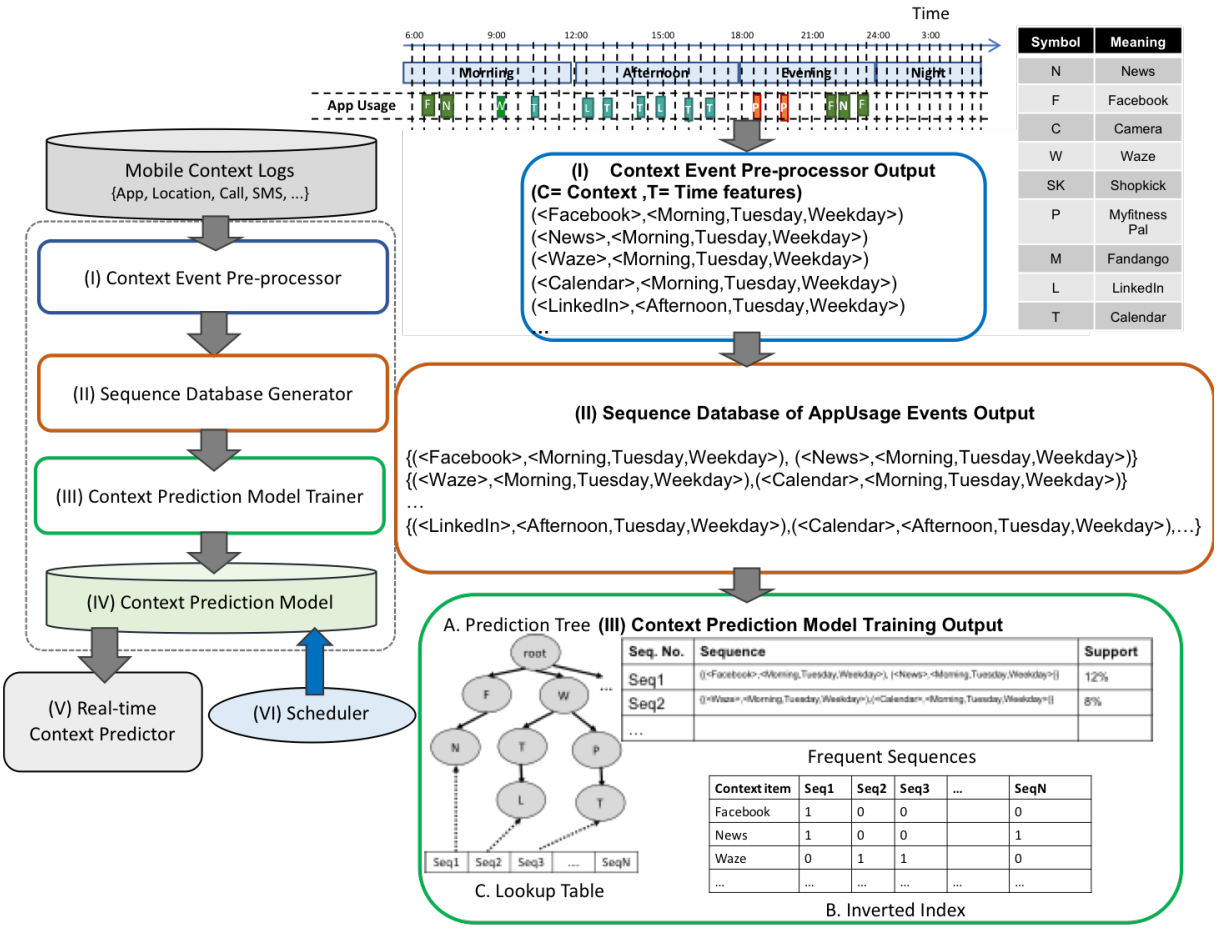


Figure 4.1: The OLAPH Architecture with a Running Example Using CPT (GFVT13).

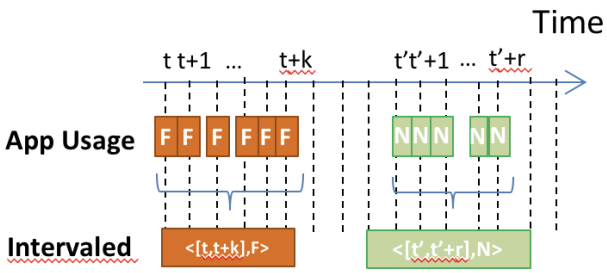


Figure 4.2: Generation of intervaled app usage events.

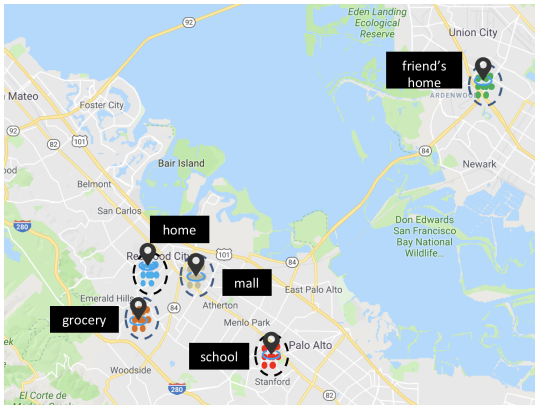


Figure 4.3: Sample clusters of frequently visited locations.

say, once a week. At each increment, once the sequence DB is generated (as shown in Figure 4.1), the *Sequence−based Context Model Trainer* (step III in Figure 4.1) builds a sequence model from the data in the sequence DB. All the above steps are scheduled as background tasks at idle times when the phone is on charging and not being used. This way, interference with user experience is avoided. The *Real-time Context Predictor* component performs context-aware predictions and recommendations using the model and current mobile context data. Various apps can then subscribe to the predictor, supplying the current context and receiving back the predicted next context.

4.2.2 Context Event Pre-processor

The purpose of the pre-processing step is to convert context data from the life-log mysql database format to a unified format of the mobile context events as described in Section 4.1, i.e., (C,T). Each probe type in the database exists in a different format. While location consists of latitude and longitude data together with other attributes, call consists of contact, call type and call duration (see Table 4.2). Each probe type thus requires slightly different pre-processing. The overall pre−processing step consists of two main tasks: (i.) generating *intervaled* context events and b. adding time features, as described

below.

4.2.2.1 Intervaled Event Generation

Below we describe how context events with time interval $[t_{start}, t_{end}]$ are generated from different logs.

App Usage. The app usage log consists of the name of the foreground app sampled at a certain rate (say, every 30 seconds) as depicted in Figure 4.2. Multiple consecutive readings over time are aggregated to achieve *intervaled* app usage events. For example, if an app is used for 10 minutes, at 30 seconds per entry, the app usage log will contain 20 consecutive entries. These are aggregated into a single *intervaled* context event with start and end times.

Location. The original location log consists of latitude and longitude values sampled from the phone GPS sensor at a certain rate (say, every 5 minutes) together with *Provider* and *MotionState*. As human's tend to stay at certain locations most of the time except moving from one place to the other, a low sampling rate is sufficient for location. The attribute *Provider* is the source of location. Phones mostly determine location using GPS but other less battery consuming options are known Wi-Fi SSIDs such as Home or Work Wi-Fi. Using the inertial sensors and GPS, the location data can also provide the *MotionState*, i.e., whether the device is stationary or moving. We also aggregate the location logs into *intervaled* context events. Consecutive latitude and longitude values may not be identical even when user is stationary. We aggregate all latitude and longitude values within a certain radius (say, 0.5 mile) to denote a specific *PlaceID* as depicted in Figure 4.3. Further, semantic tags can be attached to the aggregated *PlaceIDs* such as *home* (user spends nights and weekends) and *work* (user spends weekdays 10AM to 4PM) can be derived by using time-based logic. Therefore, the visited locations are represented as clusters $\{C_1, C_2, \ldots\}$. All (latitude, longitude) outside the 0.5 mile radius of the centroid

of the *PlaceIDs* are marked as *outside*. The *PlaceIDs* can also be generated incrementally as more user data is processed each week.

Call and SMS Logs. Call logs are the records from the calling app including outgoing, incoming, missed and voicemail calls. Calls are already intervaled whereas SMS logs are the SMSs recorded on the phone; they are sparse and infrequent for most users.

Overall, this time-intervaled aggregation of mobile context events achieves about 60-80% compression from the original life-logs. In particular, the compression is highest for the location log followed by the app usage log. The intervals of location logs depend on how long the individual user stays at each location, how often she moves and how many different places the user visits typically.

4.2.2.2 Time Features

The time feature of an intervaled context event has the format $[\mathrm{t}_{start}, \mathrm{t}_{end}]$. The intervaled event (Facebook,[10:00-10:10]) is similar to another intervaled event (Facebook,[10:05-10:15]), as both denote usage of Facebook app for 10 minutes in the morning. In order to do frequency counts over the intervaled events, time features such as *DaySegment*, *DayOfWeek* and *DayType* are added to the context events as defined in Section 4.1.

4.2.3 Sequence Database Generator

After the pre-processing step, all mobile context events e appear in the unified format (C,T) irrespective of their source. To generate sequence database $\mathcal{S}_{DB}$ over mobile context data, OLAPH uses two parameters, namely, *support threshold* $\mathcal{I}$ and *maximum allowable time gap* θ_T. All sequences in the sequence DB must satisfy a support exceeding the support threshold. The *support threshold* $\mathcal{I}$ is a domain or data driven criteria. For example, while a mobile user may typically use an app 15 times per day, they may typically only visit 3-4 places per day. Further some users may communicate with certain contacts

S = {(<Facebook>,<Morning,Tuesday,Weekday>),(<News>,<Morning,Tuesday,Weekday>)}

Meaningful Sequences ✓	Non-meaningful Sequences ✗
{(<D:A:V>,<tf1,tf2,…>),(<D:A:V>,<tf1,tf2,…>)} {(<Facebook>,<Morning>),(<News>,<Morning>)} {(<Facebook>,<Tuesday>),(<News>,<Tuesday>)} {(<Facebook>,<Weekday>),(<News>,<Weekday>)} {(<Facebook>,<Morning,Weekday>),(<News>,<Morning,Weekday>)} {(<Facebook>,<Morning,Tuesday>),(<News>,<Morning,Tuesday>)} {(<Facebook>,<Tuesday,Weekday>),(<News>,<Tuesday,Weekday>)} {(<Facebook>,<Morning,Tuesday,Weekday>),(<News>,<Morning,Tuesday,Weekday>)} {(<Facebook>,<Morning,Weekday,Tuesday>),(<News>,<Morning,Tuesday,Weekday>)} **{(<D:A:V>,<tf1,tf2,…>),(<D:A:V>,<tf2, tf1,…>)}*** {(<Facebook>,<Weekday,Tuesday>),(<News>,<Tuesday,Weekday>)} {(<Facebook>,<Morning,Weekday,Tuesday>),(<News>,<Tuesday,Weekday,Morning>)} **{(<D:A:V>),(<D:A:V>)}** {(<Facebook>),(<News>)} ***as long as the set {tf1,tf2,…} is common, the order does not matter**.	**{(D:A:V>,<tf1,tf2,…>),(<D:A:V>,<tf5,tf6,…>)}** {(<Facebook>,<Morning>),(<News>,<Tuesday>)} {(<Facebook>,<Tuesday>),(<News>,<Weekday>)} {(<Facebook>,<Weekday>),(<News>,<Morning>)} {(<Facebook>,<Morning,Weekday>),(<News>,<Morning,Tuesday>)} {(<Facebook>,<Morning,Tuesday>),(<News>,<Morning,Weekday>)} **{(<D:A:V>),(<D:A:V>,<tf1,tf2,…>)} or {(D:A:V>,<tf1,tf2,…>),(<D:A:V)}** {(<Facebook>),(<News>,<Morning,Tuesday,Weekday>)} {(<Facebook>,<Morning,Tuesday,Weekday>),(<News>)} {(<Facebook>),(<Morning,Tuesday,Weekday>)} {(<Facebook>),(<Morning>)} **{(<tf1,tf2,…>),(<D:A:V>)} or {(<D:A:V>),(<tf1,tf2,…>)}** {(<Morning>),(<News>)} {(<Facebook>),(<Morning,Tuesday,Weekday>)} **{(<tf1,tf2,…>),(<tf1,tf2,…>)}** {(<Morning,Weekday,Tuesday>),(<Morning,Weekday,Tuesday>)}

Figure 4.4: Meaningful and Non-meaningful Sequences.

multiple times a day via SMS while others may be messaging a maximum of 1-2 contacts per week. Sequential patterns on mobile context data may repeat daily or weekly. In OLAPH, the time features such as *DayOfWeek* and *DayType* capture different temporal granularities of the sequences. The highest granularity of time that OLAPH focuses on is a week; thus, a sequence α is said to be frequent if it occurs minimum of$\mathcal{J}$= 3 times in a week.

Maximum allowable time gap is the criteria used in OLAPH to assign events into different sequences. *Maximum sequence length* is an alternate criteria to be determined. For sequence DB generation, we use 10 as the value of maximum sequence length. We conducted a few empirical experiments to determine a reasonable maximum time gap θ_T as 30 minutes which is practical from mobile context data perspective. For our collected user logs, the maximum time gap assigns the context events to different sequences even if within the maximum sequence length threshold most of the time. Although more sophisticated behavior-oriented time segmentation proposed by Sarker et al. (SCKH18) may also be adapted.

4.2.4 Sequence Prediction Model Trainer

Next, we discuss how OLAPH trains the sequence prediction model over the sequences stored in the sequence DB $\mathcal{S}_{\mathcal{DB}}$. We first discuss how we introduce a *filter expression* to generate only meaningful sequence in the sequence DB and then describe the sequence prediction algorithms used for modeling.

Generating Meaningful Frequent Sequences. In OLAPH, for a sequence S = [e$_1$,. . .,e$_{i+1}$,. . ., e$_n$] to be *meaningful*, other than qualifying the parameters $\mathcal{I}$, *maximum time gap* and *maximum sequence length*, each event e$_i$ of S must contain the *primary* attribute of the context event and the same *time feature(s)* across all events in the sequence. In Figure 4.4 we present, in two columns, the lists of meaningful and non-meaningful *derived* sequences. As depicted in Figure 4.4, we are able to denote the meaningful sequences into generalized regular expressions. Therefore, we add a *filter expression* logic in the sequence modeling to generate only meaningful frequent sequences.

Sequence Prediction Algorithms. Several sequence prediction models have been proposed that capture sequential patterns in compact data structures and thus are relatively efficient for predictions. Most popular models are based on the Markov property, for example, Dependency Graph (DG) (PM96), All-K-Order-Markov (AKOM) (PP99), and Transition Directed Acyclic Graph (TDAG) (LS94). These works are inspired by the *Prediction by Partial Matching* (PPM) approach (CW84). PPM is an adaptive statistical data compression technique based on context modeling and prediction. PPM models use a set of previous symbols in the uncompressed symbol stream to predict the next symbol in the stream.

Below we describe with a running example, how a Markov-based sequence prediction model is trained. A sequence α = [e$_1$, e$_4$] denotes a consecutive occurrence of e$_1$ followed by e$_4$. in Table 4.4, the sequence DB contains the sequence α in two sequences, i.e., S$_2$ and S$_5$. Therefore, in Table 4.5, the transition count for $\{$e$_1\}$ (row) to e$_4$ (column) is

2. The **transition probability** of sequence $\{e_1, e_4\}$ is computed as the fraction of count for $\{e_1, e_4\}$ (=2) and the count of all transitions starting with e_1. Similarly, the rest of the transition probabilities are captured in 1^{st} order transition probability matrix. The sequence $\{e_1, e_4\}$ is followed by event e_5 in sequence S_5 of the sequence DB in Table 4.4. Therefore, the transition count of $\{e_1, e_4\}$ followed by e_5 is 1 in the 2^{nd} order transition probability matrix captured in Table 4.6. Similarly, for any order k, the transition probabilities are captured. So we demonstrate in this work how by increasing order k, the state complexity and the accuracy achieved change. More details about each of these approaches can be found in (FVLG$^+$16).

Sequence$_{id}$	**Sequence**
S_1	$[e_3, e_2, e_1]$
S_2	$[e_3, e_5, e_2, e_1, e_4]$
S_3	$[e_4, e_5, e_2, e_1, e_5, e_4]$
S_4	$[e_3, e_4, e_5, e_2, e_1]$
S_5	$[e_1, e_4, e_2, e_5, e_4]$

Table 4.4: Sequence DB.

1^{st} order	**e_1**	**e_2**	**e_3**	**e_4**	**e_5**
$\{e_1\}$	0	0	0	2	1
$\{e_2\}$	4	0	0	0	1
$\{e_3\}$	0	1	0	1	1
$\{e_4\}$	0	1	0	0	2
$\{e_5\}$	0	3	0	2	0

Table 4.5: 1^{st} Order Transition Probability Matrix.

2^{nd} Order	**e_1**	**e_2**	**e_3**	**e_4**	**e_5**	**2^{nd} Order**	**e_1**	**e_2**	**e_3**	**e_4**	**e_5**
$\{e_1, e_4\}$	0	1	0	0	0	$\{e_3, e_5\}$	0	1	0	0	0
$\{e_1, e_5\}$	1	0	0	1	0	$\{e_4, e_2\}$	0	0	0	0	1
$\{e_2, e_1\}$	0	0	0	1	1	$\{e_4, e_5\}$	0	2	0	0	0
$\{e_2, e_5\}$	0	0	0	1	0	$\{e_5, e_2\}$	3	0	0	0	0
$\{e_3, e_2\}$	1	0	0	0	0	$\{e_3, e_4\}$	0	0	0	0	1

Table 4.6: 2^{nd} Order Probability Matrix.

The various approaches (CW84, PM96, LS94, PP99, DK04) assume that the Markovian hypothesis holds true, i.e., each event solely depends on the previous events. Mark1 (CW84) is a Prediction by Partial Matching (PPM) approach that assumes that immediate previous event is sufficient to predict the current event. Mark1 predictions may be inaccurate over noisy data as it does not look far enough into the past. As a result, in order to obtain good predictions higher-order models must be used. Unfortunately, these higher-order models have a number of limitations associated with (i) high state-space complexity, (ii) reduced coverage, and (iii) sometimes even worse prediction accuracy. For example,

AKOM uses PPM from 1 to k order. AKOM is usually more accurate but the model can get exponentially large if k is not restricted. As illustrated in the above running example, increasing the order of Markov models induces higher state complexity, thus making sequence prediction models impractical for many real-life applications.

While most of the PPM based approaches have focused on reducing temporal and spatial complexity of these models, a recent Compact Prediction Tree (CPT) approach and its improved variant CPT+ store a compressed lossless representation of training sequences and measure the similarity of a sequence to the training sequences to perform a prediction. CPT is composed of three data structures: (1) a Prediction Tree (PT), (2) an Inverted Index (II) and (3) a Lookup Table (LT) (see Figure 4.1). The training is done using sequences from the sequence DB. Sequences are inserted one at a time in the PT and the II.

The *Prediction Tree* is recursively defined as a node. A node contains a context event e, a list of children nodes and a pointer to its parent node. A sequence is represented within the tree as a full branch or a partial branch; starting from a direct child of the root node. The second structure is the *Inverted Index*. It is designed to quickly find in which sequences a given item appears. Hence, it can also be used to find all the sequences containing a set of events. The II is defined as a hash table containing a key for each unique context event encountered during the training. Each key leads to a bitset that indicates IDs of the sequences where the context event appears. The third and last structure is the *Lookup Table*. It links the II to the PT. For each sequence ID, the LT points to the last node of the sequence in the PT. The LT purpose is to provide an efficient way to retrieve sequences from the PT using their sequence IDs

The main distinctive characteristics of CPT with respect to other prediction models are that (1) CPT stores a compressed representation of training sequences with no loss or a small loss and (2) CPT measures the similarity of a sequence to the training sequences

to perform a prediction (GFVT13, GFVRT15). In theory, CPT+ is up to 98 times smaller than CPT, performs predictions up to 4.5 times faster, and is up to 5% more accurate. These improvements are attributed to the compression strategies such as *frequent subsequence compression* (FSC) and *simple branches compression* (SBC). The *prediction with improved noise reduction* strategy further optimizes the prediction time. The different approaches optimize for the compressed structure to represent the model and on approximation of the transition scores. More details about each of these approaches can be found in (FVLG$^+$16).

In this work, our focus is to demonstrate how these existing sequence prediction models can be executed directly on mobile devices and how mobile context data can be adapted to fit these models. We further examine the trade-off between the accuracy achieved and the state complexity as well as computational complexity of model training as we vary order k of the sequence prediction modeling that practically works for mobile context data.

Real-time Predictions. Once the model is generated, in real-time the model is used to predict the current context. In order for the prediction to work, two pieces of data are prepared instantly, (1.) from the logs, the previous k (order of the model) contexts in a sequence are generated together with their time features, and (2.) using the current time, *current time features* are generated. We call the previous sequence the *prefix*. Using the *prefix* against the model, at most top-N predictions are retrieved $\{e_1^P, e_2^P, e_3^P, \ldots, e_N^P\}$. The *current time features* are then used to further filter the predictions to get the final prediction $\{e_f^P\}$. Ties are broken by selecting any one prediction at random. One way of presenting the prediction to the user is to have a widget on the home screen with the predicted values. Thus, as soon as the phone screen is turned on by user, either by pressing a key or in newer phones by face recognition, the widget containing predicted recommendations is presented to the user.

4.3 Experimental Evaluation

Here we present a thorough evaluation of OLAPH. We measure the performance and accuracy of OLAPH on Samsung S8 with snapdragon 835, 8 core and 4GB RAM. Below, we begin by describing the evaluation methodology.

4.3.1 Evaluation Methodology

We evaluate the usability and effectiveness of **OLAPH**. Two aspects that were examined are: 1. accuracy of sequence prediction models for a variety of mobile contexts, and 2. feasibility of on-device training of sequence prediction models. Below we list the highlights of our evaluation and then elaborate on each of them in detail thereafter.

1. We conducted our experiments on data collected from 85 subjects using a data collection app (API$^+$11, Sen). First, we present the data characteristics of the collected mobile context data including app usage, location, call and SMS logs.

2. We compared six state-of-the-art sequence prediction algorithms. We present interesting, and, in some cases, contrasting observations of applying sequence models over mobile context data, compared with prior use of these models in (GFVRT15). Further, as described in Section 4.2.4, order k of the Markov models is an important factor. Therefore, we study the trade-off between accuracy achieved and computational complexity of model training by varying order k of sequence prediction models for mobile context data.

3. We evaluate how many weeks of data is sufficient to train a prediction model with good accuracy and reasonable training times. Based on our analysis, we come up with a scheme for incrementally scheduling OLAPH training on phones.

4. Finally, each component of OLAPH is evaluated for its resource utilization including execution time, memory and CPU utilization as well as battery usage.

4.3.2 Context Data Collection

To evaluate OLAPH, we collected mobile context data via a commercial crowd-sourcing platform (WWBMT14). In total 85 users signed up for data collection; with a good gender and age mix; and over 100K hours of data was collected. The total cost of data collection was dramatically lower than for alternative methodologies, with total subject compensation under $3.5K US, and a total of less than 10 hours/week spent by researchers managing the study. While our entire data collection effort lasted more than 3 months, we observed that some users were more active in the data collection, while other users participated only for a few days before disabling or uninstalling the application. In Figure 4.5, we show the distribution of the number of full days of data (in increments of 14 days or 2 weeks) that we effectively collected from 71 of the users. The rest of the 14 users either discontinued participation or contributed insufficient data, so we eliminated them. Out of 71, 30 users collected more than 42 full days of data, while around 29 users collected 15-42 days of data. 12 users collected less than 15 days of data. More details of the data collection effort can be found in (WWBMT14).

For our evaluation, we include the following time-stamped context events: (1) apps used including time and duration, (2) call events including call type (incoming, outgoing, missed, voice mail), time, duration, and contact, (3) location PlaceIDs. The contact attribute for both Call and SMS logs are one-way hashed to anonymize and preserve privacy of data collection participants.

Figures 4.6 and 4.7 (a) and (b) depict the overall characteristics of the mobile context data by log types for 4 weeks of mobile context data[1]. The plots depict all 4 data logs,

[1]We establish that 4 weeks of data as sufficient to achieve good accuracy, yet have manageable resource

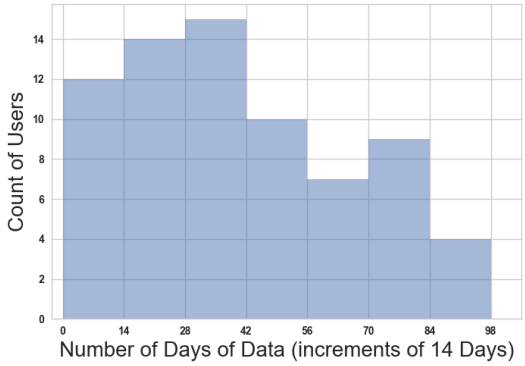


Figure 4.5: Number of Users with Days of Data Collected.

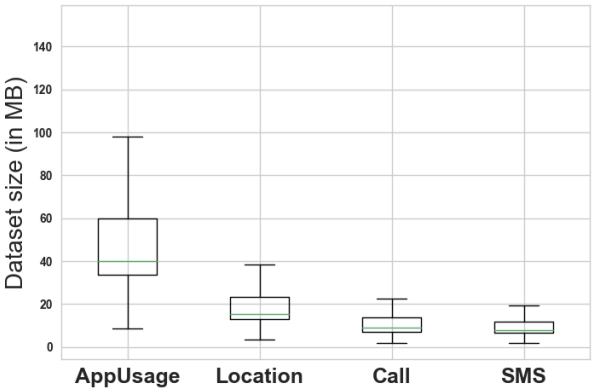
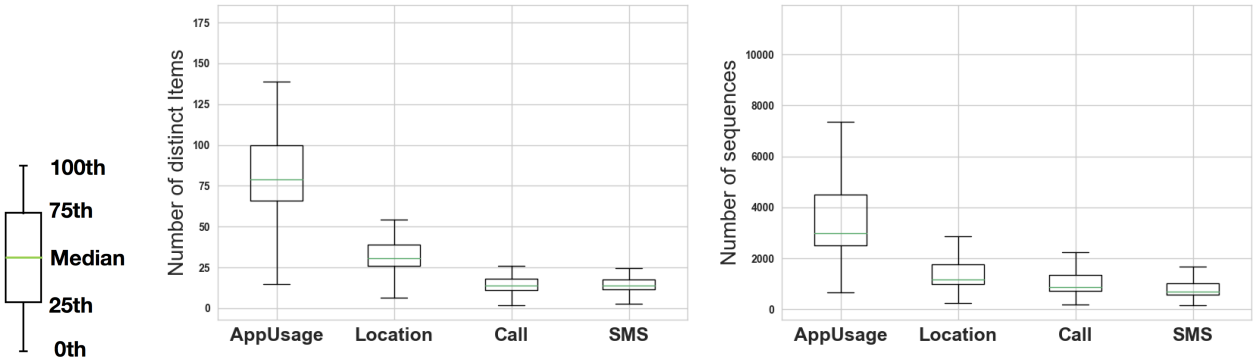


Figure 4.6: Distribution of Data Sizes Collected Across 71 Mobile Users.

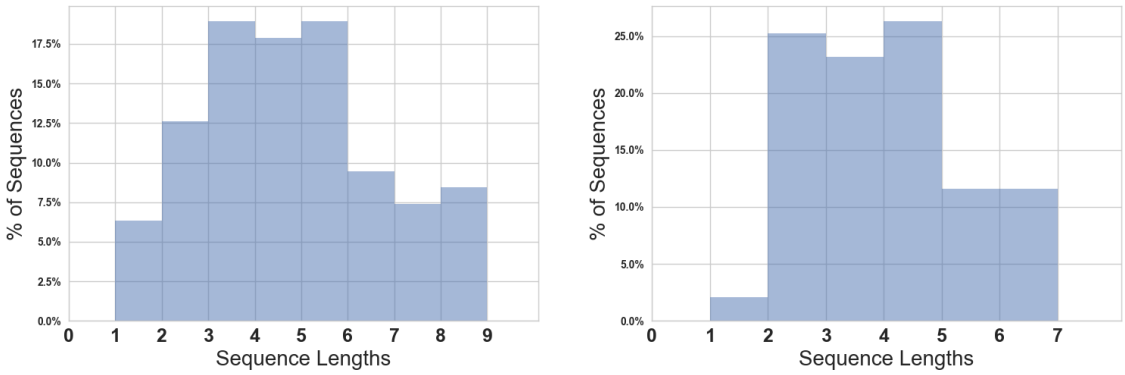


(a) Distinct Context Events Per Log Type. (b) Sequence DB Count.

Figure 4.7: Characteristics of Mobile Context Events and Generated Sequences.

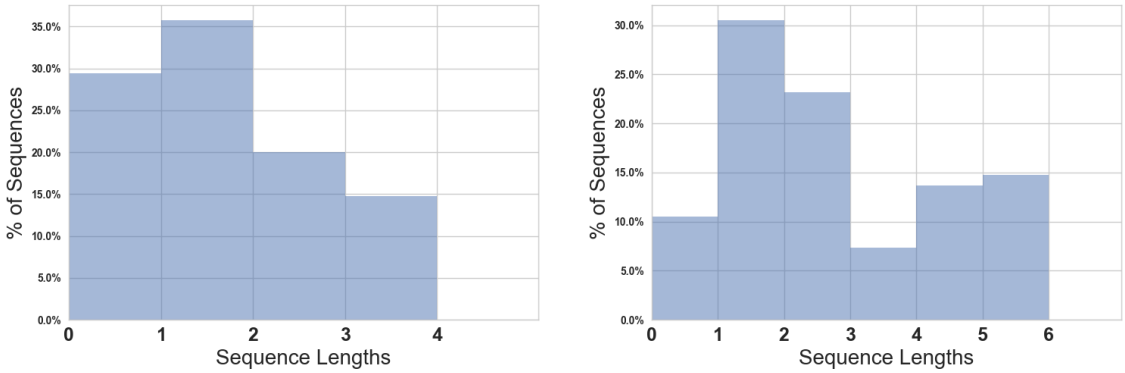
namely, app usage, location, call and SMS logs. Figure 4.6 illustrates the distribution of the data sizes per log across the users. Overall, app usage has the largest dataset size with median of 40 MB, followed by location with 18 MB whereas both call logs and SMS logs each use less than 8 MB (median) of space. After pre-processing, the median number of distinct apps installed on user devices is 78, with 75^{th} percentile of around 100, whereas median count of distinct locations visited is 27. Call and SMS logs have a median of 10-15 distinct contacts per 4 weeks (Figure 4.7 (a)). Figure 4.7 (b) depicts per log type

utilization, as shown in Section 4.3.3 III.



(a) App Usage. (b) Location.

Figure 4.8: Distribution of Sequence Lengths Per Log Type: App Usage and Location Logs.



(a) Call. (b) SMS.

Figure 4.9: Distribution of Sequence Lengths Per Log Type: Call and SMS Logs.

distribution of the counts per user of sequences in the sequence DB. While the sequence DB stores median of approximately 2800 app usage sequences over 4 weeks, location (median=1100), call (median=900) and SMS (median=420) logs generate moderate to low number of sequences.

In Figures 4.8 and 4.9, we further present the characteristics of the generated sequences, in particular, the distribution of sequence lengths for each log type using sequences from 4 weeks worth of data of 5 users. We mix the sequences into a single pool for this analysis. We find that the distribution of sequence lengths differ by log type. In

Figure 4.8 (a), 70% of the *app usage* sequences are composed of total of 3-6 app usage events. The maximum length is at times 9, but for only 8% of sequences. The *location* sequences range from 2 to 7 context events (Figure 4.8 (b)) However, majority ($\simeq$ 75%) of sequences are composed of 3 to 5 location visits. The *call* logs form shorter sequences (leq 4), with only 35% with lengths 3 or 4. About 30% are single calls and 35% are of length 2. In contrast, *SMS* logs are often series of messages ranging between 1 to 6 consecutive SMSs. However, majority of them (42%) are 2-3 consecutive SMS communications. Overall, the sequence lengths would also impact the choice of order k.

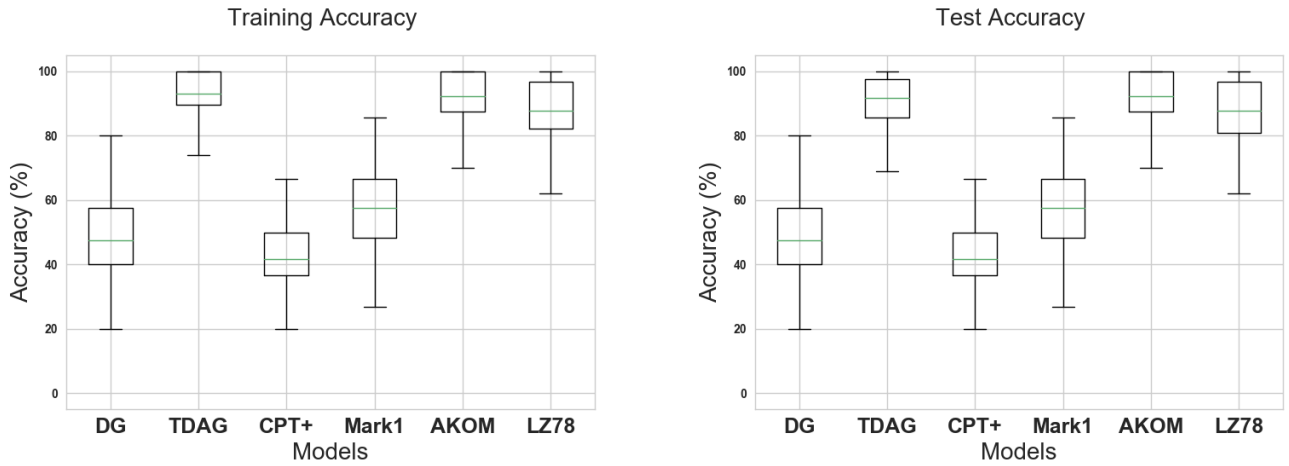
Empirical Tuning of Model Parameters. For all prediction models, we have conducted empirical evaluation to apriori set their parameters to optimal values. For the empirical experiments, we sampled a total of 10 datasets out of the 30 users with greater than 42 days of data. Mark1 and LZ78 do not have parameters. DG and AKOM have respectively a window of four and an order of five (details on the selection of order k for AKOM see Section 4.3.3 II.). Maximum depth for TDAG is restricted to 6, due to limited memory. CPT+ has six parameters; three for the FSC strategy, two for the PNR strategy and splitLength from CPT. The values of these parameters have also been empirically optimized for the 10 datasets and kept constant for the main experiments using full data.

4.3.3 System Performance

This section of the evaluation is based on the app usage log as it has the largest in storage requirements and computation overheads as shown above. We evaluated the following aspects.

I. Which sequence prediction model is best for mobile context data?

One key contribution of OLAPH is that this work adapts sequence prediction models to the paradigm of mobile context data. In other words, we prepare the mobile context data for generating a sequence prediction model. The six state-of-the-art sequence



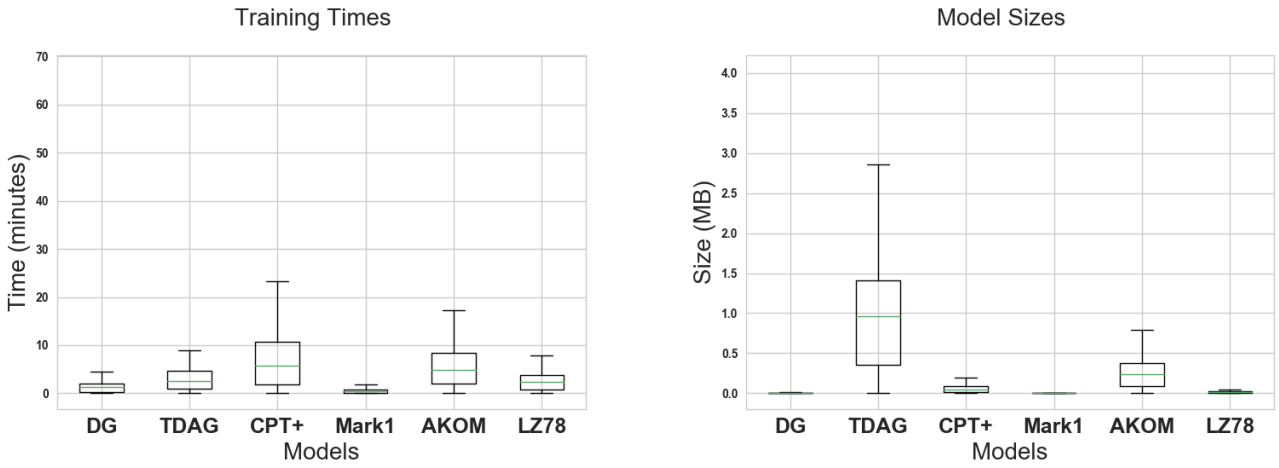
(a) Training Set. (b) Test Set.

Figure 4.10: Accuracy of Models with Order k=5.

prediction models are: (1.) Dependency Graph (DG), (2.) Transition Directed Acyclic Graph (TDAG), (3.) Compact Prediction Tree (CPT+), (4.) First-Order Markov (Mark1), (5.) All-k-Order Markov (AKOM), and (6.) Lz78. The SPMF open-source library (FVLG$^+$16) includes the java implementations of the above six algorithms, which we integrated into our Android app.

Figures 4.10 (a) and (b) depict the distribution of training and test accuracy of the 6 sequence prediction techniques for the 71 users over their consecutive 4 weeks (28 days) of app usage data. The consecutive 4 weeks are chosen starting at a random week. We use random sub-sampling to determine training accuracy. Then the following (5th) week is used as the test data. Clearly, for both training and test, TDAG and AKOM achieve 90+% median accuracy followed by LZ78. In contrast to the results for traditional benchmark data sets shown in (GFVRT15), CPT+ approach performs very poorly for mobile context data with less than 50% accuracy. First order markov (Mark1) achieves less than 60 % median accuracy. In particular, these results demonstrate that *for mobile context data, the first order is insufficient and models with higher order k are required.*

Figure 4.11 (a) illustrates the training times of the six sequence prediction models



(a) Training Time (b) Size (MB)

Figure 4.11: Training Execution Times and Model Sizes for the Six Sequence Models.

over those same 4 weeks of user data. This includes time for generating the model, given the sequence DB. Pre-processing and sequence DB generation times are the same across these techniques. Overall, all approaches have reasonable computation times of less than 10 minutes. The First order Markov (Mark1) outperforms the rest of the approaches, yet as shown in Figures 4.10 (a) and (b), the accuracy of Mark1 is poor. Both approaches with the highest accuracy, namely, TDAG and AKOM, have reasonable execution times ($\simeq$ 5 minutes); with overall TDAG being slightly faster than AKOM.

Figure 4.11 (b) illustrates the model sizes for the six approaches. TDAG has a high storage requirement, i.e., median 10 MB for model generated over 4 weeks of data. This makes it potentially unsuitable for execution on a mobile device. The rest of the models are stored compactly. All-k-order Markov (AKOM), the storage requirement is about 0.2 MB and the entire model can be reasonably loaded in-memory for context prediction. Using AKOM for modeling, approximately 40+ MB of raw data (Figure 4.6 (a)) can be compactly stored into 0.2 MB of sequence model. Therefore, *based on training and test accuracy as well as execution times and model sizes, we select AKOM as the model for our OLAPH system.*

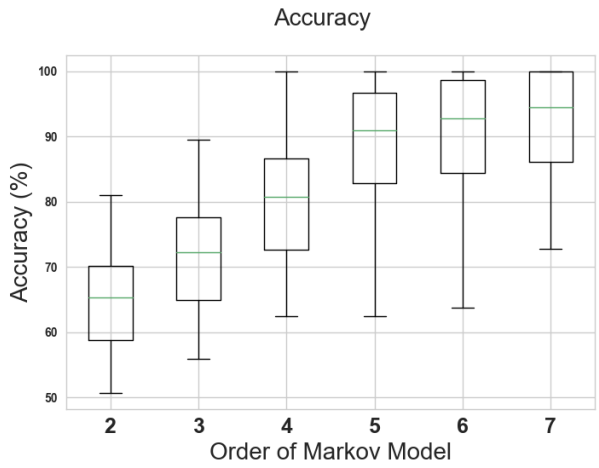
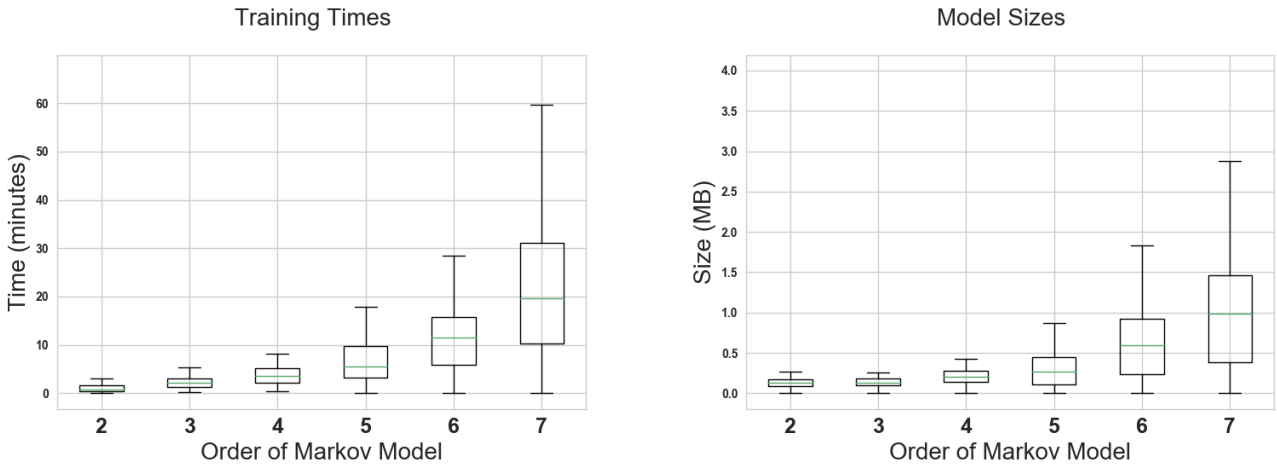


Figure 4.12: Impact of Varying Order k for AKOM: Accuracy.



(a) Training Time (b) Model Size

Figure 4.13: Impact of Varying Order k for AKOM: Training Time and Model Size.

II. What minimum order k of Markov model is sufficient to achieve reasonable sequence prediction accuracy over mobile context data?

Now that the All-k-Order Markov (AKOM) is the model of choice, we next explore the trade-off between the computational complexity of and the accuracy achieved by and varying order k of AKOM for the mobile context data. We thus compare the prediction accuracy of the AKOM with different k values. We demonstrate using a sample of 30 users' data chosen at random. Figure 4.12 depicts that as value of order k goes from 2 to 7, the median accuracy improves from 65% (k=2) to 95% (k=7). For k = {2,3}, the accu-

racy is poor (median = $<$75%). For all values of k, the 25^{th} percentile to 75^{th} percentile distribution is $>$10% indicating that trends vary depending on the user data. However, between k=4 and k=5 median accuracy increases from 81% to 91%, yet beyond k=5, the median accuracy increases slightly ($\simeq$ 3%) to 94% for k=7.

Figures 4.13 (a) and (b) depict the training time and the model sizes for the AKOM models generated by varying order k for the same 30 users as above. In Figure 4.13 (a), We observe that for low values of order (k = $\{2, 3, 4\}$), median values of training time are within 5 minutes. For k=5 the median training time is 6 minutes, and 75^{th} percentile is about 10 minutes. For k =6 and k=7, AKOM requires median of 11 and 20 minutes, respectively. In particular, for k=7, the trends reach high ($\simeq$ 30+ minutes of training times for 75^{th} percentile). Figure 4.13 (b) shows similar trends for model sizes with increase in median sizes (measured in MB) from 0.1 MB for k=2, up to 1 MB for k=7. For k=5, even the 75^{th} percentile value is within 0.5 MB.

Overall, we find that order values beyond k=5 do not provide significant gains compared to the increase in the training times and the model sizes. Therefore, we conclude that for our tested mobile context data *order k of sequence prediction models is important, and for mobile context data order k=5 achieves high accuracy while maintaining reasonably less state complexity and model size*.

III. How many weeks of data is sufficient to achieve reasonable accuracy?

To answer this question we used the data collected from another set of 30 users that have 6+ weeks ($>$42 days) of data. We use their app usage data as that has the highest number of sequences compared with the other 3 logs. Figure 4.14, illustrates the comparison between total training time required versus the training accuracy achieved with the number of weeks of data used for training the model. The training time is further divided into the 3 phases of *pre-processing*, *sequence DB generation* and *model training* (including the filter). For each user, we first divide up the data per week and select combinations of consec-

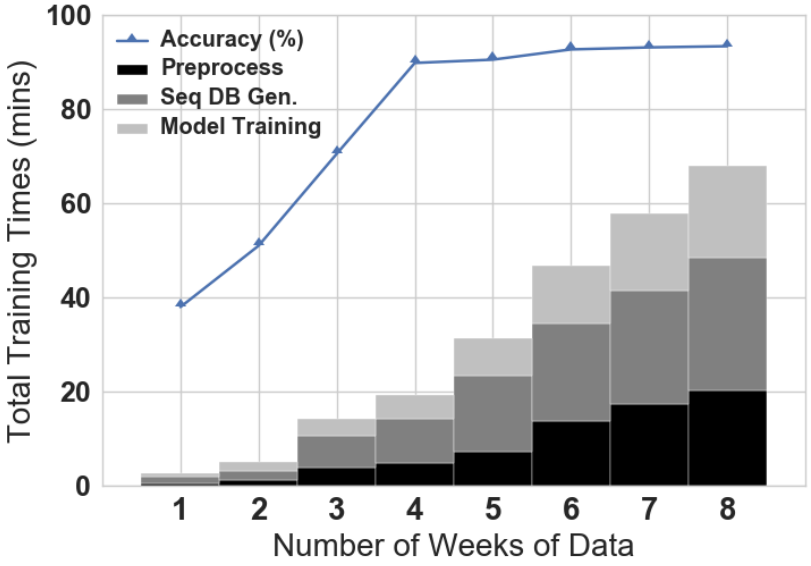


Figure 4.14: Accuracy vs. Training Times by Number of Weeks.

utive weeks of data, i.e., $\{(week_1), (week_1, week_2), \ldots, (week_1, week_2, \ldots, week_N)\}$. We use AKOM of order k=5, with random sub-sampling for cross-validation.

We observe that as the number of weeks of data grows, the accuracy improves significantly up to 4 weeks; beyond which accuracy gains are slow, maxing out at 94% at 8 weeks. Using 4 weeks of data, the accuracy achieved is 90+%. Overall, we observe that with addition of every week, the overall training time increases between 30% to 55%. At 4 weeks, the total execution time is $<$20 minutes; the time jumps to 33+ minutes when using 5 consecutive weeks of data or more. Therefore, beyond 4 weeks, the gain in accuracy is not significant given the 65+% additional training time. Thus, we conclude that *for the 30-user mobile context data, 4 weeks of data would be a good trade-off to achieve sufficient accuracy yet spend a reasonable time training the model.*

IV. How can OLAPH training be scheduled on the phone?

Modern smartphones have powerful octa-core processors (Sam) and are also typically unused for a majority of the time such as at night when the user is sleeping and the phone

is charging. OLAPH may be executed on the phone periodically during this idle time with little or no impact to the phone usage experience of the end user. We define the phone to be idle whenever it is charging, there are no foreground applications, and the battery level is at least 80%. Srinivasan et al. (SMM$^+$14) have shown that users have between 1-10 hours of idle time each week. Overall, as a sequence DB can be collected cumulatively, our proposal is to pre-process the sequence DB generation first and thereafter to incrementally update it with each new week's data. This would typically take a total of 2-3 minutes for 1 week's mobile context data. Sequence DB creation is followed by modeling over the past 4 weeks of data, which takes about 5 minutes. Therefore, in total, 7-8 minutes per week of computation is required for OLAPH. In each iteration, 3 weeks of previously generated sequences and 1 week of freshly created sequences in the sequence DB are used for training a new model each week. In future work we plan to explore incremental model training approaches. The sequences in the sequence DB and log data which are older are deleted periodically. This incremental processing could even be performed every day, however, it would be unnecessary to schedule this process daily.

V. Is it feasible to run OLAPH components on phone?

To address this question, we evaluated the execution time, memory, CPU utilization for the different components of OLAPH according to the scheduling scheme proposed above. Therefore, we evaluate metrics for pre-processing and generation of sequence DB over data processed per week. On the other hand, metrics for modeling and prediction are evaluated over 4 weeks of data. Table 4.7 illustrates the different performance metrics.

Pre-process and sequence DB generation steps combined take $<$4 minutes per week, and model training time is $<$9 minutes per week. The CPU utilization of each of the OLAPH components is less than 24%. Finally, in Table 4.7, we report the energy consumed by OLAPH as a percentage of the Galaxy S8s full battery charge (3,000mAh), assuming we perform the training once per week. For 3 sample users, we run only OLAPH

Performance Metrics	Pre-process	Seq. DB generation	Seq. Modeling: AKOM	Predictions
Execution time	$<$ 1 min	$<$ 2 mins	5 mins	1 sec
Memory	14 MB	23.2 MB	19.8 MB	0.7 MB
CPU utilization	18.1%	23.4%	20.5%	12.4%
% of full battery	0.14%	0.36%	0.18%	$<$ 0.01%

Table 4.7: On-device Performance of OLAPH.

on the phone and monitor the battery level of the phone before and after execution of each component. If we run OLAPH once per week for $<$ 10 minutes, as training is performed during the surplus time when the phone is plugged in and fully charged, the effective power consumption of OLAPH could be zero or negligible. The prediction is also light weight. Each prediction takes 1 second. If for prediction OLAPH loads the entire AKOM model, generated over 4 weeks of data, the memory requirement is approximately 0.2 MB and CPU utilization is also low.

4.4 Related Work

The relevant prior arts can be divided into two categories, namely, A. machine learning over mobile context data, and B. Sequence prediction modeling.

A. Machine Learning over Mobile Context Data. Context-aware computation on mobile devices has gained much interest recently (Nat12, SMM$^+$14, SKJ18, MSW14). Existing cloud-based services (e.g., Google Assistant) leverage instantaneous knowledge including current location, recent browsing history and calendar to present relevant weather information, directions, recommendations and appointment reminders. However, only a few recent works (Nat12, SMM$^+$14, SKJ18, MSW14) have explored using longitudinal mobile context data for mining typical user behaviors.

While Nath (Nat12) provides a cloud-based solution for mobile context rule mining

to sense the user's context in an energy-efficient manner, Srinivasan et al. (SMM+14, SKJ18) propose *MobileMiner* to explore on-device co-occurrence mining among context events for variety of use cases including app usage prediction. (SMM+14) evaluated *MobileMiner* on the same set of collected data (WWBMT14). *MobileMiner* achieves 60-70% accuracy for single prediction. To further improve prediction accuracy, *MobileMiner* explores using top 3,5 and 7 predictions. With 3-5 being acceptable number of predictions, and providing 80+% accuracy. Further, MSM (MSW14) identifies that co-occurrence patterns are inadequate as much of mobile context may not co-occur but follow a sequential pattern. (MSW14) also demonstrate that running on-device sequence mining algorithm such as PrefixSpan (PHMA+04) on mobile devices has high computational complexity, as demonstrated by (FVLG+16), and utilizing the mined results for prediction requires additional sequence prediction logic (RLM13, LRM13).

Rawassizadeh (RTWT13) propose a lightweight, configurable and extendable life-log framework, called UbiqLog, that establishes that life logging is feasible over mobile devices by efficient opportunistic sensing. Moreover, frequent human behavioral (FBP) patterns can then be mined over the collected temporal life-logs (RMD+16). However, FBP focuses on determining what the best temporal granularity to mine frequent itemsets may be. In contrast, our work follows the lead of the above works to explore compact and efficient sequence prediction approaches to identify sequential patterns in mobile context data for real-time context prediction.

B. Sequence Prediction Modeling. In contrast to sequence mining approaches such as PrefixSpan (PHMA+04) that store the sequential patterns in a compact structure, several sequence prediction models have been proposed that generate transition probabilities of sequences and store in compact data structures, ready for predictions. Most popular models are based on the Markov property, for example, Dependency Graph (DG) (PM96), All-K-Order-Markov (AKOM) (PP99), and Transition Directed Acyclic Graph

(TDAG) (LS94). These works are inspired by the *Prediction by Partial Matching* (PPM) approach (CW84). PPM is an adaptive statistical data compression technique based on context modeling and prediction. PPM models use a set of previous symbols in the uncompressed symbol stream to predict the next symbol in the stream. Besides, several compression algorithms have been adapted for sequence predictions (ZL78). While most of these prior works have focused on reducing temporal and spatial complexity of these models, few recent works, namely, CPT (GFVT13) and CPT+ (GFVRT15), attempted to increase their accuracy.

The approaches (CW84, PM96, LS94, PP99) assume that the Markovian hypothesis holds true, i.e., each event solely depends on the previous events. In contrast, AKOM (PP99) typically considers only the last k items of training sequences to perform a prediction, where k is the order of the model. One may think that a solution to this problem is to increase k to improve accuracy. However, increasing the order of Markov models induces higher state complexity, thus making sequence prediction models impractical for many real-life applications. Gueniche et al. (GFVRT15) compare these sequence prediction models over a variety of benchmark datasets to establish that these models can achieve a maximum accuracy in the range 60-74%. In this work, our focus is to demonstrate how these existing sequence models can be generated directly on mobile devices and how mobile context data can be adapted to fit these models; and thus be used for prediction of mobile context in real-time. We further identify the order k of the sequence prediction modeling that practically works for mobile context data. More details about each of these approaches can be found in (FVLG$^+$16).

4.5 Conclusion

In this work, we present OLAPH, an on-device context-aware service that learns phone usage behavioral patterns over the rich mobile context data and provides device intelligence via sequence-based predictions. We compared six sequence prediction models for performance and accuracy over data collected from 85 users. We discovered interesting, and, in some cases, contrasting observations of applying sequence models over mobile context data. The best model AKOM achieves a median of 90+% sequence prediction accuracy. As AKOM has higher state complexity compared to other first order markov approaches, we identify that AKOM with order k = 5 can achieve an overall high accuracy yet have reasonable state complexity and model size. We also demonstrate that models trained on 4 consecutive weeks of data provide good prediction accuracy with reasonable training times. Based on our analysis, we come up with a feasible scheduling scheme for OLAPH training on phones. OLAPH is generic and apps such as Google Assistant can subscribe to OLAPH to provide real-time context aware predictions to enhance mobile user experience.

5

Conclusions of This Dissertation

The theme of this dissertation is scalable and efficient data mining to improve user experience. This dissertation addresses two key challenges, namely, (i.) data mining lacks support for interactive real-time exploration, and (b.) mining algorithms are prohibitively compute-intensive to run on limited resources of a mobile device. Data analysts often need to perform numerous successive trial-and-error interactions and compare mining results with varying parameter values to find interesting rules. Further, analysts may need to interactively separate spurious rules from genuine ones. Due to high computation costs, even the fast implementations of mining algorithms are unsuitable for real-time interactive analysis.

Besides having high response times, the parameterized nature of these mining algorithms poses another challenge. Poor selection of parameters may lead to failure in discovering true patterns. Often algorithms may report spurious patterns that do not exist or overestimate the significance of the reported patterns. The task of distinguishing spurious patterns from real ones, as part of sense-making of the mined results, requires significant manual effort with little or no help from existing systems. The role of parameter selection is significant, yet appropriate parameter values are difficult to determine apriori, as

they tend to differ for different datasets, contexts and analysts' objectives. Therefore, an interactive data mining system, capable of not only answering mining queries but also providing parameter tuning recommendations at near real-time speed is important for decision making. Building on these insights, this dissertation aims to provide and end-to-end solution to support visual mining and analytics over large datasets.

Besides mining of *global* associations with the threshold measures of *minsupport* and *minconfidence* as user inputs, another requirement for interactive rule exploration is to allow the analyst to submit a *localized* mining query having two types of parameters, namely, a.) the data subset to mine rules from, which we call the *focal* subset, and b.) the items of interest to view rules on. Running the association rule mining algorithm repeatedly with varying thresholds and focal subsets can be prohibitively costly. In the pre-process-once-query-many paradigm, one-time high cost of pre-processing significantly reduces the turnaround times during online mining. Thus, in practice, POQM has been proven to be a worthwhile investment. In this dissertation, I focus on providing support for global and local association rule mining queries to enhance data mining systems.

Data mining algorithms, which provide useful insights into data sets of interest, have been traditionally considered prohibitively costly for mobile devices. Mobile sensing and context-aware computation on mobile devices has gained much interest recently. With powerful octa-core processors now available on modern smartphones, we demonstrate that such compute-intensive user pattern mining algorithms may be run on the phone itself to provide real-time, privacy preserving, context-aware predictions to enable personalized intelligent assistance.

As the first part of my dissertation, I have addressed the problem of enabling data mining at the speed of thought. The solution is achieved by breakthrough innovation ranging from the back end efficient management of mined results to the front end interactive rule exploration. In our PARAS back end framework for fast online association mining, we

propose a novel parameter space model for pre-storing rules such that a near real-time performance is guaranteed for online mining queries. In the context of the parameter space, we achieve surprisingly compact parameter space representations at pre-processing time, such that both space complexity of our proposed PSpace index and the online query processing costs are greatly reduced. In a variety of tested cases, PARAS outperforms the four competitor techniques, each by several orders of magnitude.

We then designed, implemented and evaluated an innovative visualization technology for interactive rule exploration called the **FIRE** framework. **FIRE** offers parameter recommendations and enhanced sense-making of rules and their relationships. Particularly, we propose two linked views, namely, PSpace and RSpace views. Both views are supplemented with innovative visualizations and interactions that enable analysts to effectively conduct visual rule exploration. While PSpace offers a rule distribution abstraction, RSpace facilitates detailed analysis of rules and their relations. In addition our novel RSpace glyph display enables visual comparison of rule shapes further augmented by glyph placement strategies. Our case study using the Bike sharing dataset (Bik15) illustrates the capabilities of the FIRE system and compares it with that of the state-of-the-art ARulesViz rule visualizations. Further, our user study with 22 subjects demonstrates the usability of the proposed FIRE framework using several benchmark datasets.

In the second part of my dissertation, I address the problem of *online localized rule mining*. We designed the MIP-index that makes POQM feasible for the targeted problem. Using the efficient two-layered MIP-index, we design several alternate mining plans that employ optimization strategies such as *selection pushup*, *supported R-tree filter* and *differential treatment of contained versus partially overlapped MIPs*. Our **COLARM** optimizer selects the most efficient plan for processing each mining request. Our extensive experiments using benchmark datasets establish the effectiveness of our COLARM framework in a rich variety of tested cases. We also find strong evidence of Simpson's

paradox in the context of localized rule mining. Overall, in this research work we extend the database technologies such as optimization of isolated operators and cost-based plan selection to apply them to online mining of localized rules.

Finally, we presented **OLAPH**, an on-device context-aware service that learns phone usage behavioral patterns over the rich mobile context data and provides device intelligence via sequence-based predictions. We compared 6 sequence prediction models for performance and accuracy over data collected from 85 users. We discovered interesting, and, in some cases, contrasting observations of applying sequence models over mobile context data. The best model AKOM achieves a median of 90+% sequence prediction accuracy. We also identify that sequence prediction model of order 5 can achieve an overall high accuracy yet have reasonable state complexity and model size. We also demonstrate that models trained on 4 consecutive weeks of data provide good prediction accuracy with reasonable training times. Based on our analysis, we come up with a feasible scheduling scheme for OLAPH training on phones.

In summary, this dissertation addresses key challenges of enhancing the performance and usability of data mining. Our proposed **PARAS** and **FIRE** systems provide solutions for interactive mining of *global* association rules. Further, our proposed **COLARM** system addresses the online mining of *localized* association rules. Finally, the **OLAPH** on-device service leverages mobile context data such as app usage, location, call and SMS logs to provide device intelligence via sequence-based predictions. Overall, this research encompasses significant contributions at the intersection of data mining, knowledge management and visual analytics to enhance the applicability of data mining over disparate data sources ranging from scientific benchmark datasets to mobile context data collected from several user devices.

6

Future Work

6.1 PSpace Paradigm and Online Mining Support

Here I discuss several extensions of the parameter space paradigm for exploring mined patterns in the context of association rules.

Extensions in Data Mining Technologies. PARAS/FIRE works have been the baseline for several subsequent works by PhD students in the DSRG lab at WPI. As an enhancement to PARAS, Qin et al. (QAL+14) extend PARAS to perform incremental instead of batch construction of the parameter space of rules. In (LMRW14), my fellow PhD student Xika extends the concept of parameter space-based exploration to negative rules. Further, Qin et al. (QAL+16) add a temporal dimension to the parameter space model to track changes across time. Further, the parameter space paradigm is applicable to other data mining approaches. Cao et al. (CWR14) support the interactive exploration of outliers in big data streams. Cao et al. (CWYR15, WCC+17) further propose the ONION framework that provides rich interactive support for efficiently analyzing outliers. ONION features an innovative exploration model that offers an outlier-centric panorama into big data streams.

Extending PSpace Paradigm Beyond Data Mining. In (MWB+13), we explore hypothesis and evidence in the parameter space context. The parameter space for evidence-hypothesis relationship consists of parameters such as belief and plausibility. We explore bayesian as well as DempsterShafer generalization of Bayesian theory. Therefore, the overall parameter space based paradigm proposed in my dissertation research and its extensions are well acknowledged in the research community. The

Mining of Localized Rules. Further, two additional problems related to localized association rule mining are interesting, namely, (a.) mining the subset range, support and confidence parameters from the data in an automatic and efficient way; and (b.) multi-query optimization in the context of localized association rule mining. While COLARM allows arbitrary subset of data, the key idea for automatic mining of parameters is to learn the most meaningful parameters such that mined results are of interest also for subsets of data. The idea on multi-query optimization for localized association rule mining is particularly important for shared data and infrastructure such that query results can be efficiently cached and multiple queries can be jointly resolved. The challenge will be twofold. Firstly, to apriori prepare the data efficiently not only based on just subsets but also interestingness measures such as support and confidence. Secondly, at query time, to understand the users' interest and improve the interactive experience.

6.2 On-device Mining and Prediction Modeling

As a prequel of our OLAPH service, my joint work at Samsung R&D titled MobileMiner (SMM+14) explored on-device mining of co-occurrence patterns over mobile context data. This work was well-received as a Best Paper Nominee at ACM UbiComp 2014. It was also adjudged the most innovative solution within Samsung and received the Gold medal at Samsung Best Paper Award Contest 2014. Mobile Sequence Miner

(MSW14) extends the notion of on-device mining to exploring the feasibility of running compute−intensive sequence mining (PrefixSpan) inside a mobile device.

Adding a User Study on Real-time Prediction Widget. Although adding a widget to home screen is an engineering task, learning how well the prediction is performing in real-time would be an interesting study. Further in this work we only explore generating a single prediction, however, as described in the *MobileMiner* (SMM$^+$14), the widget can display multiple (top-N) predictions. The user survey conducted in *MobileMiner* indicated 3-5 predictions are acceptable by users. The accuracy measure will then be different, less strict from the current evaluation of OLAPH as we will consider top-N predictions instead of just the most likely one.

Multi-modal Sequence Modeling. In OLAPH, I build on the insights gained from these prior works to explore sequence prediction models that are computationally less intensive than PrefixSpan, plus thus achieving a compact representation of the sequential patterns mined over the mobile context data. In this dissertation, I only explored sequence mining over single modality (e.g., AppUsage or Location or Call or SMS). In other words, I do not inter-mix events across different log types. As the next step, I will explore multi-modal sequence modeling, i.e., generating sequences across different data streams such as app usage, location, call and SMS logs. One key challenge I envision is that the frequency of different apps used on phone differs greatly from the locations. Further, for a typical user Call and SMS logs are quite infrequent. Therefore, the challenge is to solve for different granularity of support thresholds. I would further extend to additional data logs such as Screen times, battery usage and calendar. Apple launched a Screen time feature (Scr) in iOS 12 giving users insights on their mobile usage. Also battery usage (Bat) based studies are gaining popularity too. Learning of user's patterns to provide personalized recommendations on screen times and battery usage will further enable a mobile device to improve it's user experience and guide the users to utilize the mobile

device constructively.

Common Sequences Across Users. Yet another interesting study will be to analyze the patterns across users and build a common sequences knowledge base. The key challenge here would be that each user has unique set of mobile context data such as personalized set of contacts, or installed apps or visited locations. However, if certain additional knowledge about, say, relationships of locations (e.g., home, work, gym, nearby mall, and grocery store) or apps (e.g., social media, office/personal email, maps, and music) or contacts (e.g., spouse, parents, children, friends, colleagues) is built, then certain common patterns across users can be learnt. This will allow to bootstrap context aware services for a new user, until sufficient knowledge is collected for that new user.

Newer ML Technologies. I will further apply more advanced machine learning techniques such as Recurrent Neural Networks (RNN) (XLZ+18) and deep learning frameworks that run on mobile devices (LBM+17). Efficient machine learning frameworks such as tensorflow have been developed even for mobile devices (ten). Further, RNN (LSTM) and deep learning based models are well-suited for sequence modeling and known to be highly accurate given sufficient amount of training data (BVJS15, LBM+17). One limitation of applying deep learning based techniques for our collected data set would be that the data may not be large enough for deep learning to be effective. However, with multi-modal sequences the data would increase and may be well applicable.

References

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, 1993. 85, 117

[API+11] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6), 2011. 118, 128

[APY02] C. C. Aggarwal, C. Procopiuc, and P. S. Yu. Finding localized associations in market basket data. *IEEE Trans. on Knowl. and Data Eng.*, pages 51–62, 2002. 111

[ARu15] Arulesviz r package. `http://cran.r-project.org/web/packages/arulesViz/vignettes/arulesViz.pdf`, May 2015. 2, 8, 9, 13, 39, 58, 65, 66, 67, 68, 81

[AS94a] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994. 1, 2, 6, 12, 49, 69, 117

[AS94b] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th Interna-*

tional Conference on Very Large Data Bases (VLDB '94). Morgan Kaufmann, 1994. 4

[AS96] Rakesh Agrawal and John C. Shafer. Parallel mining of association rules. *IEEE Trans. Knowl. Data Eng.*, 8(6):962–969, 1996. 117

[AY01a] Charu C. Aggarwal and Philip S. Yu. A new approach to online generation of association rules. *IEEE Trans. Knowl. Data Eng.*, 13(4):527–540, 2001. 2, 9, 10, 12, 17, 21, 22, 50, 51, 56, 81, 89

[AY01b] Charu C. Aggarwal and Philip S. Yu. A new approach to online generation of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):527–540, 2001. 4

[Bat] Android battery historian. `https://developer.android.com/topic/performance/power/battery-historian`. 149

[Bik15] Bike sharing dataset. `https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset`, December 2015. 57, 58, 65, 68, 83, 145

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD*, pages 255–264, 1997. 2

[Bor15] Christian Borgelt. Efficient implementations of apriori, eclat and fp-growth. `http://www.borgelt.net`, December 2015. 1, 2, 49, 50

[BVJS15] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099, 2015. 150

[Cao13] Longbing Cao. Combined mining: Analyzing object and pattern relations for discovering and constructing complex yet actionable patterns. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 3(2):140–155, 2013. 2, 82

[CCMT90] Robert B. Cleveland, William S. Cleveland, Jean E. Mcrae, and Irma Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990. 59

[CHYN07] Olivier Couturier, Tarek Hamrouni, Sadok Ben Yahia, and Engelbert Mephu Nguifo. A scalable association rule visualization towards displaying large amounts of knowledge. In *International Conference Information Visualisation*, pages 657–663, 2007. 81

[CLN10] Surajit Chaudhuri, Hongrae Lee, and Vivek R. Narasayya. Variance aware optimization of parameterized queries. In *SIGMOD Conference*, pages 531–542, 2010. 80

[CLWY13] Longbing Cao, Jinjiu Li, Can Wang, and Philip S. Yu. Efficient selection of globally optimal rules on large imbalanced data based on rule coverage relationship analysis. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA.*, pages 216–224, 2013. 2, 9, 82

[CW84] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984. 124, 125, 141

[CWR14] Lei Cao, Qingyang Wang, and Elke A. Rundensteiner. Interactive outlier

exploration in big data streams. *Proc. VLDB Endow.*, 7(13):1621–1624, 2014. 147

[CWYR15] Lei Cao, Mingrui Wei, Di Yang, and Elke A. Rundensteiner. Online outlier exploration over large datasets. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 89–98, 2015. 147

[DK04] Mukund Deshpande and George Karypis. Selective markov models for predicting web page accesses. *ACM Trans. Internet Techn.*, 4(2):163–184, 2004. 125

[DPDK11] Mahashweta Das, Deepak P, Prasad Deshpande, and Ramakrishnan Kannan. Fast rule mining over multi-dimensional windows. In *SDM*, pages 582–593, 2011. 10, 88, 89, 104, 111

[DTB09] Songyun Duan, Vamsidhar Thummala, and Shivnath Babu. Tuning database configuration parameters with ituned. *PVLDB*, 2(1):1246–1257, 2009. 80

[FIR15] Paras/fire home page. `http://paras.cs.wpi.edu/`, December 2015. 36, 58

[FVLG+16] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. The spmf open-source data mining library version 2. In *Machine Learning and Knowledge Discovery in Databases*, pages 36–40, 2016. 10, 118, 125, 127, 133, 140, 141

[GFVRT15] Ted Gueniche, Philippe Fournier-Viger, Rajeev Raman, and Vincent S. Tseng. Cpt+: Decreasing the time/space complexity of the compact pre-

diction tree. In Tru Cao, Ee-Peng Lim, Zhi-Hua Zhou, Tu-Bao Ho, David Cheung, and Hiroshi Motoda, editors, *Advances in Knowledge Discovery and Data Mining*, pages 625–636, 2015. 11, 127, 128, 133, 141

[GFVT13] Ted Gueniche, Philippe Fournier-Viger, and Vincent S. Tseng. Compact prediction tree: A lossless model for accurate sequence prediction. In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar Zaiane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications*, pages 177–188, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. xxiii, 119, 127, 141

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, 1994. 7, 9

[HF95] Jiawei Han and Yongjian Fu. Discovery of multiple-level association rules from large databases. VLDB, pages 420–431, 1995. 3, 85, 91

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009. 8, 9, 39, 69, 81

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000. 1, 2, 49, 51

[JB02] Baptiste Jeudy and Jean-François Boulicaut. Using condensed representations for interactive association rule mining. In *PKDD*, pages 225–236, 2002. 2, 8, 9, 81

[KA08] Mehmet Kaya and Reda Alhajj. Online mining of fuzzy multidimensional weighted association rules. *Applied Intelligence*, 29:13–34, 2008. 81

[KF93] Ibrahim Kamel and Christos Faloutsos. On packing r-trees. In *CIKM*, pages 490–499. ACM, 1993. 90

[KHR+03] Miroslav Kubat, Aladdin Hafez, Vijay V. Raghavan, Jayakrishna R. Lekkala, and Wei Kian Chen. Itemset trees for targeted association querying. *IEEE Trans. Knowl. Data Eng.*, 15(6):1522–1534, 2003. 81

[KLJ+08] Seungwoo Kang, Jinwon Lee, Hyukjae Jang, Hyonik Lee, Youngki Lee, Souneil Park, Taiwoo Park, and Junehwa Song. Seemon: Scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 267–280, 2008. 113

[LBM+17] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3):82–88, 2017. 4, 6, 150

[LMR+13] Xika Lin, Abhishek Mukherji, Elke A. Rundensteiner, Carolina Ruiz, and Matthew O. Ward. PARAS: A parameter space framework for online association mining. In *PVLDB*, volume 6 (3), pages 193–204, January 2013. 3, 9, 11, 12, 17, 27, 37, 49, 81, 103

[LMRW14] Xika Lin, Abhishek Mukherji, Elke A. Rundensteiner, and Matthew O. Ward. SPIRE: supporting parameter-driven interactive rule mining and exploration. *PVLDB*, 7(13):1653–1656, 2014. 147

[LOPS04] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Fabrizio Silvestri. Webdocs: a real-life huge transac. dataset. In *FIMI*, 2004. 1, 12, 49

[LRM13] Benjamin Letham, Cynthia Rudin, and David Madigan. Sequential event prediction. *Machine Learning*, 93(2-3):357–380, 2013. 140

[LS94] Philip Laird and Ronald Saul. Discrete sequence prediction and its applications. *Machine Learning*, 15(1):43–68, Apr 1994. 124, 125, 141

[LSZ+12] Guimei Liu, Andre Suchitra, Haojun Zhang, Mengling Feng, See-Kiong Ng, and Limsoon Wong. Assocexplorer: an association rule visualization system for exploratory data analysis. In *KDD Demo.*, pages 1536–1539, 2012. 80

[MLB+13] Abhishek Mukherji, Xika Lin, Christopher R. Botaish, Jason Whitehouse, Elke A. Rundensteiner, Matthew O. Ward, and Carolina Ruiz. Paras: interactive parameter space exploration for association rule mining. In *SIGMOD*, pages 1017–1020, June 2013. 3, 12, 81

[MLT+18] Abhishek Mukherji, Xika Lin, Ermal Toto, Christopher R. Botaish, Jason Whitehouse, Elke A. Rundensteiner, and Matthew O. Ward. Fire: a two-level interactive visualization for deep exploration of association rules. *International Journal of Data Science and Analytics*, Jun 2018. 3, 12

[MLW+13] Abhishek Mukherji, Xika Lin, Jason Whitehouse, Christopher R. Botaish, Elke A. Rundensteiner, and Matthew O. Ward. Fire: interactive visual support for parameter space-driven rule mining. In *CIKM*, pages 2447–2452, 2013. 12, 13

[MRW14] Abhishek Mukherji, Elke A. Rundensteiner, and Matthew O. Ward. COLARM: cost-based optimization for localized association rule mining. In *Proceedings of the 17th International Conference on Extending Database*

Technology, EDBT 2014, Athens, Greece, March 24-28, 2014., pages 181–192, 2014. xx, 3

[MSW14] Abhishek Mukherji, Vijay Srinivasan, and Evan Welbourne. Adding intelligence to your mobile device via on-device sequential pattern mining. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 1005–1014, 2014. 6, 114, 139, 140, 149

[MWB+13] Abhishek Mukherji, Jason Whitehouse, Christopher R. Botaish, Elke A. Rundensteiner, and Matthew O. Ward. Sphinx: Rich insights into evidence-hypotheses relationships via parameter space-based exploration. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2521–2524, 2013. 148

[Nat12] Suman Nath. Ace: Exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*. ACM, 2012. 5, 113, 139

[NDD99] Biswadeep Nag, Prasad M. Deshpande, and David J. DeWitt. Using a knowledge cache for interactive discovery of association rules. In *KDD*, pages 244–253, 1999. 51

[NLHM99] Raymond Ng, Laks V. S. Lakshmanan, Jiawei Han, and Teresa Mah. Exploratory mining via constrained frequent set queries. *SIGMOD Rec.*, 1999. 111

[PHMA+04] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining sequential

patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004. 4, 118, 140

[PM96] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev.*, 26(3):22–36, July 1996. 124, 125, 140, 141

[PP99] James Pitkow and Peter Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *Proceedings of the 2Nd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 2*, USITS'99, pages 13–13, Berkeley, CA, USA, 1999. USENIX Association. 124, 125, 140, 141

[QAL+14] Xiao Qin, Ramoza Ahsan, Xika Lin, Elke A. Rundensteiner, and Matthew O. Ward. iparas: Incremental construction of parameter space for online association mining. In *Proceedings of the 3rd International Conference on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications - Volume 36*, BIGMINE'14, pages 149–165, 2014. 147

[QAL+16] Xiao Qin, Ramoza Ahsan, Xika Lin, Elke A. Rundensteiner, and Matthew O. Ward. Interactive temporal association analytics. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.*, pages 197–208, 2016. 147

[QKW+17] Xiao Qin, Tabassum Kakar, Susmitha Wunnava, Elke A. Rundensteiner, and Lei Cao. MARAS: signaling multi-drug adverse reactions. In *Pro-*

ceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017, pages 1615–1623, 2017. 2, 7, 8, 9

[RLM13] Cynthia Rudin, Benjamin Letham, and David Madigan. Learning theory analysis for association rules and sequential event prediction. *Journal of Machine Learning Research*, 14(1):3441–3492, 2013. 140

[RMD+16] Reza Rawassizadeh, Elaheh Momeni, Chelsea Dobbins, Joobin Gharibshah, and Michael Pazzani. Scalable daily human behavioral pattern mining from multivariate temporal data. *IEEE Trans. on Knowl. and Data Eng.*, 28(11):3098–3112, 2016. 6, 117, 140

[RPA+12] Lenin Ravindranath, Jitendra Padhye, Sharad Agarwal, Ratul Mahajan, Ian Obermiller, and Shahin Shayandeh. Appinsight: Mobile app performance monitoring in the wild. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI'12, pages 107–120, 2012. 4

[RTWT13] Reza Rawassizadeh, Martin Tomitsch, Katarzyna Wac, and A. Min Tjoa. Ubiqlog: A generic mobile phone-based life-log framework. *Personal Ubiquitous Comput.*, 17(4):621–637, 2013. 140

[SA95] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *VLDB*, pages 407–419, 1995. 3, 91

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. *SIGMOD Rec.*, 25(2):1–12, 1996. 3, 85, 91, 111

[Sam] Samsung galaxy s9. https://www.samsung.com/us/smartphones/galaxy-s9/. 137

[SCKH18] Iqbal H Sarker, Alan Colman, Muhammad Ashad Kabir, and Jun Han. Individualized time-series segmentation for mining mobile phone user behavior. *The Computer Journal*, 61(3):349–368, 2018. 123

[Scr] ios 12 screen time. https://www.macworld.com/article/3305557/ios/how-to-use-screen-time-in-ios.html. 149

[Sen] Sensor manager library. http://github.com/nlathia/SensorManager. 118, 128

[SH11] Maximilian Schirmer and Hagen Höpfner. Senst*: Approaches for reducing the energy consumption of smartphone-based context recognition. In Michael Beigl, Henning Christiansen, Thomas R. Roth-Berghofer, Anders Kofod-Petersen, Kenny R. Coventry, and Hedda R. Schmidtke, editors, *Modeling and Using Context*, pages 250–263, 2011. 113

[Sim51] E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society (Series B)*, 13:238–241, 1951. 3

[SKJ18] Vijay Srinivasan, Christian Koehler, and Hongxia Jin. Ruleselector: Selecting conditional action rules from user behavior patterns. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(1):35:1–35:34, 2018. 6, 139, 140

[SMM+14] Vijay Srinivasan, Saeed Moghaddam, Abhishek Mukherji, Kiran Rachuri, Chenren Xu, and Emmanuel Munguia Tapia. Mobileminer: Mining your

frequent patterns on your phone. In *Proceedings of the 2014 ACM Conference on Ubiquitous Computing (UbiComp '14)*, 2014. 6, 114, 117, 138, 139, 140, 148, 149

[SYLC15] Jingyu Shao, Junfu Yin, Wei Liu, and Longbing Cao. Actionable combined high utility itemset mining. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 4206–4207, 2015. 2, 8, 82

[ten] Tensorflow mobile. `https://www.tensorflow.org/mobile/`. 150

[TSS00] Yannis Theodoridis, Emmanuel Stefanakis, and Timos K. Sellis. Efficient cost models for spatial queries using r-trees. *IEEE Trans. Knowl. Data Eng.*, 12(1):19–32, 2000. 93, 95

[UCI15] UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, December 2015. 8, 14, 15, 38, 39, 43, 49, 70, 82, 103

[War02] Matthew O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3-4):194–210, 2002. 43, 47, 48, 83

[WCC+17] Mingrui Wei, Lei Cao, Chris Cormier, Hui Zheng, and Elke A. Rundensteiner. Interactive analytics system for exploring outliers. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 2547–2550, 2017. 147

[WCH07] Tianyi Wu, Yuguo Chen, and Jiawei Han. Association mining in large

databases: A re-examination of its measures. In *PKDD*, pages 621–628, 2007. 3, 10, 18, 82

[WCWL12] Po-Yuen Wong, Tak-Ming Chan, Man-Hon Wong, and Kwong-Sak Leung. Predicting approximate protein-dna binding cores using association rule mining. In *IEEE ICDE*, pages 965–976, 2012. 1

[WLLB05] Evan Welbourne, Jonathan Lester, Anthony LaMarca, and Gaetano Borriello. Mobile context inference using low-cost sensors. In *Location- and Context-Awareness*, pages 254–263, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 113

[WTH06] Ching-Yao Wang, Shian-Shyong Tseng, and Tzung-Pei Hong. Flexible online association rule mining based on multidimensional pattern relations. *Inf. Sci.*, pages 1752–1780, 2006. 111

[WWBMT14] Evan Welbourne, Pang Wu, Xuan Bao, and Emmanuel Munguia-Tapia. Crowdsourced mobile data collection: Lessons learned from a new study methodology. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, HotMobile '14, pages 2:1–2:6, 2014. 118, 129, 140

[XLZ+18] Shijian Xu, Wenzhong Li, Xiao Zhang, Songcheng Gao, Tong Zhan, Yongzhu Zhao, Wei-wei Zhu, and Tianzi Sun. Predicting smartphone app usage with recurrent neural networks. In Sriram Chellappan, Wei Cheng, and Wei Li, editors, *Wireless Algorithms, Systems, and Applications*, pages 532–544, 2018. 5, 150

[XMD15] Xmdvtool home page. `http://davis.wpi.edu/~xmdv/`, December 2015. 39

[YRW09] Di Yang, Elke A. Rundensteiner, and Matthew O. Ward. A shared execution strategy for multiple pattern mining requests over streaming data. *Proc. VLDB Endow.*, 2(1):874–885, August 2009. 80

[ZH02] Mohammed Javeed Zaki and Ching-Jiu Hsiao. Charm: An efficient algorithm for closed itemset mining. In *SIAM SDM*, 2002. xx, 3, 8, 88, 89, 90, 94, 95, 101, 103, 106

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978. 141

[ZPOL97] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In *SIG KDD*, pages 283–286, 1997. 1, 2, 49