HAL open science

# Constraint-Based Mining of Closed Patterns in Noisy n-ary Relations

Loïc Cerf

## To cite this version:

Loïc Cerf. Constraint-Based Mining of Closed Patterns in Noisy n-ary Relations. Other [cs.OH]. INSA de Lyon, 2010. English. NNT : . tel-00508534v2

**HAL Id: tel-00508534**

**https://theses.hal.science/tel-00508534v2**

Submitted on 21 Mar 2011

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

No d'ordre : 2010-ISAL-0050 Année 2010

**THÈSE**

*présentée devant*

L'Institut National des Sciences Appliquées de Lyon

*pour obtenir*

Le Grade de Docteur

*Spécialité*

Informatique

École Doctorale : Informatique et Mathématiques

*par*

Loïc Cerf

# CONSTRAINT-BASED MINING OF CLOSED PATTERNS IN NOISY N-ARY RELATIONS

Soutenue publiquement le 9 juillet 2010 devant le jury :

| | | |
|---|---|---|
| Jérémy Besson | University of Vilnius, LT | Examinateur |
| Francesco Bonchi | Yahoo! Research Barcelona, ES | Examinateur |
| Jean-François Boulicaut | INSA de Lyon | Directeur |
| Toon Calders | Eindhoven Technical University, NL | Examinateur |
| Bruno Crémilleux | Université de Caen | Rapporteur |
| Arno Siebes | University of Utrecht, NL | Examinateur |
| Hannu Toivonen | University of Helsinki, FI | Rapporteur |
| Christel Vrain | Université d'Orléans | Présidente |

Loïc Cerf: *Constraint-Based Mining of Closed Patterns in Noisy n-ary Relations,* PhD Thesis, © September 2007–July 2010

SUPERVISOR:
Jean-François Boulicaut

TIME FRAME:
September 2007–July 2010

# ABSTRACT

Useful knowledge discovery processes can be based on patterns extracted from large datasets. Designing efficient data mining algorithms to compute collections of relevant patterns is an active research domain. Many datasets record whether some properties hold for some objects, e. g., whether an item is bought by a customer or whether a gene is over-expressed in a biological sample. Such datasets are binary relations and can be represented as 0/1 matrices. In such matrices, a closed itemset is a maximal rectangle of '1's modulo arbitrary permutations of the lines (objects) and the columns (properties). Thus, every closed itemset supports the discovery of a maximal subset of objects sharing the same maximal subset of properties. Efficiently extracting every closed itemset satisfying user-defined relevancy constraints has been extensively studied. Despite its success across many application domains, this framework often turns out to be too narrow. First of all, many datasets are $n$-ary relations, i. e., 0/1 tensors. Reducing their analysis to two dimensions is ignoring potentially interesting additional dimensions, e. g., where a customer buys an item (localized analysis) or when a gene expression is measured (kinetic analysis). The presence of noise in most real-life datasets is a second issue, which leads to the fragmentation of the patterns to discover.

Generalizing the definition of a closed itemset to make it suit relations of higher arity and tolerate some noise is straightforward (maximal hyper-rectangle with an upper bound of '0's tolerated per hyper-plan). On the contrary, generalizing their extraction is very hard. Indeed, classical algorithms exploit a mathematical property (the Galois connection) of the closed itemsets that none of the two generalizations preserve. That is why our extractor browses the candidate pattern space in an original way that does not favor any dimension. This search can be guided by a very broad class of relevancy constraints the patterns must satisfy. In particular, this thesis studies constraints specifically designed for mining almost-persistent cliques in dynamic graphs. Our extractor is orders of magnitude faster than known competitors focusing on exact patterns in ternary relations or on noise-tolerant patterns in binary relations. Despite these results, such an exhaustive approach often cannot, in a reasonable time, tolerate as much noise as the dataset contains. In this case, complementing the extraction with a hierarchical agglomeration of the (insufficiently noise-tolerant) patterns increases the quality of the returned collection of patterns.

# RÉSUMÉ

Les processus de découverte de connaissances nouvelles peuvent être fondés sur des motifs locaux extraits de grands jeux de données. Concevoir des algorithmes de fouille de données efficaces pour calculer des collections de motifs pertinents est un domaine actif de recherche. Beaucoup de jeux de données enregistrent si des objets présentent ou non certaines propriétés; par exemple si un produit est acheté par un client ou si un gène est sur-exprimé dans un échantillon biologique. Ces jeux de données sont des relations binaires et peuvent être représentés par des matrices 0/1. Dans de telles matrices, un ensemble fermé est un rectangle maximal de '1's modulo des permutations arbitraires des lignes (objets) et des colonnes (propriétés). Ainsi, chaque ensemble fermé sous-tend la découverte d'un sous-ensemble maximal d'objets partageant le même sous-ensemble maximal de propriétés. L'extraction efficace de tous les ensembles fermés, satisfaisant des contraintes de pertinences définies par l'utilisateur, a été étudiée en profondeur. Malgré son succès dans de nombreux domaines applicatifs, ce cadre de travail se révèle souvent trop étroit. Tout d'abord, beaucoup de jeux de données sont des relations $n$-aires, c'est à dire des tenseurs 0/1. Réduire leur analyse à deux dimensions revient à ignorer des dimensions additionnelles potentiellement intéressantes; par exemple où un client achète un produit (analyse spatiale) ou quand l'expression d'un gène est mesurée (analyse cinétique). La présence de bruit dans la plupart des jeux de données réelles est un second problème qui conduit à la fragmentation des motifs à découvrir.

On généralise facilement la définition d'un ensemble fermé pour la rendre applicable à des relations de plus grande arité et tolérante au bruit (hyper-rectangle maximal avec une borne supérieure de '0's tolérés par hyper-plan). Au contraire, généraliser leur extraction est très difficile. En effet, les algorithmes classiques exploitent une propriété mathématique (la connexion de Galois) des ensembles fermés qu'aucune des deux généralisations ne préserve. C'est pourquoi notre extracteur parcourt l'espace des motifs candidats d'une façon originale qui ne favorise aucune dimension. Cette recherche peut être guidée par une très grande classe de contraintes de pertinence que les motifs doivent satisfaire. En particulier, cette thèse étudie des contraintes spécifiquement conçues pour la fouille de quasi-cliques presque-persistantes dans des graphes dynamiques. Notre extracteur est plusieurs ordres de grandeurs plus efficace que les algorithmes existants se restreignant à la fouille de motifs exacts dans des relations ternaires ou à la fouille de motifs tolérants aux erreurs dans des relations binaires. Malgré ces résultats, une telle approche exhaustive ne peut souvent pas, en un temps raisonnable, tolérer tout le bruit contenu dans le jeu de données. Dans ce cas, compléter l'extraction avec une agglomération hiérarchique des motifs (qui ne tolèrent pas suffisamment de bruit) améliore la qualité des collections de motifs renvoyées.

# PUBLICATIONS

Most ideas and figures have appeared previously in the following publications:

## INTERNATIONAL JOURNAL

[CBRB09] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Closed patterns meet n-ary relations. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1–36, March 2009. (Cited on page 53.)

## INTERNATIONAL CONFERENCES

[BCTB08] Jérémy Besson, Loïc Cerf, Rémi Thévenoux, and Jean-François Boulicaut. Tackling closed pattern relevancy in n-ary relations. In *MMD '08: Proceedings of the First International Workshop on Mining Multidimensional Data*, pages 2–16, September 2008. (Cited on page 83.)

[CBRB08] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. DATA-PEELER: Constraint-based closed pattern mining in n-ary relations. In *SDM '08: Proceedings of the Eighth SIAM International Conference on Data Mining*, pages 37–48. SIAM, April 2008. (Cited on page 53.)

[CGSB08] Loïc Cerf, Dominique Gay, Nazha Selmaoui, and Jean-François Boulicaut. A parameter-free associative classification method. In *DaWaK '08: Proceedings of the Tenth International Conference on Data Warehousing and Knowledge Discovery*, pages 293–304. Springer, September 2008. (Cited on page 4.)

[CMB09] Loïc Cerf, Pierre-Nicolas Mougel, and Jean-François Boulicaut. Agglomerating local patterns hierarchically with ALPHA. In *CIKM '09: Proceedings of the 18th International Conference on Information and Knowledge Management*, pages 1753–1756. ACM Press, November 2009. (Cited on page 83.)

[CNB09] Loïc Cerf, Tran Bao Nhan Nguyen, and Jean-François Boulicaut. Discovering relevant cross-graph cliques in dynamic networks. In *ISMIS '09: Proceedings of the 18th International Symposium on Methodologies for Intelligent Systems*, pages 513–522. Springer, September 2009. (Cited on page 121.)

[SDCP+07] Yolanda Sanchez-Dehesa, Loïc Cerf, Jose Maria Pena, Jean-François Boulicaut, and Guillaume Beslon. Artificial regulatory networks evolution. In *MLSB '07: Proceedings of the First International Workshop on Machine Learning for Systems Biology*, pages 47–52, October 2007.

## NATIONAL CONFERENCES

[CBB09] Loïc Cerf, Jérémy Besson, and Jean-Francois Boulicaut. Extraction de motifs fermés dans des relations n-aires bruitées. In *EGC '09 : Actes des neuvièmes Journées Extraction et Gestion des Connaissances*, pages 163–168. Cépaduès-Éditions, January 2009. (Cited on page 83.)

[GCSFB10] Dominique Gay, Loïc Cerf, Nazha Selmaoui-Folcher, and Jean-François Boulicaut. Un nouveau cadre de travail pour la classification associative dans les données aux classes disproportionnées. In *SFC '10 : Actes des 17èmes Rencontres de la Société Francophone de Classification*, pages 47–50. Presses Académiques, June 2010. (Cited on page 4.)

[NCB10] Kim-Ngan T. Nguyen, Loïc Cerf, and Jean-François Boulicaut. Sémantiques et calculs de règles descriptives dans une relation n-aire. In *BDA '10: Actes des 26èmes Journées Bases de Données Avancées*, October 2010. Accepted but unpublished yet. (Cited on page 156.)

## BOOK CHAPTER

[CNB10] Loïc Cerf, Tran Bao Nhan Nguyen, and Jean-François Boulicaut. Mining constrained cross-graph cliques in dynamic networks. Accepted for publication in a book coauthored by the partners of the IQ European project, 2010.

## POSTERS

[CBRB07] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Extraction d'hyper-rectangles fermés sous contraintes, July 2007. BDI '07: Atelier Bases de Données Inductives.

[CBRB08] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Constrained-based closed pattern mining in n-ary relations, April 2008. SML '08: Spring Workshop on Mining and Learning 2008.

## OTHER ARTICLES

[Cer08a] Loïc Cerf. Mining distrowatch.com logs part 1, March 2008. http://www.blue-gnu.biz/content/mining_distrowatch_com_logs_part_1. (Cited on page 121.)

[Cer08b] Loïc Cerf. Mining distrowatch.com logs part 2, March 2008. http://www.blue-gnu.biz/content/mining_distrowatch_com_logs_part_2. (Cited on page 121.)

# ACKNOWLEDGMENTS

This work was achieved within the TURING research group. Without the fruitful collaborations I was involved in, none of the results, presented in this thesis, would have been possible. That is why I want to cheerfully thank *every* member of this friendly team. The researchers who coauthored the publications listed in the two previous pages deserve special mentions. So, thank you Jérémy and Céline for your developments; they were the starting points of the generalizations this thesis exposes. Thank you Rémi, Bao, Pierre-Nicolas and Ngan for your outstanding technical contributions. Thank you Yolanda, Chema, Guillaume, Dominique and Nazha for your expertness in fascinating domains. Thank you Christophe and Marc for the constructive discussions we have had. And, above all, thank you Jean-François for your excellent management, your friendship and your titanic work. It was an honor to have this manuscript reviewed by Bruno Crémilleux and Hannu Toivonen. I am also grateful to the international members of the jury — Francesco Bonchi, Toon Calders and Arno Siebes — and to its president, Christel Vrain. Last but not least, *minha chérizinha*: her love and support is priceless.

This doctoral work was sequentially funded by the European project IST-FET FP6-516169 IQ and by the French project ANR-07-MDCO-014 BING02.

# CONTENTS


I INTRODUCTION 1

II STATE OF THE ART AND THEORETICAL BASIS 9

1 CONSTRAINT-BASED CLOSED ITEMSET MINING 13
- 1 Mining Closed Itemsets 13
  - 1.1 Context 13
  - 1.2 Definition 13
  - 1.3 Complete Extraction 14
- 2 Constraining the Itemsets 19
  - 2.1 Why Are Constraints Wanted? 19
  - 2.2 What is a Constraint? 20
  - 2.3 Classes of Constraints 21
  - 2.4 On Closedness 34
- 3 Conclusion 35

2 GENERALIZING CLOSED ITEMSET MINING 37
- 1 Mining Noise-Tolerant Itemsets 37
  - 1.1 Theoretical Aspects 37
  - 1.2 State of the Art 41
- 2 Mining Closed Patterns in $n$-ary Relations 43
  - 2.1 Theoretical Aspects 43
  - 2.2 State of the Art 46
- 3 Mining Closed Patterns in *Noisy* $n$-ary Relations 48
  - 3.1 Tolerating Noise Is a Must 48
  - 3.2 State of the Art 48
- 4 Conclusion 49

III MINING $n$-ARY RELATIONS 51

3 DATA-PEELER: THE FIRST CLOSED $n$-SET EXTRACTOR 55
- 1 DATA-PEELER 55
  - 1.1 A Closed $n$-Set Extractor 55
  - 1.2 Enumeration 55
  - 1.3 Efficient Enforcement of $\mathcal{C}_{\text{connected}}$ 56
  - 1.4 Efficient Enforcement of $\mathcal{C}_{\text{closed}}$ 57
  - 1.5 Algorithm 58
  - 1.6 Choosing the Element to Enumerate 59
- 2 Improvements to the Enumeration 60
  - 2.1 Removing Elements from $\mathcal{S}$ 60
  - 2.2 Moving Elements from $V$ to $U$ 60
  - 2.3 Improved Algorithm 62
- 3 Example of Computation 64
- 4 Data Structures 66
  - 4.1 Storing the Dataset 66
  - 4.2 Storing the Enumeration Nodes 67
  - 4.3 Space Complexity 67
- 5 Experimental Results 67
  - 5.1 QUEST-Generated Datasets 67
  - 5.2 Impact of the Enumeration Strategy 68
  - 5.3 Comparison with Competitors 68
  - 5.4 Scalability w.r.t. the Arity 69
- 6 Robustness w.r.t. Binarization 71

7 Minimizing multi-valued Logic Functions 72
7.1 Problem Setting 72
7.2 Simplifying Multi-Valued Logic Functions 73
7.3 A Global Model of $\mathcal{R}$ 73
7.4 A Closed $n$-Set Greedy Post-Processing 74
7.5 Experimental results 75
7.6 Improving Time Performances 76
8 Conclusion 79

IV MINING *noisy* $n$-ARY RELATIONS 81

4 FENSTER EXTRACTS $n$-SETS TOLERATING ERRORS IN THE RELATION 85
1 Closed ET-$n$-Sets 85
1.1 Absolute Noise-Tolerance 85
1.2 Relative Noise-Tolerance 86
2 FENSTER 87
3 Implementation 89
3.1 $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ 89
3.2 Choosing the Element to Enumerate 93
4 Space Complexity 93
5 Empirical Study 94
5.1 Synthesizing Datasets 94
5.2 Global Quality Results 94
5.3 Comparison with Competitors 97
6 Mining *Anomalously* Dense ET-$n$-Sets 100
6.1 Local Pattern 100
6.2 Strong Closedness 102
6.3 Global Quality Results 103
7 Conclusion 105

5 AGGLOMERATING LOCAL PATTERNS HIERARCHICALLY WITH ALPHA 107
1 Agglomerating Closed ET-$n$-Sets 107
1.1 A Pattern Clustering Scheme 107
1.2 Hierarchical Agglomeration 108
2 Returning the Few Relevant Patterns 110
2.1 Cluster Relevancy Measure 110
2.2 Selecting the Relevant Clusters 111
3 Empirical Study 112
3.1 Quality Measures 112
3.2 Assessing the Agglomeration 114
3.3 Assessing the Selection 115
4 Conclusion 117

V APPLICATION TO DYNAMIC GRAPH MINING 119

6 MINING DYNAMIC GRAPHS 123
1 Specializing $n$-ary Relation Mining 123
1.1 Dynamic Graph 123
1.2 A Closed ET-3-Set Under Constraints 123
1.3 Problem Setting 126
2 Related Work 126
2.1 Cross-Graph Quasi-Clique Mining 126
2.2 Contiguity 127
3 Mining $\tau$-Contiguous Closed ET-3-Set 127
3.1 A Piecewise (Anti)-Monotone Constraint... 127

3.2 ...Partially Handled in Another Way 128
3.3 Enforcing the τ-Closedness 130
4 Mining τ-Contiguous Closed ET-3-Cliques 130
4.1 A Piecewise (Anti)-Monotone Constraint... 130
4.2 ...Better Handled in Another Way 131
4.3 Constraining the Enumeration 133
4.4 Contraposition of Enumeration Constraints 134
4.5 Enforcing the Symmetric τ-Closedness 135
5 Conclusion 136
7 MINING THE VÉLO'V USAGE NETWORK 139
1 Dataset 139
2 Symmetry Between Departure and Arrival Stations 141
2.1 Avoiding False Positive Noise 141
2.2 Decreasing the Running Times 142
3 Effect of a τ-Contiguity Constraint 142
4 Agglomeration, Selection and Interpretation 143
4.1 Agglomeration and Selection 143
4.2 Seven Patterns 144
4.3 General Observations 148
5 Conclusion 150

VI CONCLUSION 151

VII BIBLIOGRAPHY 157

Figure 1 The EXTRACT closed itemset extractor (under a frequency constraint). 17
Figure 2 Enumeration of any property $e \in V^{\mathcal{P}}$. 17
Figure 3 Enumeration tree EXTRACT traverses when mining $\mathcal{B}_E$. 18
Figure 4 The EXTRACT++ closed itemset extractor (under any conjunction of monotone and anti-monotone constraints). 22
Figure 5 The EXTRACT# closed itemset extractor (under any loose anti-monotone constraint). 25
Figure 6 The EXTRACT## closed itemset extractor (under a loose anti-monotone constraint with a known max function). 26
Figure 7 The EXTRACT* closed itemset extractor (under any piecewise (anti)-monotone constraint). 31
Figure 8 Classes of constraints preserving the freedom to enumerate any property anytime. 32
Figure 9 Classes of constraints generalizing anti-monotone constraints but whose enforcements require modified enumeration principles. 33
Figure 10 Enumeration of any element $e \in V$. 56
Figure 11 Enumeration of the element $4 \in V^2$ from node $M$ (Example 12). 56
Figure 12 Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$. 56
Figure 13 Enumeration of the element $4 \in V^2$ from node $M$ (Example 13). 57
Figure 14 Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$. $\mathcal{C}_{\text{closed}}$ is checked on $U \sqcup V$ extended with every element in $\mathcal{S}$. 58
Figure 15 Enumeration of the element $4 \in V^2$ from node $M$ (Example 14). 58
Figure 16 The DATA-PEELER algorithm. 59
Figure 17 Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$ and $\mathcal{S}$. $\mathcal{C}_{\text{closed}}$ is checked on $U \sqcup V$ extended with every element in $\mathcal{S}$. 60
Figure 18 Illustration of Example 16. 61
Figure 19 Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$ and $\mathcal{S}$. The elements of $V$ that are necessarily present are moved to $U$. $\mathcal{C}_{\text{closed}}$ is checked on $U \sqcup V$ extended with every element in $\mathcal{S}$. 61
Figure 20 Illustration of Example 17. 62
Figure 21 The DATA-PEELER improved algorithm. 63
Figure 22 Part of the enumeration tree DATA-PEELER traverses when mining $\mathcal{R}_E$. 65
Figure 23 Comparing sensible enumeration strategies. 69
Figure 24 Comparison with CUBEMINER and TRIAS. 70
Figure 25 Effect of the arity on the extraction times. 70

Figure 26 Time to minimize, under $\mathcal{C}_{\text{k-summary}}$, a Quest-generated dataset. 78
Figure 27 Space to minimize, under $\mathcal{C}_{\text{k-summary}}$, a Quest-generated dataset. 78
Figure 28 Conversion from a relative noise tolerance $(r^1, r^2)$ to an absolute one $(\epsilon^1, \epsilon^2)$ depending on the region of interest. 87
Figure 29 Fenster enumerating any element $e \in V$. 88
Figure 30 Illustration of Example 21. 88
Figure 31 Illustration of Example 23. 89
Figure 32 The Fenster algorithm. 90
Figure 33 Global qualities of the closed ET-3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset. 95
Figure 34 False positive rates of the closed ET-3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset. 96
Figure 35 False negative rates of the closed ET-3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset. 96
Figure 36 False negative rates of the closed ET-3-sets with at least five elements per attribute in a $32 \times 32 \times 32$ dataset. 97
Figure 37 Global qualities of the closed ET-3-sets with at least five elements per attribute in a $32 \times 32 \times 32$ dataset. 98
Figure 38 Global qualities of the closed ET-4-sets extracted with at least two elements per attribute in a $16 \times 16 \times 16 \times 16$ dataset. 98
Figure 39 Times to extract the *exact* closed 3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset. 99
Figure 40 Global quality of the collection of patterns extracted by Fenster in a $16 \times 16$ dataset. 100
Figure 41 Global quality of the collection of patterns extracted by `AC-Close` in a $16 \times 16$ dataset. 101
Figure 42 Times to extract the collection of error-tolerant patterns with `AC-Close` or Fenster in a $16 \times 16$ dataset. 101
Figure 43 Enumeration of any element $e \in V$. 102
Figure 44 The generalized Fenster algorithm. 103
Figure 45 Illustration of Example 27. 103
Figure 46 Global quality of the $\delta$-closed ET-3-sets extracted with at least four elements per attribute in a $32 \times 32 \times 32$ dataset. 104
Figure 47 Times to extract every closed ET-3-sets with at least five elements per attribute in a $32 \times 32 \times 32$ dataset. 105
Figure 48 Two toy binary relations 109
Figure 49 KNIME dendrogram representing the hierarchical agglomeration. 110
Figure 50 KNIME workflow when experimenting Alpha on the synthetic ternary relations. 113

Figure 51 Best-ones qualities of the collections output by Fenster only and by Fenster + Alpha. 114
Figure 52 Sizes of the collections output by Fenster only and by Fenster + Alpha. 115
Figure 53 Best-ones qualities of the output collections with and without the selection of the relevant patterns. 116
Figure 54 Sizes of the output collections with and without the selection of the relevant patterns. 116
Figure 55 A dynamic (directed) graph ($\mathcal{N} = \{a, b, c, d\}$, $\mathcal{T} = \{0, 0.5, 2, 3\}$). 123
Figure 56 The Purge_$V^{\text{times}}$ procedure. 129
Figure 57 Enumeration of $0.5 \in V$ during the extraction of 1-contiguous 3-sets from the dataset represented in Table 9. 129
Figure 58 Handling the symmetry constraint. 132
Figure 59 Append_contraposition. 135
Figure 60 main. 136
Figure 61 Fenster specialization for $\tau$-contiguous closed ET-3-clique mining. 137
Figure 62 A Vélo'v station. 140
Figure 63 Effect of a $\tau$-contiguity on the number of closed 4-cliques and the time to extract them. 143
Figure 64 Mining $\mathcal{R}_{\text{Vélo'v}}$. 144
Figure 65 During the week-end from 3 pm to 8 pm. 145
Figure 66 On Fridays, Saturdays, Sundays and Mondays from 3 pm to 8 pm. 145
Figure 67 During the week-end from 3 pm to 8 pm. 146
Figure 68 During the week-end from 3 pm to 8 pm. 146
Figure 69 On Mondays, Tuesdays, Wednesdays, Thursdays and Fridays from 12 noon to 5 pm. 147
Figure 70 Everyday but Sunday from 3 pm to 8 pm. 147
Figure 71 All week long from 12 noon to 9 pm. 148
Figure 72 The Heaviside step function and two logistic functions. 155

Table 1 $\mathcal{B}_E \subseteq \{o_1, o_2, o_3, o_4\} \times \{p_1, p_2, p_3\}$. 13
Table 2 Flexible constraints. 27
Table 3 $gp' : \mathcal{B}_E \to \mathbb{R}_+$. 31
Table 4 $\mathcal{B}_{\text{hidden}E} \subseteq \{o_1, o_2, o_3, o_4\} \times \{p_1, p_2, p_3\}$. 38
Table 5 $\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$. 44
Table 6 Minimizing random multi-valued logic functions (density: 6.25%). 76
Table 7 Minimizing random multi-valued logic functions (density: 25%). 76
Table 8 Minimizing random multi-valued logic functions (density: 50%). 76
Table 9 $(A_0, A_{0.5}, A_2, A_3)$ related to the dynamic graph depicted Figure 55. 124
Table 10 Number of patterns in $\mathcal{R}_{\text{Vélo'v}}$. 141
Table 11 Running times of FENSTER on $\mathcal{R}_{\text{Vélo'v}}$. 142

# Part I

# INTRODUCTION

*Intellectus humanus fertur ad abstracta propter naturam propriam; atque ea, quae fluxa sunt, fingit esse constantia. Melius autem est naturam secare, quam abstrahere; id quod Democriti schola fecit, quae magis penetravit in naturam, quam reliquae. Materia potius considerari debet, et ejus schematismi, et meta-schematismi, atque actus purus, et lex actus sive motus; formae enim commenta animi humani sunt, nisi libeat leges illas actus formas appellare.*

— Sir Francis Bacon [4]

*The human understanding is, by its own nature, prone to abstraction, and supposes that which is fluctuating to be fixed. But it is better to dissect than abstract nature; such was the method employed by the school of Democritus, which made greater progress in penetrating nature than the rest. It is best to consider matter, its conformation, and the changes of that conformation, its own action, and the law of this action or motion, for forms are a mere fiction of the human mind, unless you will call the laws of action by that name.*

— translation by William Wood [56]

## BACKGROUND

The aphorism, starting this thesis, is part of the *Novum Organum*. This work, by Francis Bacon, is more a scientific method than a philosophy. It argues for both the *reductionism* and the *inductive reasoning*. The inductive reasoning is at work in data mining. It proceeds from facts to laws; from data to models. The reductionism, "employed by the school of Democritus", was later developed by David Hume in his *bundle theory*. In this pluralist (rather than monist) theory, an object consists of its properties and nothing else. For instance, a wine *is* its origin, its aromas, its color, its acidity, its viscosity, the grape varieties making it, etc. Considering a set of properties $\mathcal{P}$ (e. g., {"from Bourgogne","from Alsace", . . . ,"earthy","herbaceous","peppery", . . . ,"very acid", . . .}), an object (e. g., a wine) translates to a subset of $\mathcal{P}$ (e. g., a wine can be {"from Alsace","earthy","floral","garnet","made of resling","made of gewurztraminer","very acid"}). In this way, a set of objects $\mathcal{O}$, described with the properties in $\mathcal{P}$, constitutes a binary relation $\mathcal{B} \subseteq \mathcal{O} \times \mathcal{P}$, which encodes whether an object *has* a property. An interesting problem, which hopefully helps in "penetrating nature", is the search of patterns in a relation that gathers objects involved in a phenomenon to understand. Nevertheless, in the quoted aphorism, Francis Bacon warns us: "forms are a mere fiction of the human mind". Algorithms, such as the data mining methods, do not suffer from the weaknesses of the human mind. The patterns they detect are all the more trusty as their "form" is *formally specified* and the algorithms at work are *exact*. The formal specification of a data mining task is expressed in mathematical terms. Given a dataset, *exact* approaches output, without any approximation, *the* pattern(s) matching the mathematical expression.

Following the bundle theory, consider a dataset represented by a binary relation $\mathcal{B} \subseteq \mathcal{O} \times \mathcal{P}$. To "dissect" it, it may be interesting to list every subset of properties that describe, together, at least ten objects. If $\mathcal{B}$ represents wines, it may be discovered, in this way, that more than ten wines (in the dataset) are, at the same time, "from Alsace", "earthy" and "very acid". In this example, the formal specification of the patterns is $\{(O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}} \mid O \times P \subseteq \mathcal{B} \wedge |O| \geqslant 10\}$, i. e., the data mining task consists in listing every subset of properties $P \subseteq \mathcal{P}$ shared by a subset of objects $O \subseteq \mathcal{O}$ ("$O \times P \subseteq \mathcal{B}$"), which must be at least ten ("$|O| \geqslant 10$"). This problem, namely the *(10)-frequent itemset mining*, is NP-hard. Nevertheless, exact algorithms (aka *complete extractors* because they extract *every* pattern matching the formal specification) are efficient enough to be tractable on rather large relations. Other famous "forms" cannot be discovered exactly in a reasonable time. Clustering (e. g., partitioning the wines into homogeneous groups w.r.t. their properties) and classifying (e. g., learning/predicting the origin of a wine from its other properties) are famous examples of such "forms". Their optimal solutions can be formally defined but, given a large dataset, computing them takes ages. Heuristics allow to approximately solve them though. Interestingly, complete extractors may be useful in a first step towards solving these difficult problems. E. g., a classifier can be based on rules whose bodies are frequent itemsets. `fitcare` [CGSB08, GCSFB10] is such a classifier[1]. Although it is arguable, we believe that delaying the lossy heuristics as far as possible (i. e., as far as tractable) in the knowledge discovery process makes the whole process trustier. This trust is particularly important with unsupervised learning (like itemset mining or clustering; unlike classification), where, by definition, the computed pattern(s) cannot be tested.

### CONSTRAINT-BASED SPECIFICATION

By definition, constraint-based methods outline the "form" of the computed pattern(s) via constraints. Consider, again, the formal specification of 10-frequent itemsets. It consists of two constraints: (a) encompassing only couples present in the relation and (b) involving at least ten objects. The relevancy of these constraints, therefore that of the discovered patterns, is arguable. Algorithms do not suffer from the weaknesses of the human mind but the choice of the "form" they recognize does! Depending on the actual application, other constraints may match more interesting patterns. E. g., in an attempt to draw conclusions about wines, itemsets may be useless unless they contain an origin (i. e., a property "from [region]") or unless the standard deviation of their acidity remains below a user-defined threshold (what requires this numerical information for each wine). Nevertheless over-specifying the patterns is not good either: surprising patterns, which usually are the most valuable, could be missed. Anyway, beyond their ability to select the most relevant patterns, additional constraints often are indispensable to the practical computation of these patterns. Indeed a constraint, with good mathematical properties, *guides* the algorithm towards the solution(s). I. e., regions of the search space, where the constraint is violated, are pruned. That means shorter running times and the tractable discovery of patterns in larger datasets. Constraints are grouped in classes w.r.t. the enumeration principles (how the search

1. We have chosen to not present it in this thesis.

space is traversed) that are required to take advantage of them, i. e., to prune the search space. The complete extractors in this thesis handle a very expressive class of constraints. As a consequence, although these approaches are very general, they can take advantage of many constraints and mine application-specific patterns precise constraints finely outline.

Rather than itemset mining, this thesis generalizes *closed* itemset mining. The additional adjective translates to an additional constraint on the patterns. An itemset, associating a set of objects with a set of properties, is closed if and only if no other (and necessarily larger) frequent itemset has all these properties *and* describe all these objects. E. g., ten wines that are both "from Alsace" and "very acid" form a 10-frequent itemset. This pattern is not closed if there exists at least an eleventh wine which is both "from Alsace" and "very acid" (a larger pattern associates eleven wines with the properties "from Alsace" and "very acid"). It is not closed either if the ten wines are all "earthy" (a larger pattern associates the ten wines with the properties "from Alsace", "very acid" and "earthy"). The closedness constraint condenses collections of frequent itemsets by only keeping the most informative ones. Without it, the returned collections usually are too large to be interpreted. That is why, when generalizing itemset mining, a close attention is paid to the ability to the preservation of a *useful* closedness property. On one hand, its definition should still support a lossless condensation of the patterns. On the other hand, it must be efficiently computed, i. e., in the class of constraints the generalized extractors handle. These two orthogonal concerns are not specific to the closedness constraint. They are actually found across any data mining task and respectively relate to their declarative and procedural semantics.

### GENERALIZATION TOWARDS $n$-ARY RELATIONS

Discovering that more than ten wines are, at the same time, "from Alsace", "earthy" and "very acid" makes us hypothesize that these three properties are semantically related, e. g., that wines from Alsace generally are earthy and very acid. However, "the human understanding is, by its own nature, prone to abstraction, and supposes that which is fluctuating to be fixed". Are we drawing a general conclusion too quickly? Indeed, the "conformation" of a wine changes a lot w.r.t. its vintage date. A first idea consists in appending {"vintage 1988","vintage 1989", ...} to the set of properties. Nevertheless, a wine has only one vintage date, e. g., a wine "vintage 1988" cannot be, at the same time, "vintage 1989" or "vintage 1990". Notice that it also has only one origin, e. g., a wine cannot be, at the same time, "from Bourgogne" and "from Alsace". As a consequence, no itemset can support the discovery that, between 1988 and 1990, the wines from Bourgogne and from Alsace were all earthy. Indeed, by definition, no object shares two (or more) properties that are self-exclusive. Anyway, such patterns obviously are interesting. To mine them, this thesis proposes to split the one-dimensional set of properties $\mathcal{P}$ into several orthogonal dimensions of analysis $\mathcal{D}^1$, $\mathcal{D}^2$, $\mathcal{D}^3$, etc. The dataset is then expressed in terms of elements in $\mathcal{D}^1 \times \mathcal{D}^2 \times \mathcal{D}^3 \times \ldots$, i. e., it is an $n$-ary relation, where $n$ is the number of dimensions. E. g., to analyze wines w.r.t. their origins, their aromas and their vintage dates, these three orthogonal attributes

constitute three dimensions of analysis: $\mathcal{D}^{\text{origins}} = \{$"from Bourgogne", "from Alsace", ...$\}$, $\mathcal{D}^{\text{aromas}} = \{$"earthy","herbaceous","peppery", ...$\}$ and $\mathcal{D}^{\text{vintage}} = \{$"vintage 1988","vintage 1989", ...$\}$. The dataset is a ternary relation $\mathcal{R}_{\text{wines}} \subseteq \mathcal{D}^{\text{origins}} \times \mathcal{D}^{\text{aromas}} \times \mathcal{D}^{\text{vintage}}$. It contains ("from Alsace","earthy","vintage 1988") if and only if the wines harvested in Alsace in 1988 usually are earthy.

The data mining group led by Pr. Jean-François Boulicaut currently works on the systematic generalization of data mining tasks (clustering, classification, etc.) towards $n$-ary relations. This thesis exposes the first results. It considers the complete extraction of generalized closed itemsets. E. g., the algorithms presented in this thesis can discover patterns such as the one discussed in the previous paragraph, i. e., ($\{$"vintage 1988","vintage 1989","vintage 1990"$\}$, $\{$"from Bourgogne", "from Alsace"$\}$, $\{$"earthy"$\}$). Generalizing the definition of closed itemsets, to make them suit $n$-ary relations, is trivial. Generalizing their complete extraction is much harder. Indeed, an essential mathematical property, namely the Galois connection, is lost. Our proposal iteratively builds candidate patterns. At every iteration, any element, from any attribute domain, can be chosen to enlarge the current candidate. This freedom allows the design of an enumeration strategy that avoids large regions of the search space but discovers all patterns. It outperforms, by orders of magnitude, related work designed for ternary relation mining.

Real-life dynamic graphs are particular ternary relations (nodes $\times$ nodes $\times$ timestamps). They are found in many application domains. E. g., they support the kinetic analysis of biological networks such as protein-protein interactions. Cross-graph closed cliques are sets of nodes that are completely connected across several graphs. They are relevant patterns for the local analysis of dynamic graphs. This is all the more true when the graphs supporting a pattern are grouped in time, i. e., their timestamps are close to each other. It turns out that both the symmetry constraint on the two node attributes (tails and heads of the edges) and the almost-contiguity constraint on the time attribute are efficiently enforced by our extractors. I. e., the general approaches, detailed in this thesis, can efficiently mine every almost-contiguous cross-graph closed clique. This illustrates the winning combination of a general extractor and an expressive class of constraints making it useful for specific applications.

### GENERALIZATION TOWARDS NOISE TOLERANCE

Real-life datasets are noisy. In the context of a relation, tuples that should be absent from it are present and vice versa. There are many sources of noise. One of them is due to the all-or-nothing aspect of the property encoded by a relation. E. g., it was written, in the previous section, that $\mathcal{R}_{\text{wines}}$ contains ("from Alsace","earthy","vintage 1988") if and only if the wines harvested in Alsace in 1988 *usually* are earthy. If these wines *sometimes* are earthy, the related 3-tuple may (or not) end up in the relation. Among other sources of noise, let us mention erroneous measures (e. g., the oenologist was sick), subjective measures (e. g., the oenologist was in a good/bad mood), stochastic phenomena (e. g., the wine was corked), too small samples (e. g., only the best/worse wines of a kind were tasted), etc. Mining the closed itemsets in a noisy relation only allows to recover logarithmic fragments of the patterns

that would be discovered in the same relation but deprived of noise. Indeed, closed itemsets are, by definition, not allowed to encompass anything but tuples present in the relation. Their counterparts in $n$-ary relations raise even more troubles because they usually encompass more tuples. E. g., assume ({"vintage 1988","vintage 1989","vintage 1990"},{"from Bourgogne", "from Alsace"},{"earthy"}) is a pattern to discover in $\mathcal{R}_{\text{wines}}$. Imagine that both ("from Alsace","earthy","vintage 1988") and ("from Bourgogne","earthy","vintage 1990"), affected by noise, are absent from $\mathcal{R}_{\text{wines}}$. Then, instead of one, five (unconstrained) closed patterns are discovered.

To recover the patterns that were affected by noise (or, at least, larger fragments of those patterns), the formal specification of closed itemset mining (actually, of its generalization towards $n$-ary relations) cannot be kept as is. Noise tolerance parameters generalize it. They are upper-bounds of the number of $n$-tuples that every element (in every attribute) of every pattern is allowed to encompass. E. g., despite the two absent 3-tuples, ({"vintage 1988","vintage 1989","vintage 1990"},{"from Bourgogne", "from Alsace"},{"earthy"}) can be discovered in $\mathcal{R}_{\text{wines}}$. The analyst only needs to specify the tolerance of one absent 3-tuple per origin (one absent 3-tuple involves the Bourgogne, another involves the Alsace), one per vintage (both the vintages 1988 and 1989 reach this upper-bound) and two per aroma (both missing 3-tuples involve earthy wines). Because larger patterns are discovered when missing tuples are allowed, constraints on their sizes can be stronger and the supernumerary tuples, present in the relation because of noise, are avoided. From a procedural point of view, an efficient incremental computation of the quantity of noise, which candidate patterns tolerate, is not trivial. By implementing it underneath the enumeration principles developed for discovering closed itemsets generalized to $n$-ary relation, their noise tolerant counterparts are efficiently listed. Furthermore the same broad class of constraints is available to both fasten the extraction and focus it on the most relevant patterns. For example, almost-contiguous cross-graph closed *quasi*-cliques can be discovered.

Nevertheless, in large datasets, it often remains intractable to tolerate enough noise so that the *real* patterns are recovered. Following the philosophy "completeness as far as possible", the obtainable fragments are used as a basis and a heuristics complements the knowledge discovery process. It consists of a hierarchical agglomeration of the patterns followed by a selection of the relevant agglomerates. To compute the distance between two patterns, it makes sense to come back to the relation. In this way, the tuples encompassed by the considered agglomerate (minimal envelope containing the two clustered patterns) but absent from the patterns composing it, are also taken into account. The selection step is indispensable to make the returned pattern collection smaller, hence more interpretable. This collection is forced to cover the seminal pool of patterns. In this way, the completeness of the initial extraction is, somehow, preserved. The selected patterns are the ones showing the best trade-offs between a small proportion of (supposedly) noise inside them and a great distance to the outside patterns.

## ORGANIZATION OF THE THESIS

In the next chapter, the extraction of every closed itemset under constraints is surveyed. The most famous classes of constraints are

detailed along the enumeration principles that enable their efficient enforcements. In particular, the expressive class of constraints, that our complete extractors handle, is defined, studied and illustrated. Chapter 2 details the difficulties in generalizing closed itemset mining towards noise tolerance on one hand, and towards $n$-ary relations on the other hand. The state-of-the-art approaches, that tackle these problems, are presented and discussed. Chapter 3 exposes the first algorithm listing every closed pattern in arbitrary $n$-ary relations. After experiments showing its excellent time performances, a pre and a post-processing are described. Both make an original use of the supernumerary attributes (to force a certain robustness to binarization on one hand; to minimize multi-valued logic functions on the other hand). Chapter 4 discusses the additional tolerance to noise. After detailing the fundamental implementation details, the approach is shown to provide, within a reasonable time, only fragments of the hidden patterns. However, agglomerating the fragments heuristically recovers the hidden patterns. Chapter 5 details this step and the following one, i. e., the selection of the relevant agglomerates. Chapter 6 shows how the symmetry and the almost-contiguity constraints specialize our algorithms in the complete extraction of almost-contiguous cross-graph closed quasi-cliques in dynamic networks. These constraints are part of the ones that are efficiently enforceable thanks to the enumeration principles at work. Nevertheless, it is explained how and why specific implementations enable greater gains in running times. Chapter 7 details a specific application, which aims at understanding how the Vélo'v network is used. This bicycle rental service, run by the urban community of Lyon, logged, along the two studied years, more than ten millions rides. The chapter details how these data are turned into a 4-ary relation of more than 100000 tuples. Despite weak minimal size constraints, a few hours are enough for our algorithms to discover relevant patterns. Finally, a short summary and a few perspectives conclude this thesis.

# Part II

# STATE OF THE ART AND THEORETICAL BASIS

# OUTLINE

One of the earliest and most successful type of local pattern is the *itemset*. Itemsets are extracted from binary relations. Generically, a binary relation encodes Boolean properties that objects have (the couple (object,property) is in the relation) or not (the related couple is not in the relation). In such binary relations, an itemset is a subset of properties associated with the objects sharing all these properties. A closedness property allows a lossless condensation of all itemsets by keeping only the most informative patterns. Nevertheless, complete collections of closed itemsets remain huge, hence tedious to interpret, and extracting them is intractable unless additional constraints are enforced. Constraints express the relevancy of the closed itemsets to keep. The class of constraints an extractor can use to prune the search space, depends on its enumeration principles. After presenting what is a closed itemset, Chapter 1 lists the classes of constraints (definition and related enumeration principles) that are found in the literature. In particular, this chapter explains how the reverse-search paradigm enables a depth-first extraction of patterns under a loose anti-monotone constraint, shows that the primitive-based and the piecewise (anti)-monotone constraints are the same and emphasizes, through examples, how large this class is. Those are new contributions, which have not been published yet. Chapter 1 ends with a brief study of the closedness constraint. It details generalizations of it that aim at restricting the output to the anomalous patterns. The link between the strong closedness and the stability index is shown.

Noise alters most datasets. In particular, a relation may miss some tuples and closed itemsets, which cannot cover such tuples, only describe fragments of the hidden patterns. Furthermore, when available, more than two attributes should be simultaneously taken into consideration for a finer analysis. Chapter 2 presents these two generalizations of closed itemset mining: towards noise tolerance and towards $n$-ary relations. In both cases, the Galois connection is lost, what prevents simple adaptations of closed itemset extractors. To tolerate noise, the first challenge is to define the *noise tolerance*. It looks more natural to tolerate proportions of noise (w.r.t. the sizes of the patterns). Nevertheless, extractions with an absolute tolerance to noise scale much better and allow an efficient enforcement of a closedness constraint. Agglomerating itemsets is another (heuristic) way to tolerate noise. On the contrary, generalizing closed itemsets towards $n$-ary relations is straightforward. Generalizing their complete extraction is much harder. Two algorithms were specifically designed to extract closed patterns in ternary relations. Minimizing logic functions is a related topic, which is briefly presented too. In $n$-ary relations, complete collections of local patterns suffer even more from noise. A few proposals tackle both problems at a time and are discussed at the end of the chapter.

# 1 CONSTRAINT-BASED CLOSED ITEMSET MINING

## 1 MINING CLOSED ITEMSETS

### 1.1 *Context*

Given a finite set of objects $\mathcal{O}$ and a finite set of properties $\mathcal{P}$, let $\mathcal{B} \subseteq \mathcal{O} \times \mathcal{P}$ a binary relation on these domains. Table 1 represents an example of such a relation $\mathcal{B}_E \subseteq \{o_1, o_2, o_3, o_4\} \times \{p_1, p_2, p_3\}$. In this table, every '1' at the intersection of an object (a row) and a property (a column) stands for the presence of the related couple in $\mathcal{B}_E$, i. e., the objects *has* the property. For example the bold '**1**', in Table 1, is at the intersection of the object $o_1$ and the property $p_1$. It represents the presence of $(o_1, p_1)$ in $\mathcal{B}_E$, i. e., the object $o_1$ has the property $p_1$. On the contrary a '0' in Table 1 is at the intersection of two elements which form a couple absent from $\mathcal{B}_E$. For example the bold '**0**' in Table 1 means $(o_2, p_3) \notin \mathcal{B}_E$, i. e., the object $o_2$ does not have the property $p_3$.

*Binary relations associate objects with properties. They are useful across many applicative domains.*

Binary relations are present in many application domains. For instance, $\mathcal{B}_E$ could represent customers ($o_1$, $o_2$, $o_3$ and $o_4$) buying items ($p_1$, $p_2$ and $p_3$). In this context, the bold '**1**' in Table 1 would mean that the customer $c_1$ bought the item $p_1$. The bold '**0**' would be understood as "customer $c_2$ did not buy item $p_3$".

### 1.2 *Definition*

Given a binary relation $\mathcal{B} \subseteq \mathcal{O} \times \mathcal{P}$, a closed itemset is a maximal set of objects sharing the same maximal set of properties. Considering the tabular representation of the binary relation (such as Table 1), it is a maximal rectangle of '1's modulo arbitrary permutations of the rows and the columns. Here is a formal definition.

*Closed itemsets (a) cover only couples present in the binary relation; (b) cannot be enlarged without violating (a).*

**Definition 1 (Closed itemset)** $\forall (O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $(O, P)$ *is a closed itemset iff:*

- $\mathcal{C}_{connected}(O, P) \equiv O \times P \subseteq \mathcal{B}$;
- $\mathcal{C}_{closed}(O, P) \equiv \forall (O', P') \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$,
  $\Big(O \subseteq O' \wedge P \subseteq P' \wedge \mathcal{C}_{connected}(O', P')\Big) \Rightarrow (O', P') = (O, P)$.

With this definition, an itemset is a set of objects *and* a set of properties. It differs a little from the data-mining literature, where an itemset only is a subset of properties (the next section explains how the "supporting" set of objects is deducible). The unusual definition, chosen in this

| | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|
| $o_1$ | **1** | 1 | 1 |
| $o_2$ | 1 | 1 | **0** |
| $o_3$ | 0 | 1 | 0 |
| $o_4$ | 0 | 0 | 1 |

Table 1: $\mathcal{B}_E \subseteq \{o_1, o_2, o_3, o_4\} \times \{p_1, p_2, p_3\}$.

chapter, helps its generalization towards $n$-ary relations and noise tolerance. The first constraint, $\mathcal{C}_{\text{connected}}$, specifies that every object in $O$ must have all the properties in $P$, otherwise $(O, P)$ is not a closed itemset. More precisely it is not (completely) connected. The second constraint, $\mathcal{C}_{\text{closed}}$, forces any strictly larger pattern (more objects, more properties or both) to violate $\mathcal{C}_{\text{connected}}$. It is, w.r.t. $\mathcal{C}_{\text{connected}}$, a closure property on the sets of objects and properties altogether. It can easily be proved that an equivalent closedness constraint only forces the patterns with *one* more element (either an object or a property) to break $\mathcal{C}_{\text{connected}}$. Furthermore, because $\mathcal{C}_{\text{connected}}$ ensures the presence in $\mathcal{B}$ of every couple in $O \times P$, the closedness constraint can be reduced to the search of absent couples involving the additional element only.

**Definition 2 (Closed itemset (equivalent definition))** $\forall (O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $(O, P)$ *is a closed itemset iff:*

- $\mathcal{C}_{connected}(O, P) \equiv O \times P \subseteq \mathcal{B}$;
- $\mathcal{C}_{closed}(O, P) \equiv \begin{cases} \forall o \in \mathcal{O} \setminus O, \neg\mathcal{C}_{connected}(\{o\}, P), \textit{ i.e., } \{o\} \times P \not\subseteq \mathcal{B} \\ \forall p \in \mathcal{P} \setminus P, \neg\mathcal{C}_{connected}(O, \{p\}), \textit{ i.e., } O \times \{p\} \not\subseteq \mathcal{B} \end{cases}$

**Example 1** *In $\mathcal{B}_E$, represented in Table 1, $(\{o_1, o_2\}, \{p_1, p_2\})$ is a closed itemset:*

- $\{o_1, o_2\} \times \{p_1, p_2\} \subseteq \mathcal{B}_E$ *(in Table 1 there are '1's at the intersection of all the related rows and columns);*
- *Every pattern with one more element violates $\mathcal{C}_{connected}$:*
  - $\neg\mathcal{C}_{connected}(\{o_3\}, \{p_1, p_2\})$, *i.e.,* $\{o_3\} \times \{p_1, p_2\} \not\subseteq \mathcal{B}_E$;
  - $\neg\mathcal{C}_{connected}(\{o_4\}, \{p_1, p_2\})$, *i.e.,* $\{o_4\} \times \{p_1, p_2\} \not\subseteq \mathcal{B}_E$;
  - $\neg\mathcal{C}_{connected}(\{o_1, o_2\}, \{p_3\})$, *i.e.,* $\{o_1, o_2\} \times \{p_3\} \not\subseteq \mathcal{B}_E$.

$(\{o_1, o_4\}, \{p_3\})$ *and* $(\{o_1, o_2, o_3, o_4\}, \emptyset)$ *are other examples of closed itemsets in $\mathcal{B}_E$.*

If, again, the binary relation stands for customers buying items, a closed itemset is a maximal subset of customers buying the same maximal subset of items. Such a pattern is useful for analyzing buying behaviors. The closedness constraint filters out all strict "sub-patterns" (i. e., patterns where some elements are removed and none are added) of the largest ones that are extracted. It reduces the size of the output collection (what is necessary when it comes to interpreting it). Whatever the dataset, two arguments justify the choice for closed patterns. The first argument is a theorem stating that the closed patterns always are more informative (lower p-value) that any of its "sub-patterns" (proof in [26], which extends [25]). The second argument is the fact that all (closed and unclosed) connected itemsets are deducible from all closed itemsets only [13], i. e., the latter collection is a *condensed representation* [53] of the former. It means that, given any subset of properties $P' \subseteq \mathcal{P}$ (resp. objects $O' \subseteq \mathcal{O}$), *all* objects (resp. properties) that share these properties (resp. objects) can be derived from the closed itemsets only. They are the largest set of objects (resp. properties) a closed itemset associates with a superset of $P$ (resp. $O$).

*The closedness constraint provides a* lossless condensation *of all itemsets by only keeping the* most informative *ones.*

## 1.3 *Complete Extraction*

### 1.3.1 *Research Directions*

One of the oldest (if not *the* oldest) algorithms, that list every closed itemset in a binary relation, was published in 1969 [20]. The *formal*

*concept analysis*, introduced in 1982 [91] (see [27] for a state-of-the-art reference), studies the mathematical properties of the closed itemsets (aka *formal concepts*). After discovering efficient strategies for enumerating (both closed and unclosed) itemsets under frequency constraints (in particular Apriori [1] in 1994), data miners (re)discovered, in 1999 [61], the relevancy of a condensed representation of such collections by listing the closed itemsets only. This research community focuses on problems such as scalability, constraint handling, generalization towards noise and $n$-ary relations (that do not preserve the mathematical foundation of formal concept analysis, i. e., the *Galois connection*). They are topics this thesis treats. That is why this state of the Art focuses on data mining approaches. Nevertheless, the two communities are not ignoring each other and interesting closed itemset extractors, such as TITANIC [81], were designed at the interface between formal concept analysis and data mining.

### 1.3.2 *Enumerating the Subsets of One Domain*

This section does not aim at detailing the differences, in efficiency, between the many closed itemset miners. In the opposite, it focuses on their similarities. Indeed, the fundamental mechanisms, brought into the complete (i. e., *every* closed itemset is found) extraction of the closed itemsets, usually remain the same. This observation was formalized in [7]. The closed itemset extractors, like most local pattern complete extractors (such as frequent itemset miners), traverse the candidate pattern space, $2^{\mathcal{O}} \times 2^{\mathcal{P}}$, by only enumerating the subsets of one of the two domains (either the subsets of objects or the subsets of properties). Despite its title, it can even be argued that [38] implicitly proceeds to such an enumeration. Traditionally, the subsets of properties are enumerated. The mathematical reason behind this ability in reducing the traversal of the candidates in $2^{\mathcal{O}} \times 2^{\mathcal{P}}$ to that of $2^{\mathcal{P}}$ (or $2^{\mathcal{O}}$) is a bijection between the closed itemsets $(O, P)$ and their sets of properties $P$ (or objects $O$). Stated with the terminology of *formal concept analysis*, an upper/lower adjoint of a Galois connection uniquely determines the other. In our context, the Galois connection is the pair of functions $(f, g)$ defined as follows:

*There is a bijection between the closed itemsets and their subsets of properties (or objects). Thus, almost all closed itemset extractors enumerate subsets of one domain.*

- $\forall O \subseteq \mathcal{O}, f(O) = \{p \in \mathcal{P} \mid O \times \{p\} \subseteq \mathcal{B}\}$.
- $\forall P \subseteq \mathcal{P}, g(P) = \{o \in \mathcal{O} \mid \{o\} \times P \subseteq \mathcal{B}\}$;

It is easily proved that if $(O, P)$ is a closed itemset then $g(P) = O$ (and $f(O) = P$). That is why enumerating the subsets $P$ of $\mathcal{P}$ is equivalent to enumerating the patterns $(g(P), P)$ among which are found every closed itemset (those that have $f(g(P)) = P$).

**Example 2** *In $\mathcal{B}_E$, represented in Table 1, when $\{p_1\}$ is enumerated, it is associated with $g(\{p_1\}) = \{o_1, o_2\}$. However $(\{o_1, o_2\}, \{p_1\})$ is not a closed itemset because $f(\{o_1, o_2\}) = \{p_1, p_2\} \neq \{p_1\}$. A closed itemset is found when $\{p_1, p_2\}$ is enumerated: $(\{o_1, o_2\}, \{p_1, p_2\})$.*

With the publication of [59] and [71] in 2003, the data mining community rediscovered that the extraction of every closed itemset actually is symmetric w.r.t. the domain in which subsets are enumerated. In other terms, applying a closed itemset extractor on a 0/1 matrix such as Table 1 or on its transpose provides the same collection of closed itemsets (the couples $(P, O)$ being reversed in $(O, P)$). As a consequence, it is faster to extract every closed itemset by enumerating the subsets of

the smallest domain. Indeed, they are less numerous ($|2^{\mathcal{D}}|$ increasing with $|\mathcal{D}|$). From now on, let us assume that there are less properties than objects. As a consequence, the subsets of $\mathcal{P}$ are chosen to be enumerated, i. e., the considered patterns are of the form $(g(P), P)$.

The closed itemset extractors do not actually aim at listing every closed itemset but only those satisfying a relevancy constraint. Since the first local pattern miners, and until today, one constraint has been clearly favored to play this role: the frequency constraint. Given a user-defined threshold $\gamma \in \mathbb{N}$, a closed itemset $(g(P), P)$ is frequent if and only if $|g(P)| \geqslant \gamma$. The success of this constraint is twofold: (a) it actually keeps relevant patterns (the discovered conjunction of properties, shared by a great number of objects, is more relevant) and (b) it allows to prune large regions of the search space (hence a reduction of the running times) when the enumerated subsets of properties are larger and larger along the computation. That is why most closed itemset extractors actually are *frequent* closed itemset extractors and enumerate growing subsets of properties. If a candidate pattern $(g(P), P)$ is not frequent, every closed itemset $(g(P'), P')$ with $P \subseteq P'$ is not frequent either because $g(P') \subseteq g(P)$. The region of the search space where candidate patterns have a superset of $P$ is empty of frequent closed itemset. That is why the extractors do not traverse it. It is said to be pruned.

*By enumerating growing subsets of properties, the closed itemsets involving at least $\gamma$ objects can be listed while ignoring many candidate patterns.*

It is important to understand what can mean "enumerating larger and larger subsets of properties". It can mean a breadth-first traversal of the search space (à la Apriori [1]) that is space-consuming. For example, the first closed itemset extractor, Close [61], does so. Nevertheless, it can mean a depth-first traversal of the search space (à la $\mathcal{DF}$ [65]) too. Indeed, to optimally take advantage of the frequency constraint, when a candidate $(g(P), P)$ violates it, the patterns $(g(P'), P')$ with $P \subseteq P'$ should not have been traversed earlier. This is the case with a depth-first traversal of the search space too. The first depth-first closed itemset extractor was ChARM [95].

### 1.3.3 *A Typical Frequent Closed Itemset Extractor*

Extract is an example of a simple closed itemset extractor based on the fundamental principles detailed in the previous section and shared by most closed itemset extractors. This recursive algorithm enumerates subsets of properties in a depth-first way and forces the closed itemsets to have at least $\gamma \in \mathbb{N}$ objects (frequency constraint). Figure 1 expresses it with a formalism that will be used all along this thesis. Extract is initially called with $(U^{\mathcal{P}}, V^{\mathcal{P}}) = (\emptyset, \mathcal{P})$. Here is the semantics behind these two variables:

- $U^{\mathcal{P}} \subseteq \mathcal{P}$ contains properties that will always be present in every closed itemset recursively discovered from the current call of Extract. Previous works sometimes talk about a "conditional base".
- $V^{\mathcal{P}} \subseteq \mathcal{P}$ contains properties that may or may not be present in the closed itemset recursively discovered from the current call of Extract. In other terms, $2^{V^{\mathcal{P}}}$ is the search space given the "conditional base". If $V^{\mathcal{P}} = \emptyset$ then the search space is reduced to $|2^{\emptyset}| = 1$ pattern, $(g(U^{\mathcal{P}}), U^{\mathcal{P}})$, that is output if it is closed.

$U^{\mathcal{P}}$ is the smallest possible set of properties that may be output from the current call of Extract. It is considered in the branch of the

**Input:** $(U^{\mathcal{P}}, V^{\mathcal{P}}) \in (2^{\mathcal{P}})^2$
**Output:** Every closed itemset having all properties in $U^{\mathcal{P}}$, potentially some properties in $V^{\mathcal{P}}$ and satisfying $\mathcal{C}_{\gamma\text{-frequent}}$
**if** $|g(U^{\mathcal{P}})| \geqslant \gamma \wedge \forall s \in \mathcal{P} \setminus (U^{\mathcal{P}} \cup V^{\mathcal{P}}), g(U^{\mathcal{P}}) \times \{s\} \not\subseteq \mathcal{B}$ **then**
  **if** $V^{\mathcal{P}} = \emptyset$ **then**
    **output**$((g(U^{\mathcal{P}}), U^{\mathcal{P}}))$
  **else**
    Choose $e \in V^{\mathcal{P}}$
    EXTRACT$(U^{\mathcal{P}} \cup \{e\}, V^{\mathcal{P}} \setminus \{e\})$
    EXTRACT$(U^{\mathcal{P}}, V^{\mathcal{P}} \setminus \{e\})$
  **end if**
**end if**

Figure 1: The EXTRACT closed itemset extractor (under a frequency constraint).

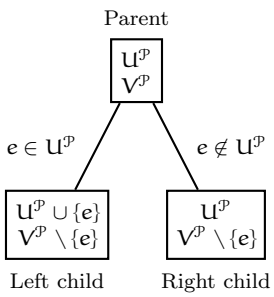
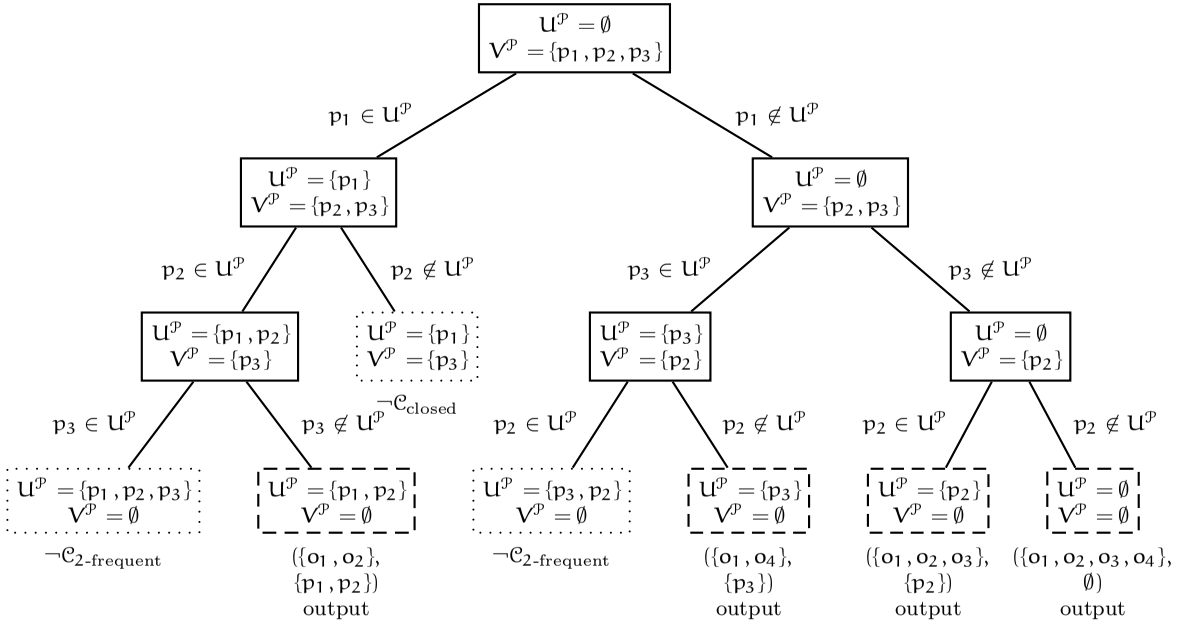



Figure 2: Enumeration of any property $e \in V^{\mathcal{P}}$.

enumeration tree where every property in $V^{\mathcal{P}}$ is refused in $U^{\mathcal{P}}$ (second recursive call of EXTRACT in Figure 1). $U^{\mathcal{P}} \cup V^{\mathcal{P}}$ is the largest possible set of properties that may be output from the current call of EXTRACT. It is considered in the branch of the enumeration tree where every property in $V^{\mathcal{P}}$ is moved to $U^{\mathcal{P}}$ (first recursive call of EXTRACT in Figure 1). $\mathcal{C}_{\text{closed}}$ is tested against the extensions of $U^{\mathcal{P}}$ with every property $s$ that is neither in $U^{\mathcal{P}}$ nor in $V^{\mathcal{P}}$. If $s$ extends $(g(U^{\mathcal{P}}), U^{\mathcal{P}})$ without violating $\mathcal{C}_{\text{connected}}$ (i. e., $g(U) \times \{s\} \not\subseteq \mathcal{B}$) then it extends any pattern recursively discovered from the current call of EXTRACT. Indeed, its objects always are a subset of $g(U^{\mathcal{P}})$. It can be written that $\mathcal{C}_{\text{closed}}$, like the frequency constraint, prunes the search space. This will be further discussed in Section 2.4.1.

A binary tree can represent the enumeration (of the subsets of properties) performed by EXTRACT. A left (resp. right) child relates to the first (resp. second) recursive call where the closed itemsets having (resp. not having) the last enumerated property $e$ will be listed. Figure 2 depicts, in this way, the partition of the search space performed by EXTRACT. Figure 3 is an enumeration tree EXTRACT could traverse when applied on $\mathcal{B}_E$ (represented in Figure 1) with a frequency constraint, $\mathcal{C}_{2\text{-frequent}}$, forcing at least two objects. The previous sentence uses the conditional mood because the function "Choose" in Figure 1 was not specified. Notice that this choice of the property to enumerate does not need to rely on a global ordering of $\mathcal{P}$. For example, the enumeration order is different in the different branches of the tree in Figure 3. In this tree, the dashed leaves are enumeration nodes where a closed itemset is output. The dotted leaves are pruned.

*At every iteration,* any *property can be chosen to enlarge the current candidate pattern.*




| | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|
| $o_1$ | 1 | 1 | 1 |
| $o_2$ | 1 | 1 | 0 |
| $o_3$ | 0 | 1 | 0 |
| $o_4$ | 0 | 0 | 1 |

$\mathcal{B}_E \subseteq \{o_1, o_2, o_3, o_4\} \times \{p_1, p_2, p_3\}$.

Figure 3: Enumeration tree EXTRACT traverses when mining $\mathcal{B}_E$.

An original algorithm, named COBBLER [60], proposes to further exploit the freedom to "choose" the element to enumerate at every recursive call. Because the collection of closed itemsets mathematically is stable by transpose of the binary relation $\mathcal{B}$, COBBLER dynamically switches between the enumeration of properties and objects. The work presented in this thesis reuses this idea that becomes a key for the tractable extraction of closed patterns in $n$-ary relations ($n \geqslant 2$). COBBLER uses a switching condition, between enumerating properties and objects, that is based on a number of enumeration nodes estimated in *simplified* enumeration trees. We believe this is the cause for a time performance that could have been much better. Our approach uses another switching condition.

Figure 1 only aims at presenting, with the notations of this thesis, a simple algorithm based on the fundamental principles shared by most closed itemset extractors. EXTRACT is not a state-of-the-art algorithm. It can be improved. In particular:

- Because $g(U^{\mathcal{P}})$ only loses properties at every recursive call, it could be recursively computed (it would become an additional argument of EXTRACT). This would avoid the scan of all objects, $\mathcal{O}$, at every recursive call.
- When enforcing $\mathcal{C}_{\text{closed}}$, every tested extension could be taken in $\{s \in \mathcal{P} \setminus (U^{\mathcal{P}} \cup V^{\mathcal{P}}) \mid \mathcal{C}_{\text{connected}}(g(U^{\mathcal{P}} \cup V^{\mathcal{P}}), \{s\})\}$ (the other properties in $\mathcal{P} \setminus (U^{\mathcal{P}} \cup V^{\mathcal{P}})$ cannot prevent the closedness of any recursively computed pattern) and this set could be recursively computed too. This would reduce the time spent enforcing $\mathcal{C}_{\text{closed}}$.
- Every property $v' \in V^{\mathcal{P}}$ such that $\mathcal{C}_{\text{connected}}(g(U^{\mathcal{P}}), \{v'\})$ could be directly moved to $U^{\mathcal{P}}$. Indeed, such a property $v'$ must be in every closed itemset recursively discovered from the current call otherwise this itemset would be extensible with $v'$, hence unclosed. This improvement would reduce the enumeration tree and the running time.

Analog improvements will be discussed in the thesis in the more general context of closed pattern mining in arbitrary $n$-ary relations.

## 2 CONSTRAINING THE ITEMSETS

### 2.1 *Why Are Constraints Wanted?*

#### 2.1.1 *Focusing on Relevant Itemsets*

The collection of all closed itemsets in a binary relation $\mathcal{B} \subseteq \mathcal{O} \times \mathcal{P}$ usually is huge. In the worst case it has $2^{\min(|\mathcal{O}|,|\mathcal{P}|)}$ patterns [1]. Stating a minimal number of objects $\gamma \in \mathbb{N}$, under which the closed itemsets are not listed, is a progress. Indeed, conjunctions of properties must apply to enough objects to be statistically relevant. Nevertheless the frequency constraint is not the only relevant constraint. Depending on the actual semantics behind $\mathcal{B}$, many various constraints are useful. For example, in the context of customers buying products, the analyst could be interested in subsets of items:

*Constraints provide a declarative semantics of the* relevant *(closed) itemsets.*

- that are often bought together (by at least $\gamma \in \mathbb{N}$ customers),
- but not too often either (at most $\Gamma \in \mathbb{N}$ customers involved),
- have an average gross profit (for the retailer) above 1€,

1. This maximum derives from the bijection between the closed itemsets and either their sets of objects ($\in 2^{\mathcal{O}}$) or their sets of properties ($\in 2^{\mathcal{P}}$). See Section 1.3.2.

– and with at least one item whose gross profit is below 1€ and at least one item whose gross profit is above 2€.

In this specification of what is a relevant (closed) itemsets, the first part, "often bought together", is a frequency constraint. The second part, "not too often bought together", is an infrequency constraint. The two last parts are more complex constraints. The "average gross profit above 1€" is based on the mean of values that are functions of the items or even of both the items and the customers if the gross profit of a same item varies from one sell in $\mathcal{B}$ to another. Considering all applications that may be of interest, an infinite quantity of relevancy constraints can be imagined.

### 2.1.2 *Reducing Extraction Times*

Obviously, any constraint may be handled as a post-processing step, i. e., the collection of all closed itemsets is extracted and a sub-collection, on which the constraint is satisfied, is filtered afterwards. Unless there are few (unconstrained) closed itemsets of the binary relation $\mathcal{B}$, such an approach is not tractable. Section 2.1.1 mentioned that, at worst, there are $2^{\min(|\mathcal{O}|,|\mathcal{P}|)}$ closed itemsets of $\mathcal{B}$. The problem of listing them all is NP-hard [94] (and so is the verification of a constraint on them all). Beyond the relevancy intrinsically expressed by a constraint, such constraints must be handled at extraction to prune the pattern search space, reduce the extracted collection of closed itemsets (while keeping all those satisfying the constraint) and decrease the running time. Ideally, *only* the closed itemsets satisfying the constraint should be extracted. The frequency constraint is such an ideal constraint, i. e., it is integrated into the extractor that directly returns the closed itemsets having enough objects and only them. In fact, the possible integration of a constraint in an extractor depends on the enumeration principles of this extractor. In other terms, classes of constraints are defined w.r.t. the enumeration principles that allow to integrate them so that the search space is pruned and the running time lowered.

*For some constraints, regions of the search space where it is violated can be avoided, hence* faster *extractions.*

## 2.2 *What is a Constraint?*

A constraint is a propositional function of the patterns, i. e., a statement that uses, as a variable, a pattern $(O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$ and returns either *true* or *false*. For example, the frequency constraint forcing at least $\gamma \in \mathbb{N}$ objects in every extracted closed itemset is formally defined as the propositional function $\mathcal{C}_{\gamma\text{-frequent}}$ below:

$$\mathcal{C}_{\gamma\text{-frequent}}(O, P) \equiv |O| \geqslant \gamma \ .$$

The infrequency constraint forcing at most $\Gamma \in \mathbb{N}$ objects in every closed itemset is:

$$\mathcal{C}_{\Gamma\text{-infrequent}}(O, P) \equiv |O| \leqslant \Gamma \ .$$

The constraint forcing every closed itemset to represent an average gross profit above 1€ (see Section 2.1.1) requires the use of a function $gp : \mathcal{P} \to \mathbb{R}_+$ that returns the gross profit on any item. The constraint is defined as:

$$\mathcal{C}_{\text{avg-gp}\geqslant 1}(O, P) \equiv \frac{\sum_{p \in P} gp(p)}{|P|} \geqslant 1 \ .$$

If the gross profit varies from one sell in $\mathcal{B}$ to another, there is a need for a function $gp' : \mathcal{B} \to \mathbb{R}_+$ that returns the gross profit made when any customer $o \in \mathcal{O}$ bought an item $p \in \mathcal{P}$ ($gp'$ is defined on $O \times P$ because the closed itemset $(O, P)$ satisfies $\mathcal{C}_{\text{connected}}$). The constraint $\mathcal{C}_{\text{avg-gp}\geqslant 1}$ becomes $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$:

$$\mathcal{C}_{\text{avg-gp}'\geqslant 1}(O, P) \equiv \frac{\sum_{(o,p)\in O\times P} gp'(o, p)}{|O \times P|} \geqslant 1 \ .$$

The constraint forcing every closed itemset to contain at least one item whose gross profit is under 1€ and at least one item whose gross profit is above 2€ only accommodates a function gp of the items:

$$\mathcal{C}_{\exists gp\leqslant 1\wedge\exists gp\geqslant 2}(O, P) \equiv \exists(o, o') \in O^2 \mid gp(o) \leqslant 1 \wedge gp(o') \geqslant 2 \ .$$

## 2.3 *Classes of Constraints*

### 2.3.1 *Monotone and Anti-Monotone Constraints*

Like the frequency constraint, several constraints allow, when, at some point of the extraction, they are violated by $(g(P), P)$, to prune every pattern $(g(P'), P')$ with $P \subseteq P'$. These constraints are said anti-monotone:

*If an itemset violates an anti-monotone constraint then itemsets with additional properties violate it too.*

**Definition 3 (Anti-monotonicity)** *A constraint $\mathcal{C}$ is said anti-monotone iff* $\forall(P, P') \in (2^{\mathcal{P}})^2, \Big(P \subseteq P'\Big) \Rightarrow \Big(\mathcal{C}(g(P'), P') \Rightarrow \mathcal{C}(g(P), P)\Big)$.

For example, $\mathcal{C}_{\text{2-frequent}}$ is anti-monotone, i. e., if a pattern is frequent then every pattern having a subset of its properties is frequent as well.

To prune the search space thanks to an anti-monotone constraint $\mathcal{C}_{\text{anti-monotone}}$, a closed itemset extractor that enumerates larger and larger subset of properties (like EXTRACT in Figure 1) considers the smallest possible property set that may be recursively considered from the current call. Using the notations of EXTRACT, $\mathcal{C}_{\text{anti-monotone}}$ is tested on $(g(U^{\mathcal{P}}), U^{\mathcal{P}})$. The recursive computation can safely be aborted if the test fails.

**Example 3** *Consider the execution of* EXTRACT *on $\mathcal{B}_E$ and under the anti-monotone constraint $\mathcal{C}_{\text{2-frequent}}$ (see Figure 3). At every call,* EXTRACT *tests $\mathcal{C}_{\text{2-frequent}}(g(U^{\mathcal{P}}), U^{\mathcal{P}})$. When, at the bottom-left corner of Figure 3,* EXTRACT *is called with $(U^{\mathcal{P}}, V^{\mathcal{P}}) = (\{p_1, p_2\}, \{p_3\})$, this test succeeds (because $|g(\{p_1, p_2\})| = |\{o_1, o_2\}| \geqslant 2$). As a consequence, a closed itemset satisfying $\mathcal{C}_{\text{2-frequent}}$ may recursively be extracted and the computation goes on. Indeed, $(\{o_1, o_2\}, \{p_1, p_2\})$ is discovered in a descendant enumeration node.*

*If an itemset violates a monotone constraint then itemsets with some properties removed violate it too.*

On the contrary, if a pattern $(g(P), P)$ allows, when it violates a constraint, to affirm its violation by every pattern $(g(P''), P'')$ with $P' \subseteq P$, then the constraint is said monotone:

**Definition 4 (Monotonicity)** *A constraint $\mathcal{C}$ is said monotone iff* $\forall(P, P'') \in (2^{\mathcal{P}})^2, \Big(P'' \subseteq P\Big) \Rightarrow \Big(\mathcal{C}(g(P''), P'') \Rightarrow \mathcal{C}(g(P), P)\Big)$.

For example, $\mathcal{C}_{\text{2-infrequent}}$ is monotone, i. e., if a pattern is infrequent then every pattern having a superset of its properties is infrequent as well. Notice that negations of anti-monotone constraints are monotone and vice versa.

**Input:** $(U^{\mathcal{P}}, V^{\mathcal{P}}) \in (2^{\mathcal{P}})^2$
**Output:** Every closed itemset having all properties in $U^{\mathcal{P}}$, potentially some properties in $V^{\mathcal{P}}$ and satisfying $\mathcal{C}_{\text{monotone}} \wedge \mathcal{C}_{\text{anti-monotone}}$
**if** $\mathcal{C}_{\text{anti-monotone}}(g(U^{\mathcal{P}}), U^{\mathcal{P}}) \wedge \mathcal{C}_{\text{monotone}}(g(U^{\mathcal{P}} \cup V^{\mathcal{P}}), U^{\mathcal{P}} \cup V^{\mathcal{P}}) \wedge \forall s \in \mathcal{P} \setminus (U^{\mathcal{P}} \cup V^{\mathcal{P}}), g(U^{\mathcal{P}}) \times \{s\} \not\subseteq \mathcal{B}$ **then**
  **if** $V^{\mathcal{P}} = \emptyset$ **then**
    **output**$((g(U^{\mathcal{P}}), U^{\mathcal{P}}))$
  **else**
    Choose $e \in V^{\mathcal{P}}$
    EXTRACT++$(U^{\mathcal{P}} \cup \{e\}, V^{\mathcal{P}} \setminus \{e\})$
    EXTRACT++$(U^{\mathcal{P}}, V^{\mathcal{P}} \setminus \{e\})$
  **end if**
**end if**

Figure 4: The EXTRACT++ closed itemset extractor (under any conjunction of monotone and anti-monotone constraints).

To prune the search space thanks to a monotone constraint $\mathcal{C}_{\text{monotone}}$ and without changing the enumeration, $(g(U^{\mathcal{P}} \cup V^{\mathcal{P}}), U^{\mathcal{P}} \cup V^{\mathcal{P}})$ must be considered, i. e., the pattern with the largest possible property set (every property in $V^{\mathcal{P}}$ is accepted in $U^{\mathcal{P}}$). If this pattern satisfies $\mathcal{C}_{\text{monotone}}$ then the enumeration must go on, otherwise the search space can be pruned.

**Example 4** *Consider the execution of* EXTRACT *on* $\mathcal{B}_E$ *and under the monotone constraint* $\mathcal{C}_{\textit{2-infrequent}}$. *When* EXTRACT *is called with* $(U^{\mathcal{P}}, V^{\mathcal{P}}) = (\emptyset, \{p_2\})$, $\mathcal{C}_{\textit{2-infrequent}}(g(U^{\mathcal{P}} \cup V^{\mathcal{P}}), U^{\mathcal{P}} \cup V^{\mathcal{P}})$ *is false. As a consequence, recursive calls would not allow the extraction of any closed itemset satisfying* $\mathcal{C}_{\textit{2-infrequent}}$ *and the search space can be pruned, i. e., the two enumeration nodes in the bottom-right corner of Figure 3 are not to be traversed. Indeed, neither* $(\{o_1, o_2, o_3\}, \{p_2\})$ *nor* $(\{o_1, o_2, o_3, o_4\}, \emptyset)$ *(that are extracted when* $\mathcal{C}_{\textit{2-infrequent}}$ *is not enforced) satisfies* $\mathcal{C}_{\textit{2-infrequent}}$.

Obviously any conjunction of monotone (resp. anti-monotone) constraints is a monotone (resp. anti-monotone) constraint. As a consequence, any conjunction of monotone and anti-monotone constraints can be reduced to a conjunction $\mathcal{C}_{\text{monotone}} \wedge \mathcal{C}_{\text{anti-monotone}}$. The extraction of every closed itemset under $\mathcal{C}_{\text{monotone}} \wedge \mathcal{C}_{\text{anti-monotone}}$ is achieved by the algorithm EXTRACT++ in Figure 4. This extractor generalizes EXTRACT and enforces $\mathcal{C}_{\text{monotone}} \wedge \mathcal{C}_{\text{anti-monotone}}$ as explained in the previous paragraphs.

*If the itemset with the smallest (resp. greatest) possible set of properties violates an anti-monotone (resp. monotone) constraint, the search space can be pruned.*

The duality between $(g(U^{\mathcal{P}}), U^{\mathcal{P}})$ and $(g(U^{\mathcal{P}} \cup V^{\mathcal{P}}), U^{\mathcal{P}} \cup V^{\mathcal{P}})$ was understood and exploited in [16]. The presented algorithm, DualMiner, extracts (not necessarily closed) itemset by simultaneously taking advantage of both monotone and anti-monotone constraints to prune the search space. The work presented in this thesis extends the class of constraints DualMiner can efficiently handle but the same duality is exploited.

A constraint that is neither monotone nor anti-monotone is harder to integrate into the extraction (to reduce extraction times). Automatically pre-processing such a constraint to turn it into a Boolean expression of monotone and anti-monotone constraints is a key to a theory of data-mining but remains an open problem. See [70] for a seminal paper on the subject and [55] for a specific case study (though, in both cases, the considered patterns are strings rather than itemsets). Even if a

relaxation of the tough constraint is strictly weaker than the original one, its integration may greatly reduce the extraction times and the original constraint can then filter the supernumerary patterns in a post-processing step.

[58] is a key article in the definition of classes of constraints for itemset mining. It introduced the concepts anti-monotonicity and succinctness (defined in the next section). The enforcement of monotone constraints along the extraction was achieved later [34].

### 2.3.2 *Succinct Constraints*

Succinct constraints reduce the search space before the extraction starts, i. e., the candidate pattern set is not $2^{\mathcal{O}} \times 2^{\mathcal{P}}$ anymore. In other terms, the satisfaction of a succinct constraint does not depend on the binary relation $\mathcal{B}$ and redefines what is a syntactically relevant pattern. The enforcement of a succinct constraint is handled by a modified enumeration that only generates the candidate patterns that satisfy the constraint. The considered enumeration is that of most closed itemset extractors, i. e., the enumeration of the subsets of one property domain (see Section 1.3.2), traditionally the properties. As a consequence succinct constraints relate to selections of relevant elements in $2^{\mathcal{P}}$.

*A succinct constraint defines "positive" and "negative" subsets of properties. It forces the properties of a closed itemset to be included in one "positive" set and not included in any "negative" set.*

**Definition 5 (Succinctness)** *A constraint $\mathcal{C}$ is said succinct iff there exists $(k, l) \in \mathbb{N}^2$, $(P_i)_{i=1..k} \in (2^{\mathcal{P}})^k$ and $(Q_j)_{j=1..l} \in (2^{\mathcal{P}})^l$ such that the patterns $(g(P), P)$ satisfying $\mathcal{C}$ are those with $P \in \cup_{i=1}^{k} 2^{P_i} \setminus \cup_{j=1}^{l} 2^{Q_j}$.*

For example, $\mathcal{C}_{\exists gp \leqslant 1 \wedge \exists gp \geqslant 2}$ (defined in Section 2.2) is succinct. Indeed the patterns $(g(P), P)$ satisfying it are those with $P \in 2^{\mathcal{P}} \setminus (2^{\sigma_{gp \leqslant 2}(\mathcal{P})} \cup 2^{\sigma_{gp \geqslant 1}(\mathcal{P})})$, where $\sigma_{gp \leqslant 2}(\mathcal{P}) = \{p \in \mathcal{P} \mid gp(p) \leqslant 2\}$ (the items with a gross profit below 2€) and $\sigma_{gp \geqslant 1}(\mathcal{P}) = \{p \in \mathcal{P} \mid gp(p) \geqslant 1\}$ (the items with a gross profit above 1€).

The succinct constraints are, historically, handled by a modified enumeration. However, a closed itemset extractor, such as EXTRACT++, able to enforce conjunctions of monotone and anti-monotone constraints, can handle any succinct constraint. Indeed, a succinct constraint $\mathcal{C}$, as defined above, is equivalent to $\mathcal{C}_{\text{succinct anti-monotone}} \wedge \mathcal{C}_{\text{succinct monotone}}$, where:

- $\mathcal{C}_{\text{succinct anti-monotone}}(O, P) \equiv P \in \cup_{i=1}^{k} 2^{P_i}$ is anti-monotone;
- $\mathcal{C}_{\text{succinct monotone}}(O, P) \equiv P \notin \cup_{j=1}^{l} 2^{Q_j}$ is monotone.

### 2.3.3 *Convertible Constraints*

Convertible constraints were introduced in [62]. This class of constraints lies on abandoning the freedom to enumerate any remaining property (in $V^{\mathcal{P}}$ according to the notations of Figure 4) at any recursive call. Ordering the properties makes, on every branch of the enumeration tree, the sequence of larger and larger subsets of properties deterministic. Some constraints, which are not monotone (resp. anti-monotone), can become monotone (resp. anti-monotone) for a particular order (i. e., a particular Choose function in Figure 4) of the enumerated properties. In other terms, the properties are enumerated in a well-chosen order that makes the constraint monotone (resp. anti-monotone) on every branch of the enumeration tree. Such constraints are called convertible monotone (resp. anti-monotone).

*The convertibility is the monotonicity (resp. anti-monotonicity) for an arbitrary total order on the subsets of properties.*

**Definition 6 (Convertible monotonicity)** *A constraint $\mathcal{C}$ is said convertible monotone iff there exists a total order $\preceq$ of the properties in $\mathcal{P}$ such*

*that* $\forall(P, P'') \in (2^{\mathcal{P}})^2, \Big(P'' \subseteq P \wedge \forall(p'', p) \in P'' \times P \setminus P'', p'' \preceq p\Big) \Rightarrow \Big(\mathcal{C}(g(P''), P'') \Rightarrow \mathcal{C}(g(P), P)\Big)$.

**Definition 7 (Convertible anti-monotonicity)** *A constraint* $\mathcal{C}$ *is said convertible anti-monotone iff there exists a total order* $\preceq$ *of the properties in* $\mathcal{P}$ *such that* $\forall(P, P') \in (2^{\mathcal{P}})^2, \Big(P \subseteq P' \wedge \forall(p, p') \in P \times P' \setminus P, p \preceq p'\Big) \Rightarrow \Big(\mathcal{C}(g(P'), P') \Rightarrow \mathcal{C}(g(P), P)\Big)$.

$\mathcal{C}_{\text{avg-gp}\geqslant 1}$ is an example of a convertible monotone constraint. Indeed, by enumerating the items by increasing gross profit, i. e., $\forall(p, p') \in \mathcal{P}^2, p \preceq p' \Leftrightarrow gp(p) \leqslant gp(p')$, $\mathcal{C}_{\text{avg-gp}\geqslant 1}$ is monotone on every branch of the enumeration tree, i. e., once satisfied at some point of the enumeration tree it remains true in the whole sub-tree recursively built from this point. Indeed, the average only increases when a greater value (than those previously considered) is added. Notice that $\mathcal{C}_{\text{avg-gp}\geqslant 1}$ is convertible anti-monotone too (items ordered by decreasing gross profit). However not every convertible monotone constraint is convertible anti-monotone as well.

When the analyst specifies a convertible constraint, the Choose function of Figure 4 must be the one that always enumerates the smallest property w.r.t. the order $\preceq$ related to the constraint, i. e., Choose returns $e \in V^{\mathcal{P}}$ such that $\forall f \in V^{\mathcal{P}}, e \preceq f$. Then, the constraint is treated as a monotone constraint if it is convertible monotone; as an anti-monotone constraint if it is convertible anti-monotone. This way of handling convertible constraints has two drawbacks:

- It is impossible to enforce several convertible constraints unless they rely on the same order of properties (in other terms, the convertibility is not preserved by conjunction);
- Because the Choose function is imposed by the convertible constraint, it cannot be defined so that the extraction time is heuristically lowered.

*Convertibility fixes the search space traversal. Losing this degree of freedom prevents some performance improvements.*

When no convertible constraint is specified, a popular heuristic lowering the extraction time is the enumeration of properties by increasing density, i. e., Choose returns a property $e \in V^{\mathcal{P}}$ minimizing the expression below:

$$|(\mathcal{O} \times \{e\}) \cap \mathcal{B}\}| \ .$$

This heuristic was rediscovered in [43] for (not necessarily closed) itemset mining. However, it was already used in [76] for the older related problem of identifying prime implicants in CNF propositional logic expressions. The algorithms presented in this thesis generalizes this heuristic when it comes to choose an element to enumerate.

### 2.3.4 *Loose Anti-Monotone Constraints*

Loose anti-monotonicity was introduced in [12]. If a candidate pattern $(g(P), P)$ with $|P| \geqslant 2$ satisfies a loose anti-monotone constraint then one of its properties $p$ can be removed and $(g(P \setminus \{p\}), P \setminus \{p\})$ satisfies it too.

*If an itemset satisfies a loose anti-monotone constraint then at least one itemset with one property removed satisfies it too.*

**Definition 8 (Loose anti-monotonicity)** *A constraint* $\mathcal{C}$ *is said loose anti-monotone iff* $\forall P \subseteq \mathcal{P}, \Big(|P| \geqslant 2 \wedge \mathcal{C}(g(P), P)\Big) \Rightarrow \exists p \in P \mid \mathcal{C}(g(P \setminus \{p\}), P \setminus \{p\})$.

**Input:** $L \subseteq 2^{\mathcal{P}}$
**Output:** Every closed itemset having a superset of the properties of any element in $L$ and satisfying $\mathcal{C}_{\text{anti-monotone}} \wedge \mathcal{C}_{\text{LAM}}$
$L' \leftarrow \emptyset$
**for all** $U^{\mathcal{P}} \in L$ **do**
  **if** $\mathcal{C}_{\text{anti-monotone}}(g(U^{\mathcal{P}}), U^{\mathcal{P}}) \wedge \mathcal{C}_{\text{LAM}}(g(U^{\mathcal{P}}), U^{\mathcal{P}})$ **then**
    **if** $\forall s \in \mathcal{P} \setminus U^{\mathcal{P}}, g(U^{\mathcal{P}}) \times \{s\} \not\subseteq \mathcal{B}$ **then**
      **output**$((g(U^{\mathcal{P}}), U^{\mathcal{P}}))$
    **end if**
    **for all** $e \in \mathcal{P} \setminus U^{\mathcal{P}}$ **do**
      $L' \leftarrow L' \cup \{U^{\mathcal{P}} \cup \{e\}\}$
    **end for**
  **end if**
  $L \leftarrow L \setminus \{U^{\mathcal{P}}\}$
**end for**
**if** $L' \neq \emptyset$ **then**
  EXTRACT#$(L')$
**end if**

Figure 5: The EXTRACT# closed itemset extractor (under any loose anti-monotone constraint).

Consider the constraint $\mathcal{C}_{\text{std-gp}\leqslant 1}$ specifying a standard deviation below 1€ for the gross profits of the items involved in a relevant pattern.

$$\mathcal{C}_{\text{std-gp}\leqslant 1}(O, P) \equiv \sqrt{\frac{\sum_{p \in P}(gp(p) - \mu)^2}{|P|}} \leqslant 1, \text{ where } \mu = \sum_{p' \in P} \frac{gp(p')}{|P|} \ .$$

This constraint is loose anti-monotone: if a pattern $(g(P), P)$ satisfies $\mathcal{C}_{\text{std-gp}\leqslant 1}$ then there exists an item $p \in P$ such that $(g(P \setminus \{p\}), P \setminus \{p\})$ satisfies $\mathcal{C}_{\text{std-gp}\leqslant 1}$ as well. Indeed, removing the item with the gross profit that is (one of) the farthest from $\mu$ always decreases the standard deviation. More formally, this item $p \in P$ is such that $\forall p' \in P, |gp(p') - \mu| \leqslant |gp(p) - \mu|$.

With a loose anti-monotone constraint $\mathcal{C}_{\text{LAM}}$ the candidate patterns is traversed breadth-first. In this way, it is natural to check the existence of a pattern with a subset of the properties and satisfying $\mathcal{C}_{\text{LAM}}$. Figure 5 gives a simple closed itemset extractor EXTRACT# that handles such a loose anti-monotone constraint $\mathcal{C}_{\text{LAM}}$ (and an anti-monotone constraint $\mathcal{C}_{\text{anti-monotone}}$). It is initially called with the first level of the enumeration tree, i. e., $L = \{\{\emptyset\}\}$. Notice that, whatever $U^{\mathcal{P}} \subseteq \mathcal{P}$, any property $e \in \mathcal{P} \setminus U^{\mathcal{P}}$ may be appended to it. As a consequence, there is no need to store the search space $V^{\mathcal{P}}$ anymore. However the number of generated patterns potentially is much larger and duplicates may be considered. To address this issue, $L'$ only keeps one occurrence of identical sets of properties.

*An enumeration only taking advantage of loose anti-monotonicity is far less efficient.*

In the literature, the few proposals that handle a loose anti-monotone constraint prune the search space in a more complex way than EXTRACT#. To avoid the costly change in candidate enumeration (i. e., to maintain a search space $V^{\mathcal{P}} \subseteq \mathcal{P}$), they assume the patterns are always mined under an anti-monotone constraint too. In this way, pruning thanks to the loose anti-monotone constraint can be done on top of the classical enumeration, where the anti-monotone constraint is fully exploited (i. e., contrary to EXTRACT#, if $(g(P'), P')$ is considered then *every* $(g(P), P)$

**Input:** $U^{\mathcal{P}} \subseteq \mathcal{P}$
**Output:** Every closed itemset having all properties in $U^{\mathcal{P}}$ and satisfying $\mathcal{C}_{\text{anti-monotone}} \wedge \mathcal{C}_{\text{LAM}}$
**if** $\mathcal{C}_{\text{anti-monotone}}(g(U^{\mathcal{P}}), U^{\mathcal{P}}) \wedge \mathcal{C}_{\text{LAM}}(g(U^{\mathcal{P}}), U^{\mathcal{P}})$ **then**
  **if** $\forall s \in \mathcal{P} \setminus U^{\mathcal{P}}, g(U^{\mathcal{P}}) \times \{s\} \not\subseteq \mathcal{B}$ **then**
    **output**$((g(U^{\mathcal{P}}), U^{\mathcal{P}}))$
  **end if**
  **for all** $e \in \mathcal{P} \setminus U^{\mathcal{P}}$ **do**
    **if** $\max(U^{\mathcal{P}} \cup \{e\}) = e$ **then**
      EXTRACT##$(U^{\mathcal{P}} \cup \{e\})$
    **end if**
  **end for**
**end if**

Figure 6: The EXTRACT## closed itemset extractor (under a loose anti-monotone constraint with a known max function).

with $P \subseteq P'$ satisfies the anti-monotone constraint). Moreover, they can enforce monotone constraints. Nevertheless, their enumeration remains breadth-first, hence potential space issues (dominated by the storage of the greatest level denoted L in Figure 5). A more fundamental drawback of the class of loose anti-monotone constraints is that, like that of convertible constraints, it is not stable under conjunction. Thus, it is generally impossible to efficiently enforce conjunctions of loose anti-monotone constraints.

Notice that any convertible anti-monotone constraint is loose anti-monotone. Indeed, a consequence of the definition of a convertible anti-monotone constraint $\mathcal{C}$ is that there exists a total order $\preceq$ of the properties such that $\forall P \subseteq \mathcal{P}, \mathcal{C}(g(P), P) \Rightarrow \mathcal{C}(g(P \setminus \{p\}), P \setminus \{p\})$, where $p \in P$ is the greatest property in $P$ w.r.t. $\preceq$, i. e., $\forall p' \in P, p' \preceq p$. This statement still holds if the greatest property w.r.t. $\preceq$ is defined locally (i. e., w.r.t. a specific $P \subseteq \mathcal{P}$) rather than globally, i. e., if there exists a function $\max : 2^{\mathcal{P}} \to \mathcal{P}$ that takes as input any $P \subseteq \mathcal{P}$ and such that $\mathcal{C}(g(P), P) \Rightarrow \mathcal{C}(g(P \setminus \{\max(P)\}), P \setminus \{\max(P)\})$. The related class of constraints actually is the loose anti-monotonicity. Nevertheless, such a reformulation presupposes an a priori knowledge of the function max (and the ability to quickly compute it). It is often the case. For example the max function related to $\mathcal{C}_{\text{std-gp}\leqslant 1}$ is:

$$\max : P \to \text{argmax}_{p \in \mathcal{P}}(|gp(p) - \mu|), \text{ where } \mu = \sum_{p' \in P} \frac{gp(p')}{|P|} \quad .$$

More precisely, there is a need for an additional arbitrary order on the properties such that max returns a unique element (for example the smallest w.r.t. this arbitrary order) among those that minimize $|gp(p) - \mu|$.

*A depth-first candidate enumeration is possible for some loose anti-monotone constraints.*

Integrating to the algorithm the knowledge of the function max, related to the loose anti-monotone constraint $\mathcal{C}_{\text{LAM}}$, helps the extraction. In particular, it becomes possible to traverse the search space depth-first by only considering as a child a pattern where the newly added element is the maximal one. This technique is called *reverse search* [3]. Figure 6 gives such an extractor, namely EXTRACT##.

| Flexible constraints $\mathcal{F}$ | Operators | Argument(s) |
|---|---|---|
| $\mathcal{C}_1 \theta \mathcal{C}_2$ | $\theta \in \{\wedge, \vee\}$ | $(\mathcal{C}_1, \mathcal{C}_2) \in \mathcal{F}^2$ |
| $\neg \mathcal{C}$ | - | $\mathcal{C} \in \mathcal{F}$ |
| $e_1 \theta e_2$ | $\theta \in \{<, \leqslant\}$ | $(e_1, e_2) \in \mathcal{E}^2$ |
| $X_1 \theta X_2$ | $\theta \in \{\subset, \subseteq\}$ | $(X_1, X_2) \in \mathcal{S}^2$ |
| constant $b$ | - | $b \in \{\text{true,false}\}$ |
| **Aggregate expressions $\mathcal{A}$** | **Operators** | **Argument(s)** |
| $a_1 \theta a_2$ | $\theta \in \{+, -, \times, /\}$ | $(a_1, a_2) \in \mathcal{A}^2$ |
| $\lvert X \rvert$ | - | $X \in \mathcal{S}$ |
| $\theta_{x \in X} val(x)$ | $\theta \in \{\sum, \max, \min\}$ | $X \in \mathcal{S}$ |
| constant $r$ | - | $r \in \mathbb{R}_+$ |
| **Syntactic expressions $\mathcal{S}$** | **Operators** | **Argument(s)** |
| $X_1 \theta X_2$ | $\theta \in \{\cup, \cap, \setminus\}$ | $(X_1, X_2) \in \mathcal{S}^2$ |
| $g(X)$ | - | $X \in \mathcal{S}$ |
| variable $P$ | - | $P \subseteq \mathcal{P}$ |
| constant $P$ | - | $P \subseteq \mathcal{P}$ |

Table 2: Flexible constraints.

### 2.3.5 *Flexible constraints*

The flexible constraints were introduced in [78]. They are recursively defined.

*A fixed set of primitives recursively defines flexibility.*

**Definition 9 (Flexibility)** *A constraint $\mathcal{C}$ is said flexible if it is in $\mathcal{F}$ that Table 2 recursively defines.*

Contrary to convertible or loose anti-monotone constraints, the flexible constraints are, by definition (the first line defining $\mathcal{F}$ in Table 2), stable under conjunction (and disjunction). The class of flexible constraints is very broad. Although it was not previously shown, even $\mathcal{C}_{\text{std-gp}\leqslant 1}$ is flexible. With its expression based on the definition of the variance, it does not look so:

$$\mathcal{C}_{\text{std-gp}\leqslant 1}(O, P) \equiv \sqrt{\frac{\sum_{p \in P}(gp(p) - \mu)^2}{|P|}} \leqslant 1, \text{ where } \mu = \sum_{p' \in P} \frac{gp(p')}{|P|} \ .$$

However, the theorem of König-Huyghens proves the equivalence of this definition with the following:

$$\mathcal{C}_{\text{std-gp}\leqslant 1}(O, P) \equiv \sqrt{\frac{\sum_{p \in P} gp(p)^2}{|P|} - \mu^2} \leqslant 1, \text{ where } \mu = \sum_{p' \in P} \frac{gp(p')}{|P|} \ .$$

Both sides of the inequality are positive. As a consequence, they can be raised to the power 2 and an equivalent expression is obtained:

$$\mathcal{C}_{\text{std-gp}\leqslant 1}(O, P) \equiv \frac{\sum_{p \in P} gp(p)^2}{|P|} - \mu^2 \leqslant 1, \text{ where } \mu = \sum_{p' \in P} \frac{gp(p')}{|P|} \ .$$

This time, a flexible constraint is recognized. Here is how to recursively build it from the primitives in Table 2 (the first level lists primitives; then,

every expression has at least one argument taken from the previous level):

1. – variable $P_1 \subseteq \mathcal{P}$ ($\in \mathcal{S}$).
   – variable $P_2 \subseteq \mathcal{P}$ ($\in \mathcal{S}$).
   – variable $P_3 \subseteq \mathcal{P}$ ($\in \mathcal{S}$).
   – variable $P_4 \subseteq \mathcal{P}$ ($\in \mathcal{S}$).
   – constant $1 \in \mathbb{R}_+$ ($\in \mathcal{A}$).
2. – $\sum_{x \in P_1} val(x)$, with $val : x \to gp(x)^2$ ($\in \mathcal{A}$).
   – $|P_2|$ ($\in \mathcal{A}$).
   – $\sum_{x \in P_3} val(x)$, with $val : x \to gp(x)$ ($\in \mathcal{A}$).
   – $|P_4|$ ($\in \mathcal{A}$).
3. – $\left(\sum_{x \in P_1} val(x)\right)/|P_2|$, with $val : x \to gp(x)^2$ ($\in \mathcal{A}$).
   – $\mu = \left(\sum_{x \in P_3} val(x)\right)/|P_4|$, with $val : x \to gp(x)$ ($\in \mathcal{A}$).
4. – $\mu^2 = \mu \times \mu$ ($\in \mathcal{A}$).
5. – $\left(\left(\sum_{x \in P_1} val(x)\right)/|P_2|\right) - \mu^2$, with $val : x \to gp(x)^2$ ($\in \mathcal{A}$).
6. – $\left(\left(\sum_{x \in P_1} val(x)\right)/|P_2|\right) - \mu^2 \geqslant 1$, with $val : x \to gp(x)^2$ ($\in \mathcal{F}$, i. e., $\mathcal{C}_{\text{std-gp}\leqslant 1} \in \mathcal{F}$).

### 2.3.6 *Primitive-Based/Piecewise (Anti)-Monotone Constraints*

In [79], the authors notice that the flexible constraints only are a subset of a more general class constraints, which are analogously enforced. These constraints, namely the primitive-based constraints, are those recursively defined by a list of primitives (such as Table 2) that are increasing or decreasing w.r.t. each of their arguments (the others considered fixed). A constraint is increasing or decreasing w.r.t. the order $\{\text{false} \prec \text{true}\}$. An aggregate expression is increasing or decreasing w.r.t. the order $\leqslant$ on real numbers. Finally, a syntactic expression is increasing or decreasing w.r.t. the order $\subseteq$ on sets. Every primitive in Table 2 is increasing or decreasing on each of their arguments. As a consequence, the flexible constraints are primitive based. The converse does not hold.

*Primitive-based constraints are recursively defined from arbitrary primitives that either increase or decrease w.r.t. each of their arguments.*

Consider the constraint $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$ defined in Section 2. It is not flexible because the function $gp'$ is defined over $\mathcal{B} \subseteq \mathcal{O} \times \mathcal{P}$ and Table 2 does not allow such a multivariable calculus. Nevertheless, $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$ is primitive-based. Consider, the following aggregate expression:

$$\sum_{(x_1,x_2) \in X^2} val(x_1, x_2) \ (X \in \mathcal{S} \text{ and } val : (\mathcal{O} \cup \mathcal{P})^2 \to \mathbb{R}_+)$$

This primitive can be added to Table 2 to define a set of constraints $\mathcal{F}' \supset \mathcal{F}$. Every constraint in $\mathcal{F}'$ is primitive based because, like the primitives in Table 2, the additional aggregate expression is increasing with $X$, i. e., $\forall (X_1, X_2) \in (2^{\mathcal{O} \cup \mathcal{P}})^2$, $\left(X_1 \subseteq X_2 \Rightarrow \sum_{(x_1,x_2) \in X_1^2} val(x_1, x_2) \leqslant \sum_{(x_1,x_2) \in X_2^2} val(x_1, x_2)\right)$. It can now be proved that $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$ is in $\mathcal{F}'$, hence primitive-based:

1. – variable $P_1 \subseteq \mathcal{P}$ ($\in \mathcal{S}$).
   – variable $P_2 \subseteq \mathcal{P}$ ($\in \mathcal{S}$).
   – constant $1 \in \mathbb{R}_+$ ($\in \mathcal{A}$).
2. – $g(P_1)$ ($\in \mathcal{S}$).

   - $g(P_2)$ ($\in \mathcal{S}$).
   - $|P_2|$ ($\in \mathcal{A}$).
3. - $g(P_1) \cup P_1$ ($\in \mathcal{S}$).
   - $|g(P_2)|$ ($\in \mathcal{A}$).
4. - $\sum_{(x_1,x_2)\in(g(P_1)\cup P_1)^2} val(x_1,x_2)$,
   with $val : (x_1,x_2) \rightarrow \begin{cases} gp'(x_1,x_2) \text{ if } (x_1,x_2) \in \mathcal{B} \\ 0 \text{ otherwise} \end{cases}$ ($\in \mathcal{A}$).
   - $|P_2| \times |g(P_2)|$ ($\in \mathcal{A}$).
5. - $\Big(\sum_{(x_1,x_2)\in(g(P_1)\cup P_1)^2} val(x_1,x_2)\Big) / \Big(|P_2| \times |g(P_2)|\Big)$,
   with $val : (x_1,x_2) \rightarrow \begin{cases} gp'(x_1,x_2) \text{ if } (x_1,x_2) \in \mathcal{B} \\ 0 \text{ otherwise} \end{cases}$ ($\in \mathcal{A}$).
6. - $\Big(\sum_{(x_1,x_2)\in(g(P_1)\cup P_1)^2} val(x_1,x_2)\Big) / \Big(|P_2| \times |g(P_2)|\Big) < 1$,
   with $val : (x_1,x_2) \rightarrow \begin{cases} gp'(x_1,x_2) \text{ if } (x_1,x_2) \in \mathcal{B} \\ 0 \text{ otherwise} \end{cases}$ ($\in \mathcal{F}'$).
7. - $\neg\Big(\Big(\sum_{(x_1,x_2)\in(g(P_1)\cup P_1)^2} val(x_1,x_2)\Big) / \Big(|P_2| \times |g(P_2)|\Big) < 1\Big)$,
   with $val : (x_1,x_2) \rightarrow \begin{cases} gp'(x_1,x_2) \text{ if } (x_1,x_2) \in \mathcal{B} \\ 0 \text{ otherwise} \end{cases}$ ($\in \mathcal{F}'$, i. e., $\mathcal{C}_{\text{avg-gp}'\geqslant 1} \in \mathcal{F}'$).

As illustrated above, it may be tedious to prove that a particular constraint is primitive-based. Indeed, the required primitives must be intuitively found, proved increasing or decreasing w.r.t. each of their arguments (the others considered fixed) and must be combined so that an equivalent expression of the constraint is found (e. g., the last expression proving $\mathcal{C}_{\text{avg-gp}'\geqslant 1} \in \mathcal{F}'$ is equivalent to $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$ but is not syntactically identical).

Piecewise (anti)-monotone constraints apply to patterns that hold in arbitrary $n$-ary relations ($n \geqslant 2$). Although it was not understood earlier, this class of constraints actually is, when restricted to binary contexts, that of primitive-based constraints. At first sight, this equality is not obvious because the piecewise (anti)-monotonicity was defined so that it is quite easy to prove a particular constraint belongs to this class. The basic common principle is the monotonicity/anti-monotonicity per argument. To introduce the piecewise (anti)-monotonicity, this only needs to be defined on constraints.

*Piecewise (anti)-monotone and primitive-based are synonymous.*

**Definition 10 ((Anti)-monotonicity per argument)** *A constraint $\mathcal{C}$ is said monotone (resp. anti-monotone) w.r.t. the $i^{th}$ argument iff it is monotone (resp. anti-monotone) when all its arguments but the $i^{th}$ are considered constant.*

The definition of piecewise (anti)-monotonicity relies on attributing a separate argument to every occurrence of every variable and, then, proving that the obtained constraint is (anti)-monotone w.r.t. each of its arguments. It can be written that the definition of piecewise (anti)-monotonicity is the top-down counterpart of the bottom-up definition of primitive-based constraints.

*Piecewise (anti)-monotone constraints are either monotone or anti-monotone w.r.t. each occurrence of a variable in their expressions.*

**Definition 11 (Piecewise (anti)-monotonicity)** *A constraint $\mathcal{C}$ is piecewise (anti)-monotone iff the rewritten constraint $\mathcal{C}'$, attributing a separate*

*argument to every occurrence of every variable in the expression of $\mathcal{C}$, is (anti)-monotone w.r.t. each of its arguments.*

Given a particular constraint, the definition of piecewise (anti)-monotonicity makes it easier to prove it primitive-based/piecewise (anti)-monotone. Consider $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$ again. Its expression is:

$$\mathcal{C}_{\text{avg-gp}'\geqslant 1}(O, P) \equiv \frac{\sum_{(o,p)\in O\times P} gp'(o,p)}{|O \times P|} \geqslant 1 \ .$$

By attributing a separate argument to every occurrence of $O$ and $P$, $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$ is rewritten as follows:

$$\mathcal{C}'_{\text{avg-gp}'\geqslant 1}(O_1, O_2, P_1, P_2) \equiv \frac{\sum_{(o,p)\in O_1\times P_1} gp'(o,p)}{|O_2 \times P_2|} \geqslant 1 \ .$$

$\mathcal{C}'_{\text{avg-gp}'\geqslant 1}$ is monotone on its first and third arguments. It is anti-monotone on its second and fourth arguments. As a consequence, $\mathcal{C}_{\text{avg-gp}'\geqslant 1}$ is, by definition, primitive-based/piecewise (anti)-monotone. In the remaining of this thesis, the term "piecewise (anti)-monotone" is preferred because its definition translates the approach the thesis follows: useful constraints are first identified, then proved to be part of the class of constraints the proposed algorithms efficiently handle. The definition of primitive-based constraints translates the reversed approach, i. e., designing a SQL-like language to query patterns.

EXTRACT*, in Figure 7, extracts every closed itemset under a piecewise (anti)-monotone constraint $\mathcal{C}_{\text{P(A)M}}$. Apart from the enforcement of $\mathcal{C}_{\text{P(A)M}}$, it is identical to EXTRACT++ (in Figure 4). To enforce $\mathcal{C}_{\text{P(A)M}}$, EXTRACT* needs the constraint $\mathcal{C}'_{\text{P(A)M}}$ where every occurrence of every variable in $\mathcal{C}_{\text{P(A)M}}$ is attributed a separate argument on which $\mathcal{C}'_{\text{P(A)M}}$ is (anti)-monotone. At any call, every argument on which $\mathcal{C}'_{\text{P(A)M}}$ is monotone is instantiated:

- if the argument ranges in $2^{\mathcal{O}}$, with the largest object set that may be considered from the current call, i. e., $g(U^{\mathcal{P}})$ ($g$ is decreasing w.r.t. its argument and $U^{\mathcal{P}}$ is the smallest property set that may be considered from the current call);
- if the argument ranges in $2^{\mathcal{P}}$, with the largest property set that may be considered from the current call, i. e., $U^{\mathcal{P}} \cup V^{\mathcal{P}}$.

Dually, every argument on which $\mathcal{C}'_{\text{P(A)M}}$ is anti-monotone is instantiated:

- if the argument ranges in $2^{\mathcal{O}}$, with the smallest object set that may be considered from the current call, i. e., $g(U^{\mathcal{P}} \cup V^{\mathcal{P}})$ ($g$ is decreasing w.r.t. its argument and $U^{\mathcal{P}} \cup V^{\mathcal{P}}$ is the largest property set that may be considered from the current call);
- if the argument ranges in $2^{\mathcal{P}}$, with the smallest property set that may be considered from the current call, i. e., $U^{\mathcal{P}}$.

*A piecewise (anti)-monotone constraint is instantiated for some variables with the smallest, for the others with the greatest, possible set of properties. If violated the search space can be pruned.*

If this instantiation is false then no pattern recursively considered from the current call satisfies $\mathcal{C}_{\text{P(A)M}}$ (and EXTRACT* prunes the search space). This implication directly follows the definition of piecewise (anti)-monotonicity and the fact that $g(U^{\mathcal{P}} \cup V^{\mathcal{P}})$ and $U^{\mathcal{P}}$ are the smallest object and property sets that may be considered from the current call; $g(U^{\mathcal{P}})$ and $U^{\mathcal{P}} \cup V^{\mathcal{P}}$ the largest.

**Example 5** *Consider the execution of* EXTRACT* *on $\mathcal{B}_E$ and under the piecewise (anti)-monotone constraint $\mathcal{C}_{avg\text{-}gp'\geqslant 1}$, where Table 3 defines the function*

**Input:** $(U^{\mathcal{P}}, V^{\mathcal{P}}) \in (2^{\mathcal{P}})^2$
**Output:** Every closed itemset having all properties in $U^{\mathcal{P}}$, potentially some properties in $V^{\mathcal{P}}$ and satisfying $\mathcal{C}_{\text{P(A)M}}$
**if** $\mathcal{C}'_{\text{P(A)M}}$ is satisfied when instantiated as detailed in the text $\wedge \forall s \in \mathcal{P} \setminus (U^{\mathcal{P}} \cup V^{\mathcal{P}}), g(U^{\mathcal{P}}) \times \{s\} \not\subseteq \mathcal{B}$ **then**
  **if** $V^{\mathcal{P}} = \emptyset$ **then**
    **output**$((g(U^{\mathcal{P}}), U^{\mathcal{P}}))$
  **else**
    Choose $e \in V^{\mathcal{P}}$
    EXTRACT*$(U^{\mathcal{P}} \cup \{e\}, V^{\mathcal{P}} \setminus \{e\})$
    EXTRACT*$(U^{\mathcal{P}}, V^{\mathcal{P}} \setminus \{e\})$
  **end if**
**end if**

Figure 7: The EXTRACT* closed itemset extractor (under any piecewise (anti)-monotone constraint).

| | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|
| $o_1$ | 0.2 | 1 | 0.2 |
| $o_2$ | 0 | 0.5 | (0) |
| $o_3$ | (0) | 2 | (0) |
| $o_4$ | (0) | (0) | 1.5 |

Table 3: $gp' : \mathcal{B}_E \to \mathbb{R}_+$.

$gp' : \mathcal{B}_E \to \mathbb{R}_+$. *At every call,* EXTRACT* *tests* $\mathcal{C}'_{avg\text{-}gp' \geqslant 1}(g(U^{\mathcal{P}}), g(U^{\mathcal{P}} \cup V^{\mathcal{P}}), U^{\mathcal{P}} \cup V^{\mathcal{P}}, U^{\mathcal{P}})$, *where* $\mathcal{C}'_{avg\text{-}gp' \geqslant 1}$ *is the constraint, (anti)-monotone on each of its arguments, that was introduced earlier to prove the piecewise (anti)-monotonicity of* $\mathcal{C}_{avg\text{-}gp' \geqslant 1}$. *When* EXTRACT* *is called with* $(U^{\mathcal{P}}, V^{\mathcal{P}}) = (\{p_1, p_2\}, \{p_3\})$, *this tests fails:*

$$\begin{aligned}
& \frac{\sum_{(o,p) \in g(\{p_1,p_2\}) \times \{p_1,p_2,p_3\}} gp'(o,p)}{|g(\{p_1,p_2,p_3\}) \times \{p_1,p_2\}|} \geqslant 1 \\
\equiv \quad & \frac{\sum_{(o,p) \in \{o_1,o_2\} \times \{p_1,p_2,p_3\}} gp'(o,p)}{|\{o_1\} \times \{p_1,p_2\}|} \geqslant 1 \\
\equiv \quad & \frac{0.2+1+0.2+0+0.5+0}{2} \geqslant 1 \\
\equiv \quad & 0.95 \geqslant 1 \\
\equiv \quad & \textit{false}
\end{aligned}$$

*As a consequence, recursive calls would not allow the extraction of any closed itemset satisfying* $\mathcal{C}_{avg\text{-}gp' \geqslant 1}$ *and the search-space can be pruned, i. e., the two enumeration nodes in the bottom-left corner of Figure 3 are not to be traversed. Interestingly, at the parent node, where* $(U^{\mathcal{P}}, V^{\mathcal{P}}) = (\{p_1\}, \{p_2, p_3\})$, *the piecewise (anti)-monotone constraint* $\mathcal{C}_{avg\text{-}gp' \geqslant 1}$ *does not prune the search space although no connected pattern, that may be recursively considered from it, satisfies* $\mathcal{C}_{avg\text{-}gp' \geqslant 1}$. *The reason is* $\mathcal{C}'_{avg\text{-}gp' \geqslant 1}$ *is not instantiated with a real pattern. Instead, each of its arguments is instantiated with the objects or the attributes of either the smallest or the largest pattern that may be recursively considered from the current call.*

In fact, the enforcement of piecewise (anti)-monotone constraints generalizes that of conjunctions of monotone and anti-monotone constraints (see Section 2.3.1). Like EXTRACT++, EXTRACT* enforces a monotone (resp. anti-monotone) constraint by instantiating its first argument with

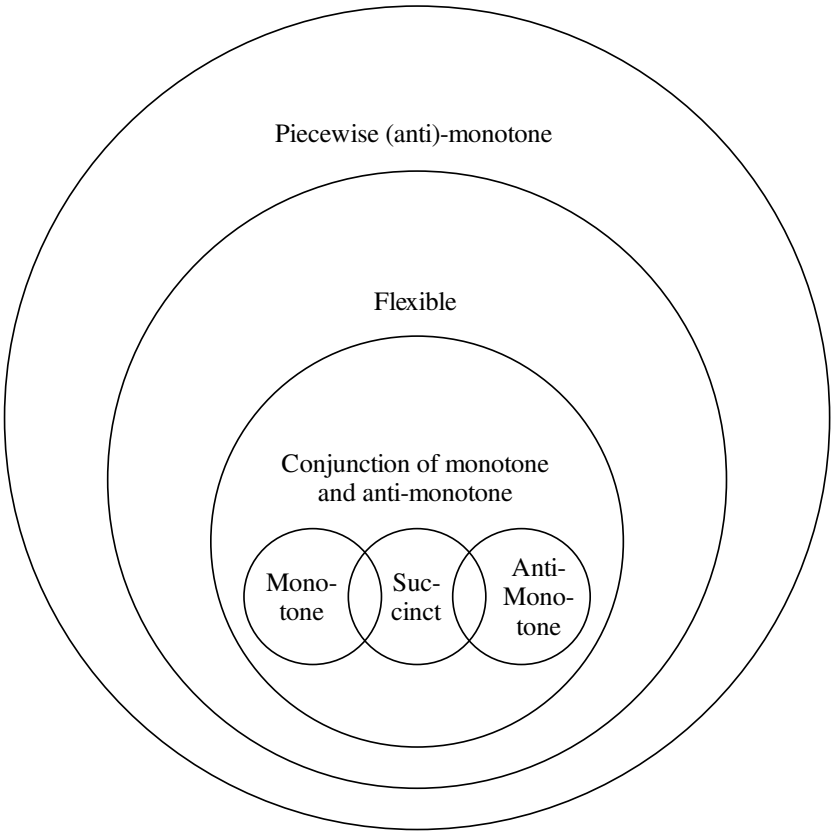



Figure 8: Classes of constraints preserving the freedom to enumerate any property anytime.

$g(U^{\mathcal{P}} \cup V^{\mathcal{P}})$ (resp. $g(U^{\mathcal{P}})$) and its second with $U^{\mathcal{P}} \cup V^{\mathcal{P}}$ (resp. $U^{\mathcal{P}}$). Indeed, an anti-monotone constraint (see Definition 3) is a two-variable constraint that is monotone w.r.t. the first argument (the objects) and anti-monotone w.r.t. the second argument (the properties). Conversely, a monotone constraint (see Definition 4) is anti-monotone w.r.t. the first argument and monotone w.r.t. the second argument.

Interestingly, both constraints $\mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\text{closed}}$ are piecewise (anti)-monotone. $\mathcal{C}_{\text{connected}}$ is monotone w.r.t. both its arguments and $\mathcal{C}_{\text{closed}}$ is proved piecewise (anti)-monotone in the next section. Because conjunctions of piecewise (anti)-monotone constraints are piecewise (anti)-monotone, "being a closed itemset" is piecewise (anti)-monotone too. As a consequence, the separation between the definition of a closed itemset, and an additional relevancy constraint $\mathcal{C}$ it must satisfies, somehow disappears if $\mathcal{C}$ only needs to be piecewise (anti)-monotone.

*"Being a closed itemset" is piecewise (anti)-monotone.*

### 2.3.7 *Relations Between the Classes*

Inclusions between classes of constraints were mentioned earlier. Figure 8 and 9 depict them. The enforcement of a convertible or of the loose (anti)-monotone constraint require more constrained enumeration principles. That prevents the use of some heuristics to improve the extraction performance. Furthermore, that make their respective classes unstable under conjunction. That is why they were, graphically, separated from the other classes, which do not impose any constraints on the enumeration.

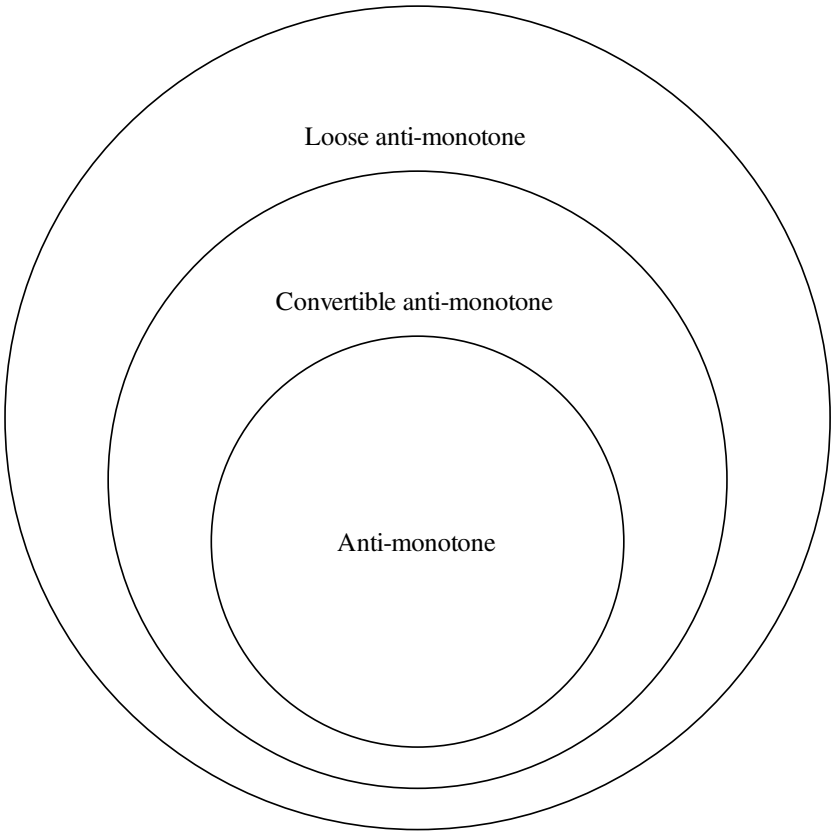



Figure 9: Classes of constraints generalizing anti-monotone constraints but whose enforcements require modified enumeration principles.

## 2.4 *On Closedness*

### 2.4.1 *Closedness*

Because only subsets of properties are enumerated and the supporting objects are *all* obtained thanks to the $g$ function (see Section 1.3.2), the closedness constraint $\mathcal{C}_{\text{closed}}$, as stated in Definition 2, can be reduced to testing whether a property can extend the pattern:

$$\mathcal{C}_{\text{closed}}(g(P), P) \equiv \forall s \in \mathcal{P} \setminus P, \neg\mathcal{C}_{\text{connected}}(g(P), \{s\}), \text{ i. e., } g(P) \times \{s\} \not\subseteq \mathcal{B} \ .$$

In Extract (see Figure 1), Extract++ (see Figure 4) and Extract* (see Figure 7), this constraint prunes the search space if it is false when instantiated as follows:

- the first occurrence of $P$ ($\forall s \in \mathcal{P} \ldots$) is instantiated with the greatest subset of properties that may be recursively considered from the current call, i. e., $U^{\mathcal{P}}$;
- the second occurrence of $P$ ($\ldots g(P) \times \{s\} \not\subseteq \mathcal{B}$) is instantiated with the smallest subset of properties that may be recursively considered from the current call, i. e., $U^{\mathcal{P}} \cup V^{\mathcal{P}}$.

This procedure exactly follows the enforcement of a piecewise (anti)-monotone constraint. Indeed, $\mathcal{C}_{\text{closed}}$ belongs to this class of constraint and is neither monotone, nor anti-monotone, nor succinct, nor convertible, nor loose (anti)-monotone.

CCI Miner [11] is an efficient closed itemset extractor which takes advantage of conjunctions of monotone and anti-monotone constraints. Discussing the technical reasons behind its efficiency is out of the scope of this thesis. Beyond these aspects, this article points out the ambiguity behind the quest for closed itemsets under constraints. Indeed, two interpretations are possible:

- The returned collection must be the one that would be obtained by listing every (unconstrained) closed itemset and, then, removing those that do not satisfy the constraints;
- The returned collection must be the one that would be obtained by listing every (not necessarily closed) itemset under constraint and, then, removing those that have both subsets (of objects and properties) included in another extracted pattern.

*In this thesis, the closedness constraint is a closure property w.r.t. to all unconstrained patterns.*

The toy algorithms illustrating this chapter, like most closed itemset extractors, adopt the first interpretation. CCI Miner uses the second interpretation. In $\mathcal{B}_E$, represented in Table 1, the complete collection of the closed itemsets under the anti-monotone constraint "having at most one property" includes $(\{o_1, o_2\}, \{p_1\})$ with the second interpretation but not with the first one. The authors of CCI Miner talk about a loss of information. However, in this example, the analyst does not want to look at $(\{o_1, o_2\}, \{p_1\})$. Indeed, the objects $o_1$ and $o_2$ actually share strictly more than one property. Furthermore, it is interesting to notice that the stability of the constrained pattern collection by transpose of the binary relation (see Section 1.3.2) is preserved by the first interpretation but not by the second one. To sum up, $\mathcal{C}_{\text{closed}}$ is the closedness constraint w.r.t. $\mathcal{C}_{\text{connected}}$ alone and a different closedness constraint can be defined with respect to another constraint, e. g., $\mathcal{C}_{\text{connected}} \wedge \mathcal{C}_{\leqslant 1 \text{ property}}$.

### 2.4.2 *Strong Closedness and Stability*

According to David J. Hand, here is the definition of a local pattern [39]:

> A local pattern is a data vector serving to describe an *anomalously* high local density of data points.

The closedness constraints on itemsets loosely enforces the anomalous aspect. Indeed, in practice, many of them are very similar to each other. Furthermore, although the closed itemsets are a condensed representation of all (closed and unclosed) itemsets (see Section 1.2), complete collections of closed itemsets remain huge. [22] proposed to strengthen the closedness on properties. In this way, the *anomalous* aspect of a local is enforced and the size of the extracted collection of pattern is reduced. A closed itemset $(O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$ is $\delta$-*tolerant closed* if, whatever $p \in \mathcal{P} \setminus P$, there always is strictly more than a proportion $\delta$ of $O$ that does not have $p$. If $\delta = 0$, the $\delta$-tolerant closed itemsets are closed itemsets. In other terms, Definition 2 is generalized.

*Strongly closed itemsets are* isolated *from the other patterns.*

**Definition 12 ($\delta$-tolerance closed itemset)** $\forall \delta \in [0, 1[$ *and* $\forall (O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $(O, P)$ *is a $\delta$-tolerance closed itemset iff:*

- $\mathcal{C}_{connected}(O, P) \equiv O \times P \subseteq \mathcal{B}$;
- $\mathcal{C}_{rel\text{-}\delta\text{-}closed}(O, P) \equiv \begin{cases} \forall o \in \mathcal{O} \setminus O, \{o\} \times P \not\subseteq \mathcal{B} \\ \forall p \in \mathcal{P} \setminus P, |(O \times \{p\}) \setminus \mathcal{B}| > \delta |O| \end{cases}$.

[9] proposed a similar reinforcement of the closedness on properties but with an absolute parametrization. A closed itemset $(O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$ is $\delta$-closed if, whatever $p \in \mathcal{P} \setminus P$, there is strictly more than $\delta$ objects in $O$ that do not have $p$. Again, if $\delta = 0$, the $\delta$-closed itemsets are closed itemsets and Definition 2 is generalized.

*The strong closedness is defined absolutely or in proportion to the number of objects in the itemset.*

**Definition 13 ($\delta$-closed itemset)** $\forall \delta \in \mathbb{N}$ *and* $\forall (O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $(O, P)$ *is a $\delta$-closed itemset iff:*

- $\mathcal{C}_{connected}(O, P) \equiv O \times P \subseteq \mathcal{B}$;
- $\mathcal{C}_{\delta\text{-}closed}(O, P) \equiv \begin{cases} \forall o \in \mathcal{O} \setminus O, \{o\} \times P \not\subseteq \mathcal{B} \\ \forall p \in \mathcal{P} \setminus P, |(O \times \{p\}) \setminus \mathcal{B}| > \delta \end{cases}$.

Notice that allowing $\delta < 0$ would make the $\delta$-closedness always satisfied. In this way, the itemsets, which are matched, are connected but not necessarily closed. The same notice can be made with Definition 12 modified so that $\delta < 0$ is allowed.

The $\delta$-closedness of a closed itemset $(O, P)$ can alternatively be defined as a stability index of the $(|O| - \delta)^{\text{th}}$ level equal to 1.

**Definition 14 (Stability index)** $\forall j \in \mathbb{N}$ *and* $(O, P)$ *a closed itemset, its stability index of the $j^{th}$ level is:*

$$\frac{|\{O' \subseteq O \mid |O'| = j \wedge f(O') = P\}|}{\binom{|O|}{j}}.$$

[47] first defined and studied the stability index. It quantifies to what extent the $\delta$-closedness is satisfied. Every definition of this section strengthen/quantify the closedness w.r.t. the properties. Applying them to the transpose of $\mathcal{B}$ would strengthen/quantify the closedness w.r.t. the objects.

## 3 CONCLUSION

This chapter mainly aims at presenting related works and at easing the introduction of generalizations of closed itemset mining this thesis

propose. However, it is also, by itself, a contribution. First of all, some of the presented results (the reverse-search paradigm helping extractions under any loose anti-monotone constraint, piecewise (anti)-monotone as a synonym of primitive-based, link between strong-closedness and stability) are, to the best of our knowledge, new. Then, this chapter hopefully clarifies a global view of the extraction of closed itemsets under constraints. In this regard, it follows a tradition of thesis, such as that of Hannu Toivonen [83], Francesco Bonchi [10], Sau Dan Lee [48] and Arnaud Soulet [77].

# 2 GENERALIZING CLOSED ITEMSET MINING

## 1 MINING NOISE-TOLERANT ITEMSETS

### 1.1 *Theoretical Aspects*

#### 1.1.1 *False Positive/Negative Noise*

It was mentioned that many of the closed itemsets are very similar to each other (see Section 2.4.2 in Chapter 1). This problem stems from the noisy aspect of most real-life datasets. Noise is an alteration of the data that prevents pattern discovery tasks from directly returning the relevant regularities. There are many possible sources of noise. It may be intrinsic to the studied system (e. g., stochastic biological processes) or be the result of erroneous measures. It can be imputed to mis-parameterized (or even mis-chosen) pre-processing steps too. In particular, when a binary relation $\mathcal{B} \subseteq \mathcal{O} \times \mathcal{P}$ is derived from a numerical dataset, there is a cumbersome need to choose a threshold beneath/beyond which an encoded Boolean property is claimed satisfied. [82] theoretically and empirically shows that the number of frequent itemsets exponentially grows with the level of noise while their sizes exponentially decrease with it.

The noise can have two opposite effects on the Boolean properties $\mathcal{B}$ encodes:

FALSE POSITIVE NOISE Boolean properties are satisfied but should be violated (supernumerary couples in $\mathcal{B}$);

FALSE NEGATIVE NOISE Boolean properties are violated but should be satisfied (missing couples in $\mathcal{B}$).

*Noise adds or removes couples from the relation. Large enough closed itemsets do not encompass the additional couples but should tolerate a few removed couples.*

Size constraints (e. g., a minimal number of objects *and* a minimal number of properties) usually are enough to avoid encompassing false positive couples. Indeed, the noise is, by definition, randomly distributed. As a consequence, false positive couples usually are in small patterns only. On the opposite, the definition of a closed itemset (more precisely, of $\mathcal{C}_{\text{connected}}$) prevents a pattern from encompassing false negative couples.

**Example 6** *Assume that $\mathcal{B}_E$, represented in Table 1, actually is affected by noise. Deprived of noise, this relation could be $\mathcal{B}_{hidden E}$, represented in Table 4. Extracting, in $\mathcal{B}_E$, closed itemsets with at least two objects and as many properties, provides only $(\{o_1, o_2\}, \{p_1, p_2\})$. It is a fragment of the hidden pattern $(\{o_1, o_2, o_3\}, \{p_1, p_2\})$, which would be found if $(o_3, p_2)$ was not affected by false negative noise (or if the pattern would tolerate some false negative noise).*

*The false positive noise inserted the couple $(o_1, p_3)$ in $\mathcal{B}_E$. Nevertheless, thanks to the frequency constraint "at least two objects", $(o_1, p_3)$ is not enough to append $p_3$ to the patterns. $(o_2, p_3)$ would be required too. However, thanks to the minimal size constraints and assuming a random distribution of the noise, this is unlikely $o_4$ or $p_4$ would extend a fragment of $(\{o_1, o_2, o_3\}, \{p_1, p_2\})$. Unfortunately, this might happen if there is much noise in the relation. In such a context, the fragments of the hidden patterns*

| | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|
| $o_1$ | 1 | 1 | 0 |
| $o_2$ | 1 | 1 | 0 |
| $o_3$ | 1 | 1 | 0 |
| $o_4$ | 0 | 0 | 1 |

Table 4: $\mathcal{B}_{\text{hiddenE}} \subseteq \{o_1, o_2, o_3, o_4\} \times \{p_1, p_2, p_3\}$.

*are small. Strong size constraints would discard them all. As a consequence, these constraints must remain weak and false positive noise may be encompassed. Interestingly, in the presence of noise, a small hidden pattern, such as* $(\{o_4, p_3\})$ *in* $\mathcal{B}_{hiddenE}$*, cannot be discovered. Indeed, it is of the size of the meaningless patterns the false positive noise randomly generates.*

### 1.1.2 *Absolute vs. Relative Tolerance*

*Couples absent from the relation are tolerated either absolutely or in proportion to the size of the closed itemset.*

To tolerate false negative noise, closed itemsets (see Definition 1) needs to be generalized in closed ET-itemsets [1]. Either an absolute or a relative tolerance to noise can be chosen. With a relative tolerance to noise, every object (resp. property) in a closed ET-itemset $(O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$ can miss at most a proportion $\epsilon^{\mathcal{P}}_{\text{rel}}$ (resp. $\epsilon^{\mathcal{O}}_{\text{rel}}$) of the properties in $P$ (resp. objects in $O$).

**Definition 15 (Closed (relative) ET-itemset)** $\forall \epsilon_{rel} = (\epsilon^{\mathcal{O}}_{rel}, \epsilon^{\mathcal{P}}_{rel}) \in [0, 1]^2$ *and* $\forall (O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $(O, P)$ *is a closed (relative) ET-itemset iff:*

- $\mathcal{C}_{rel\text{-}\epsilon\text{-}connected}(O, P) \equiv \begin{cases} \forall o \in O, |(\{o\} \times P) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{P}}_{rel}|P| \\ \forall p \in P, |(O \times \{p\}) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{O}}_{rel}|O| \end{cases}$ ;
- $\mathcal{C}_{rel\text{-}\epsilon\text{-}closed}(O, P) \equiv \forall (O', P') \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $\Big(O \subseteq O' \wedge P \subseteq P' \wedge \mathcal{C}_{rel\text{-}\epsilon\text{-}connected}(O', P')\Big) \Rightarrow (O', P') = (O, P)$.

With an absolute tolerance to noise, every object (resp. property) in a closed ET-itemset $(O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$ can miss at most $\epsilon^{\mathcal{P}}$ (resp. $\epsilon^{\mathcal{O}}$) properties in $P$ (resp. objects in $O$).

**Definition 16 (Closed (absolute) ET-itemset)** $\forall \epsilon = (\epsilon^{\mathcal{O}}, \epsilon^{\mathcal{P}}) \in \mathbb{N}^2$ *and* $\forall (O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $(O, P)$ *is a closed (absolute) ET-itemset iff:*

- $\mathcal{C}_{\epsilon\text{-}connected}(O, P) \equiv \begin{cases} \forall o \in O, |(\{o\} \times P) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{P}} \\ \forall p \in P, |(O \times \{p\}) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{O}} \end{cases}$ ;
- $\mathcal{C}_{\epsilon\text{-}closed}(O, P) \equiv \forall (O', P') \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $\Big(O \subseteq O' \wedge P \subseteq P' \wedge \mathcal{C}_{\epsilon\text{-}connected}(O', P')\Big) \Rightarrow (O', P') = (O, P)$.

Considering the tabular representation of the binary relation $\mathcal{B}$ (such as Table 1), an ET-itemset is a rectangle (modulo permutations of the rows/columns), with upper-bounded proportions (resp. numbers in Definition 16) of '0's on any row and on any column. The upper-bounds on rows and on columns can be different. The closedness constraint further forces any extension of a pattern to exceeds these noise tolerance thresholds. With $\epsilon_{\text{rel}} = (0, 0)$ or $\epsilon = (0, 0)$, both definitions are equivalent to Definition 1. In other terms, closed ET-itemsets generalize closed itemsets.

1. Like in [93], ET stands for Error-Tolerant.

**Example 7** *Consider a relative tolerance to noise* $\epsilon_{rel} = (0.4, 0.5)$. *In* $\mathcal{B}_E$, *represented in Table 1,* $(\{o_1, o_2, o_3\}, \{p_1, p_2\})$ *is a closed ET-itemset:*

- *Every object in* $\{o_1, o_2, o_3\}$ *(resp. property in* $\{p_1, p_2\}$*) misses, at most, half of the properties in* $\{p_1, p_2\}$ *(resp. 40% of the objects in* $\{o_1, o_2, o_3\}$*):*
  - $|(\{o_1\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = |\emptyset| = 0 \leqslant 0.5|\{p_1, p_2\}|$;
  - $|(\{o_2\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = |\emptyset| = 0 \leqslant 0.5|\{p_1, p_2\}|$;
  - $|(\{o_3\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = |\{(o_3, p_1)\}| = 1 \leqslant 0.5|\{p_1, p_2\}|$;
  - $|(\{o_1, o_2, o_3\} \times \{p_1\}) \setminus \mathcal{B}_E| = |\{(o_3, p_1)\}| = 1 \leqslant 0.4|\{o_1, o_2, o_3\}|$;
  - $|(\{o_1, o_2, o_3\} \times \{p_2\}) \setminus \mathcal{B}_E| = |\emptyset| = 0 \leqslant 0.4|\{o_1, o_2, o_3\}|$.
- *Every "super-pattern" violates* $\mathcal{C}_{rel\text{-}\epsilon\text{-}connected}$*:*
  - $\neg\mathcal{C}_{rel\text{-}\epsilon\text{-}connected}(\{o_1, o_2, o_3, o_4\}, \{p_1, p_2\})$ *(because* $|(\{o_4\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = 2 > 0.5|\{p_1, p_2\}|$*);*
  - $\neg\mathcal{C}_{rel\text{-}\epsilon\text{-}connected}(\{o_1, o_2, o_3\}, \{p_1, p_2, p_3\})$ *(because, e.g.,* $|(\{o_3\} \times \{p_1, p_2, p_3\}) \setminus \mathcal{B}_E| = 2 > 0.5|\{p_1, p_2\}|$*);*
  - $\neg\mathcal{C}_{rel\text{-}\epsilon\text{-}connected}(\{o_1, o_2, o_3, o_4\}, \{p_1, p_2, p_3\})$ *(because, e.g.,* $|(\{o_3\} \times \{p_1, p_2, p_3\}) \setminus \mathcal{B}_E| = 2 > 0.5|\{p_1, p_2\}|$*).*

$(\{o_1, o_2, o_4\}, \{p_2, p_3\})$ *and* $(\{o_1, o_3, o_4\}, \{p_2, p_3\})$ *are two other closed ET-itemsets in* $\mathcal{B}_E$.

Following the example above, the verification of $\mathcal{C}_{\text{rel-}\epsilon\text{-closed}}$ on an ET-itemset $(O, P)$ involves $2^{|(\mathcal{O}\setminus O)\cup(\mathcal{P}\setminus P)|}$ checks of $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$. With a relative tolerance to noise, it is impossible to define the closedness w.r.t. the patterns with one more element (either an object or a property). Indeed a closed ET-itemset (according to Definition 7) does not imply each of its "sub-patterns" satisfies $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$. As a consequence, enforcing $\mathcal{C}_{\text{rel-}\epsilon\text{-closed}}$ does not provide a lossless condensation of all ET-itemsets. That is why, with a relative tolerance to noise, most extractors of ET-itemsets do not force them to be closed. Therefore, the output collections of ET-itemsets are very large, carry much redundant information and interpreting them is tedious.

*With absolute numbers (resp. proportions) of absent couples,* closed *ET-itemsets are (resp. are not) a lossless condensation of all ET-itemsets.*

On the contrary, with an absolute tolerance to noise, $\mathcal{C}_{\epsilon\text{-closed}}$ provides a lossless condensation of the collection of all (closed and unclosed) ET-itemsets. It can easily be proved that an equivalent closedness constraint only forces the patterns with one more element (either an object or a property) to violate $\mathcal{C}_{\epsilon\text{-connected}}$.

**Definition 17 (Closed (absolute) ET-itemset (equivalent definition))** $\forall \epsilon = (\epsilon^{\mathcal{O}}, \epsilon^{\mathcal{P}}) \in \mathbb{N}^2$ *and* $\forall (O, P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $(O, P)$ *is a closed (absolute) ET-itemset iff:*

- $$\mathcal{C}_{\epsilon\text{-}connected}(O, P) \equiv \begin{cases} \forall o \in O, |(\{o\} \times P) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{P}} \\ \forall p \in P, |(O \times \{p\}) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{O}} \end{cases} ;$$
- $$\mathcal{C}_{\epsilon\text{-}closed}(O, P) \equiv \begin{cases} \forall o \in \mathcal{O} \setminus O, \begin{cases} |(\{o\} \times P) \setminus \mathcal{B}| > \epsilon^{\mathcal{P}} \\ \textit{or} \\ \exists p \in P \,|\, |((O \cup \{o\}) \times \{p\}) \setminus \mathcal{B}| > \epsilon^{\mathcal{O}} \end{cases} \\ \forall p \in \mathcal{P} \setminus P, \begin{cases} |(O \times \{p\}) \setminus \mathcal{B}| > \epsilon^{\mathcal{O}} \\ \textit{or} \\ \exists o \in O \,|\, |(\{o\} \times (P \cup \{p\})) \setminus \mathcal{B}| > \epsilon^{\mathcal{P}} \end{cases} \end{cases}$$

**Example 8** *Consider an absolute tolerance to noise* $\epsilon = (1, 1)$. *In* $\mathcal{B}_E$, *represented in Table 1,* $(\{o_1, o_2, o_3\}, \{p_1, p_2\})$ *is a closed ET-itemset:*

- *Every object in* $\{o_1, o_2, o_3\}$ *(resp. property in* $\{p_1, p_2\}$*) misses, at most, one properties in* $\{p_1, p_2\}$ *(resp. one object in* $\{o_1, o_2, o_3\}$*):*

  - $|(\{o_1\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = |\emptyset| = 0 \leqslant 1$;
  - $|(\{o_2\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = |\emptyset| = 0 \leqslant 1$;
  - $|(\{o_3\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = |\{(o_3, p_1)\}| = 1 \leqslant 1$;
  - $|(\{o_1, o_2, o_3\} \times \{p_1\}) \setminus \mathcal{B}_E| = |\{(o_3, p_1)\}| = 1 \leqslant 1$;
  - $|(\{o_1, o_2, o_3\} \times \{p_2\}) \setminus \mathcal{B}_E| = |\emptyset| = 0 \leqslant 1$.
- *Every pattern with one more element violates* $\mathcal{C}_{\epsilon\text{-connected}}$*:*
  - $\neg\mathcal{C}_{\epsilon\text{-connected}}(\{o_1, o_2, o_3, o_4\}, \{p_1, p_2\})$
    *(because* $|(\{o_4\} \times \{p_1, p_2\}) \setminus \mathcal{B}_E| = 2 > 1$*);*
  - $\neg\mathcal{C}_{\epsilon\text{-connected}}(\{o_1, o_2, o_3\}, \{p_1, p_2, p_3\})$
    *(because, e.g.,* $|(\{o_3\} \times \{p_1, p_2, p_3\}) \setminus \mathcal{B}_E| = 2 > 1$*);*

$(\{o_1, o_2, o_4\}, \{p_2, p_3\})$ *and* $(\{o_1, o_3, o_4\}, \{p_2, p_3\})$ *are two other closed ET-itemsets in* $\mathcal{B}_E$*.*

*"Being a closed ET-itemset" is piecewise (anti)-monotone.*

The constraints $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$, $\mathcal{C}_{\text{rel-}\epsilon\text{-closed}}$, $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ are piecewise (anti)-monotone. Because conjunctions of piecewise (anti)-monotone constraints are piecewise (anti)-monotone, "being a closed ET-itemset" is piecewise (anti)-monotone. A relative tolerance to noise looks more natural. Nevertheless, ET-itemsets are much easier to extract with an absolute tolerance to noise. There are several ways to understand it. One of them is comparing the enforcement of $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-connected}}$ as piecewise (anti)-monotone constraints. $\mathcal{C}_{\epsilon\text{-connected}}$ does not need to be rewritten to be proved piecewise (anti)-monotone. Indeed, it is anti-monotone w.r.t. each of its arguments. As a consequence, when an extractor enforces $\mathcal{C}_{\epsilon\text{-connected}}$, it instantiates its two arguments with a pattern $(O, P)$ that may be recursively considered from the current call. On the opposite, to be proved piecewise (anti)-monotone $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$ needs to be rewritten in $\mathcal{C}'_{\text{rel-}\epsilon\text{-connected}}$:

$$\begin{aligned} & \mathcal{C}'_{\text{rel-}\epsilon\text{-connected}}(O_1, O_2, O_3, P_1, P_2, P_3) \\ \equiv & \begin{cases} \forall o \in O_1, |(\{o\} \times P_1) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{P}}_{\text{rel}} |P_2| \\ \forall p \in P_3, |(O_2 \times \{p\}) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{O}}_{\text{rel}} |O_3| \end{cases} . \end{aligned}$$

$\mathcal{C}'_{\text{rel-}\epsilon\text{-connected}}$ is monotone w.r.t. its third and sixth arguments and anti-monotone w.r.t. its other arguments. As a consequence, $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$ is, by definition, piecewise (anti)-monotone. When an extractor enforces it, depending on the argument, $\mathcal{C}'_{\text{rel-}\epsilon\text{-connected}}$ is partly instantiated with the smallest subset of objects (or properties), that may be considered from the current call, and partly with the largest. In other terms, the instantiation is not related to a pattern and $\mathcal{C}'_{\text{rel-}\epsilon\text{-connected}}$ may be satisfied even if none of the patterns, that may be recursively considered from the current call, satisfies $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$.

### 1.1.3 *Loss of the Galois Connection*

Contrary to closed itemset mining, extracting every closed ET-itemset cannot rely on a Galois connection $(f, g)$ (see Section 1.3.2 in Chapter 1). Indeed, generalizing $f$ (resp. $g$) towards noise tolerance makes it possibly return several subsets of properties (resp. objects) from one subset of objects (respectively properties). For instance, both Examples 7 and 8 mention that, with the chosen noise tolerances, $(\{o_1, o_2, o_4\}, \{p_2, p_3\})$ and $(\{o_1, o_3, o_4\}, \{p_2, p_3\})$ are closed ET-itemsets in $\mathcal{B}_E$. As a consequence, $g(\{p_2, p_3\})$ must be both $\{o_1, o_2, o_4\}$ and $\{o_1, o_3, o_4\}$.

### 1.1.4 *Other Definitions*

The first ET-itemset complete extractors defined the connectedness in other ways. [93] proposed these two definitions:

**Definition 18 (Weak (relative) connection)** $\forall \epsilon_{rel} \in [0,1]$ *and* $\forall (O,P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $\mathcal{C}_{rel\text{-}\epsilon\text{-}weakly\text{-}connected}(O,P) \equiv |(O \times P) \setminus \mathcal{B}| \leqslant \epsilon_{rel}|O \times P|$.

**Definition 19 (Strong (relative) $\mathcal{P}$-connection)** $\forall \epsilon^{\mathcal{P}}_{rel} \in [0,1]$, $\forall (O,P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $\mathcal{C}_{rel\text{-}\epsilon^{\mathcal{P}}_{rel}\text{-}strongly\text{-}\mathcal{P}\text{-}connected}(O,P) \equiv \forall o \in O, |(\{o\} \times P) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{P}}_{rel}|P|$.

The following analog definition could be added:

**Definition 20 (Strong (relative) $\mathcal{O}$-connection)** $\forall \epsilon^{\mathcal{O}}_{rel} \in [0,1]$, $\forall (O,P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $\mathcal{C}_{rel\text{-}\epsilon^{\mathcal{O}}_{rel}\text{-}strongly\text{-}\mathcal{O}\text{-}connected}(O,P) \equiv \forall p \in P, |(O \times \{p\}) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{O}}_{rel}|O|$.

$\mathcal{C}_{\text{rel-}\epsilon\text{-weakly-connected}}$ forces a maximal proportion $\epsilon_{\text{rel}}$ of couples absent from $\mathcal{B}$ among those any ET-itemset covers. $\mathcal{C}_{\text{rel-}\epsilon^{\mathcal{P}}_{\text{rel}}\text{-strongly-}\mathcal{P}\text{-connected}}$ and $\mathcal{C}_{\text{rel-}\epsilon^{\mathcal{O}}_{\text{rel}}\text{-strongly-}\mathcal{O}\text{-connected}}$ actually are special cases of $\mathcal{C}_{\text{rel-}\epsilon\text{-connected}}$ (see Definition 15):

- $\forall \epsilon^{\mathcal{P}}_{\text{rel}} \in [0,1]$, $\mathcal{C}_{\text{rel-}\epsilon^{\mathcal{P}}_{\text{rel}}\text{-strongly-}\mathcal{P}\text{-connected}} \equiv \mathcal{C}_{\text{rel-}(1,\epsilon^{\mathcal{P}})\text{-connected}}$;
- $\forall \epsilon^{\mathcal{O}}_{\text{rel}} \in [0,1]$, $\mathcal{C}_{\text{rel-}\epsilon^{\mathcal{O}}_{\text{rel}}\text{-strongly-}\mathcal{O}\text{-connected}} \equiv \mathcal{C}_{\text{rel-}(\epsilon^{\mathcal{P}},1)\text{-connected}}$.

None of these definitions is satisfactory. Indeed, they match patterns with objects (resp. properties) that do not have any property (resp. object) of the pattern.

*If the tolerated noise is bounded per object (resp. attribute) only, some irrelevant closed itemsets are extracted.*

**Example 9** *In $\mathcal{B}_E$, represented in Table 1, consider $(\{o_1, o_2, o_4\}, \{p_1, p_2\})$. The object $o_4$ obviously has nothing to do with $\{p_1, p_2\}$ since it does not have any of these properties. However, $(\{o_1, o_2, o_4\}, \{p_1, p_2\})$ satisfies both $\mathcal{C}_{rel\text{-}0.4\text{-}weakly\text{-}connected}$ and $\mathcal{C}_{rel\text{-}0.4\text{-}strongly\text{-}\mathcal{O}\text{-}connected}$.*

[74] worded this critic. In fact, with any of these three definitions and large enough patterns, completely disconnected elements (either objects or attributes) can be part of an ET-itemset whatever the noise tolerance (as far as it is not 0). With an absolute tolerance to noise, the analog definitions raise the same issue. Several works, mentioned in the next section, use the absolute counterpart of Definition 19:

**Definition 21 (Strong (absolute) $\mathcal{P}$-connection)** $\forall \epsilon^{\mathcal{P}} \in \mathbb{N}$, $\forall (O,P) \in 2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $\mathcal{C}_{\epsilon^{\mathcal{P}}\text{-}strongly\text{-}\mathcal{P}\text{-}connected}(O,P) \equiv \forall o \in O, |(\{o\} \times P) \setminus \mathcal{B}| \leqslant \epsilon^{\mathcal{P}}$.

## 1.2 *State of the Art*

### 1.2.1 *ET-Itemset Complete Extractors*

Several research groups have considered the complete extraction of ET-itemsets in binary relations (see [37] for a survey). None of them is fully satisfactory. As explained in Section 1.1.2, a relative tolerance to noise makes the extraction task very hard and a closedness constraint does not provide a lossless condensation of the ET-itemsets. As a consequence, the extractors tolerating *proportions* of noise, suffer from great scalability issues and they output large collections of ET-itemsets carrying much redundant information, hence a difficult interpretation. Furthermore, until the very recent publication of [68] (excluded), every proposal was either relying on lossy heuristics or imposing additional constraints. Among the extractors relying on lossy heuristics, AFI [52]

*Every state-of-the-art ET-itemset extractor suffers from several of the following troubles: discovery of some irrelevant patterns, no (or strange) closedness constraint, scalability issues, compulsory additional constraints, lossy heuristics.*

was the first algorithm to use the connection constraint of Definition 15 (but no closedness constraint) but approximates the number of objects involved in a pattern. Among the approaches that impose additional constraints, `AC-Close` [21] is significant. Indeed, it is the only *closed* ET-itemset extractor with a relative tolerance to noise (Definition 15 is used). Furthermore the experiments show it runs much faster than AFI. These results are possible thanks to a frequency constraint on an *exact* closed itemset every ET-itemset must contain. Thus, `AC-Close` requires an awkward minimal number of objects *exact* closed itemsets (by opposition to *ET-itemset*) must involve. It extracts them and, in a second step, extends them in closed ET-itemsets. Despite the performance improvement w.r.t. AFI, Section 5.3 in Chapter 4 empirically shows `AC-Close` is intractable on medium-size relations.

To completely extract ET-itemsets in larger relations, the algorithms with an absolute tolerance to noise remains. `FT-`*Apriori* [63] extracts every frequent ET-itemset matching Definition 21. As detailed in Section 1.1.4, such a definition raises issues. To avoid extracting ET-itemsets with some properties that almost every object (in the pattern) misses, each of these properties is forced to hold for at least $\gamma \in \mathbb{N}$ objects of the pattern. This frequency constraint, restricted to the objects of the ET-itemset, filters out some of the least relevant patterns. However, because it deals with couples present in (rather than absent from) $\mathcal{B}$, ET-itemsets with many objects may still gather properties that many objects of the pattern do not have. Furthermore, because no closedness constraint is enforced, the extracted collections contain redundant information. VB-FT-Mine [46] builds upon `FT-`*Apriori* and significantly improves both the running times (thanks to the use of bit vectors) and the space requirements (thanks to a depth-first enumeration). The authors of [66] consider different definitions of an ET-itemset. All of them tolerate noise in an absolute way. To extract them, the BIAS framework enumerates growing subsets of properties and words the noise tolerance w.r.t. every enumerated property as an inequality. Integer linear programming allows to derive (one of) the largest set of objects that satisfies every inequality. [67] optimizes this procedure and generalizes the definition of an ET-itemset. Its objects are either those that have (or miss) at most, at least, or exactly a given number of properties among those in the pattern. Thus, Definition 21 is generalized (but remains the definition of interest to tolerate noise). The authors propose a recursive equation that computes the number of objects in an ET-itemset from those of the ET-itemsets with one property less. Beyond the absence of a constraint bounding the noise tolerance w.r.t. properties and the absence of a closedness constraint, this approach requires much space. Indeed, it stores all subsets of properties that were previously considered and the associated numbers of objects. On the positive side, the approach can easily be implemented in any frequent itemset extractor and its running time does not increase much w.r.t. the extraction of *exact* itemsets. DR-Miner [6] extracts complete collections of closed ET-itemsets (Definition 7) but its closedness constraint is differently and, somehow, oddly defined. It forces every element (either an object or a property) "outside" a closed ET-itemset to gather strictly more couples absent from the mined relation than any element "inside" it. This excludes some ET-itemsets that would be closed if some additional couples were missing *outside* the pattern.

### 1.2.2 *Agglomerating Itemsets*

A popular heuristic to tolerate noise consists in extracting exact patterns that are, in a second step, clustered. The most fundamental parameter is the choice of a relevant metric. [84] proposes to cluster complete collection of association rules. The involved distance simply is the number of objects covered by one rule (i. e., by the itemset union of its left and right parts) and not the other. [36] somehow makes this measure relative to the number of objects covered by both association rules. [8] also proposes the use of such a relative measure but on both the objects and the properties. A generalization of this metric enables a fuzzy hierarchical clustering. Initially, every closed itemset is associated with two Boolean vectors where a '0' (respectively '1') stands for the absence (respectively the presence) of an element (either an object or a property) in the closed itemset. Then classical fuzzy operators and a 1-norm are used to compute the distance between two clusters and a fuzzy union agglomerates them if they are the closest pair at the current iteration. MicroCluster [98] discovers complete collections of local patterns in real-valued matrices. In particular, it can extract the closed itemsets. A post-processing step is proposed to handle noise by deleting or merging some of the extracted patterns. The involved distances are based on counts of couples. For example, two patterns are merged when the number of couples covered by at least one of the two patterns is greater than $\gamma$ times the number of couples belonging to the envelope of the two patterns but not to any of the two patterns. Until now, every approach presented in this section uses a metric (in the pattern space) that is solely based on the patterns, i. e., the agglomeration ignores the regions of the data that are not in the relation subspaces described by any of the two itemsets. In particular, the number and the repartition of the couples absent from $\mathcal{B}$, but covered by the agglomerated patterns, is not taken into consideration. On the contrary, [92] argues for using not only the information the patterns directly express but the binary relation they were extracted from too. The proposed metrics quantifies the entropy in the clusters considered as sets of independent Boolean vectors. Thus, the two attributes are not symmetric.

*Agglomerating exact patterns to tolerate noise is a popular heuristics. It requires the definition of a metric.*

## 2 MINING CLOSED PATTERNS IN $n$-ARY RELATIONS

### 2.1 *Theoretical Aspects*

#### 2.1.1 N-*ary Relations*

Given an arity $n \in \mathbb{N}$ and $n$ finite sets $(\mathcal{D}^i)_{i=1..n}$, let $\mathcal{R} \subseteq \times_{i=1..n}\mathcal{D}^i$ the $n$-ary relation where patterns are to be discovered. All along the remaining of this thesis, $\mathcal{R}$ denotes this dataset. Table 5 represents an example of such a relation $\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$, hence a ternary relation. In this table, every '1' at the intersection of three elements stands for the presence of the related triplet in $\mathcal{R}_E$. For example the bold '**1**', in Table 5, is at the intersection of the elements $\alpha$, 1 and A. It represents the presence of $(\alpha, 1, A)$ in $\mathcal{R}_E$. On the contrary a '0' in Table 5 is at the intersection of three elements which form a triplet absent from $\mathcal{R}_E$. For example the bold '**0**' in Table 5 means $(\alpha, 2, C) \notin \mathcal{R}_E$. In practice, the relations we are interested in are

N-*ary relations generalize binary relations by associating elements from* $n$ *attribute domains.*

| | A | B | C | A | B | C | A | B | C |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 2 | 1 | 1 | **0** | 1 | 0 | 0 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| | $\alpha$ | | | $\beta$ | | | $\gamma$ | | |

Table 5: $\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$.

much larger than $\mathcal{R}_E$. Anyway, $\mathcal{R}_E$ is enough to illustrate most of the examples in Parts iii and iv.

N-ary relations are present in many application domains. For instance, $\mathcal{R}_E$ could represent customers (1, 2, 3 and 4) buying items (A, B and C) along three months ($\alpha$, $\beta$ and $\gamma$). In this context, the bold '**1**' in Table 5 would mean that the customer 1 bought the item A during the first month. The bold '**0**' would be understood as "customer 2 did not buy item C during the first month".

### 2.1.2 *A Natural Definition*

The patterns in $\times_{i=1..n} 2^{\mathcal{D}^i}$ are called $n$-sets. They associate $n$ subsets of elements from the $n$ domains of the relation. To simplify the exposition of this thesis, an $n$-set $(S^i)_{i=1..n}$ will often be assimilated with $\cup_{i=1..n} S^i$ (without loss of generality, the attribute domains $\mathcal{D}^i$ are considered disjoint). For example, given an $n$-set $A = (A^i)_{i=1..n}$ and an element $e \in \cup_{i=1..n} \mathcal{D}^i$, we write:

- $e \in A$ instead of $e \in \cup_{i=1..n} A^i$;
- $A \setminus \{e\}$ instead of $\begin{cases} (A^1 \setminus \{e\}, A^2, \ldots, A^n) \text{ if } e \in \mathcal{D}^1 \\ \vdots \\ (A^1, \ldots, A^{n-1}, A^n \setminus \{e\}) \text{ if } e \in \mathcal{D}^n \end{cases}$.

Nevertheless, to avoid too much abuse of notation, the union and the inclusion of $n$-sets are formally defined below.

**Definition 22 ($n$-set union $\sqcup$)** $\forall A = (A^i)_{i=1..n} \in \times_{i=1..n} 2^{\mathcal{D}^i}$ *and* $\forall B = (B^j)_{j=1..n} \in \times_{j=1..n} 2^{\mathcal{D}^j}$, $A \sqcup B = (A^1 \cup B^1, \ldots, A^n \cup B^n)$.

**Definition 23 ($n$-set inclusion $\sqsubseteq$)** $\forall A = (A^i)_{i=1..n} \in \times_{i=1..n} 2^{\mathcal{D}^i}$ *and* $\forall B = (B^j)_{j=1..n} \in \times_{j=1..n} 2^{\mathcal{D}^j}$, $A \sqsubseteq B \Leftrightarrow A^1 \subseteq B^1 \wedge \cdots \wedge A^n \subseteq B^n$.

Notice that, by definition, the union of two $n$-sets is the $n$-set with the minimal envelope enclosing both of them. The inclusion of $n$-sets is useful to define a closedness constraint. Closed $n$-sets in $n$-ary relations ($n \geqslant 2$) generalize closed itemsets in binary relations. In other terms, closed 2-sets *are* closed itemsets. The generalization of Definition 1 towards $n$-ary relations is natural. Considering the 0/1 representation of the $n$-ary relation (such as Table 5), a closed $n$-set is a maximal hyper-rectangle of '1's modulo arbitrary permutations of the hyper-plans. Here is a more formal definition:

*Closed $n$-sets (a) cover only tuples present in the relation; (b) cannot be enlarged without violating (a).*

**Definition 24 (Closed $n$-set)** $\forall X = (X^i)_{i=1..n} \in \times_{i=1..n} 2^{\mathcal{D}^i}$, $X$ *is a closed $n$-set iff:*

- $\mathcal{C}_{connected}(X) \equiv \times_{i=1..n} X^i \subseteq \mathcal{R}$;

– $\mathcal{C}_{closed}(X) \equiv \forall X' \in \times_{j=1..n} 2^{\mathcal{D}^j}, \left(X \sqsubseteq X' \wedge \mathcal{C}_{connected}(X')\right) \Rightarrow X' = X.$

According to the first constraint, $\mathcal{C}_{\text{connected}}$, taking one element from each of the subsets constituting a closed $n$-set is constructing an $n$-tuple that is in $\mathcal{R}$. The second constraint, $\mathcal{C}_{\text{closed}}$, tells that $X$ is closed if any strictly larger pattern (more elements from any domains) violates $\mathcal{C}_{\text{connected}}$. It is, for $\mathcal{C}_{\text{connected}}$, a closure property on the $n$ subsets of $\mathcal{D}^1$, $\mathcal{D}^2, \ldots$ and $\mathcal{D}^n$ altogether. It can easily be proved that an equivalent closedness constraint only forces the patterns with *one* more element (from any domain) to break $\mathcal{C}_{\text{connected}}$. Furthermore, because $\mathcal{C}_{\text{connected}}$ ensures the presence in $\mathcal{R}$ of every $n$-tuple in $\times_{i=1..n} X^i$, checking the closedness constraint can be reduced to searching for absent $n$-tuples involving one additional element only.

**Definition 25 (Closed $n$-set (equivalent definition))** $\forall X = (X^i)_{i=1..n} \in \times_{i=1..n} 2^{\mathcal{D}^i}$, *$X$ is a closed $n$-set iff:*
- $\mathcal{C}_{connected}(X) \equiv \times_{i=1..n} X^i \subseteq \mathcal{R}$;
- $\mathcal{C}_{closed}(X) \equiv \forall i = 1..n, \forall s \in \mathcal{D}^i \setminus X^i,$
  $\neg\mathcal{C}_{connected}(X^1, \ldots, \{s\}, \ldots, X^n)$, *i.e.*, $X^1 \times \cdots \times \{s\} \times \cdots \times X^n \not\subseteq \mathcal{R}$.

**Example 10** *In $\mathcal{R}_E$, represented in Table 5, $(\{\alpha, \gamma\}, \{1, 2\}, \{A, B\})$ is a closed 3-set:*
- *$\{\alpha, \gamma\} \times \{1, 2\} \times \{A, B\} \subseteq \mathcal{R}_E$ (in Table 5 there are '1's at the intersection of all the related hyper-plans);*
- *Every pattern with one more element violates $\mathcal{C}_{connected}$:*
  - *$\neg\mathcal{C}_{connected}(\{\beta\}, \{1, 2\}, \{A, B\})$, i.e., $\{\beta\} \times \{1, 2\} \times \{A, B\} \not\subseteq \mathcal{R}_E$;*
  - *$\neg\mathcal{C}_{connected}(\{\alpha, \gamma\}, \{3\}, \{A, B\})$, i.e., $\{\alpha, \gamma\} \times \{3\} \times \{A, B\} \not\subseteq \mathcal{R}_E$;*
  - *$\neg\mathcal{C}_{connected}(\{\alpha, \gamma\}, \{4\}, \{A, B\})$, i.e., $\{\alpha, \gamma\} \times \{4\} \times \{A, B\} \not\subseteq \mathcal{R}_E$;*
  - *$\neg\mathcal{C}_{connected}(\{\alpha, \gamma\}, \{1, 2\}, \{C\})$, i.e., $\{\alpha, \gamma\} \times \{1, 2\} \times \{C\} \not\subseteq \mathcal{R}_E$.*

*$(\{\alpha, \beta, \gamma\}, \{1, 2\}, \{A\})$ and $(\{\alpha, \beta, \gamma\}, \{1, 2, 3, 4\}, \emptyset)$ are two other examples of closed 3-sets.*

If, again, a ternary relation stands for customers buying items along three months, a closed 3-set is a maximal subset of customers buying the same maximal subset of items during a maximal subset of months. Such a pattern is useful for analyzing buying behaviors. The closedness constraint filters out all strict "sub-patterns" (i.e., patterns where some elements are removed and none are added) of the largest ones that are extracted. The justification for this constraint is the same as with itemset mining: a lossless (and necessary w.r.t. the interpretation) reduction of the output, which keeps the most informative pattern of every equivalence class. With collections of $n$-sets, a "lossless condensation" means that, whatever $j = 1..n$ and given any $(n-1)$-set $X \in \times_{i=1..n \wedge i \neq j} 2^{\mathcal{D}^i}$, *all* elements in $\mathcal{D}^j$ that relate with every combination of $n-1$ elements taken from the $n-1$ subsets of $X$ can be derived from the closed $n$-sets only. They are the largest set of elements in $\mathcal{D}^j$ a closed $n$-set associates with an $(n-1)$-set larger than $X$ (w.r.t. the $\sqsubseteq$ order).

*The closedness constraint provides a* lossless condensation *of all $n$-sets by only keeping the* most informative *ones.*

### 2.1.3 *Loss of the Galois Connection*

Unless $\mathcal{R}$ is a binary relation (i.e., $n = 2$), extracting every closed $n$-set cannot rely on a Galois connection (see Section 1.3.2 in Chapter 1). Indeed, several closed $n$-sets can share a same subset of elements from one attribute domain. For instance, Example 10 mentions both $(\{\alpha, \gamma\}, \{1, 2\}, \{A, B\})$ and $(\{\alpha, \beta, \gamma\}, \{1, 2\}, \{A\})$ are closed 3-sets in $\mathcal{R}_E$. They both involve the subset $\{1, 2\}$ of the second attribute domain.

Nevertheless, $n-1$ "components" of a closed $n$-set uniquely determines the last one. For instance, in $\mathcal{R}_E$, there exists only one closed 3-set that involves exactly $\{\alpha, \gamma\}$ and $\{1, 2\}$ (or $\{\alpha, \gamma\}$ and $\{A, B\}$ or $\{1, 2\}$ and $\{A, B\}$). The related functions are not injective. As a consequence, they are not part of Galois connections.

*Extracting every closed $n$-set does not look reducible to mining closed itemsets or multi-relational patterns.*

There does not seem to exist any "simple" bijection between $n$-ary relations and binary ones helping for the extraction of closed $n$-sets. Such a transformation would certainly lead to a combinatorial explosion of the number of elements. Indeed the attributes of the binary relation should combine several elements of the different domains to encompass the $n$-ary relation. It is trivial to turn an $n$-ary relation into $n$ binary relations: give an id to every $n$-tuple and relate this id with each of the $n$ elements. This transformation does not help the extraction of patterns. Indeed, in any of the constructed binary relations, every id is related to one and only one element in $\mathcal{D}^i$ ($i = 1..n$). In particular, in these binary relations, the only closed itemsets, with frequencies strictly greater than 0, have at most one element in $\mathcal{D}^i$ ($i = 1..n$). Anyway, the complete extraction of closed patterns in multi-relational settings is an interesting but difficult task. In particular, defining the closedness is problematic (see, e. g., [69] and [18]).

## 2.2 *State of the Art*

### 2.2.1 *Ad-hoc Methods for Ternary Relations*

[41] proposes two algorithms to extract every closed 3-set in ternary relations. The first one, called *Representative slice mining*, is simple and very inefficient. It consists in enumerating all subsets of the smallest attribute domain. For each of them, a binary relation is computed by bitwise AND operations between the elements of the subset. Then, any closed itemset extractor can be used on each of these relations and a post-processing step removes the 3-sets (the closed itemsets associated with the subset of elements that were used to generate the binary relation) that are not closed. The second algorithm, called CubeMiner, directly operates on the ternary relation $\mathcal{R}$. It consists in using the 3-sets $(X, Y, Z)$, called *cutters*, presenting the following particularity: none of the 3-tuples they cover are in $\mathcal{R}$. Thus, the authors generalize the notion of *cutter* introduced in [5] for closed itemset mining. CubeMiner first considers the whole ternary relation as a candidate pattern. Along a depth-first enumeration, the cutters are recursively applied to generate three candidate children containing less tuples absent from $\mathcal{R}$ than the parent: a first one without the elements in $X$, a second one without the elements in $Y$ and a third one without the elements in $Z$. For each child pattern, several checks are required to ensure its closedness and uniqueness. For the pattern to be unique, its newly removed elements must not be included in a cutter previously applied on this branch of the enumeration tree. To verify this, every formerly applied cutter is intersected with the current one. For the pattern to be closed, the elements of these formerly applied cutters should not extend it. Thus, every candidate pattern is twice compared to the formerly applied cutters. The related computational cost, at every enumeration node, grows linearly with the height of the enumeration tree (and, naturally, most of its nodes are at the bottom of it). Furthermore, the cutters

*CubeMiner and Trias extract every closed 3-set satisfying minimal size constraints. Both suffer from scalability issues.*

are not ordered in a particular way that could reduce the size of the enumeration tree and the running time.

TRIAS [40] extracts every closed 3-set in a ternary relations $\mathcal{R} \subseteq \mathcal{D}^1 \times \mathcal{D}^2 \times \mathcal{D}^3$. It relies on closed itemset extractions. Assuming $\mathcal{D}^1$ is the attribute domain with the smallest cardinality, TRIAS first constructs the binary relation $\mathcal{B}_1 \subseteq \mathcal{D}^1 \times (\mathcal{D}^2 \times \mathcal{D}^3)$ where $(x^1, x^2, x^3) \in \mathcal{R} \Leftrightarrow (x^1, (x^2, x^3)) \in \mathcal{B}$. Every closed itemset $(X^1, \mathcal{B}_2)$ is extracted from this relation. Let us name $D^2$ (resp. $D^3$) the elements in $\mathcal{D}^2$ (resp. $\mathcal{D}^3$) that are in at least one couple of $\mathcal{B}_2$. Usually, $\mathcal{B}_2$ is different from $D^2 \times D^3$, i. e., $(X^1, D^2, D^3)$ is not connected. That is why, in a second step, TRIAS extracts every closed itemset in $\mathcal{B}_2$ (considered as a binary relation on $D^2 \times D^3$). Every closed itemset $(X^2, X^3)$ in $\mathcal{B}_2$ form, with $X^1$, a connected 3-set in $\mathcal{R}$. However, $(X^1, X^2, X^3)$ is not necessarily closed w.r.t. $\mathcal{D}^1$. To finally output or filter out the 3-set, TRIAS checks whether $(X^2, X^3)$ is connected in any of the binary relations related to the elements in $\mathcal{D}^1 \setminus X^1$. The more elements, in average, in $\mathcal{D}^1 \setminus X^1$, the more unclosed 3-set that are generated before being discarded. That is why the running time of TRIAS is much dependent on the size of the smallest attribute domain.

### 2.2.2 *Minimizing Relations*

*Closed $n$-sets somehow summarize parts of the relation. That is why mining them relates to simplifying multi-valued logic functions.*

Considering the $n$ sets $(\mathcal{D}^i)_{i=1..n}$ as the domains of $n$ multi-valued variables, the relation $\mathcal{R}$ can be seen as the truth table of a multi-valued logic function with $\{0, 1\}$ as a range. Boolean functions are a specialization of this framework where every domain gathers two elements (usually bound to the semantics "true" and "false"). The *Karnaugh map* [44] is a tool to simplify such Boolean functions. This method is to be applied by hand ("by eye" would be more correct since it exploits the human capability to discern geometrical patterns). For this reason, it works well for up to four variables but becomes unpractical for more than six variables. It relies on organizing the truth table in such a way that every maximal rectangle of '1' gives a *prime implicant* (a disjunction of conjunctions) *tiling* the part of the Boolean function responsible for the '1's in the rectangle. Once every '1' is covered by at least one prime implicant, the disjunction of the prime implicants is a simplification of the original Boolean function. The *Quine-McCluskey algorithm* [54] was designed to deal with more variables. The procedure basically remains the same. However, the organization of the truth table, used in the *Karnaugh map*, is substituted by a tabular form, which better suits computers' way of processing data. This algorithm always returns the minimal form of the Boolean function to the cost of finding all prime implicants. ESPRESSO [14] uses a different approach. The returned function is not always the minimal form (but close to it) and the computation is reduced (in both space and time) by orders of magnitude. It is still heavily used, in particular in Programmable Logic Devices. ESPRESSO was generalized in ESPRESSO-MV [73] to deal with multi-valued logic functions. Indeed, multiple-valued logic functions have applications in several areas. The most successful one probably is the enhancement of circuit performances in terms of chip area, operation speed and power consumption (see, e. g., [24]).

The data mining community recently granted some attention to tiling binary relations (i. e., 2-variable multi-valued logic functions). In particular, [30] defined several problems related to the minimization of

binary relations through collections of large enough (area constraint) patterns. "Itemsets that compress" [75] now are a new paradigm of data mining. They were used to classify [85], to derive emerging patterns [88], to anonymize data [89], to treat missing values [87], to find groups [86], etc.

## 3 MINING CLOSED PATTERNS IN *noisy* $n$-ARY RELATIONS

### 3.1 *Tolerating Noise Is a Must*

*On relations of higher arity, noise scourges more and more the collections of closed $n$-sets.*

When mining relations of higher arity, noise becomes more and more problematic. First of all, the patterns to discover encompass exponentially more $n$-tuples. As a consequence, there is a higher probability that some of them were affected. Moreover, at a fixed number of false negative $n$-tuples, the number of closed $n$-sets linearly increases with $n$ ($\mathcal{C}_{\text{closed}}$ is a maximality constraint w.r.t. *every* attribute). When $n$-ary relations are derived from numerical data, the samples of data that decide whether a property holds (i. e., whether the related $n$-tuple should be in the relation) gets smaller. For example, in a transactional context, deciding whether a category of customers frequently buys an item (binary relation binding categories of customers and items) is less prone to noise than deciding whether a category of customers *from a particular country* frequently buys an item (ternary relation binding categories of customers, countries and items) because the former decision is based on more data. Despite this growing necessity to tolerate noise when mining relations of higher arity, the topic has not been much studied in the literature yet.

### 3.2 *State of the Art*

#### 3.2.1 *Complete Extractors*

*DCE extracts dense patterns in tensors. The chosen definition matches some irrelevant patterns and the condensation by closure is lossy.*

DCE [31] extracts, from real-valued tensors, every *dense $n$-set*. Having a density greater than $\alpha \in \mathbb{R}$ is a constraint similar to $\mathcal{C}_{\text{avg-gp}'\geqslant\alpha}$, defined in Section 2.2 of Chapter 1 and proved piecewise (anti)-monotone in Section 2.3.6 of the same chapter. This constraint can also be seen as the weak connection (see Definition 18) generalized to real-valued tensors. It suffers from the same relevancy problem, i. e., an element can be part of a pattern but disconnected or very weakly connected with the other attributes of the pattern. DCE does not exploit the piecewise (anti)-monotonicity of the minimal density constraint but some kind of generalized loose anti-monotonicity (see Section 2.3.4 in Chapter 1). Indeed, given a dense $n$-set, there exists a "sub-pattern" with one element less that is dense. This element, from any attribute domain (hence, the "generalization"), relates to the hyper-plan of the pattern with the smallest sum of the real values it contains, hence the definition of the max function (see Section 2.3.4 in Chapter 1). Thus, DCE is depth-first and, at every recursive call, enumerates $n$-sets with such an additional element. In this way, every candidate pattern is traversed once, the minimal density constraint becomes anti-monotone on any enumeration branch and prunes the search space. Because the tolerance to noise is relative, a closedness constraint would not provide a lossless condensation of the dense $n$-sets and reducing its test to the patterns with one additional element (from any attribute domain) is

not correct. Anyway, the authors propose such a procedure to reduce the problematic sizes of the output collections.

Given an $n$-ary relation, a *subspace cluster* is a *local pattern* (rather than *cluster*) of the form $(X^i)_{i=1..m}$ where every $X^i$ is a subset of a different attribute domain and $m \leqslant n$. It constrains the data restricted to every *pair* of elements to be *strongly connected*. Besides the number of tests required to be a subspace cluster (product of the number of elements per attribute), these tests involve the whole relation. More precisely, in a subspace cluster, every pair of elements (whatever the attribute domain(s) they belong to) must frequently appear together in the relation. This frequency is defined w.r.t. its expected value obtained by making the assumption of a uniform distribution of the $n$-tuples in the relation. CACTUS [28] and CLICKS [96] extract *maximal* (closedness constraint) subspace clusters from arbitrary $n$-ary relations. The latter generalizes the former that only mines a restricted class of subspace clusters.

*Closed ET-$n$-sets are less constrained than subspace clusters (with one constraint per* pair *of elements). However, the latter does not necessarily have $n$ dimensions.*

### 3.2.2 *Agglomerating 3-Sets*

TRICLUSTER [99] is an extension of MicroCluster, described in Section 1.2.2. It discovers complete collections of local patterns in real-valued tensors. In particular, it can extract the closed 3-sets. A post-processing step is proposed to handle noise by deleting or merging some of the extracted patterns. The involved distances are based on counts of 3-tuples. For example, two patterns are merged when the number of 3-tuples covered by at least one of the two patterns is greater than $\gamma$ times the number of tuples belonging to the envelope of the two patterns but not to any of the two patterns. Notice that this metric (in the pattern space) is solely based on the patterns, i. e., the agglomeration ignores the regions of the data that are not in the relation subspaces described by any of the two patterns. In particular, the number and the repartition of the couples absent from $\mathcal{R}$, but covered by the agglomerated patterns, is not taken into consideration.

*Heuristic agglomerations of closed itemsets are easily generalized to closed $n$-sets.*

## 4 CONCLUSION

Generalizing closed itemset mining towards noise tolerance and $n$-ary relations ($n \geqslant 2$) is challenging. To avoid matching irrelevant patterns, the noise tolerance sets upper-bounds for the number (or the proportion) of couples absent from the relation in *every* restriction of the patterns to its individual elements (objects and properties). On the contrary, defining a closed $n$-set is rather natural. Nevertheless, in both cases, complete extractors cannot rely on Galois connections and running times are problematic. To tolerate more noise than what complete extractors can achieve (while remaining tractable), patterns can be heuristically agglomerated. To do so, a notion of distance between the patterns must be defined.

# Part III

# MINING N-ARY RELATIONS

# OUTLINE

The previous chapter argued for two needed generalizations of closed itemset mining: towards noise tolerance and towards $n$-ary relations. This thesis eventually proposes a solution to both issues together. For the time, this part focus on generalizing (exact) closed itemset mining towards relations of higher arity. Thus, it deals with the complete extraction of closed patterns that hold in $n$-ary relations. This type of pattern, namely the closed $n$-set, was defined in Section 2 of the previous chapter. This section also emphasized that none of the previous work on local pattern extraction can efficiently deal with relations having an arity beyond 3. Data-Peeler is the first algorithm that was designed to extract closed $n$-sets whatever $n \geqslant 2$. It does not make any assumption on the proportions of the dataset either. In fact, Data-Peeler does not favor, a priori, any attribute or set of attributes. Despite its generic scope, Data-Peeler turns out to be, on ternary relations, orders of magnitude faster than its competitors. Section 2 in Chapter 1 emphasized that the enumeration principles of an extractor determines the class of constraints it can efficiently enforce. In the case of Data-Peeler, this class, namely the piecewise (anti)-monotone constraints, is very broad and the analyst can choose among many useful relevancy constraints. Both Data-Peeler and the class of piecewise (anti)-monotone constraints were first published in [CBRB08]. However some improvements to Data-Peeler's enumeration were added in an extended article [CBRB09].

When the original data are numerical, they need, in a pre-processing step, to be converted into Boolean properties so that Data-Peeler can be used. Different binarization methods lead to different perspectives on the data. Considering several methods altogether (they form an additional dimension to the dataset) and mining, with Data-Peeler, patterns that are frequent across them, is a way to specify a certain robustness w.r.t. binarization. [CBRB09] introduced this idea. This article also presents a heuristic method to globally model the dataset by post-processing the local patterns it contains. It is about obtaining a tiling, i. e., a coverage of the whole dataset by a minimal collection of patterns. In the context of $n$-ary relations, this problem is equivalent to the minimization of multi-valued logic functions. Some previous works ensure the optimality of the minimization but are intractable on medium-size datasets. That is why a greedy procedure is proposed. It provides, in reasonable times, tilings (of arbitrary $n$-ary relations) that are better than the state of the Art Espresso-MV.

# 3 DATA-PEELER: THE FIRST CLOSED N-SET EXTRACTOR

## 1 DATA-PEELER

### 1.1 *A Closed n-Set Extractor*

DATA-PEELER extracts every closed $n$-set in an arbitrary $n$-ary relation $\mathcal{R} \subseteq \times_{i=1..n}\mathcal{D}^i$. Section 2.1.2 in Chapter 2 discussed the (natural) definition of a closed $n$-set. Let us recall it. An $n$-set $X = (X^i)_{i=1..n} \in \times_{i=1..n}2^{\mathcal{D}^i}$ is a closed $n$-set if and only if:

- $\mathcal{C}_{\text{connected}}(X) \equiv \times_{i=1..n}X^i \subseteq \mathcal{R}$;
- $\mathcal{C}_{\text{closed}}(X) \equiv \forall i = 1..n, \forall s \in \mathcal{D}^i \setminus X^i$, $\neg\mathcal{C}_{\text{connected}}(X^1, \ldots, \{s\}, \ldots, X^n)$, i. e., $X^1 \times \cdots \times \{s\} \times \cdots \times X^n \not\subseteq \mathcal{R}$.

*DATA-PEELER extracts every closed n-set.*

### 1.2 *Enumeration*

Like many complete algorithms for constraint-based local pattern mining, DATA-PEELER is based on enumerating candidates in a way that can be represented by a binary tree where (a) at every node, an element $e$ is enumerated; (b) every pattern extracted from the left child *does contain* $e$; (c) every pattern extracted from the right child does *not* contain $e$. This leads to a partition of the search space, i. e., the union of the closed $n$-sets found in both enumeration sub-trees are exactly the closed $n$-sets to be extracted from the parent node (*correctness*) *and* each of these closed $n$-sets is found only once (*uniqueness*). In the case of DATA-PEELER, the enumerated element $e$ can always be freely chosen among all the elements (from all attribute domains $\mathcal{D}^1$, $\mathcal{D}^2$, ..., $\mathcal{D}^n$) remaining in the search space.

*The enumeration follows a binary tree. At every node, both the smallest and the greatest n-set in the sub-tree are known and any element from any attribute domain can be chosen to extend the smallest n-set.*

Each node $N$ in the enumeration tree is a pair $(U, V)$ where $U$ and $V$ are two $n$-sets. $N$ *represents* all the $n$-sets containing all the elements of $U$ and a subset of the elements of $V$. In other words, this is the search space of the $n$-sets $(X^1, \ldots, X^n)$ s.t. $\forall i = 1..n$, $U^i \subseteq X^i \subseteq U^i \cup V^i$. The root node, $((\emptyset, \ldots, \emptyset), (\mathcal{D}^1, \ldots, \mathcal{D}^n))$, represents all possible $n$-sets. On the contrary, nodes such that $\forall i = 1..n$, $V^i = \emptyset$ represent a single $n$-set, $(U^1, \ldots, U^n)$. More generally, a node $(U, V)$ represents $2^{\sum_{i=1}^{n}|V^i|}$ $n$-sets.

**Example 11** *The node* $M = (U, V) = (\{\{\alpha\}, \emptyset, \{C\}\}, (\{\gamma\}, \{1, 4\}, \{A, B\}))$ *represents* $2^5$ *(i. e., 32) 3-sets. E. g., it represents the 3-sets* $(\{\alpha\}, \emptyset, \{C\})$, $(\{\alpha\}, \{4\}, \{C\})$ *and* $(\{\alpha, \gamma\}, \{1, 4\}, \{A, B, C\})$. *On the contrary, it represents neither* $(\{\alpha\}, \emptyset, \emptyset)$ *(*$C$ must *be in the 3-set) nor* $(\{\alpha, \beta, \gamma\}, \{4\}, \{C\})$ *(*$\beta$ must not *be in the 3-set).*

At a node $N = (U, V)$, DATA-PEELER chooses an element $e$ from $V$ (the selection criterion is discussed in Section 1.6) and generates two new nodes, $N_L = (U_L, V_L) = (U \cup \{e\}, V \setminus \{e\})$ and $N_R = (U_R, V_R) = (U, V \setminus \{e\})$. $N_L$ (resp. $N_R$) represents the $n$-sets of $N$ that contain (resp. do not contain) $e$. Figure 10 depicts this simple partitioning of the search space.

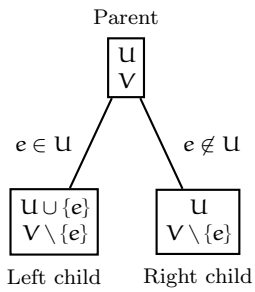



Figure 10: Enumeration of any element $e \in V$.

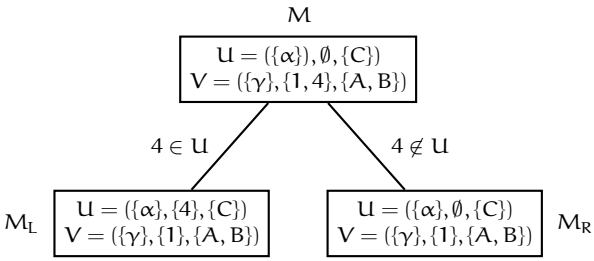



Figure 11: Enumeration of the element $4 \in V^2$ from node $M$ (Example 12).

**Example 12** *Considering the node $M$ of Example 11, the selection of the element $4 \in V^2$ leads to the two nodes $M_L = (\{\{\alpha\}, \{4\}, \{C\}\}, \{\{\gamma\}, \{1\}, \{A, B\}\})$ and $M_R = (\{\{\alpha\}, \emptyset, \{C\}\}, \{\{\gamma\}, \{1\}, \{A, B\}\})$ (see Figure 11).*

### 1.3 *Efficient Enforcement of $\mathcal{C}_{connected}$*

*$\mathcal{C}_{connected}$ reduces the search space when an element is claimed present.*

$\mathcal{C}_{\text{connected}}$ allows to reduce the search space of the left child, i. e., the size of $V_L$. In other terms, some $n$-sets, that are represented by the parent node, are not to be represented by the left child. Indeed, the elements of $V$ that violate $\mathcal{C}_{\text{connected}}$ if added to $U_L$ can be removed from $V_L$. Formally, these elements are $\{v \in V_L \mid U_L^1 \times \cdots \times \{v\} \times \cdots \times U_L^n \not\subseteq \mathcal{R}\}$. They are found in the following way: $\forall v \in V$, whenever an element $e$ is moved from $V$ to $U$, if $U^1 \times \cdots \times \{e\} \times \cdots \times \{v\} \times \cdots \times U^n \not\subseteq \mathcal{R}$ then $v$ is removed from $V$. In this way, at any enumeration node, $U$ can "receive" any element from $V$ without violating $\mathcal{C}_{\text{connected}}$. Figure 12 depicts this enforcement of $\mathcal{C}_{\text{connected}}$, which ensures that every $n$-set satisfying this constraint is browsed once (and only once).

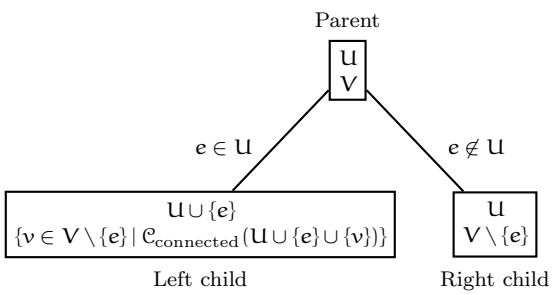



Figure 12: Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$.

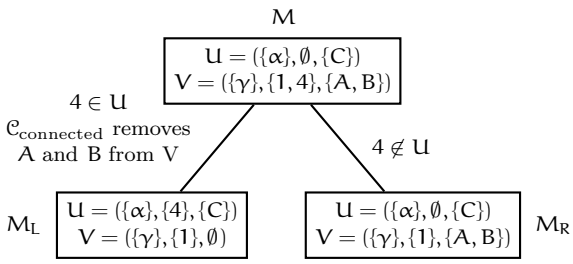



Figure 13: Enumeration of the element $4 \in V^2$ from node $M$ (Example 13).

**Example 13** *In our running example, according to $\mathcal{R}_E$ (see Table 5), neither the element $A$ nor the element $B$ can be added to $U_L = (\{\alpha\}, \{4\}, \{C\})$ to form a 3-set satisfying $\mathcal{C}_{connected}$. Indeed, $(\alpha, 4, A) \notin \mathcal{R}_E$ and $(\alpha, 4, B) \notin \mathcal{R}_E$. As a consequence, $\mathcal{C}_{connected}$ removes those two elements from $V^3$: we finally obtain $M_L = ((\{\alpha\}, \{4\}, \{C\}), (\{\gamma\}, \{1\}, \emptyset))$ (see Figure 13).*

Until now, we discussed how to extract all $n$-sets satisfying $\mathcal{C}_{\text{connected}}$ in $n$-ary relations. We now need to enforce the closedness property.

### 1.4 *Efficient Enforcement of $\mathcal{C}_{closed}$*

For a better performance, the closedness constraint must be handled during the enumeration process (safe pruning) rather than in a post-processing phase. At a given enumeration node $N$, if there exists an element $s \in \mathcal{D}^j \setminus (U^j \cup V^j)$ such that $\mathcal{C}_{\text{connected}}(U \sqcup V \cup \{s\})$ is satisfied, then every $n$-set represented by $N$ can be extended with $s$ to form a larger $n$-set satisfying $\mathcal{C}_{\text{connected}}$. Indeed, $\forall V' \sqsubseteq V, \forall s \in \mathcal{D}^j \setminus (U^j \cup V^j)$, $\mathcal{C}_{\text{connected}}(U \sqcup V \cup \{s\}) \Rightarrow \mathcal{C}_{\text{connected}}(U \sqcup V' \cup \{s\})$. None of the $n$-sets $N$ represents being closed, the whole enumeration sub-tree rooted by $N$ is safely pruned. DATA-PEELER does not miss the closed $n$-sets "containing" $U$: they are found in the part of the enumeration tree where $s \in U$.

*An enumeration sub-tree is pruned when its largest closed $n$-set can be extended by an element out of the search space.*

Given an element $s$ that potentially extends every $n$-set represented by the current node, there is no need to browse the whole subspace of the relation related to $U \sqcup V \cup \{s\}$. Indeed, because $\mathcal{C}_{\text{connected}}(U \sqcup V)$ is always true (see Section 1.3), all its $n$-tuples absent from $\mathcal{R}$ involve $s$. As a consequence, browsing $(U^1 \cup V^1) \times \cdots \times \{s\} \times \cdots \times (U^n \cup V^n)$ is enough. Furthermore, as soon as one of its $n$-tuples is found missing from $\mathcal{R}$, the check is aborted ($s$ does not prevent the closure of every $n$-set represented by the current node).

Furthermore, one the most interesting advantage of our enumeration strategy is that there is actually no need to check whether every element in $(\cup_{i=1..n} \mathcal{D}^i) \setminus (\cup_{i=1..n} U^i \cup V^i)$ may prevent the closure. Indeed, any element that has been removed from $V$ thanks to $\mathcal{C}_{\text{connected}}$ (see Section 1.3) cannot. Indeed, the reason of the removal of such an element $f$ from $V$ is that $\mathcal{C}_{\text{connected}}(U \cup \{f\})$ is false. In such circumstances, $\mathcal{C}_{\text{connected}}(U \sqcup V \cup \{f\})$ cannot be true either. When checking $\mathcal{C}_{\text{closed}}$ the only elements that need to be tried as extensions are those that were previously chosen to be enumerated but refused (right child). These elements are stored in an $n$-set that will always be denoted $\mathcal{S}$. Figure 14 complements Figure 12 with this $n$-set $\mathcal{S}$. Given the three $n$-sets $U$, $V$

*Only the elements refused "by enumeration" may prevent the closedness.*

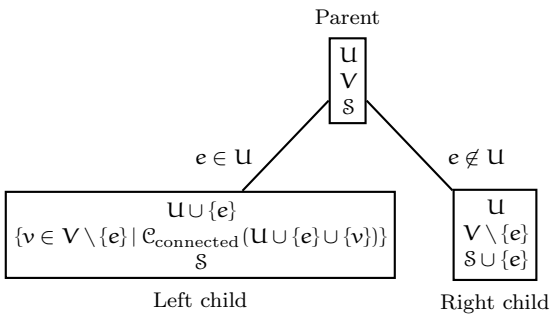



Figure 14: Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$. $\mathcal{C}_{\text{closed}}$ is checked on $U \sqcup V$ extended with every element in $S$.

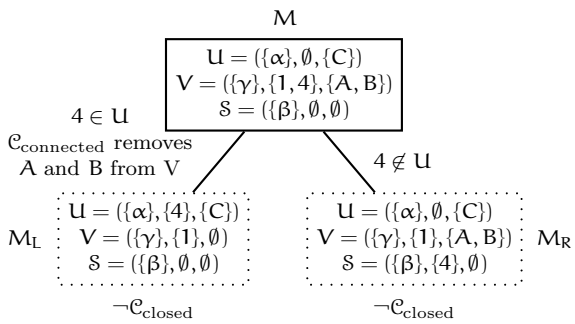



Figure 15: Enumeration of the element $4 \in V^2$ from node $M$ (Example 14).

and $S$ attached to an enumeration node, the closedness constraint is checked as follows:

$$\forall s \in S, U^1 \cup V^1 \times \cdots \times \{s\} \times \cdots \times U^n \cup V^n \not\subseteq \mathcal{R} \ .$$

**Example 14** *Still using the running example, assume that $S = (\{\beta\}, \emptyset, \emptyset)$ is bound to the enumeration node $M$. Neither $M_L$ nor $M_R$ satisfies $\mathcal{C}_{closed}$. Indeed $(\{\beta\}, \{1, 4\}, \{C\})$ is connected and so is $(\{\beta\}, \{1\}, \{A, B, C\})$ (see Table 5). Figure 15 depicts the enumeration aborted for the two children nodes. To conclude, among the 32 3-sets represented by $M$, none are both connected and closed.*

## 1.5 *Algorithm*

Data-Peeler is a depth-first search algorithm. It takes three arguments: $U$, $V$ and $S$. It starts with $U_0 = (\emptyset, \ldots, \emptyset)$, $V_0 = (\mathcal{D}^1, \ldots, \mathcal{D}^n)$ and $S_0 = (\emptyset, \ldots, \emptyset)$. Its major steps are presented in the pseudo-code of Figure 16, which can be seen as a translation of the diagram of Figure 14. First of all, the closedness property is checked (see Section 1.4). If it is satisfied and no element remains to be enumerated, the $n$-set $U$ is output. Otherwise an element $e$ of $V$ is chosen (Section 1.6 discusses this step) and the search space is split between the $n$-sets that contain $e$ and those that do not (see Section 1.2). Finally, Data-Peeler is recursively called on the two related enumeration nodes. Notice that the $n$-set $S$ is only fed by the elements $e$ that are chosen to be enumerated but refused (right child). The elements that $\mathcal{C}_{\text{connected}}$ removes from the search space $V$ are not moved to $S$ since they cannot prevent the descendant nodes to be closed.

**Input:** $U, V, \mathcal{S}$
**Output:** Every closed $n$-set represented by $(U, V)$
**if** $\mathcal{C}_{\text{closed}}(U \sqcup V)$ **then**
  **if** $V = (\emptyset, \ldots, \emptyset)$ **then**
    **output**$(U)$
  **else**
    Choose $e \in V$
    DATA-PEELER$(U \cup \{e\}, \{v \in V \setminus \{e\} \mid \mathcal{C}_{\text{connected}}(U \cup \{e\} \cup \{v\})\}, \mathcal{S})$
    DATA-PEELER$(U, V \setminus \{e\}, \mathcal{S} \cup \{e\})$
  **end if**
**end if**

Figure 16: The DATA-PEELER algorithm.

## 1.6 *Choosing the Element to Enumerate*

As explained in Section 1.2, an element $e \in V$ must be chosen to be enumerated. Its choice determines the two nodes $N_L$ and $N_R$ deriving from the current one. The more elements their $V$ $n$-sets contain, the greater the remaining search space. $V_R$ always contain $|V| - 1$ elements. That is why DATA-PEELER's selection strategy for $e$ focuses on minimizing the number of elements in $V_R$, i. e., it aims at maximizing the number of elements $\mathcal{C}_{\text{connected}}$ removes from the search space when $e$ is set present.

*The enumerated element is heuristically chosen such that the search space at the left node is minimized.*

Whenever an element is enumerated, $\mathcal{C}_{\text{connected}}$ removes some elements if (a) they are in $V$ and (b) elements from the $n-1$ other attributes are in $U$. The following formula gives the maximum number of $n$-tuples in $\mathcal{R}$ that are browsed when enforcing $\mathcal{C}_{\text{connected}}$ after an element from $V^d$ is enumerated:

$$\sum_{k \neq d} \left( |V^k| \times \prod_{l \notin \{d,k\}} |U^l| \right) .$$

DATA-PEELER enumerates an element on the attribute domain $d$ maximizing this formula. The choice for an element $e \in V^d$ remains. It is the one (or one of those) presenting the lowest density in $\mathcal{R}$, i. e., an element $e \in V^d$ minimizing the expression below:

$$|(\mathcal{D}^1 \times \cdots \times \{e\} \times \cdots \times \mathcal{D}^n) \cap \mathcal{R}| .$$

This heuristic generalizes the one presented in Section 2.3.3 of Chapter 1 and can be justified as follows: the less elements are connected in $\mathcal{R}$, the more likely $\mathcal{C}_{\text{connected}}$ removes elements from $V$ to build $V_L$. The experiment in Section 5.2 empirically shows that the proposed choice criterion outperforms other sensible criteria.

**Example 15** *In our running example, at node M, the choice of enumerating $4 \in V^2$ actually follows the heuristic stated above:*

CHOICE OF $v^2$: $\sum_{k \neq d} \left( |V^k| \times \prod_{l \notin \{d,k\}} |U^l| \right)$ *is maximized for* $d = 2$*:*

$d = 1$: $(|V^2| \times |U^3|) + (|V^3| \times |U^2|) = (2 \times 1) + (2 \times 0) = 2$;

$d = 2$: $(|V^1| \times |U^3|) + (|V^3| \times |U^1|) = (1 \times 1) + (2 \times 1) = 3$;

$d = 3$: $(|V^1| \times |U^2|) + (|V^2| \times |U^1|) = (1 \times 0) + (2 \times 1) = 2$.

CHOICE OF 4: $|(\{\alpha, \beta, \gamma\} \times \{e\} \times \{A, B, C\}) \cap \mathcal{R}|$ *is minimized for* $e = 4$*:*

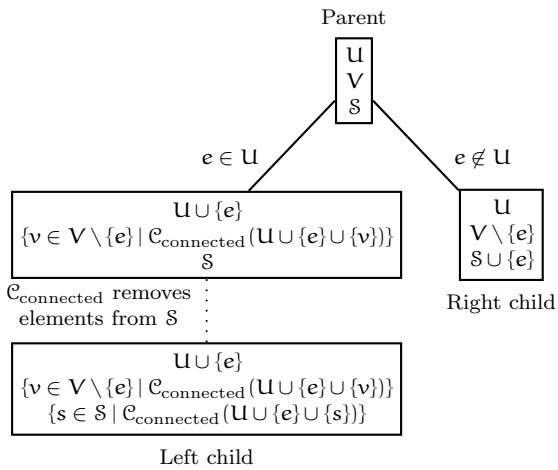



Figure 17: Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$ and $\mathcal{S}$. $\mathcal{C}_{\text{closed}}$ is checked on $U \sqcup V$ extended with every element in $\mathcal{S}$.

$$e = 1\colon |\{(\alpha,1,A),(\alpha,1,B),(\alpha,1,C),(\beta,1,A),(\beta,1,B),(\beta,1,C),(\gamma,1,A),(\gamma,1,B)\}| = 8;$$

$$e = 4\colon |\{(\alpha,4,C),(\beta,4,A),(\beta,4,C),(\gamma,4,A),(\gamma,4,B),(\gamma,4,C)\}| = 6.$$

## 2 IMPROVEMENTS TO THE ENUMERATION

### 2.1 *Removing Elements from $\mathcal{S}$*

*Given an enumeration sub-tree, the elements that cannot extend its smallest n-set, cannot prevent any of them to be closed.*

Every $n$-set represented by a node $N = (U, V)$ "contains" $U$. As a consequence, the elements in $\mathcal{S}$ that violate $\mathcal{C}_{\text{connected}}$ when added to $U$, will not enlarge any $n$-set represented by $N$. They can be removed from $\mathcal{S}$. Formally, these elements, which are safely removed from $\mathcal{S}$, are $\{s \in \mathcal{S} \mid U^1 \times \cdots \times \{s\} \times \cdots \times U^n \not\subseteq \mathcal{R}\}$. They are found in the following way: $\forall s \in \mathcal{S}$, whenever an element $e$ is moved from $V$ to $U$, if $U^1 \times \cdots \times \{e\} \times \cdots \times \{s\} \times \cdots \times U^n \not\subseteq \mathcal{R}$ then $s$ is removed from $\mathcal{S}$. This process is similar to the enforcement of $\mathcal{C}_{\text{connected}}$ (see Section 1.3) but applied on $\mathcal{S}$ instead of $V$. This optimization speeds up the enforcement of $\mathcal{C}_{\text{closed}}$ for all the nodes deriving from $N$. The gain is two-fold:

- $\mathcal{S}$ containing less elements, the global cost pertaining to the enforcement of $\mathcal{C}_{\text{closed}}$ is lowered;
- When enforcing $\mathcal{C}_{\text{closed}}$, there is no need to browse $\{U^1 \times \cdots \times \{s\} \times \cdots \times U^n \mid s \in \mathcal{S}\}$: all these tuples are present otherwise some $s \in \mathcal{S}$ would have been removed by this optimization.

This improvement slightly modifies DATA-PEELER's enumeration. The enumeration taking advantage of it is depicted in Figure 17.

**Example 16** *Given the node* $(U, V) = (\{\alpha,\gamma\},\{1,2\},\{B\}),(\emptyset,\emptyset,\{A\}))$ *and the relation* $\mathcal{R}_E$ *represented in Table 5, assume that* $\mathcal{S} = (\{\beta\},\emptyset,\emptyset)$. $\beta$ *is removed from* $\mathcal{S}^1$ *because* $\mathcal{C}_{connected}(\{\beta\},\{1,2\},\{B\})$ *is false (see Figure 18).*

### 2.2 *Moving Elements from $V$ to $U$*

Every $n$-set represented by a node $N = (U, V)$ is "included in" (see Definition 23) $U \sqcup V$. As a consequence, an element of $V^i$ which, in $\mathcal{R}$,

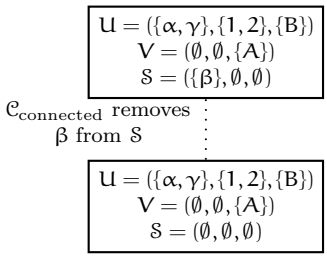



Figure 18: Illustration of Example 16.

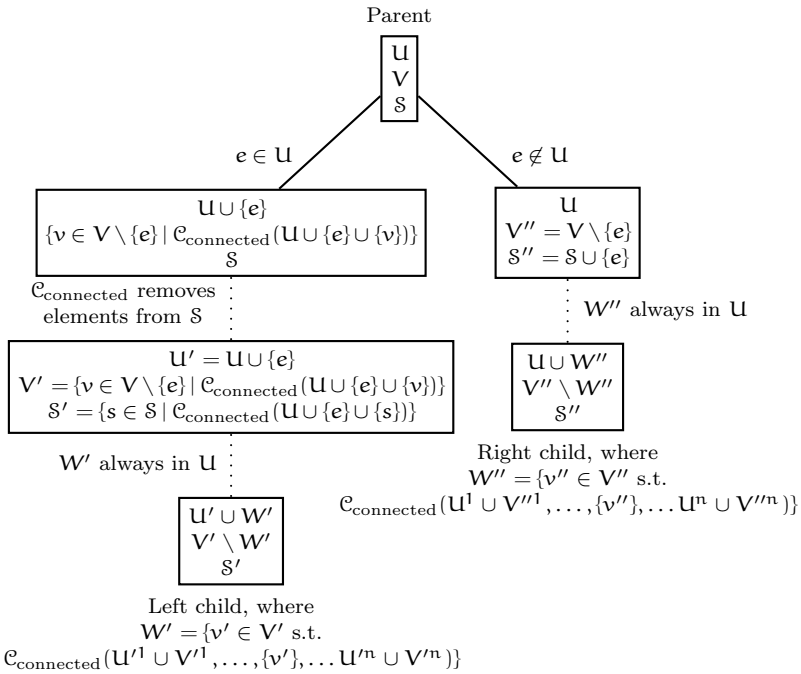



Figure 19: Enumeration of any element $e \in V$. $\mathcal{C}_{\text{connected}}$ removes elements from $V$ and $S$. The elements of $V$ that are necessarily present are moved to $U$. $\mathcal{C}_{\text{closed}}$ is checked on $U \sqcup V$ extended with every element in $S$.

is associated to all the elements of $\times_{j \neq i} U^j \cup V^j$ is necessarily element of every closed $n$-set represented by $N$. It can be moved to $U$. Thus, these elements, which can be moved to $U$, are $\{v \in V \mid U^1 \cup V^1 \times \cdots \times \{v\} \times \cdots \times U^n \cup V^n \subseteq \mathcal{R}\}$. In the worst case (all the elements of $V$ can extend $N$), given $N$, finding the elements of this set means checking the presence in $\mathcal{R}$ of this number of $n$-tuples:

*Given an enumeration sub-tree, the elements that extend its largest $n$-set are in every closed $n$-set.*

$$\sum_{k=1}^{n} \left( |V^k| \times \prod_{l \neq k} |U^l \cup V^l| \right) .$$

This cost may look high. However, recall that the enumeration sub-tree whose root is $N$ contains, at worse (no pruning), $2^{1+\sum_{i=1}^{n} |V^i|} - 1$ nodes. That is why removing elements from $V$ as soon as possible significantly reduces the number of nodes to consider and, as a consequence, the running time of DATA-PEELER.

Again, this improvement modifies the enumeration of DATA-PEELER. Figure 19 depicts the enumeration taking advantage of the two improvements that have just been described.

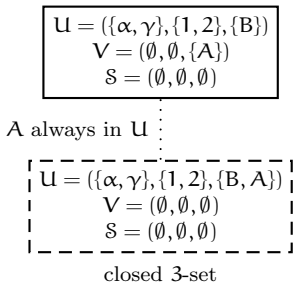



Figure 20: Illustration of Example 17.

**Example 17** *Consider the node* $((\{\alpha, \gamma\}, \{1, 2\}, \{B\}), (\emptyset, \emptyset, \{A\}))$ *obtained in Example 16. The element* $A$ *is safely moved from* $V^3$ *to* $U^3$ *(see Figure 20). Indeed,* $(\{\alpha, \gamma\}, \{1, 2\}, \{A\})$ *satisfies* $\mathcal{C}_{connected}$ *(see Table 5). Once this second improvement applied,* $(\{\alpha, \gamma\}, \{1, 2\}, \{B\})$ *is claimed closed 3-set. Indeed* $V$ *does not contain any element and neither does* $S$*, hence the closedness.*

## 2.3 *Improved Algorithm*

Figure 21 details, at a high level of abstraction, how DATA-PEELER recursively extracts every closed $n$-set. It is similar to Figure 16 but includes the two improvements that have just been presented. In this respect, it can be seen as a pseudo-code translation of Figure 19. A constraint $\mathcal{C}_{P(A)M}$ was added too. This user-defined constraint aims at focusing DATA-PEELER on the relevant closed $n$-sets, i. e., every closed $n$-set that satisfies $\mathcal{C}_{P(A)M}$. Furthermore, along the extraction, $\mathcal{C}_{P(A)M}$ guides the search, i. e., the search space is pruned when it is certain the related region (the $n$-sets represented by the current enumeration node) is empty of $n$-set satisfying $\mathcal{C}_{P(A)M}$. In many practical settings with large domain sizes and/or high densities, the use of constraints is a key to extraction tractability. At any call the smallest and the largest $n$-set that may be recursively considered are known (respectively $U$ and $U \sqcup V$). As a consequence DATA-PEELER can handle any piecewise (anti)-monotone constraint (see Section 2.3.6 in Chapter 1), i. e., $\mathcal{C}_{P(A)M}$ can be any constraint in this very broad class.

DATA-PEELER *efficiently handles any piecewise (anti)-monotone constraint.*

The definition of piecewise (anti)-monotonicity (i. e., Definition 11), given in Chapter 1, is sufficiently generic to apply to $n$-sets. It is not recalled here. The enforcement of $\mathcal{C}_{P(A)M}$ in DATA-PEELER is analog with the one presented in Section 2.3.6 of Chapter 1, i. e., $\mathcal{C}'_{P(A)M}$ is $\mathcal{C}_{P(A)M}$ rewritten such that every occurrence of its variables is attributed a separate argument that is instantiated with the largest (if $\mathcal{C}'_{P(A)M}$ is monotone w.r.t. this argument) or the smallest (if $\mathcal{C}'_{P(A)M}$ is anti-monotone w.r.t. this argument) subset of the same attribute domain that may be recursively considered from the current call. In other terms, every argument of $\mathcal{C}'_{P(A)M}$ that ranges in $2^{\mathcal{D}^i}$ ($i = 1..n$) is instantiated with:

- $U^i \cup V^i$ if $\mathcal{C}'_{P(A)M}$ is monotone w.r.t. this argument;
- $U^i$ if $\mathcal{C}'_{P(A)M}$ is anti-monotone w.r.t. this argument.

**Example 18** *Consider the following constraint forcing the extracted closed* $n$*-sets to be globally large:*

$$\mathcal{C}_{\nu\text{-}volume}(X^1, \ldots, X^n) \equiv |X^1 \times \cdots \times X^n| \geqslant \nu \ .$$

**Input:** $U, V, S$
**Output:** Every closed $n$-set represented by $(U, V)$ and satisfying $\mathcal{C}_{P(A)M}$
**if** $\mathcal{C}'_{P(A)M}$ is satisfied when instantiated as detailed in the text $\wedge \mathcal{C}_{closed}(U \sqcup V)$ **then**
  **if** $V = (\emptyset, \ldots, \emptyset)$ **then**
    **output**$(U)$
  **else**
    Choose $e \in V$
    $U' \leftarrow U \cup \{e\}$
    $V' \leftarrow \{v \in V \setminus \{e\} \mid \mathcal{C}_{connected}(U \cup \{e\} \cup \{v\})\}$
    $S' \leftarrow \{s \in S \mid \mathcal{C}_{connected}(U \cup \{e\} \cup \{s\})\}$
    $W' \leftarrow \{v' \in V' \mid \mathcal{C}_{connected}(U'^1 \cup V'^1, \ldots, \{v'\}, \ldots U'^n \cup V'^n)\}$
    DATA-PEELER$(U' \cup W', V' \setminus W', S')$
    $V'' \leftarrow V \setminus \{e\}$
    $S'' \leftarrow S \cup \{e\}$
    $W'' \leftarrow \{v'' \in V'' \mid \mathcal{C}_{connected}(U^1 \cup V''^1, \ldots, \{v''\}, \ldots U^n \cup V''^n)\}$
    DATA-PEELER$(U \cup W'', V'' \setminus W'', S'')$
  **end if**
**end if**

Figure 21: The DATA-PEELER improved algorithm.

*Because every argument of $\mathcal{C}_{v\text{-}volume}$ occurs exactly once in the above expression, attributing a separate argument to every occurrence actually provides the same expression. Anyway, without any transformation, $\mathcal{C}_{v\text{-}volume}$ is (anti)-monotone w.r.t. each of its arguments. For example, it is monotone w.r.t. the first argument: $\forall (X^1, \ldots, X^n) \in 2^{\mathcal{D}^1} \times \cdots \times 2^{\mathcal{D}^n}$ and $\forall X^{1\prime} \subseteq \mathcal{D}^1, \left(X^1 \subseteq X^{1\prime} \Rightarrow |X^1 \times \cdots \times X^n| \geqslant v \Rightarrow |X^{1\prime} \times \cdots \times X^n| \geqslant v\right)$. As a consequence $\mathcal{C}_{v\text{-}volume}$ is piecewise (anti)-monotone. Because, it actually is monotone w.r.t. each of its arguments,* DATA-PEELER *tests, at every recursive call, $\mathcal{C}_{v\text{-}volume}(U \sqcup V)$. If the test succeeds at least $U \sqcup V$, the current node represents, satisfies $\mathcal{C}_{v\text{-}volume}$. If the test fails the search space is pruned. Indeed, every $n$-set, the current node represents, violates $\mathcal{C}_{v\text{-}volume}$.*

**Example 19** *Consider the following constraint forcing the extracted closed $n$-sets to approximately gather the same number of elements in the first attribute as in the second attribute.*

$$\mathcal{C}_{\epsilon\text{-}square}(X^1, X^2) \equiv \frac{|X^1|}{|X^2|} - \frac{|X^2|}{|X^1|} \leqslant \epsilon \wedge \frac{|X^2|}{|X^1|} - \frac{|X^1|}{|X^2|} \leqslant \epsilon \wedge X^1 \neq \emptyset \wedge X^2 \neq \emptyset$$

*The parameter $\epsilon \in \mathbb{R}_+$ tunes the approximation: the smaller $\epsilon$, the stronger the constraint ($\epsilon = 0$ forces $|X^1| = |X^2|$). $\mathcal{C}_{\epsilon\text{-}square}$ has two arguments, $X^1$ and $X^2$, but each of them occurs four times in the expression of the constraint. To prove the piecewise (anti)-monotonicity of $\mathcal{C}_{\epsilon\text{-}square}$, the constraint is rewritten in $\mathcal{C}'_{\epsilon\text{-}square}$ with a separate argument for each of the eight occurrences:*

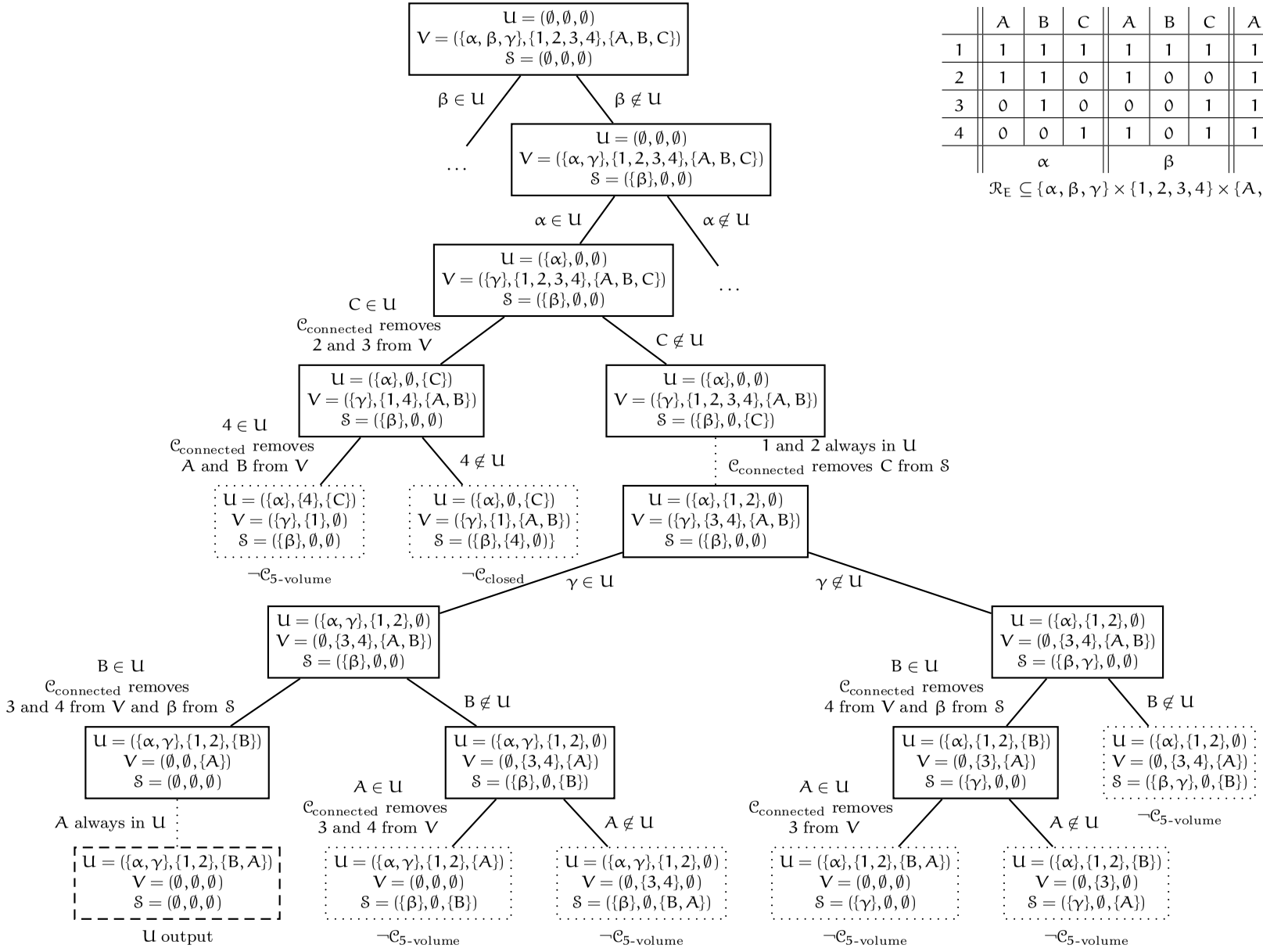
$$\begin{aligned} & \mathcal{C}'_{\epsilon\text{-}square}(X^1_1, X^1_2, X^1_3, X^1_4, X^1_5, X^2_1, X^2_2, X^2_3, X^2_4, X^2_5) \\ \equiv \quad & \frac{|X^1_1|}{|X^2_1|} - \frac{|X^2_2|}{|X^1_2|} \leqslant \epsilon \wedge \frac{|X^2_3|}{|X^1_3|} - \frac{|X^1_4|}{|X^2_4|} \leqslant \epsilon \wedge X^1_5 \neq \emptyset \wedge X^2_5 \neq \emptyset \ . \end{aligned}$$

*$\mathcal{C}'_{\epsilon\text{-}square}$ is (anti)-monotone w.r.t. each of its arguments, what proves, by definition, the piecewise (anti)-monotonicity of $\mathcal{C}_{\epsilon\text{-}square}$. More precisely, $\mathcal{C}'_{\epsilon\text{-}square}$ is monotone w.r.t. $X^1_3$, $X^1_4$, $X^1_5$, $X^2_1$, $X^2_2$ and $X^2_5$ and anti-monotone w.r.t. $X^1_1$,*

$X_2^1$, $X_3^2$ *and* $X_4^2$. *As a consequence, to enforce* $\mathcal{C}_{\epsilon\text{-}square}$, DATA-PEELER *tests, at every recursive call,* $\mathcal{C}'_{\epsilon\text{-}square}(U^1, U^1, U^1 \cup V^1, U^1 \cup V^1, U^1 \cup V^1, U^2 \cup V^2, U^2 \cup V^2, U^2, U^2, U^2 \cup V^2)$. *If the test fails the search space is pruned. Indeed, every* $n$*-set, the current node represents, violates* $\mathcal{C}_{\epsilon\text{-}square}$.

## 3 EXAMPLE OF COMPUTATION

Figure 22 depicts a part of the computation of DATA-PEELER on $\mathcal{R}_E$ (represented in Table 5) where every closed 3-set satisfying $\mathcal{C}_{5\text{-volume}}$ (introduced in Example 18) is to be extracted. The dashed leaf is a such a pattern. The dotted leaves are pruned. The choice of the element to enumerate follows the rule enunciated in Section 1.6.




| | A | B | C | A | B | C | A | B | C |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 2 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| | $\alpha$ | | | $\beta$ | | | $\gamma$ | | |

$\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$.

Figure 22: Part of the enumeration tree Data-Peeler traverses when mining $\mathcal{R}_E$.

## 4 DATA STRUCTURES

In this section, the size (in bits) of an element ID is denoted $a$ and the size (in bits) of a pointer is denoted $b$.

### 4.1 *Storing the Dataset*

Unlike for binary relation mining algorithms, it is not possible to store the projection (usually called "tidset") of the input dataset $\mathcal{R}$ on each element $e \in \mathcal{D}^1 \cup \cdots \cup \mathcal{D}^n$. The use of sophisticated data structures like FP-trees [38] remains an open problem because of the multiple attributes to consider and the required ability to enumerate any of them all along the enumeration. As a consequence, the whole dataset must be stored in main memory so that $\mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\text{closed}}$ can be enforced.

Two classes of data structures were investigated, namely a bitset-based structure, and a list-based structure. In both cases, the dataset is stored in a complete prefix tree of height $n-1$ corresponding to the $n-1$ first attributes. The nodes at depth $i = 0..n-2$ always have $|\mathcal{D}^{i+1}|$ children, one for every element of $\mathcal{D}^{n+1}$. From depth $0$ to $n-2$, the edges binding a node to its children are pointers. Each leaf stands for a prefix of size $n-1$ of every element of $\mathcal{D}^1 \times \cdots \times \mathcal{D}^{n-1}$. The difference between the two studied structures relies in how the last attribute elements are stored.

#### 4.1.1 *Bitset-Based Structure*

In such a structure, every leaf of the prefix tree points to a bitset representing the last attribute elements. A "0" (respectively "1") in the bitset stands for the absence (respectively the presence) of the related element of $\mathcal{R}$. The presence of such an element is tested in constant time. The space occupied by the dataset is:

$$\underbrace{b \sum_{i=0}^{n-1} \prod_{j=1}^{i} |\mathcal{D}^i|}_{\text{the depths from } 0 \text{ to } n-1} + \underbrace{\prod_{j=1}^{n} |\mathcal{D}^j|}_{\text{the bitsets}} .$$

#### 4.1.2 *List-Based Structure*

Here, every leaf points to a list of IDs of elements of $\mathcal{D}^n$. Each of them represents an element of $\mathcal{R}$. The presence of such an element is tested in $O(\log |\mathcal{D}^n|)$. Choosing $\mathcal{D}^n$ to be the smallest attribute domain minimizes the access time. If $d = \frac{|\mathcal{R}|}{\prod_{i=1}^{n} |\mathcal{D}^i|}$ denotes the density of the dataset, the space requirement is:

$$\underbrace{b \sum_{i=0}^{n-1} \prod_{j=1}^{i} |\mathcal{D}^j|}_{\text{the depths from } 0 \text{ to } n-1} + \underbrace{a \times d \prod_{j=1}^{n} |\mathcal{D}^j|}_{\text{the lists}} .$$

Compared to the bitset-based structure, a space gain occurs if and only if $d < \frac{1}{a}$. Taking $a = 64$ (size of an integer on modern hardware), the density of the dataset must be under 1.56% for the list-based structure to present a space advantage over the bitset-based structure. Thus,

the bitset-based structure is always better in data access time and, in most cases, in space requirement too. Therefore, this structure was chosen for our implementation.

*The dataset is stored in a bitset-based structure for faster accesses and space gains unless the relation is very sparse.*

Notice that other sparser structures were theoretically investigated. They consist in using an incomplete prefix tree. Of course, the time access cost increases ($O(\sum_{i=1}^{n} \log|\mathcal{D}^i|)$ for a totally sparse tree). Furthermore, the space requirement can be greater since we need to add an element ID to each node. Indeed the child node addressed by a pointer cannot be identified from the position of the child in the list of children (some are "missing"). It can be shown that a space gain occurs only when, in average, a node at depth $i$ has less than $\frac{b}{a+b}|\mathcal{D}^{i+1}|$ children. Unless the dataset is very sparse and/or non-homogeneous, even depth $n-2$ does not satisfy such a property. This justifies the fact that we focused on the *list-based structure* where only the deepest level is sparse.

### 4.2 *Storing the Enumeration Nodes*

Both $U$ and $S$ can be statically stored in stacks. At every recursive call, one single element is pushed in either $U$ (when constructing $N_L$) or $S$ (when constructing $N_R$) and popped once this recursive call is completed.

Any element of $V$ can be removed when $\mathcal{C}_{\text{connected}}$ is enforced. As a result, $V$ cannot be statically stored. The construction of the enumeration tree being depth-first, the worst case is bound to reaching the deepest node. At worst, the depth of the enumeration tree is $\sum_{i=1}^{n} |\mathcal{D}^i|$ where each recursive call removes only one element from $V$. In this case, the required space to store $V$ is:

$$a \sum_{i=1}^{\sum_{j=1}^{n}|\mathcal{D}^j|} i = \frac{a}{2} \sum_{j=1}^{n} |\mathcal{D}^j| \times (\sum_{j=1}^{n} |\mathcal{D}^j| - 1) \ .$$

### 4.3 *Space Complexity*

Combining the results from Section 4.1 and Section 4.2, the space complexity of DATA-PEELER is linear in the size of the input relation ($\prod_{i=1}^{n} |\mathcal{D}^i|$). More precisely it is:

*Unless the relation is binary, its storage dominates the space complexity.*

- $O((|\mathcal{D}^1| + |\mathcal{D}^2|)^2)$ if $n = 2$ (the space requirement for the $V$ set predominates);
- $O(\prod_{i=1}^{n} |\mathcal{D}^i|)$ if $n > 2$ (the space requirement for the dataset predominates).

## 5 EXPERIMENTAL RESULTS

Every experiment, this section describes, has been performed on a GNU/Linux™ system equipped with an AMD Sempron™ 2600+ processor and 512 MB of RAM. DATA-PEELER is implemented in C++ and compiled with GCC 4.1.2.

### 5.1 QUEST-*Generated Datasets*

To study the behavior of DATA-PEELER and compare it to competitors in different situations, we have used the IBM QUEST data generator [2]. Various synthetic basket datasets with predefined attributes

and densities have been generated. Three attributes are considered: the customers, the bought items, and the time periods (in months).

To test the scalability of DATA-PEELER w.r.t. the arity of the relation (the size of the input data remaining constant), three kinds of uniformly random datasets are generated:

1. 16 attributes with 2-valued domains (Boolean attributes);
2. 8 attributes with 4-valued domains;
3. 4 attributes with 16-valued domains.

In such a relation, every tuple has a given probability to be in $\mathcal{R}$. When generating a large dataset, its density $d = \frac{|\mathcal{R}|}{\prod_{i=1}^{n} |\mathcal{D}^i|}$ is close to this probability. Datasets built in this way usually do not contain any large closed $n$-sets: the extraction problem is known to be hard.

## 5.2 *Impact of the Enumeration Strategy*

Let us first empirically compare the enumeration strategy presented in Section 1.6 with two other sensible strategies:

1. For each node $(U, V)$, the attribute $j$ is chosen such that it has the smallest non-empty $V^j$. Among the elements in $V^j$, the element with the smallest density in $\mathcal{R}$ is enumerated.
2. For each node $(U, V)$, the enumerated element $e \in V$ is chosen such that $(\mathcal{D}^1 \times \ldots \times \{e\} \times \ldots \times \mathcal{D}^n) \setminus \mathcal{R}$ has the largest cardinality.

The first strategy enumerates every element of the $n-2$ domains with the smallest cardinalities. Then, when enumerating elements from the two remaining attributes, $\mathcal{C}_{\text{connected}}$ may finally succeed in reducing the $V$ $n$-set. Indeed, $n-1$ attributes need to be set ($U^i \neq \emptyset$) for $\mathcal{C}_{\text{connected}}$ to, hopefully, remove elements from the last attribute. The second strategy globally sorts the elements of all domains. If every attribute domain has the same cardinality, this order follows a growing density. Otherwise, an element $e$ from a small attribute domain size is favored since $\mathcal{D}^1 \times \ldots \times \{e\} \times \ldots \times \mathcal{D}^n$ is larger.

Tests have been performed on the datasets generated by QUEST. Whereas DATA-PEELER's enumeration strategy scales very well, the other strategies force us to choose small size attributes to be able to plot results: 36 customers buying in average 6 items out of 18 (density of about 33%) per month. The number of months vary from 6 to 36 and we enforce the constraint that every closed 3-set must involve at least three customers, two items and three months.

*Compared to other sensible choices,* DATA-PEELER*'s enumeration strategy supports extractions that are orders of magnitude faster.*

Figure 23 presents the results. DATA-PEELER's enumeration strategy largely outperforms the two other strategies. The performance of Enumeration 1 mainly depends on the size of the smallest attribute domain (above 18 months, the smallest attribute domain becomes the set of items that is constant). As mentioned earlier, the complete enumeration of the smallest domain causes this behavior. The performance of Enumeration 2 emphasizes the need, when choosing the element to enumerate, to take into account the characteristics of the current node.

## 5.3 *Comparison with Competitors*

DATA-PEELER is compared to both CUBEMINER [41] and TRIAS [40] on 3-ary relations. Their implementations were kindly provided by their respective authors. The comparison is achieved on QUEST-generated

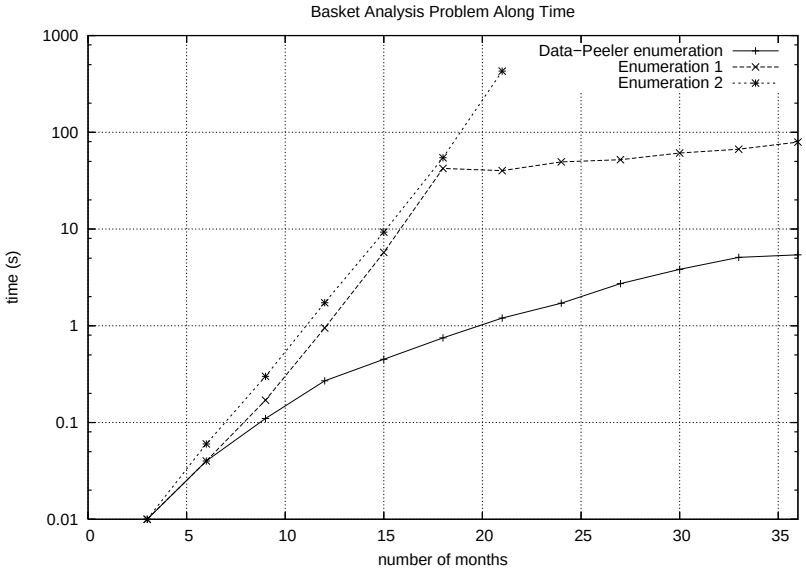



Figure 23: Comparing sensible enumeration strategies.

datasets. 144 customers buying in average 6 items out of 72 (density of about 8.3%) per month have been generated. We make the number of months vary from 6 to 66 and constrain every closed 3-set to involve at least two customers, two items and two months.

Figure 24 presents the results. DATA-PEELER outperforms its competitors by several orders of magnitude. The growing number of months (the smallest domain) significantly alters TRIAS' performance, whereas it has less effect on CUBEMINER. For example, considering data along 48 months, to extract all the 5801 closed 3-sets, CUBEMINER takes 1 hour and 50 minutes, TRIAS 3 hours and 14 minutes, whereas DATA-PEELER only needs 2.5 seconds. Unlike its competitors, even with 600 months, DATA-PEELER is still able to extract all closed 3-sets in a reasonable time, i. e., 1 minute and 21 seconds for 431892 closed 3-sets.

DATA-PEELER *outperforms its competitors by several orders of magnitude.*

## 5.4 *Scalability w.r.t. the Arity*

The three kinds of uniformly random datasets, presented in Section 5.1, are generated with densities varying between 0 and 0.5. Given a density, the sizes of the input data are the same for all three datasets. The extracted closed n-sets are forced to contain at least four tuples (this constraint depends neither on the arity of the relation nor on the sizes of its domains). The results are plotted in Figure 25. When the datasets are sparse (e. g., a 0.05 density), a high arity has a negative impact on DATA-PEELER's performance. With greater densities, the extraction times, on the three datasets, are of the same order of magnitude.

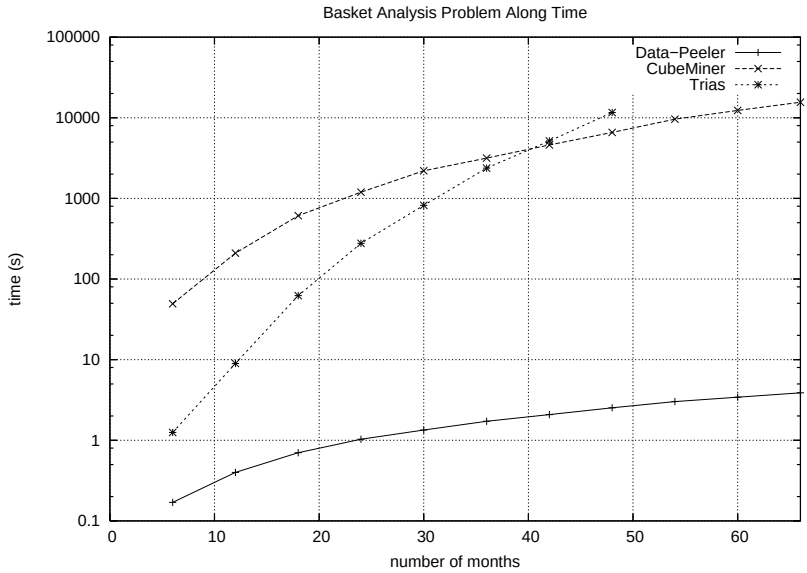



Figure 24: Comparison with CubeMiner and Trias.

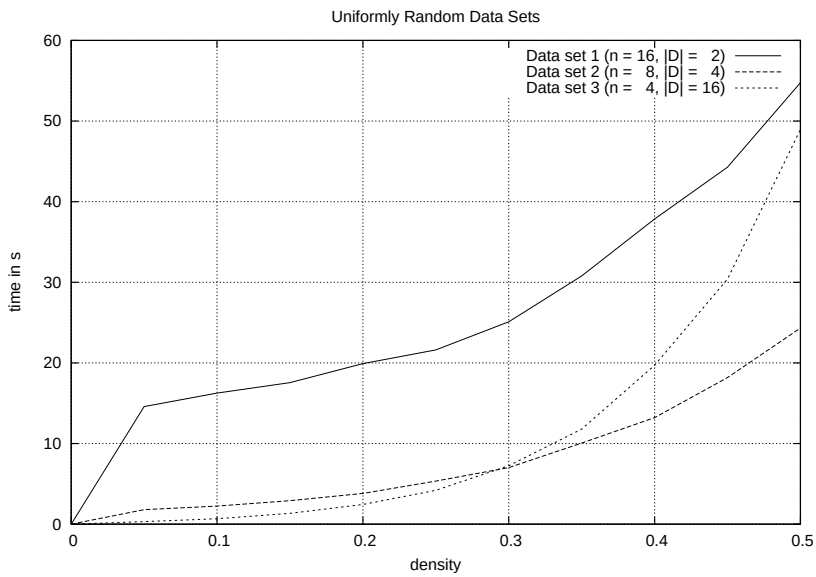



Figure 25: Effect of the arity on the extraction times.

## 6 ROBUSTNESS W.R.T. BINARIZATION

Many relations are derived from numerical datasets. To decide which tuples eventually are in $\mathcal{R}$, several binarization methods exist and/or, for a given method, various parameter settings are possible. Consider, e. g., gene expression data analysis. This application domain has motivated the design of many closed pattern extractors (e. g., [59] and [99]). The gene expression is a real number that needs to be turned into a Boolean value, i. e., a yes/no answer to a question like "Is this gene over-expressed in this experiment?". Transformations from numerical datasets to Boolean ones are called *binarizations*.

*Binarizations turn numerical datasets into relations.*

Given a numerical dataset (that may have been normalized earlier), a simple binarization consists in deriving a value $\alpha$ from the whole dataset and defining $\mathcal{R}$ as the tuples associated to a value greater (or smaller) than $\alpha$. For example, $\alpha$ could be chosen as one quarter of the maximal value in the dataset and $\mathcal{R}$ would contain every tuple associated with a value greater than $\frac{\alpha}{4}$. The same kind of procedure can be separately applied on different subspaces of the dataset. For example, to decide whether a given gene is over-expressed, there is no reason to consider the expressions of the other genes, and $\mathcal{R}$ could be the couples $(\text{gene}, \text{sample})$ such that "gene" has, in "sample", an expression level above its own average. Many other transformations, from numerical datasets to a Boolean ones, have been proposed [64]. Anyway, the simple examples, that have just been given, already provide many different sensible choices depending on whether or how to normalize the numerical data, how local/global the binarization should be, how to derive a threshold, etc.

*Choosing a binarization is taking up a perspective on the data.*

Given a numerical dataset, it remains unclear which method to pick up and how to parametrize it. Different binarized versions of a same dataset actually are different perspectives on the data. All these perspectives may be interesting. Focusing on one perspective may hide relevant patterns which would be present with many other. As a consequence, the analyst, who wants to discover unexpected patterns, often does not have any reason to prefer a perspective to another. We actually suggest him/her not to make any choice. To be more precise, we suggest him/her to make all the choices at a time and use DATA-PEELER to extract closed patterns that are relevant across several perspectives on the data.

The most interesting patterns are (at least partially) found across several versions of the same dataset binarized in different ways. Tagging the $n$-tuples with the binarization methods that select it is like adding a new attribute to our dataset. Geometrically, this process can be imagined as "stacking" the different Boolean versions of the initially numerical dataset. In this way, if the initial dataset is a two-dimensional matrix, the different 0/1 versions of it are stacked in a 0/1 cube whose height is the number of different binarizations that were applied. More generally, if the initial dataset is an $n$-dimensional tensor, an $(n+1)$-ary relation is built. Extracting, with DATA-PEELER, the closed $(n+1)$-sets under a minimal size constraint on the binarization attribute, provides patterns that are relevant across many perspectives on the data. In other terms, these closed $(n+1)$-sets are robust to the binarization method. Beyond "real" $n$-ary relations, this original approach emphasizes how mining closed $n$-sets can improve the knowledge discovery processes on classical numerical matrices.

*A minimal size constraint on an attribute gathering several binarizations is a formal specification of "robustness w.r.t. perspectives".*

# 7 MINIMIZING MULTI-VALUED LOGIC FUNCTIONS

## 7.1 *Problem Setting*

A relation is usually represented as a set of tuples. Storing and retrieving a relation by listing all its tuples one by one is both time and space consuming. That is why minimizing the expression of an arbitrary $n$-ary relation $\mathcal{R}$ is an interesting problem. Since a collection of $n$-tuples is not a satisfactory solution, we may look for relevant collections of patterns that, in this context, are generically called *tiles*. The *tiling task* consists in finding a collection of (possibly overlapping) tiles that is as compact as possible but still entirely expresses $\mathcal{R}$, i. e., the union of the $n$-tuples in all tiles equals $\mathcal{R}$.

*Closed $n$-sets summarize parts of the relations.*

Choosing the tiles to be closed $n$-sets looks like a clever idea. Indeed, a closed $n$-set can be seen as a syntactical summary of a part of $\mathcal{R}$. Indeed, it is shorter to write a closed $n$-set than to list all the tuples it encompasses. For example, without any loss of information, we can write that the relation $\mathcal{R}_E$, represented in Table 5, contains $(\{\beta,\gamma\},\{1,2,4\},\{A\})$ instead of listing all the tuples this closed 3-set encompasses:

$$(\beta,1,A),\ (\beta,2,A),\ (\beta,4,A),\ (\gamma,1,A),\ (\gamma,2,A),\ (\gamma,4,A)$$

A collection of *well-chosen* closed 3-sets shortly expresses the whole relation $\mathcal{R}_E$:

$$\begin{gathered}(\{\alpha,\gamma\},\{1,2\},\{A,B\})\\(\{\beta,\gamma\},\{3,4\},\{C\})\\(\{\alpha,\beta\},\{1\},\{A,B,C\})\\(\{\gamma\},\{1,2,4\},\{A,B\})\\(\{\beta\},\{1,2,4\},\{A,C\})\\(\{\alpha,\beta,\gamma\},\{4\},\{C\})\\(\{\alpha\},\{1,2,3\},\{B\})\end{gathered}$$

*The closedness constraint does not make sense when tiling.*

By definition, closed $n$-sets satisfy both $\mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\text{closed}}$. However, for the sake of minimizing the expression of $\mathcal{R}$, constraining the tiles to be closed does not make sense. Indeed, in some situations, when two (or more) closed $n$-sets are overlapping, one of them can be "cropped" so that the relation is expressed in a shorter way. For example, in the collection above, $(\{\beta\},\{1,2,4\},\{A,C\})$ can be "cropped" in $(\{\beta\},\{2,4\},\{A,C\})$ if $(\{\alpha,\beta\},\{1\},\{A,B,C\})$ is kept (unaltered) in the collection. Indeed, this closed 3-set already encompasses $\{\beta\}\times\{1\}\times\{A,C\}$. That is why the tiles are advantageously chosen among the $n$-sets rather than the closed $n$-sets. The $n$-set domain is a superset of the closed $n$-sets domain where $\mathcal{C}_{\text{connected}}$ still needs to be checked but where $\mathcal{C}_{\text{closed}}$ does not necessarily hold. With tiles in this larger pattern domain, the previous tiling of $\mathcal{R}_E$ can be improved into the following one:

$$\begin{gathered}(\{\alpha,\gamma\},\{1,2\},\{A,B\})\\(\{\beta,\gamma\},\{3,4\},\{C\})\\(\{\alpha,\beta\},\{1\},\{A,B,C\})\\(\{\gamma\},\{1,2,4\},\{A,B\})\\(\{\beta\},\{2,4\},\{A\})\\(\{\alpha\},\{4\},\{C\})\\(\{\alpha\},\{3\},\{B\})\end{gathered}$$

### 7.2 *Simplifying Multi-Valued Logic Functions*

Interestingly, the attributes $(A^i)_{i=1..n}$ can be seen as variables of a multi-valued logic function whose truth table is given by $\mathcal{R}$. Notice that Boolean functions are a specialization of this framework ($\forall i = 1..n, |\mathcal{D}^i| = 2$). In this perspective, tiling $\mathcal{R}$ is equivalent to *minimizing* (*simplifying* is used too) the related multi-valued logic function. For example, the tiling of $\mathcal{R}_E$ written above provides this simplified expression of the related multi-valued logic function:

*Minimizing a multi-valued logic function is tiling its truth table.*

$$
\begin{array}{ll}
 & (A^1 = \alpha \vee A^1 = \gamma) \wedge (A^2 = 1 \vee A^2 = 2) \wedge (A^3 = A \vee A^3 = B) \\
\vee & (A^1 = \beta \vee A^1 = \gamma) \wedge (A^2 = 3 \vee A^2 = 4) \wedge (A^3 = C) \\
\vee & (A^1 = \alpha \vee A^1 = \beta) \wedge (A^2 = 1) \wedge (A^3 = A \vee A^3 = B \vee A^3 = C) \\
\vee & (A^1 = \gamma) \wedge (A^2 = 1 \vee A^2 = 2 \vee A^2 = 4) \wedge (A^3 = A \vee A^3 = B) \\
\vee & (A^1 = \beta) \wedge (A^2 = 2 \vee A^2 = 4) \wedge (A^3 = A) \\
\vee & (A^1 = \alpha) \wedge (A^2 = 4) \wedge (A^3 = C) \\
\vee & (A^1 = \alpha) \wedge (A^2 = 3) \wedge (A^3 = B)
\end{array}
$$

Given a tiling, its quality can be measured with the number of logic operators ($\vee$ and $\wedge$) in the simplified expression of the related multi-valued logic function, the smaller the better. For example, 18 $\vee$ and 14 $\wedge$ are present in our simplified expression of the multi-valued logic function related to $\mathcal{R}_E$. The quality of this tiling is $18 + 14 = 32$.

### 7.3 *A Global Model of $\mathcal{R}$*

The simplification of multi-valued logic functions is an interesting application of tiling. Nevertheless, this coverage of $\mathcal{R}$ can be seen as a solution to a machine learning problem too. Consider $\mathcal{R}$ as a set of positive examples and $(\times_{i=1..n}\mathcal{D}^i) \setminus \mathcal{R}$ as a set of negative examples. A tiling is a *consistent* hypothesis that complies with the observed data, i. e., it covers every positive example and no negative example. With this perspective, minimizing the tiling is searching for the *shortest* hypothesis that remains consistent. This relates to the famous minimum description length principle coined in [72] (see [35] for a comprehensive and modern reference):

*Tiling can be seen as searching for the shortest hypothesis that is consistent with the data.*

> Choose the model that gives the shortest description of data.

In this respect, a tiling is a global model of $\mathcal{R}$. Taking up a framework "From Local Patterns to Global Models" (see, e. g., [45]), we propose to derive it from the closed n-sets DATA-PEELER lists.

When a user-defined relevancy constraint $\mathcal{C}$ is applied to the local patterns, the n-tuples in $\mathcal{R}$ that are covered by none of the closed n-sets may be considered spurious, i. e., they are false-positive examples. Then, if $\mathcal{R}$ is deprived of these n-tuples before being tiled, the obtained result is a shorter hypothesis that remains *correct* (it covers no negative example) but becomes *incomplete* (it does not cover the false-positive examples). The "force" of the chosen relevancy constraint $\mathcal{C}$ tunes the trade-off between briefness and completeness. The incompleteness of a shorter model can be seen as tolerating positive noise and becomes, to some extent, a wanted feature.

Interestingly, the same post-processing, presented in the next section, but based on error-tolerant patterns, i. e., patterns that can cover a few negative examples, also allows to obtain a shorter hypothesis to the

cost of losing the consistency. More precisely, the hypothesis remains *complete* (it covers every positive example) but becomes *incorrect* (some negative examples are covered as well). Error-tolerance thresholds tune the trade-off between briefness and correctness. The incorrectness of a shorter model can be seen as tolerating negative noise and becomes, to some extent, a wanted feature. The next part will define such error-tolerant patterns that hold in $n$-ary relations.

## 7.4 *A Closed $n$-Set Greedy Post-Processing*

The set of all closed $n$-sets returned by DATA-PEELER (without enforcing any constraint but $\mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\text{closed}}$) is a tiling of the relation since it integrally covers it. Its quality is very poor though: most of the time, it is far worse than listing every tuple one by one. Nevertheless, the closed $n$-sets are here considered a starting point. Post-processing DATA-PEELER will take care of removing useless information from the computed closed $n$-sets to obtain a tiling of $\mathcal{R}$ with a good quality.

### 7.4.1 *Removing the Complete Sets*

Consider a tile $(X^i)_{i=1..n}$. If one of the $X^i$ set equals the whole attribute domain $\mathcal{D}^i$, listing its elements one by one is useless. For example, in $\mathcal{R}_E$, $(\{\alpha, \beta\}, \{1\}, \{A, B, C\})$ can be shortened into $(\{\alpha, \beta\}, \{1\}, -)$ meaning that, whatever $v \in \{A, B, C\}$, the 3-tuples $(\alpha, 1, v)$ and $(\beta, 1, v)$ are in $\mathcal{R}_E$. In this way, when a tile has $X^i = \mathcal{D}^i$ ($i = 1..n$), the number of required logic operators to express it is lowered (i. e., the quality is improved) by $|\mathcal{D}^i|$. Let us express the previous example using the multi-valued logic form. $(A^1 = \alpha \vee A^1 = \beta) \wedge (A^2 = 1) \wedge (A^3 = A \vee A^3 = B \vee A^3 = C)$ requires 5 logic operators to be written. Once turned into $(A^1 = \alpha \vee A^1 = \beta) \wedge (A^2 = 1)$, only $5 - |\mathcal{D}^3| = 2$ logic operators are needed.

*If an entire attribute domain is involved in a tile, its elements do not need to be written.*

### 7.4.2 *Tightening the Tiles*

Given an element in any set of a tile, consider the tuples encompassed by the tile and involving this element. Verifying whether the previously output tiles encompass them is straightforward. Here is how it is achieved: whenever a tile is output, the tuples it encompasses are removed from the relation. In this way, the elements of another tile can be considered one by one and removed if, in the related hyper-plans, none of the tuples remain in the relation. E. g., a tile $(\{\beta, \gamma\}, \{1, 2, 4\}, \{A\})$ can be tightened into $(\{\beta\}, \{2, 4\}, \{A\})$ if $(\{\alpha, \beta\}, \{1\}, -)$ and $(\{\gamma\}, \{1, 2, 4\}, \{A, B\})$ were previously output. Indeed they already encompass both $\{\beta, \gamma\} \times \{1\} \times \{A\}$ (hyper-plan related to the element 1) and $\{\gamma\} \times \{1, 2, 4\} \times \{A\}$ (hyper-plan related to the element $\gamma$). The quantity of information safely removed in this way greatly depends on the order in which the tiles are processed.

*Given a tile, its hyper-plans that were previously encompassed are cropped.*

### 7.4.3 *Ordering the Tiles*

Relying on the order in which $n$-sets are discovered by DATA-PEELER does not provide a good minimization of $\mathcal{R}$. It is advantageously replaced by the following heuristic:

**Heuristic 1** *Output first the tile presenting the best ratio between the number of newly encompassed (i. e., not encompassed by previously output tiles) tuples and the number of logic operators ($\vee$ and $\wedge$) needed to express it.*

*Tiles with the greatest ratio* `newly encompassed tuples / description size` *are favored.*

To do so, the closed n-sets are stored in main memory instead of being directly output. Whenever a tile is output, the part of $\mathcal{R}$ it encompasses is removed. Thus a large tile may be moved towards the end of the sequence (and even never be output) if there are larger tiles encompassing many of its tuples. The algorithms terminates when $\mathcal{R}$ is completely covered.

Notice that the sequence of remaining tiles is not maintained ordered at any time. Instead, only the first tile is considered. If the quantity of tuples it encompasses (initialized at extraction time) has decreased it is moved down the sequence. Otherwise, it is output.

### 7.4.4 *Don't Care Set*

*Don't care set* is the name given to a set $\mathcal{R}_{DCS}$ of tuples that can either be considered as elements of $\mathcal{R}$ or not. Typically they are impossible combinations of values for the n attributes. The don't care set plays an interesting role in the minimization problem: its elements can enable bigger tiles even though they are not required to be part of a tile. Tiling with DATA-PEELER easily takes advantage of a don't care set. The closed n-sets are extracted on $\mathcal{R} \cup \mathcal{R}_{DCS}$. Then the tuples of $\mathcal{R}_{DCS}$ are removed (i. e., they are considered covered) and the post-process, detailed in this section, is performed unchanged.

*A don't care set can be specified.*

## 7.5 *Experimental results*

Here again, the experiments have been performed on a GNU/Linux™ system equipped with an AMD Sempron™ 2600+ processor and 512 MB of RAM. The performance in minimizing a multi-valued logic function are evaluated with respect to:

- The time it takes to tile (extraction of the closed n-sets included);
- The number of logic operators ($\vee$ and $\wedge$) in the tiling.

### 7.5.1 *Comparision with* ESPRESSO-MV

An implementation of the ESPRESSO-MV algorithm [73] (shipped with the MVSIS 3.0 package [29] for logic synthesis and verification) was used as a reference in our tests. ESPRESSO-MV is a generalization to multi-valued functions of the ESPRESSO-II algorithm [14], which only simplifies Boolean functions (see Section 2.2.2 in Chapter 2).

DATA-PEELER and ESPRESSO-MV are compared on the uniformly random datasets presented in Section 5.1. The dimensions of these datasets are recalled:

1. 16 attributes with 2-valued domains (Boolean attributes);
2. 8 attributes with 4-valued domains;
3. 4 attributes with 16-valued domains.

Typical results (no significant variation from a random generation to another) are listed in the Tables 6, 7 and 8 for three different densities (6.25%, 25% and 50%).

| | Time performances in s | | | Quality in number of $\vee$ and $\wedge$ | | |
|---|---|---|---|---|---|---|
| Dataset | E-MV | D-P | Var. rate | E-MV | D-P | Var. rate |
| 1 | 2.21 | 10.62 | +380.54% | 46068 | 46518 | +0.97% |
| 2 | 146.88 | 1.95 | -98.67% | 23662 | 23166 | -2.09% |
| 3 | 103.62 | 0.52 | -99.49% | 10734 | 10035 | -6.51% |

Table 6: Minimizing random multi-valued logic functions (density: 6.25%).

| | Time performances in s | | | Quality in number of $\vee$ and $\wedge$ | | |
|---|---|---|---|---|---|---|
| Dataset | E-MV | D-P | Var. rate | E-MV | D-P | Var. rate |
| 1 | 36.09 | 19.75 | -45.27% | 107869 | 109722 | +1.71% |
| 2 | 957.75 | 6.34 | -99.33% | 69460 | 57014 | -17.91% |
| 3 | 552.11 | 4.39 | -99.20% | 28118 | 23082 | -17.91% |

Table 7: Minimizing random multi-valued logic functions (density: 25%).

### 7.5.2 *Discussion*

*In the general case, our approach minimizes more and runs orders of magnitude faster then* ESPRESSO-MV.

BOOLEAN ATTRIBUTES DATA-PEELER's simplification is between 0 and 3% worse than ESPRESSO-MV's. On the positive side, DATA-PEELER performs faster when the density of the dataset is 15% or more.

MULTI-VALUED ATTRIBUTES When the attributes take more than 2 values, DATA-PEELER significantly outperforms ESPRESSO-MV both in quality and running time. The gain in quality grows with the number of values per attribute and the density of the dataset. With a 50% density, DATA-PEELER builds minimized expressions of the relation that are more than one third smaller than what ESPRESSO-MV achieves. The gain in time is impressive: DATA-PEELER performs the task in about 1% of the time required by ESPRESSO-MV.

## 7.6 *Improving Time Performances*

### 7.6.1 $\mathcal{C}_{k-summary}$

*By translating background knowledge into piecewise (anti)-monotone constraints, the tiling can be fasten without too much degradation of its quality.*

With some background knowledge about $\mathcal{R}$, some constraints most closed $n$-sets will satisfy may be known. If these constraints are piecewise (anti)-monotone, DATA-PEELER more quickly extracts the closed $n$-sets. If every closed $n$-set actually satisfies the specified constraints, the computed tiling is identical. Otherwise, the constrained closed $n$-sets may not totally cover $\mathcal{R}$. In this case, the collection of tiles is completed by a linear procedure browsing the uncovered dataset and outputting aggregates of tuples along the largest attribute domain. Clearly, the quality of the tiling gradually decreases with the stringiness of the constraints until no closed $n$-set satisfies them. In this extreme

| | Time performances in s | | | Quality in number of $\vee$ and $\wedge$ | | |
|---|---|---|---|---|---|---|
| Dataset | E-MV | D-P | Var. rate | E-MV | D-P | Var. rate |
| 1 | 122.21 | 48.81 | -60.06% | 120170 | 122452 | +1.89% |
| 2 | 1781.62 | 25.31 | -98.57% | 99763 | 66512 | -33.32% |
| 3 | 1064.33 | 50.81 | -95.22% | 45060 | 27886 | -38.11% |

Table 8: Minimizing random multi-valued logic functions (density: 50%).

case, the tiling is the collection of aggregates of tuples along the largest attribute domain. Depending on the mined relation, some specific constraints may greatly reduce the extraction times while providing a collection of closed $n$-sets that allow a good minimization of the dataset.

With large relations, specifying a constraint may even be compulsory so that the tiling problem is tractable. This tractability relates to time but also to space requirements. Indeed, the proposed post-process stores every tile in main memory. Without any background knowledge about the relation $\mathcal{R}$ to tile, the more natural constraint to enforce is related to the order in which the tiles are stored: the ratio between the volume of a closed $n$-set and the number of logic operators ($\vee$ and $\wedge$) needed to express it must exceed a given threshold $k \in \mathbb{R}$. Expressed formally, the constraint is:

*Without any background knowledge, a minimal ratio `volume` / `description size` decreases time and space requirements.*

$$\mathcal{C}_{\text{k-summary}} \equiv \prod_{i=1}^{n} |X^i| \geqslant k \sum_{i=1}^{n} f(X^i), \text{ where } f(X^i) = \begin{cases} 0 \text{ if } X^i = \mathcal{D}^i \\ |X^i| \text{ otherwise} \end{cases}$$

$\mathcal{C}_{\text{k-summary}}$ ($k \in \mathbb{R}$) is piecewise (anti)-monotone. In $\mathcal{R}_E$, the six closed 3-sets satisfying $\mathcal{C}_{\text{1.5-summary}} \wedge \mathcal{C}_{\text{1-volume}}$ encompass 17 tuples out of 23. They are completed with aggregates of the remaining tuples along the attribute having largest domain (the second one for $\mathcal{R}_E$), i. e., with tiles restricted to one element in every other attribute. Once the steps detailed in Section 7.4 applied, the tiling is:

$$\begin{gathered}
(\{\alpha, \beta\}, \{1\}, -) \\
(-, \{1, 2\}, \{A\}) \\
(\{\gamma\}, -, \{A\}) \\
(-, \{4\}, \{C\}) \\
(\{\gamma\}, \{1, 2, 4\}, \{B\}) \\
(\{\alpha\}, \{2, 3\}, \{B\}) \\
(\{\gamma\}, \{3\}, \{C\}) \\
(\{\beta\}, \{3\}, \{C\}) \\
(\{\beta\}, \{4\}, \{A\})
\end{gathered}$$

The four first tiles come from the extracted closed 3-sets. The five last tiles are aggregates of tuples along the second attribute. In this example, two closed 3-sets do not generate any tile (see Section 7.4.3).

### 7.6.2 *Experimental Results*

A synthetic QUEST-generated dataset (144 customers buying in average 6 items out of 72 during 144 months) was tiled under the $\mathcal{C}_{\text{k-summary}} \wedge \mathcal{C}_{\text{1-volume}}$ constraint. The running time (extraction included) and the space requirements (estimated by the number of extracted closed 3-sets) are respectively plotted in Figures 26 and 27.

The QUEST-generated dataset is not much prone to be tiled. In other terms, the 3-sets it contains do not make good summaries. Indeed, none of them satisfy $\mathcal{C}_{\text{2.8-summary}} \wedge \mathcal{C}_{\text{1-volume}}$. As a consequence the quality of the tiling of the QUEST-generated dataset is not much altered when $k$ increases. Indeed, when no 3-set is extracted, the tiling (the conjunction of every aggregate of tuples along the largest attribute domain), is less than 14% bigger. Thus, enforcing $\mathcal{C}_{\text{1.5-summary}}$ divides by two the space requirements against a minor alteration of the quality of the tiling (+0.84% logic operators).

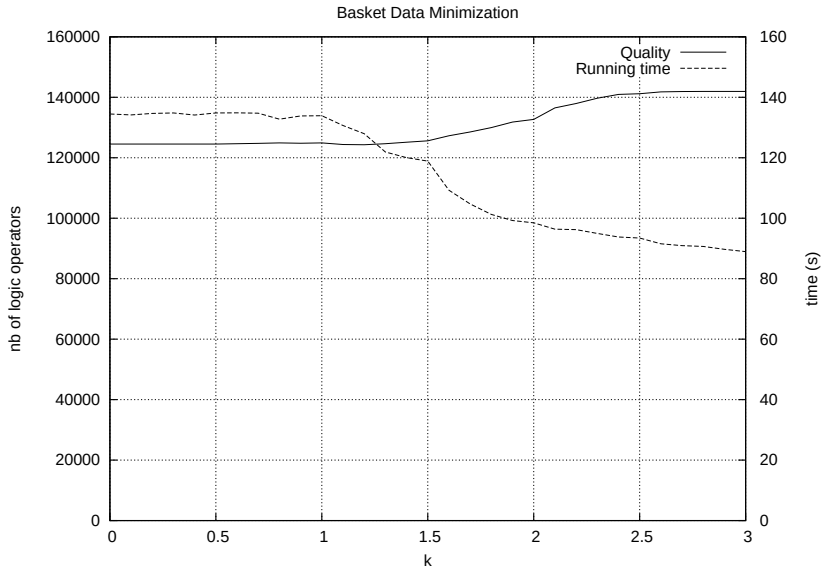



Figure 26: Time to minimize, under $\mathcal{C}_{\text{k-summary}}$, a QUEST-generated dataset.

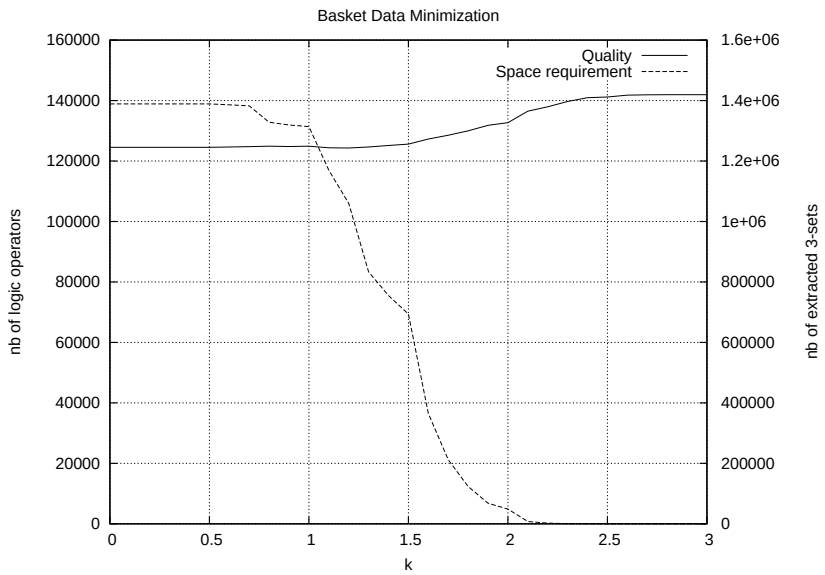



Figure 27: Space to minimize, under $\mathcal{C}_{\text{k-summary}}$, a QUEST-generated dataset.

## 8 CONCLUSION

Whatever the arity of the relation, DATA-PEELER extracts, under constraints, every closed $n$-set in it. Its enumeration principles, which do not favor any attribute, make it orders of magnitude faster than its competitors focusing on ternary relations. Furthermore, any piecewise (anti)-monotone constraint can guide this enumeration towards the most relevant patterns. Although many datasets naturally are $n$-ary relations, many others are numerical and require a binarization step before using DATA-PEELER. An additional attribute, gathering different perspectives on the data, and a minimal size constraint on its elements allow the discovery of patterns that are robust w.r.t. binarization. Another original use of DATA-PEELER is as a first step to minimize multi-valued logic functions. A post-process outputting, at every iteration, the pattern encompassing the more tuples (that have not been encompassed yet) was designed. It provides better results than the state-of-the-art ESPRESSO-MV.

# Part IV

# MINING *NOISY* N-ARY RELATIONS

The previous part dealt with generalizing closed itemset mining towards $n$-ary relations. An additional challenge is tackled in Chapter 2: noise tolerance. The good performance of Data-Peeler motivated us to reuse its principles and add noise tolerance on top of it. Thus, instead of mining noise tolerant itemsets, the proposed approach, Fenster, completely extracts noise tolerant closed patterns from $n$-ary relations, i. e., both needed generalizations to closed itemset mining (see Chapter 2) are addressed at a time. This approach was presented in [CBB09]. Chapter 4 defines the closed ET-$n$-set and details how to efficiently mine them. The principles behind Data-Peeler are found unchanged at a high level of abstraction. The constraints $\mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\text{closed}}$ are only redefined to tolerate noise. Nevertheless much work was required to efficiently implement the enforcement of these constraints. This will be emphasized by theoretically comparing the time complexity of Fenster w.r.t. a similar extractor that would naively verify these constraints. Empirically, Fenster is fast. No comparison can be made on relations with an arity greater than three because Fenster is the only approach tackling such a general task. In the particular case of binary relations, Fenster is shown to perform orders of magnitude faster than a state of the Art algorithm.

The collections of patterns Fenster computes have good global qualities, i. e., the closed ET-$n$-sets altogether cover the dataset deprived of noise. Ideally, every extracted pattern would be an *anomalously* high local density of $n$-tuples present in the relation. This follows observations made in [BCTB08] and relates to a strengthened closedness constraint that filters out the patterns that can be extended without introducing many $n$-tuples absent from the relation (see Section 2.4.2 in Chapter 1). However, reinforcing the closedness significantly decreases the global quality of the collection of patterns Fenster extracts because individual closed ET-$n$-sets only are fragment of an hidden pattern the analyst would like to find. Increasing the quantity of noise they tolerate would theoretically solve this issue. Unfortunately, more noise tolerance means longer extraction times. Even with relations suffering from rather low levels of noise, it turns out to be impossible to tolerate it all while preserving reasonable running times for Fenster.

A solution to this problem was detailed in [CMB09] and is further developed in Chapter 5. It suggests to revise our completeness demand downwards. Fenster provides fragments of the patterns dissimulated under some noise but these patterns globally are relevant. This brings the idea of agglomerating the fragments. A merging operator and a metric, which makes use of the relation (and not only of the closed ET-$n$-sets), define a hierarchical agglomeration of the patterns. An additional step selects, among the agglomerates, those that are relevant according to the intuition behind a local pattern (an anomalously high local density of $n$-tuples present in the relation). The experiments show that the whole approach, namely Alpha, returns small collections of patterns that are not only globally but also individually of a high quality.

# 4 FENSTER EXTRACTS N-SETS TOLERATING ERRORS IN THE RELATION

## 1 CLOSED ET-$n$-SETS

### 1.1 *Absolute Noise-Tolerance*

The definition of a closed $n$-set is too strict to enable the discovery of relevant patterns in *noisy* $n$-ary relations. A closed ET-$n$-set is a relaxation of the definition of a closed $n$-set. It is based on absolute noise-tolerance parameters $\epsilon = (\epsilon^i)_{i=1..n} \in \mathbb{N}^n$. Given those parameters, the type of pattern that is to be mined is defined by a conjunction of two constraints, $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$.

*Closed ET-$n$-sets tolerate absent $n$-tuples up to absolute upper-bounds per element.*

**Definition 26 ($\mathcal{C}_{\epsilon\text{-connected}}$)** $\forall X = (X^1, \dots, X^n) \in 2^{\mathcal{D}^1} \times \dots \times 2^{\mathcal{D}^n}$, $\mathcal{C}_{\epsilon\text{-connected}}(X) \equiv \forall i = 1..n, \forall e \in X^i, |(X^1 \times \dots \times \{e\} \times \dots \times X^n) \setminus \mathcal{R}| \leqslant \epsilon^i$.

**Definition 27 ($\mathcal{C}_{\epsilon\text{-closed}}$)** $\forall X = (X^1, \dots, X^n) \in 2^{\mathcal{D}^1} \times \dots \times 2^{\mathcal{D}^n}$, $\mathcal{C}_{\epsilon\text{-closed}}(X) \equiv \forall X' = (X'^1, \dots, X'^n) \in 2^{\mathcal{D}^1} \times \dots \times 2^{\mathcal{D}^n}$, $\Big((\forall i = 1..n, X^i \subseteq X'^i) \wedge \mathcal{C}_{\epsilon\text{-connected}}(X')\Big) \Rightarrow X = X'$.

**Definition 28 (Closed ET-$n$-set)** $\forall X \in 2^{\mathcal{D}^1} \times \dots \times 2^{\mathcal{D}^n}$, $X$ *is a closed ET-$n$-set iff* $\mathcal{C}_{\epsilon\text{-connected}}(X) \wedge \mathcal{C}_{\epsilon\text{-closed}}(X)$.

Let us discuss the meaning of the noise-tolerance parameters on a closed ET-$n$-set $(X^1, \dots, X^n)$. The parameter $\epsilon^i$ quantifies, on any element in $X^i$, the maximal number of $n$-tuples that are allowed to be absent from $\mathcal{R}$. In other terms, with a spatial vision of a pattern (an $n$-dimensional rectangle in $\mathcal{R}$ modulo permutations of the elements), $\epsilon^i$ is the maximal number of '0's on any hyper-plan of the $i^{\text{th}}$ dimension. Furthermore $\mathcal{C}_{\epsilon\text{-closed}}$ forces $(X^1, \dots, X^n)$ to be closed, i. e., any extension of it will break $\mathcal{C}_{\epsilon\text{-connected}}$. If $\forall i = 1..n, \epsilon^i = 0$ then $\mathcal{C}_{\epsilon\text{-connected}} \equiv \mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}} \equiv \mathcal{C}_{\text{closed}}$. It can be written that the closed ET-$n$-set is a generalization of the closed $n$-set.

Since the definition of a closed ET-$n$-set uses an absolute noise-tolerance (it considers numbers rather than proportions of '0's), the following function helps in referring to counts of $n$-tuples absent from $\mathcal{R}$ on any element of an $n$-set.

**Definition 29 (Function $0$)** $\forall X = (X^1, \dots, X^n) \in 2^{\mathcal{D}^1} \times \dots \times 2^{\mathcal{D}^n}$, $\forall i = 1..n, \forall e \in \mathcal{D}^i$, $0(X, e) = |(X^1 \times \dots \times \{e\} \times \dots \times X^n) \setminus \mathcal{R}|$.

Let us use this function to rewrite Definition 26 and 27:

**Definition 30 ($\mathcal{C}_{\epsilon\text{-connected}}$)** $\forall X \in 2^{\mathcal{D}^1} \times \dots \times 2^{\mathcal{D}^n}$, $\mathcal{C}_{\epsilon\text{-connected}}(X) \equiv \forall i = 1..n, \forall e \in X^i, 0(X, e) \leqslant \epsilon^i$.

**Definition 31 ($\mathcal{C}_{\epsilon\text{-closed}}$)** $\forall X \in 2^{\mathcal{D}^1} \times \dots \times 2^{\mathcal{D}^n}$, $\mathcal{C}_{\epsilon\text{-closed}}(X) \equiv \forall i = 1..n$,

$$\forall e \in \mathcal{D}^i \setminus X^i, \begin{cases} 0(X, e) > \epsilon^i \\ \textit{or} \\ \exists j \neq i, \exists f \in X^j \textit{ s.t. } 0((X^1, \dots, X^i \cup \{e\}, \dots, X^n), f) > \epsilon^j \end{cases}.$$

Definition 30 is a direct rewrite of Definition 26. Definition 31 is more than that. It is equivalent to Definition 27 because if a pattern can be extended without violating $\mathcal{C}_{\epsilon\text{-connected}}$, then there is such an extension with *one* element only (and the reverse obviously is true too). Furthermore, Definition 31 details the two ways to break $\mathcal{C}_{\epsilon\text{-connected}}$. Either the element to extend the closed ET-n-set gathers, when projected on the pattern, too many n-tuples absent from $\mathcal{R}$ or this additional element makes the number of '0's on an orthogonal element (an element from another domain of the relation) exceed the related noise-tolerance parameter. Examples taken from $\mathcal{R}_E$ (see Table 5) help in understanding that:

**Example 20** *Let* $\epsilon = (1, 1, 1)$. $X = (\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ *is a closed ET-3-set in* $\mathcal{R}_E$. $X$ *satisfies* $\mathcal{C}_{\epsilon\text{-connected}}$ *since each of its hyper-plans contains, at most, one 3-tuple absent from* $\mathcal{R}_E$*:* $0(X, \alpha) = 0$, $0(X, \gamma) = 1$, $0(X, 1) = 0$, $0(X, 2) = 0$, $0(X, 3) = 1$ *and* $0(X, B) = 1$. $X$ *satisfies* $\mathcal{C}_{\epsilon\text{-closed}}$ *because extending it with any additional element either means that the hyper-plan of* $X$ *on this element contains strictly more than one 3-tuple absent from* $\mathcal{R}_E$ *(e. g.,* $0(X, \beta) = 2$*) or at least one of the hyper-plans on an orthogonal element in* $X$ *would contain strictly more than one 3-tuple absent from* $\mathcal{R}_E$ *(e. g., 4 cannot extend* $X$ *because* $0((\{\alpha, \gamma\}, \{1, 2, 3, 4\}, \{B\}), B) = 2$*).* $(\alpha, \beta, \gamma\}, \{1, 2\}, \{A, B\})$ *is another closed ET-3-set in* $\mathcal{R}_E$.

## 1.2 *Relative Noise-Tolerance*

The reader may wonder why our definition of a closed ET-n-set is based on absolute parameters and not relative ones. The reasons are the same as those cited in the binary case (see Section 1.1.2 in Chapter 2), i. e., using a relative tolerance to noise makes the extraction suffer from great scalability issues and the closedness constraint does not provide a lossless condensation of all ET-connected n-sets (and, without it, output collections are very large and much redundant). Nevertheless, thanks to its enumeration principles inherited from DATA-PEELER, our closed ET-n-set extractor, named FENSTER, can efficiently enforce any piecewise (anti)-monotone constraints. Such a constraint allows to focus on a tight range of sizes (for the extracted closed ET-n-sets) where a user-defined relative parametrization can be converted into an absolute one. Using this constraint, FENSTER returns the same collection of patterns as one based on a relative parametrization (in the chosen range of sizes). Given user-defined relative parameters $(r^i)_{i=1..n} \in [0, 1]^n$ and absolute tolerances to errors $(\epsilon^i)_{i=1..n} \in \mathbb{N}^n$, here is this constraint:

*Closed ET-n-sets are a lossless condensation of all ET-n-sets.*

*FENSTER efficiently handles any piecewise (anti)-monotone constraint. E. g., it can focus on the region of the search space where the absolute noise tolerance is equivalent to a relative one.*

$$\mathcal{C}_{\text{in-region-of-interest}}(X) \equiv \bigwedge_{i=1}^{n} \left(\epsilon^i \leqslant r^i \prod_{j \neq i} |X^j| < \epsilon^i + 1\right) .$$

The proof of the piecewise (anti)-monotonicity of $\mathcal{C}_{\text{in-region-of-interest}}$ is based on, first, splitting the double inequalities, then, showing that the left ones are anti-monotone and the right ones monotone.

Given a relative parametrization $(r^i)_{i=1..n}$ and an interesting range of sizes (i. e., a *region of interest*), the absolute parameters $(\epsilon^i)_{i=1..n}$ are easily computed. Figure **??** depicts these regions in the case of a binary relation, i. e., it plots the contour lines of every region related to every absolute parametrization $(\epsilon^1, \epsilon^2)$ (restricted to $\{0, 1, 2\} \times \{0, 1\}$ in the figure). The x-axis (resp. the y-axis) relates to the size of the extracted closed ET-n-sets on the first (resp. second) attribute. The

| | | |
|---|---|---|
| $\lvert X^2\rvert \in \left[\frac{2}{r^1}, \frac{3}{r^1}\right[$ | $(2,0)$ | $(2,1)$ |
| $\lvert X^2\rvert \in \left[\frac{1}{r^1}, \frac{2}{r^1}\right[$ | $(1,0)$ | $(1,1)$ |
| $\lvert X^2\rvert \in \left[0, \frac{1}{r^1}\right[$ | $(0,0)$ | $(0,1)$ |
| | $\lvert X^1\rvert \in \left[0, \frac{1}{r^2}\right[$ | $\lvert X^1\rvert \in \left[\frac{1}{r^2}, \frac{2}{r^2}\right[$ |

Figure 28: Conversion from a relative noise tolerance $(r^1, r^2)$ to an absolute one $(\epsilon^1, \epsilon^2)$ depending on the region of interest.

couple of integers inside a region are the absolute parameters FENSTER uses to extract all ET-n-sets whose "geometry" fits in the region. When $n \geqslant 3$, analog (but not rectangular anymore) regions can be drawn in an n-dimensional coordinate system $(|X^1|, \ldots, |X^n|)$. When several regions contain (relatively defined) closed ET-n-sets of interest, several extractions, under the strong constraint $\mathcal{C}_{\text{in-region-of-interest}}$, allow, by union of their returned collections of patterns, to list them all.

## 2 FENSTER

FENSTER builds upon the powerful enumeration principles of DATA-PEELER to exhaustively list the closed ET-n-sets. This allows, in particular, to efficiently enforce any piecewise (anti)-monotone constraint. From an abstract perspective, FENSTER looks like DATA-PEELER with $\mathcal{C}_{\epsilon\text{-connected}}$ instead of $\mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ instead of $\mathcal{C}_{\text{closed}}$. Nevertheless, as it will be detailed in Section 3.1, the efficient enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ requires, at every enumeration node to reuse counts of absent n-tuples that were previously made and that need to be updated.

*From an abstract perspective,* FENSTER *proceeds like* DATA-PEELER.

Similarly to DATA-PEELER, three n-sets $U = (U^1, \ldots, U^n)$, $V = (V^1, \ldots, V^n)$, and $S = (S^1, \ldots, S^n)$ are attached to every enumeration node of FENSTER. We recall that all the elements in the n-set $U \in 2^{\mathcal{D}^1} \times \cdots \times 2^{\mathcal{D}^n}$ are contained in any closed ET-n-set extracted from the node. The n-set $V \in 2^{\mathcal{D}^1} \times \cdots \times 2^{\mathcal{D}^n}$ contains the elements that may be present in the closed ET-n-sets extracted from the node, i. e., the search space. Finally, the n-set $S \in 2^{\mathcal{D}^1} \times \cdots \times 2^{\mathcal{D}^n}$ contains the elements that may prevent the ET-n-sets, extracted from this node, from being closed.

The enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$ is analog to that of $\mathcal{C}_{\text{closed}}$ by DATA-PEELER (see Section 1.4). Figure 29 depicts the enumeration of FENSTER. The n-sets $U$, $V$ and $S$ attached to the children nodes are computed from the parent's analogous n-sets, the enumerated element and the data (for the left child only). In particular, in a left child, FENSTER ensures that $U$ can receive any element from $V$ without violating $\mathcal{C}_{\epsilon\text{-connected}}$. Hence, at every enumeration node, the n-set $U$ is $\epsilon$-connected, i. e., $\mathcal{C}_{\epsilon\text{-connected}}(U)$. Furthermore the reduction of $S$ performed by DATA-PEELER (see Section 2.1) is applied by FENSTER too. In the end, Figure 29 is very similar to Figure 17.

**Example 21** *Let* $\epsilon = (1,1,1)$. *Consider that* FENSTER, *working on the relation* $\mathcal{R}_E$ *(represented in Table 5), reaches the enumeration node where* $U = (\{\alpha, \gamma\}, \emptyset, \{B\})$, $V = (\emptyset, \{1,2,3,4\}, \{A\})$ *and* $S = (\{\beta\}, \emptyset, \{C\})$. FENSTER *chooses to enumerate the element* $A \in V^3$ *and generates the two children depicted in Figure 30. In the left child,* 3 *and* 4 *are removed from* $V^2$ *because*

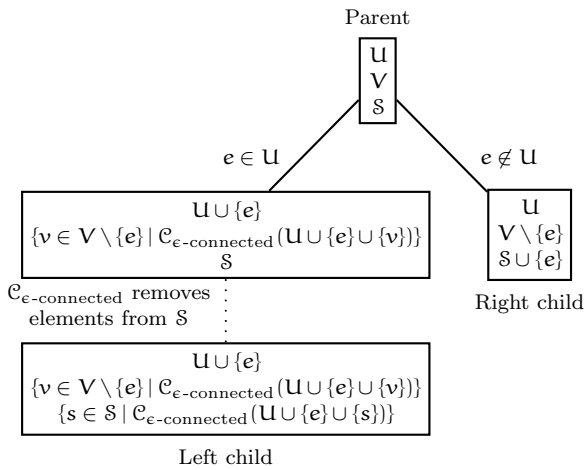



Figure 29: FENSTER enumerating any element $e \in V$.

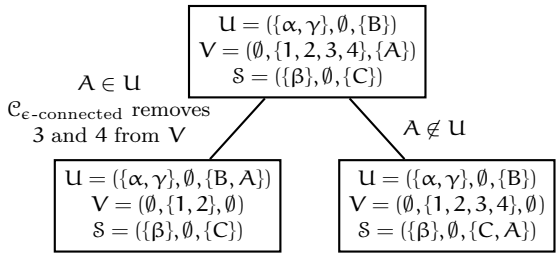



Figure 30: Illustration of Example 21.

*neither* $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{3\}, \{B\})$ *nor* $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{4\}, \{B\})$ *is true. In this example, none of the elements in the* $S$ *n-set of the left child can be removed.*

At this point the reader may wonder why FENSTER does not take advantage of the last improvement to DATA-PEELER's enumeration, i. e., why an element $v \in V$ cannot be moved to $U$ when $\mathcal{C}_{\epsilon\text{-connected}}(U^1 \cup V^1, \ldots, \{v\}, \ldots, U^n \cup V^n)$ holds. In fact, if $\forall i = 1..n, \epsilon^i \neq 0$, then $\mathcal{C}_{\epsilon\text{-connected}}(U^1 \cup V^1, \ldots, \{v\}, \ldots, U^n \cup V^n)$ is not sufficient to entail the presence of $v$ in every closed ET-n-set represented by the current enumeration node $(U, V)$. Stated in mathematical terms, the $\epsilon$-closedness of an n-set ($U$ in our extraction context) is *not unique* (but all of them are listed by FENSTER). Indeed, given a domain $i = 1..n$, while $\mathcal{C}_{\epsilon\text{-connected}}(U^1 \cup V^1, \ldots, \{v\}, \ldots, U^n \cup V^n)$ may be true for several $v \in V^i$, an orthogonal element $f \in V^{j \neq i}$ may gather more than $\epsilon^j$ n-tuples absent from $\mathcal{R}$ in the n-set extended by *several* of these elements in $V^i$. As a consequence, such an element $v$ does not belong to every closed ET-n-set represented by $(U, V)$. This is easier to understand via an example on $\mathcal{R}_E$ (see Table 5).

*Counts of absent n-tuples involving an element in the search space never allow to claim this element present.*

**Example 22** *Let* $\epsilon = (1, 1, 1)$. *Consider the enumeration node where* $U = (\{\alpha, \gamma\}, \{1, 2\}, \{B\})$ *and* $V = (\emptyset, \{3, 4\}, \emptyset)$. *Both* $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{3\}, \{B\})$ *and* $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{4\}, \{B\})$ *are true. However neither* 3 *nor* 4 *will be part of every closed ET-n-set the current node represents:* $(\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ *does contain* 3 *but not* 4*;* $(\{\alpha, \gamma\}, \{1, 2, 4\}, \{B\})$ *does* 4 *but not* 3.

Nevertheless, a weaker improvement was implemented. It consists in checking whether the elements of $V$ *altogether* can extend $U$ while

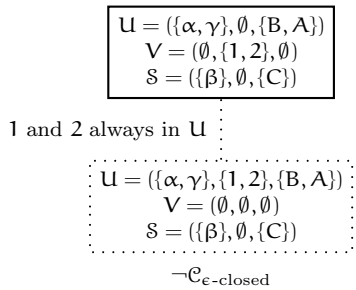



Figure 31: Illustration of Example 23.

preserving $\mathcal{C}_{\epsilon\text{-connected}}$. In other terms, $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V)$ is tested. If it is satisfied, a direct jump to the leftmost leaf of the enumeration sub-tree (rooted by the current node) is performed. This jump is safe, i. e., it does not "jump over" closed ET-n-sets. Indeed $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V)$ implies that every n-set strictly "included in" (see Definition 23) $U \sqcup V$ can be extended by the missing elements. By definition, they are not $\epsilon$-closed. Because most of the nodes are at the bottom of the enumeration tree (in a complete binary tree, half of the nodes are leaves), this improvement, though weaker than its analog in DATA-PEELER, significantly reduces the extraction times.

*Given an enumeration sub-tree, if its largest n-set is ET-connected,* FENSTER *jumps to it.*

**Example 23** *In the left child of Example 21,* $U \sqcup V = (\{\alpha, \gamma\}, \{1, 2\}, \{B, A\})$ *and* $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{1, 2\}, \{B, A\})$ *is true. The improvement is applied (see Figure 31) but the obtained node is not* $\epsilon$*-closed:* $\beta \in S^1$ *can extend it.*

In summary, at a high level of abstraction, FENSTER is DATA-PEELER with $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ respectively substituting $\mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\text{closed}}$ and the second improvement to DATA-PEELER's enumeration weakened. The pseudo-code for FENSTER is displayed in Figure 32. FENSTER, like DATA-PEELER, recursively traverses the search space depth-first, is initially called is with $U = (\emptyset, \ldots, \emptyset)$, $V = (\mathcal{D}^1, \ldots, \mathcal{D}^n)$, and $S = (\emptyset, \ldots, \emptyset)$ and $\mathcal{C}_{P(A)M}$ can be any piecewise (anti)-monotone constraint the relevant closed ET-n-sets satisfy.

## 3 IMPLEMENTATION

### 3.1 $\mathcal{C}_{\epsilon\text{-connected}}$ *and* $\mathcal{C}_{\epsilon\text{-closed}}$

#### 3.1.1 *Performance Issue*

Even though the enumeration of FENSTER was inspired by that of DATA-PEELER, FENSTER is not a trivial extension of DATA-PEELER. A naive enforcements of the new constraints $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ would lead to disastrous extraction times. Contrary to DATA-PEELER, FENSTER cannot traverse small subspaces of the dataset in search of *one* n-tuple absent from $\mathcal{R}$. When an element $e$ is chosen, the absence in $\mathcal{R}$ of *one* n-tuple with $e$, i. e., on one hyper-plan, is not enough to enforce $\mathcal{C}_{\epsilon\text{-connected}}$, whereas it is when enforcing $\mathcal{C}_{\text{connected}}$. Searching for *several* n-tuples absent from $\mathcal{R}$ in this hyper-plan is not enough either. FENSTER needs to know the other n-tuples absent from $\mathcal{R}$ that were previously tolerated in every n-set represented by the current node, i. e., the n-tuples in $(\times_{i=1..n} U^i) \setminus \mathcal{R}$. It needs to know *where*, i. e., on which hyper-plans, they are and how many of them are found on

*Contrary to* DATA-PEELER*, a naive implementation of* FENSTER *would access large parts of the dataset at every iteration.*

**Input:** $U, V, S$
**Output:** Every closed ET-$n$-set containing every element in $U$, possibly some elements in $V$, and satisfying $\mathcal{C}_{\text{P(A)M}}$
**if** $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V)$ **then**
  $U \leftarrow U \sqcup V$
  $V \leftarrow (\emptyset, \ldots, \emptyset)$
**end if**
**if** $\mathcal{C}_{\text{P(A)M}}$ may be satisfied by an $n$-set descending from this node $\wedge \mathcal{C}_{\epsilon\text{-closed}}(U \sqcup V)$ **then**
  **if** $V = (\emptyset, \ldots, \emptyset)$ **then**
    **output**$(U)$
  **else**
    Choose $e \in V$
    FENSTER$(U \cup \{e\}$,
    $\{v \in V \setminus \{e\} \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{v\})\}$,
    $\{s \in S \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{s\})\}$)
    FENSTER$(U, V \setminus \{e\}, S \cup \{e\})$
  **end if**
**end if**

Figure 32: The FENSTER algorithm.

each of these hyper-plans. The enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$ raises the same trouble: given $U$, $V$ and $S$, the $\epsilon$-closedness of some $n$-set represented by $(U, V)$ cannot be proved by only consulting with the $n$-tuples in $\times_{i=1..n} U^i \cup V^i$ involving the elements in $S$. As a consequence, a naive enforcement of $\mathcal{C}_{\text{connected}}$ (resp. $\mathcal{C}_{\epsilon\text{-closed}}$) would, at every iteration, count the numbers of $n$-tuples absent from $\mathcal{R}$ in every hyper-plan of $U$ (resp. $U \sqcup V$) and on each of its projections on the elements in $V$ (resp. $S$). Such an implementation would be intractable even on rather small relations.

### 3.1.2 *Noise Counters in Relevant Subspaces of the Relation*

FENSTER *incrementally computes counters helping the verification of the constraints defining a closed ET-n-set.*

To drastically improve the performance, FENSTER relies on the following observation: from a parent enumeration node to its children, $U$ and $U \sqcup V$ do not change much. $U$ only grows by one element in the left child and $U \sqcup V$ looses one element in the right child, potentially more in the left child. Instead of traversing $\times_{i=1..n} U^i \cup V^i \cup S^i$ at every iteration, FENSTER updates counters of absent $n$-tuples. Focusing on the symmetric differences between the $n$-sets $U_P$ and $V_P$ at the parent node and the respective $n$-sets $U$ and $V$ at the child node is enough to updates such counters. This means a much better time performance (than the naive approach) to the cost of a worse memory consumption (to store the counters). Later, it will be formally shown that the time gain is huge while the space complexity is, in fact, dominated by the dataset when $n \geqslant 4$.

Let us finally list the counters that are relevant when enforcing $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$. Keeping in mind their definitions while looking at Figure 32 provides the following list:

- To check $\mathcal{C}_{\epsilon\text{-closed}}(U \sqcup V)$:
  - $\forall s \in S, \mathcal{O}(U \sqcup V, s)$;
  - $\forall s \in S, \forall u \in U, \mathcal{O}(U \sqcup V \cup \{s\}, u)$.

- Given $e \in V$ and $U_L = U \cup \{e\}$ (the elements that are present in every $n$-set descendant of the left child), to compute $\{v \in V \setminus \{e\} \mid \mathcal{C}_{\epsilon\text{-connected}}(U_L \cup \{v\})\}$:
  - $\forall v \in V, \mathcal{O}(U_L, v)$;
  - $\forall v \in V, \forall u \in U_L, \mathcal{O}(U_L \cup \{v\}, u)$.
- Given $e \in V$ and $U_L = U \cup \{e\}$ (the elements that are present in every $n$-set descendant of the left child), to compute $\{s \in \mathcal{S} \mid \mathcal{C}_{\epsilon\text{-connected}}(U_L \cup \{s\})\}$:
  - $\forall s \in \mathcal{S}, \mathcal{O}(U_L, s)$;
  - $\forall s \in \mathcal{S}, \forall u \in U_L, \mathcal{O}(U_L \cup \{s\}, u)$.

By factorizing the last two points, four families of counters are useful:
- $\forall f \in \mathcal{S}, \mathcal{O}(U \sqcup V, f)$;
- $\forall f \in \mathcal{S}, \forall u \in U, \mathcal{O}(U \sqcup V \cup \{f\}, u)$;
- $\forall f \in V \sqcup \mathcal{S}, \mathcal{O}(U_L, f)$;
- $\forall f \in V \sqcup \mathcal{S}, \forall u \in U_L, \mathcal{O}(U_L \cup \{f\}, u)$.

Because any element in $V$ may, in the descendant nodes, belong to a $U$ or a $\mathcal{S}$ $n$-set, these counters, $\mathcal{O}(U \sqcup V, f)$, $\mathcal{O}(U \sqcup V \cup \{f\}, u)$, $\mathcal{O}(U_L, f)$ and $\mathcal{O}(U_L \cup \{f\}, u)$, are maintained updated for every $(f, u) \in (U \sqcup V \sqcup \mathcal{S})^2 \setminus U^2$. In this way, some counters are only used when the element defining the hyper-plan is in a specific set (e. g., a counter $\mathcal{O}(U \sqcup V, f)$ is not used until $f \in \mathcal{S}$). Anyway, it is advantageous to maintain them updated for every element that may reach a state where they would be useful (e. g., $\mathcal{O}(U \sqcup V, f)$ is maintained updated even if $f \in V$). An alternative strategy would be to initialize a counter when required. It would be less efficient because, along the enumeration tree, there are exponentially many states where a given counter is useful. As a consequence, the cost of an on-demand initialization of the counter (scan of part of the dataset) multiplied by this number of states exceeds the cost of maintaining them all updated until used or useless. Thus, all counters are initialized while storing the dataset and, whenever elements are moved or removed from $V$, the counters are updated by only traversing the symmetric differences between the $n$-sets $U_P$ and $V_P$ at the parent node and the respective $n$-sets $U$ and $V$ at the child node. In FENSTER, this update is further improved: the counters of the type $\mathcal{O}(U_L \cup \{f\}, u)$ and $\mathcal{O}(U \sqcup V \cup \{f\}, u)$ ($(f, u) \in (U \sqcup V \sqcup \mathcal{S})^2 \setminus U^2$) are replaced by:

*FENSTER incrementally updates counters of absent $n$-tuples involving elements or pairs of elements.*

- $\mathcal{O}(U_L, f, u) = |(U_L^1 \times \cdots \times \{f\} \times \cdots \times \{u\} \times \cdots \times U_L^n) \setminus \mathcal{R}|$;
- $\mathcal{O}(U \sqcup V, f, u) = |(U^1 \cup V^1 \times \cdots \times \{f\} \times \cdots \times \{u\} \times \cdots \times U^n \cup V^n) \setminus \mathcal{R}|$.

The desired quantities $\mathcal{O}(U_L \cup \{f\}, u)$ and $\mathcal{O}(U \sqcup V \cup \{f\}, u)$ can be computed, still without any access to the relation:
- $\mathcal{O}(U_L \cup \{f\}, u) = \mathcal{O}(U_L, u) + \mathcal{O}(U_L, f, u)$;
- $\mathcal{O}(U \sqcup V \cup \{f\}, u) = \mathcal{O}(U \sqcup V, u) + \mathcal{O}(U \sqcup V, f, u)$.

The new counters involving much smaller subspaces (one dimension less) than the original ones, they do not need to be updated as often, hence the additional time gain.

**Example 24** *Consider, like in Example 22, that* FENSTER, *working on the relation $\mathcal{R}_E$ (represented in Table 5), reaches the enumeration node where $U = (\{\alpha, \gamma\}, \{1, 2\}, \{B\})$ and $V = (\emptyset, \{3, 4\}, \emptyset)$. Consider, moreover, that $\mathcal{S} = (\{\beta\}, \emptyset, \{A\})$. Although this enumeration node is rather small, it is associated with too many counters to list them all here. Among them, $\mathcal{O}(U, \alpha) = 0$, $\mathcal{O}(U \sqcup V, \alpha) = 1$, $\mathcal{O}(U, 3) = 1$, $\mathcal{O}(U \sqcup V, 3) = 1$, $\mathcal{O}(U, A) = 0$, $\mathcal{O}(U \sqcup V, A) = 2$, $\mathcal{O}(U, \alpha, 3) = 0$, $\mathcal{O}(U \sqcup V, \alpha, 3) = 0$, $\mathcal{O}(U, \alpha, A) = 0$, $\mathcal{O}(U \sqcup V, \alpha, A) = 2$, $\mathcal{O}(U, 3, A) = 1$, $\mathcal{O}(U \sqcup V, 3, A) = 1$, etc.*

### 3.1.3 *Time Gain*

Consider an enumeration node $(U, V)$, its $n$-set $\mathcal{S}$ and the last enumerated element $e$. With the naive enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$, $|\times_{i=1..n} U^i| + |\{\times_{i=1..n} U^i \cup V^i \cup \{s\} \mid s \in \mathcal{S}\}|$ $n$-sets would be traversed at a left child (the first term to enforce $\mathcal{C}_{\epsilon\text{-connected}}$, the second to enforce $\mathcal{C}_{\epsilon\text{-closed}}$) and $|\{\times_{i=1..n} U^i \cup V^i \cup \{s\} \mid s \in \mathcal{S}\}|$ at a right child (where only $\mathcal{C}_{\epsilon\text{-closed}}$ is enforced). If, in a left child, $\times_{i=1..n} U^i$ is traversed only once to enforce both $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$, the time cost is $O(|\{\times_{i=1..n} U^i \cup V^i \cup \{s\} \mid s \in \mathcal{S}\}|)$ for any enumeration node.

The use of the counters, we came up with in the previous section, restricts the number of $n$-sets traversed when updating them to $|U^1 \times \cdots \times \{e\} \times \cdots \times U^n| + |\{U^1 \cup V^1 \times \cdots \times \{e\} \times \cdots \times \{s\} \times \cdots \times U^n \cup V^n \mid s \in \mathcal{S}\}|$ at a left child and $|\{U^1 \cup V^1 \times \cdots \times \{e\} \times \cdots \times \{s\} \times \cdots \times U^n \cup V^n \mid s \in \mathcal{S}\}|$ at a right child. Since, in a left child, FENSTER actually traverses $U^1 \times \cdots \times \{e\} \times \cdots \times U^n$ only once for updating both the counters related to $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$, the time FENSTER takes traversing the dataset to update counters is $O(|\{U^1 \cup V^1 \times \cdots \times \{e\} \times \cdots \times \{s\} \times \cdots \times U^n \cup V^n \mid s \in \mathcal{S}\}|)$ for every enumeration node. By comparison with the naive approach and supposing $e \in \mathcal{D}^d$, this is $|U^d \cup V^d|$ times less. The enumeration strategy, hence the number of enumeration nodes, being the same in both cases, it could be written that the use of counters allows an equivalent division of the total extraction times the naive implementation would provide.

Nevertheless, this is true only if the time spent using the counters (to actually enforce $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$) is dominated by the time spent updating them. To study that, the number of counters accessed at every node is computed. To enforce $\mathcal{C}_{\epsilon\text{-connected}}$, it is, at worst (none of the elements in $V$ are removed), $|V| + 2\sum_{i=1..n} |V^i| \sum_{j \neq i} |U^j|$. To enforce $\mathcal{C}_{\epsilon\text{-closed}}$, it is, at worst (every element in $\mathcal{S}$ extends $U \sqcup V$), $|\mathcal{S}| + 2\sum_{i=1..n} |\mathcal{S}^i| \sum_{j \neq i} |U^j \cup V^j|$. In both cases, the first term relates to checking whether every hyper-plan $v \in V$ (resp. $s \in \mathcal{S}$) contains too many (resp. enough) $n$-tuples absent from $\mathcal{R}$ to satisfy $\mathcal{C}_{\epsilon\text{-connected}}$ (resp. $\mathcal{C}_{\epsilon\text{-closed}}$) and the second term relates to checking whether an hyper-plan $v \in V$ (resp. $s \in \mathcal{S}$), if added to $U$ (resp. $U \sqcup V$) would make any orthogonal element exceed its noise tolerance threshold. The total number of counters used to enforce $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ is dominated by $2\sum_{i=1..n} |\mathcal{S}^i| \sum_{j \neq i} |U^j \cup V^j|$. This number is now compared to the number of $n$-sets traversed when updating the counters. When $n = 2$ it takes more time to use the counters than to update them. The time taken to use the counters is $O(|((U^1 \cup V^1) \times \{\mathcal{S}^2\}) \cup (\{\mathcal{S}^1\} \times (U^2 \cup V^2))|)$, i. e., similar to the time the naive implementation would take to enforce $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$. When $n = 3$, the times to update and use the counters are on the same order, whereas the update dominates when $n \geqslant 4$. This is deduced from comparing the numbers computed above but here is a more intuitive way to understand it: the number of $n$-sets traversed, at every iteration, to update the counters is an hyper-plan (related to $e$) of the dataset, i. e., an $(n-1)$-dimensional subspace, whereas the number of used counters, whatever the arity of $\mathcal{R}$, is on the order of a 2-dimensional subspace of the dataset (the finest counters are numbers of $n$-tuples absent from $\mathcal{R}$ at the intersection of *two* orthogonal hyper-plans). To conclude, the counters present an advantage over the naive implementation when $n \geqslant 3$. FENSTER, working on such an $n$-ary relation, is as fast as the naive implementation processing a relation on domains of the same sizes but with an arity of $n-1$.

*On $n$-ary relations with $n \geqslant 3$, using counters makes* FENSTER *as fast as the naive implementation on $(n-1)$-ary relations.*

### 3.2 *Choosing the Element to Enumerate*

At every recursive call, any element, from any attribute, can be enumerated (function *Choose* in Figure 32). Section 5.2 in Chapter 3 empirically showed that different sensible strategies produce different enumeration trees whose sizes (hence, the time required to traverse them) varies between several orders of magnitude. Compared to DATA-PEELER, FENSTER profits from more information at its disposal when it comes to choose an element to enumerate: the counters allow a finer choice. In this way, FENSTER builds smaller enumeration trees. Like DATA-PEELER (see Section 1.6), FENSTER chooses the enumerated element in two stages:

*The attribute domain of the enumerated element is chosen like* DATA-PEELER *does. In this domain, the chosen element introduces as many absent $n$-tuples as possible.*

1. The attribute domain, in which the element will be enumerated, is chosen.
2. The element itself is chosen.

The first stage is that of DATA-PEELER: the chosen attribute domain $\mathcal{D}^d$ maximizes the following function that increases with the average number of elements that may be, in the left child, removed from $V$.

$$\sum_{k \neq d} \left( |V^k| \times \prod_{l \notin \{d,k\}} |U^l| \right) .$$

Understood with FENSTER's way of enforcing $\mathcal{C}_{\epsilon\text{-connected}}$, this function actually computes the number of $n$-tuples that are browsed to update the counters $\mathcal{O}(U_L, f)$ and $\mathcal{O}(U_L, f, u)$ ($f \in V^{k \neq d}, u \in U^{l \notin \{d,k\}}$) involved in this process.

Then, FENSTER takes advantage of the counters $\mathcal{O}(U_L, f)$. It chooses the element $f \in V^d$ providing the greatest $\mathcal{O}(U_L, f)$. The justification for this choice is simple: the more $n$-tuples in $U^1 \times \cdots \times U^n$ that are absent from $\mathcal{R}$, the less room for others, hence the smaller the search space of the left child. For the same reason applied to the left grandchild (and beyond), when several elements in $V^d$ maximizes $\mathcal{O}(U, f)$, an element leading to a greater $\mathcal{O}(U \sqcup V, f)$ is preferred.

**Example 25** *In the Example 21, illustrated by Figure 30, the choice of enumerating $A \in V^3$ actually follows the heuristic stated above:*

CHOICE OF $v^3$: $\sum_{k \neq d} \left( |V^k| \times \prod_{l \notin \{d,k\}} |U^l| \right)$ *is maximized for* $d = 3$*:*

$$d = 1\text{: } (|V^2| \times |U^3|) + (|V^3| \times |U^2|) = (4 \times 1) + (1 \times 0) = 4;$$
$$d = 2\text{: } (|V^1| \times |U^3|) + (|V^3| \times |U^1|) = (0 \times 1) + (1 \times 2) = 2;$$
$$d = 3\text{: } (|V^1| \times |U^2|) + (|V^2| \times |U^1|) = (0 \times 0) + (4 \times 2) = 8.$$

CHOICE OF $a$: *Among the elements in $V^3$, $e = A$ maximizes the value of $\mathcal{O}((\{\alpha, \gamma\}, \emptyset, \{B\}), e)$ (this value is $0$ but $V^3$ only contains $A$).*

## 4 SPACE COMPLEXITY

The $n$-sets $U$, $V$ and $\mathcal{S}$ and all the counters associated with the elements they gather need to be copied whenever a left child enumeration node is built. In spite of that, the depth-first traversal of the search space makes the space complexity of FENSTER be dominated by the storage of the relation, similar to that of DATA-PEELER (see Section 4.1 in Chapter 3), when $n \geqslant 4$. In this case, it is $O(\prod_{i=1}^{n} |\mathcal{D}^i|)$. When $n \in \{2, 3\}$, the counters occupy most of the memory and the space

*Unless the relation is binary or ternary, its storage dominates the space complexity. The time and space costs of copying the counters is significantly reduced if the right children overwrite the parents.*

complexity is $O(|D^i|^2 \times |D^j|)$, where $D^i$ is the largest attribute domain and $D^j$ the second largest.

The time and space requirements are significantly reduced by making the right child enumeration nodes overwrite their parent. In this way, the counters do not need to be copied. Overwriting the parent enumeration nodes with their left children would not provide as much gain. Indeed, in a right child enumeration node, the search space $V$ is only reduced by one element and $U$ stays unchanged. Because of that, the enumeration sub-tree rooted by a right child node is far less often pruned (by $\mathcal{C}_{\epsilon\text{-closed}}$ or $\mathcal{C}_{P(A)M}$) than that of a left child, where, in particular, the search space $V$ may be greatly reduced. As a consequence, the recursive calls of FENSTER (see Figure 32), down to a leaf, usually involve far more right children than left ones and, in practical settings, overwriting the parent enumeration nodes with their right children significantly decreases the average number of nodes to be kept in memory. It even provides substantial gains in terms of average extraction time because the cost of copying all counters is taken off.

## 5 EMPIRICAL STUDY

FENSTER was coded in C++ and compiled with GCC 4.3.2. Most of the following experiments were performed on an Intel® processor cadenced at 2.8GHz, 3 Gb of RAM and running a GNU/Linux™ operating system. Because `AC-Close` only runs on Windows™, the experiment involving it was performed on another computer equipped with an Intel® processor cadenced at 2.26GHz and 3 Gb of RAM. The implementations of CUBEMINER [41], TRIAS [40] and `AC-Close` [21] were kindly provided by their respective authors.

### 5.1 *Synthesizing Datasets*

Four, possibly overlapping, $n$-sets are randomly placed in a *cubic* dataset, i. e., all attribute domains have the same cardinality. The obtained relation is named $\mathcal{R}_{\text{hidden}}$. Some noise was added to $\mathcal{R}_{\text{hidden}}$. In this way we obtain the relation $\mathcal{R}$ that is mined. The noise follows a Bernoulli distribution, i. e., every $n$-tuple has the same probability (called "noise level") to be switched (an $n$-tuple absent from $\mathcal{R}_{\text{hidden}}$ becomes present in $\mathcal{R}$ or the opposite). The experiments are performed with relations whose noise level varies between 0 and 0.45 (0.5 corresponds to purely random datasets). The mining task being symmetric w.r.t. the attributes, every tested parametrization satisfies $\forall i = 1..n, \epsilon^i = \epsilon \in \mathbb{N}$.

### 5.2 *Global Quality Results*

Let $\mathcal{E}$ the set of $n$-tuples encompassed by at least one closed ET-$n$-set. The *global quality* of the mined collection of closed ET-$n$-sets is measured by: $\frac{|\mathcal{R}_{\text{hidden}} \cap \mathcal{E}|}{|\mathcal{R}_{\text{hidden}} \cup \mathcal{E}|}$. It will be shown later that this measure does not reflect the quality as perceived by analyst. However the good global qualities FENSTER obtains allow, via a post process detailed in the next chapter, to derive collections of patterns the analyst actually perceives as good.

*FENSTER obtains good* global *quality results, i. e., the n-tuples encompassed by* all *closed ET-n-sets it extracts are those in the hidden patterns.*

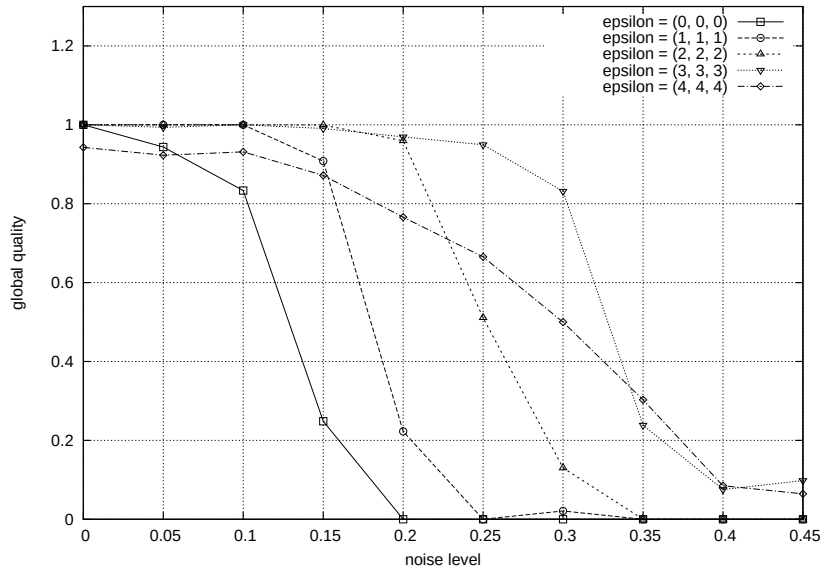



Figure 33: Global qualities of the closed ET-3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset.

The global qualities obtained with ternary relations are plotted in Figure 33. In this setting, the hidden patterns gather eight elements in every attribute domain (of 32 elements) and FENSTER constrains the closed ET-3-sets to have at least four elements per attribute. The best parametrization is $\epsilon = (2, 2, 2)$ for levels of noise below 0.15. At this point, the quality of the extracted collection of closed ET-3-sets is almost perfect, whereas the collection of exact closed patterns shows a quality of 0.25. The noisiest settings are advantageously mined with $\epsilon = (3, 3, 3)$. This confirms that greater noise tolerances are preferred to mine relations suffering from higher levels of noise.

*Closed ET-n-sets in noisier relations benefit from greater tolerances to noise.*

Figures 34 and 35 provide a finer analysis of these results. In Figure 34, $\frac{|\mathcal{E} \setminus \mathcal{R}_{\text{hidden}}|}{|\mathcal{E}|}$, i. e., the proportion of false positive 3-tuples (encompassed by $\mathcal{E}$ and not $\mathcal{R}_{\text{hidden}}$), is plotted, whereas Figure 35, represents $\frac{|\mathcal{R}_{\text{hidden}} \setminus \mathcal{E}|}{|\mathcal{R}_{\text{hidden}}|}$, i. e., the proportion of false negative 3-tuples (encompassed by $\mathcal{R}_{\text{hidden}}$ and not $\mathcal{E}$). It becomes clear why, in this experiment, $\epsilon = (2, 2, 2)$ is always better than weaker tolerances to noise: it lowers the false negative rate while keeping the false positive rate null. More generally, the false positive rate increases with $\epsilon$, whereas the false negative rate decreases. The quality measure benefits from a good trade-off between these two tendencies. Nevertheless, in critical applications, lowering the false negative rate may be more important than maximizing the quality measure. In such cases, it is worth using greater noise tolerance parameters.

Too loose size constraints provide high false positive rates. Indeed, it is easier for a small pattern to include an element, outside the hidden pattern, that is altered by the false positive noise. Reinforcing the size constraints filters out these small patterns, i. e., lowers the false positive rate. To confirm that, the same datasets were mined under minimal size constraints raised to five elements per attribute. Thanks these reinforced minimal size constraints, no closed ET-3-set encompasses 3-tuples absent from $\mathcal{R}_{\text{hidden}}$, i. e., the size constraints have the forecast filtering effect on the collection of patterns. Unfortunately, these

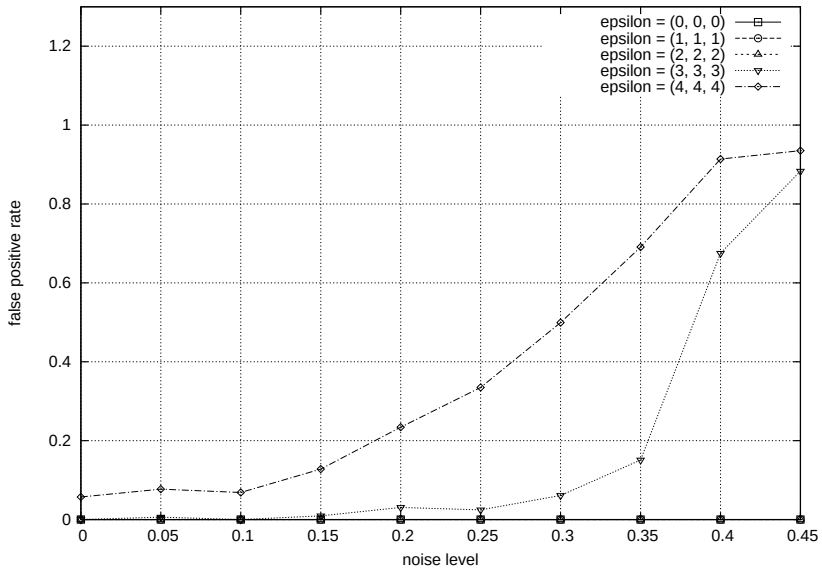



Figure 34: False positive rates of the closed ET-3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset.

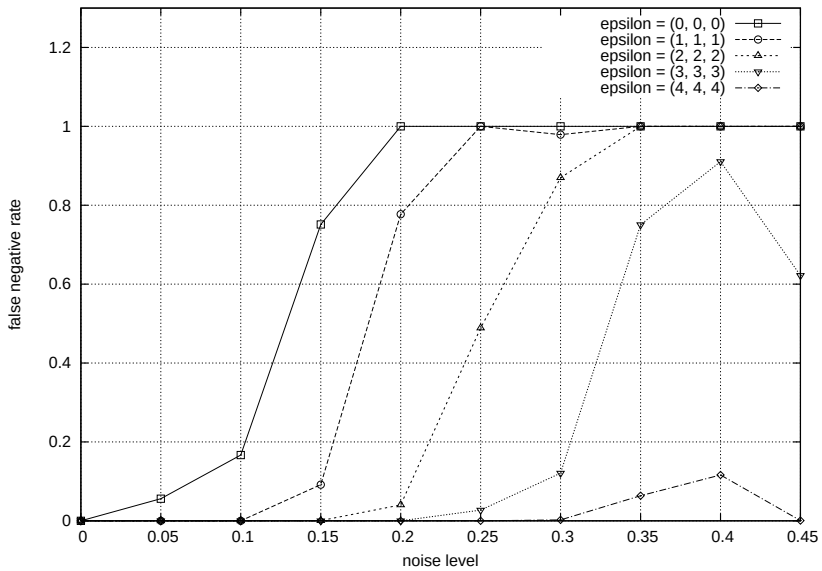



Figure 35: False negative rates of the closed ET-3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset.

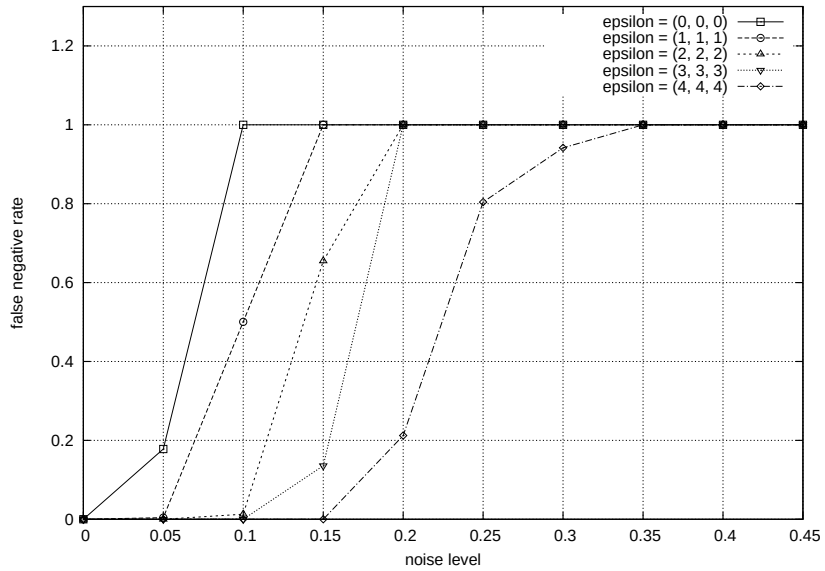



Figure 36: False negative rates of the closed ET-3-sets with at least five elements per attribute in a $32 \times 32 \times 32$ dataset.

constraints not only filter out the closed ET-3-sets that go outside the hidden patterns, but some closed ET-3-sets that remain inside them too. As a consequence, the false negative rates in Figure 36 are always worse than their counterparts of Figure 35 and the resulted global qualities usually are worse too (see Figure 37).

FENSTER was tested on 4-ary relations too. The experimental protocol still follows what was explained in Section 5.1. The hidden patterns gather four elements in every attribute domain (of 16 elements) and the closed ET-3-sets are forced to have at least two elements per attribute. Figure 38 gives the global qualities obtained in this setting. Whatever the level of noise, the collections of closed ET-4-sets obtained with $\epsilon = (1, 1, 1, 1)$ have a better quality than the collections of *exact* closed 4-sets (i. e., $\epsilon = (0, 0, 0, 0)$). With a noise level of 0.25, the quality of the collection of exact closed 4-sets is below 0.3, whereas it reaches 0.65 when $\epsilon = (1, 1, 1, 1)$. Furthermore, because of the loose minimal frequency constraint, $\epsilon = (2, 2, 2, 2)$ is too high.

## 5.3 *Comparison with Competitors*

Since FENSTER is, to the best of our knowledge, the only algorithm able to deal with both error-tolerance and arbitrary $n$-ary relations ($n \geqslant 3$), it is compared to CUBEMINER [41], TRIAS [40] and DATA-PEELER in the particular context of *exact* closed 3-set mining. Thus, the ternary setting, presented in the previous section, was used to compare the time performance of FENSTER with that of these three approaches. The extraction times are depicted in Figure 39. Even though FENSTER has been designed to tolerate noise, it is about one order of magnitude faster than both CUBEMINER and TRIAS. Its enumeration principles, shared with DATA-PEELER, grants these good results. Comparing the performance of DATA-PEELER and FENSTER allows to quantify the overhead brought by the tolerance to noise (in particular the update and

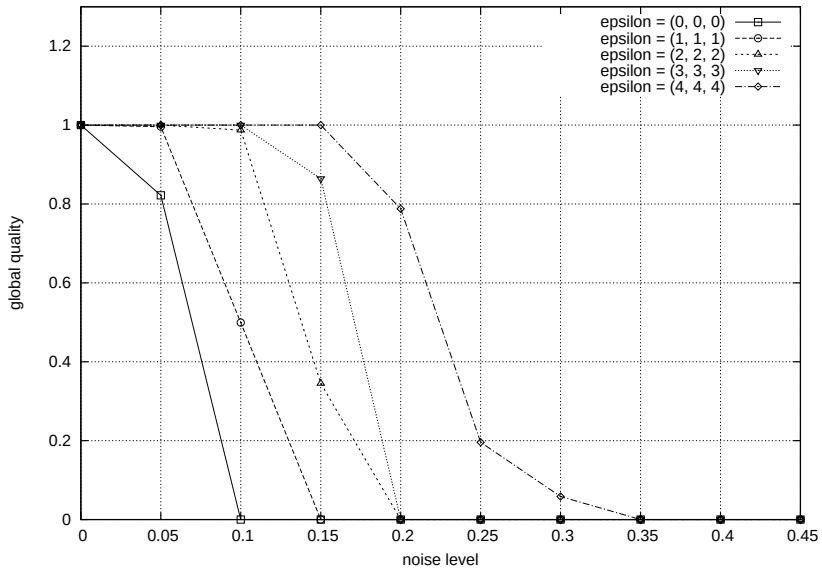



Figure 37: Global qualities of the closed ET-3-sets with at least five elements per attribute in a $32 \times 32 \times 32$ dataset.

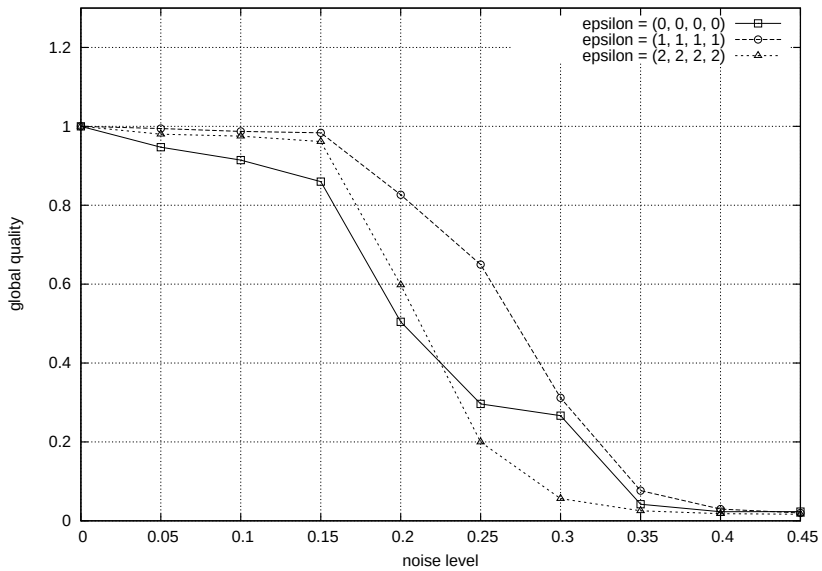



Figure 38: Global qualities of the closed ET-4-sets extracted with at least two elements per attribute in a $16 \times 16 \times 16 \times 16$ dataset.

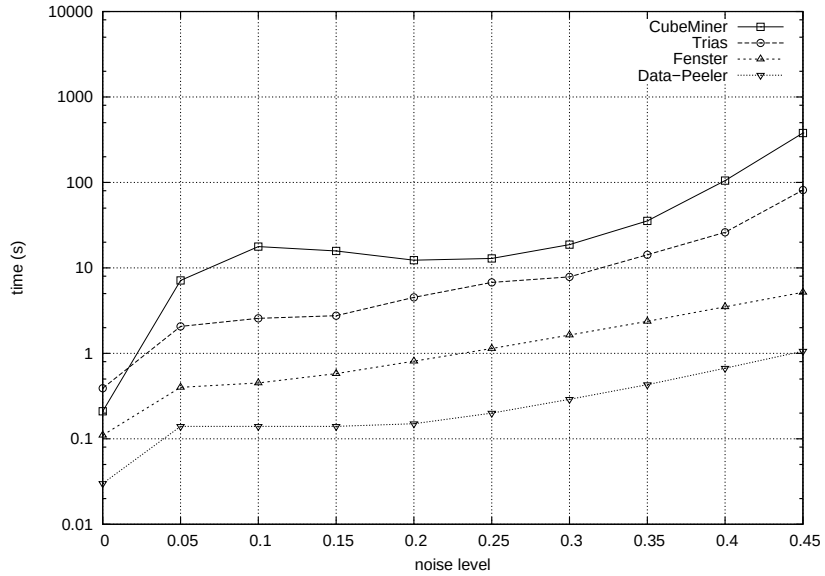



Figure 39: Times to extract the *exact* closed 3-sets with at least four elements per attribute in a $32 \times 32 \times 32$ dataset.

the use of the counters presented in Section 3.1), useless in this exact context. This overhead remains below one order of magnitude.

*With no tolerance to noise, the performance overhead of* FENSTER *w.r.t.* DATA-PEELER *remains below one order of magnitude.*

Several approaches were designed to extract error-tolerant patterns from noisy *binary* relations (see Section 1.2.1). FENSTER was compared to `AC-Close` [21] on this particular (w.r.t. what FENSTER can achieve) task. Two main reasons justified the choice of `AC-Close` as a competitor. Like FENSTER, this algorithm mines *closed* ET-patterns, whereas most of the other approaches do not force the returned itemsets to be closed. Moreover, by constraining the cardinality of the *exact* support of a pattern to exceed $\alpha s$ (where $s$ is a minimal size constraint on the ET-pattern and $\alpha \in [0, 1]$ is a user-defined parameter), `AC-Close` somehow circumvents the performance issues the other approaches go through. Indeed, in their experimental section, the authors claim that `AC-Close` runs much faster than AFI described in [52]. Nevertheless, when using `AC-Close` with $\alpha = 0.75$ and $s = 4$, the task proposed in this section was intractable on a $32 \times 32$ dataset containing four $8 \times 8$ hidden patterns. As a consequence, smaller $16 \times 16$ datasets containing four overlapping $4 \times 4$ patterns were built. The closed patterns extracted by both FENSTER and `AC-Close` are constrained to gather at least two elements per attribute [1]. $\alpha$ was set to 0.5 and various (relative) noise tolerance levels $\epsilon$ were tested. Like with FENSTER, such a level is applied to both attributes. Figures 40 and 41 respectively plot the quality results of FENSTER and `AC-Close`. Because the hidden patterns are small, the best results are obtained with no tolerance to noise. In this setting `AC-Close` and FENSTER compute the same collections of patterns. It is noticeable that a relative tolerance to noise allows more subtle variations of the returned collections. Figure 42 compares the time performance of FENSTER (absent points were measured at 0s and cannot be plot on the chosen logarithmic scale) and `AC-Close`. On this small dataset, FENSTER already runs three orders of magnitude faster than

1. `AC-Close` cannot enforce a minimal size constraint on both attributes. As a consequence the minimal size constraint on one of them is enforced in a post-processing step. Without this post-process, worse quality measures are obtained.

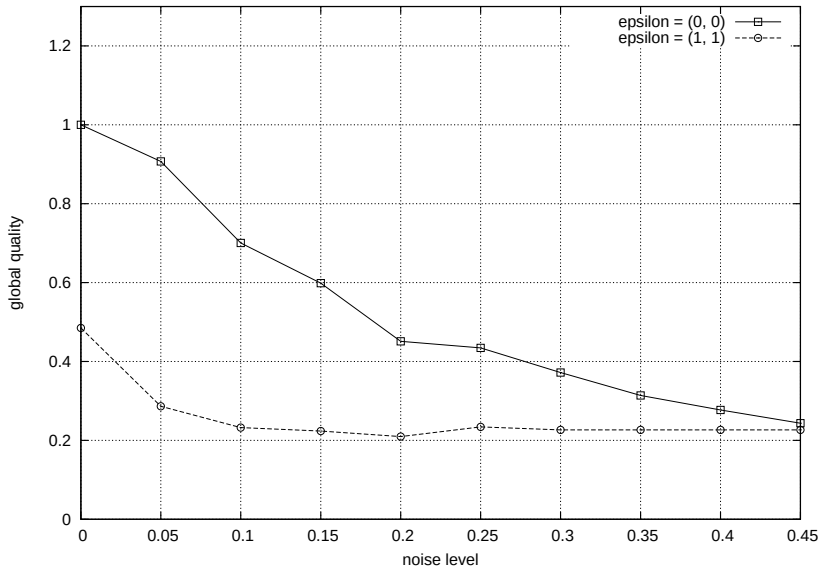



Figure 40: Global quality of the collection of patterns extracted by FENSTER in a $16 \times 16$ dataset.

*Compared to one of the fastest ET-itemset extractors with a relative tolerance to noise,* FENSTER *outperforms it by orders of magnitude.*

`AC-Close`. As mentioned earlier, this difference increases with the size of the dataset (even if the same ratio `size of an attribute domain` / `minimal size constraint` is kept). The choice for an absolute tolerance to noise is here empirically validated: even the fastest approaches with a relative tolerance to noise relatively do not scale-up to medium-size datasets.

## 6 MINING *anomalously* DENSE ET-n-SETS

### 6.1 *Local Pattern*

When extended, a closed ET-n-sets that does not suffer from the introduction of many n-tuples absent from $\mathcal{R}$ may be considered irrelevant. Indeed, it does not respect David J. Hand's definition of a local pattern [39]:

> A local pattern is a data vector serving to describe an *anomalously* high local density of data points.

According to this definition, the closed ET-n-sets that are fragments of larger regions of the same "density" (of n-tuples present in $\mathcal{R}$) are undesirable. To avoid their extraction the closedness constraint must be strengthened, i. e., an ET-n-set should not be considered closed when some extension of it by one element $s$ does not introduce many more n-tuples absent from $\mathcal{R}$, either on $s$ itself or on any orthogonal element. Formally, this change only consists in using different noise tolerances for the two constraints defining a closed ET-n-set (Definitions 26/30 and 27/31). Let us consider $\epsilon = (\epsilon^i)_{i=1..n} \in \mathbb{N}^n$, the noise tolerance used to define the $\epsilon$-connectedness of ET-n-sets, and $\delta = (\delta^i)_{i=1..n} \in \mathbb{Z}^n$, the noise tolerance used to define their $\delta$-closedness. The particular case $\epsilon = \delta$ corresponds to $\mathcal{C}_{\delta\text{-closed}}$ being the "natural" closedness w.r.t. $\mathcal{C}_{\epsilon\text{-closed}}$. It was the setting adopted in this chapter until now. Given $i = 1..n$, setting $\delta^i > \epsilon^i$ strengthens the closedness constraint on the $i^{\text{th}}$ attribute, i. e., filters out the patterns that can be extended by an

*Setting higher noise tolerance parameters for the closedness constraint only is strengthening the closedness.*

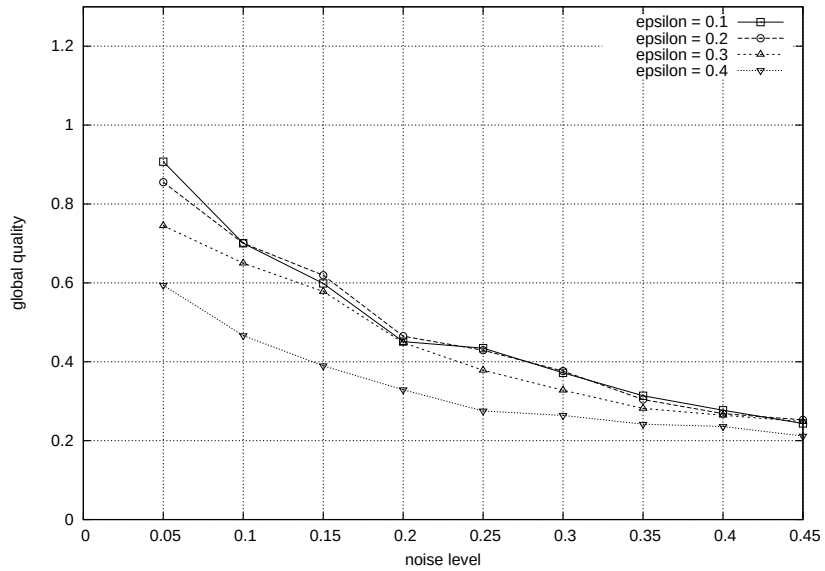



Figure 41: Global quality of the collection of patterns extracted by `AC-Close` in a $16 \times 16$ dataset.

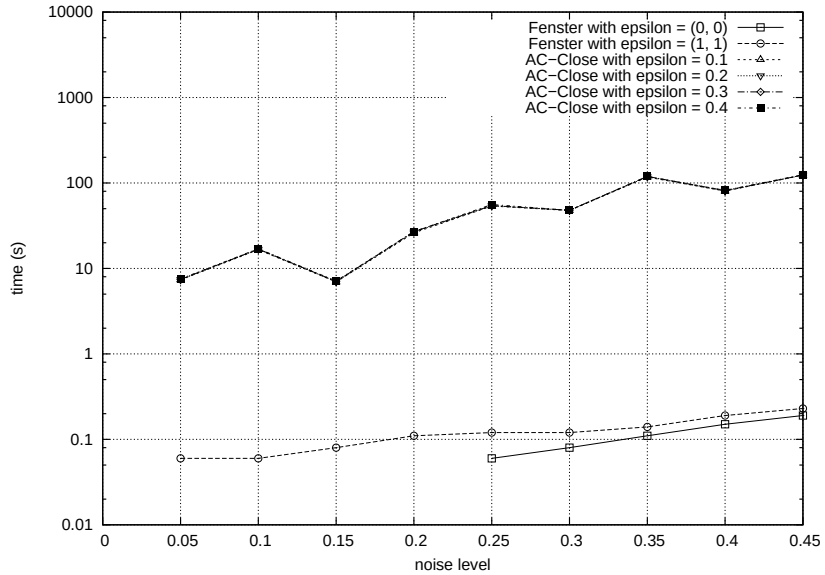



Figure 42: Times to extract the collection of error-tolerant patterns with `AC-Close` or FENSTER in a $16 \times 16$ dataset.

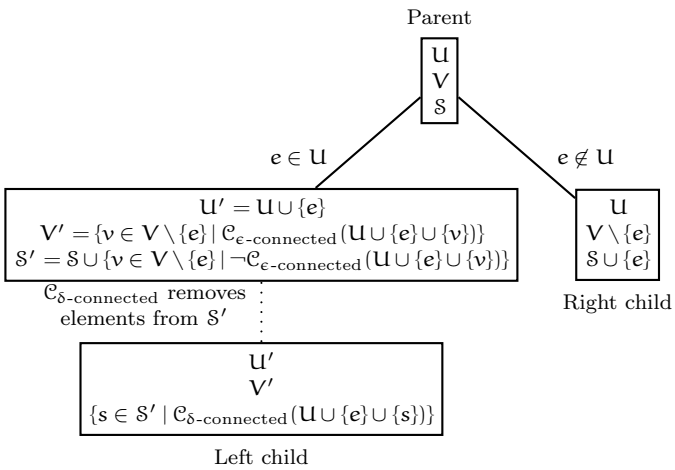



Figure 43: Enumeration of any element $e \in V$.

element (from any attribute domain) without introducing strictly more than $\delta^i$ $n$-tuples absent from $\mathcal{R}$ on any element in the $i^{th}$ dimension of the extended pattern.

**Example 26** *Consider the parametrization* $\epsilon = \delta = (1, 1, 1)$. *It was shown in Example 20* $(\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ *is a closed ET-3-set in* $\mathcal{R}_E$. *Setting* $\delta^3$ *to 2 instead of 1 filters this pattern out. Indeed, when extending it with 4,* $O((\{\alpha, \gamma\}, \{1, 2, 3, 4\}, \{B\}), B) = 2$ *is not strictly greater than* $\delta^3 = 2$.

Choosing $\delta^i < \epsilon^i$ looks useless. However, notice that negative values for every $\delta^i$ $(i = 1..n)$ makes the definition of a $\delta$-closed ET-$n$-set match every $\epsilon$-connected (but not necessarily closed) $n$-set. Therefore, a $\delta$-closed ET-$n$-set generalizes this type of pattern too and FENSTER can extract complete collections of, for instance, frequent ET-$n$-sets (minimal size constraints on some attribute).

## 6.2 *Strong Closedness*

*An element that cannot extend an n-set may still prevent its* strong *closedness.*

Extracting every closed ET-$n$-set with a strong closedness forces modifications in the enumeration of FENSTER and, as a consequence, in the algorithm. More precisely, at a given enumeration node $(U, V)$, an element in $V$ that cannot extend $U$ without violating $\mathcal{C}_{\epsilon\text{-connected}}$ may still prevent the $\delta$-closedness of some ET-$n$-sets $(U, V)$ represents. As a consequence, such an element is, like before, removed from $V$ but, this time, it is inserted in $S$ too. With the same argument, in a left child node where $e$ has just been enumerated, an element $s \in S$ can be removed from $S$ when $\neg\mathcal{C}_{\delta\text{-connected}}(U \cup \{e\} \cup \{s\})$ (notice the use of $\delta$ instead of $\epsilon$). These are the only differences that must be brought to FENSTER to make it able to extract $\delta$-closed ET-$n$-sets defined with $\epsilon \neq \delta$. The modified enumeration is depicted in Figure 43. The related pseudo-code is given in Figure 44. At the implementation level, the counters presented in Section 3.1 still fulfill the task.

**Example 27** *Let* $\epsilon = (1, 1, 1)$ *and* $\delta = (2, 1, 1)$. *Contrary to Example 21 (where* $\epsilon = \delta$*), the elements 3 and 4 from* $V^2$ *are, in the left child, moved to* $S^2$ *instead of being only removed from* $V^2$. *In a second step, element 3 is removed from* $S^2$ *because* $\mathcal{C}_{\delta\text{-connected}}(\{\alpha, \gamma\}, \{3\}, \{B, A\})$ *is false. Indeed* $O((\{\alpha, \gamma\}, \{3\}, \{B, A\}), 3) = 2$ *is strictly greater than* $\delta^2 = 1$. *In the opposite,*

**Input:** $U, V, S$
**Output:** Every $\delta$-closed ET-$n$-set containing every element in $U$, possibly some elements in $V$, and satisfying $\mathcal{C}_{P(A)M}$
**if** $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V)$ **then**
  $U \leftarrow U \sqcup V$
  $V \leftarrow (\emptyset, \ldots, \emptyset)$
**end if**
**if** $\mathcal{C}_{P(A)M}$ may be satisfied by an $n$-set descending from this node $\wedge \mathcal{C}_{\delta\text{-closed}}(U \sqcup V)$ **then**
  **if** $V = (\emptyset, \ldots, \emptyset)$ **then**
    **output**$(U)$
  **else**
    Choose $e \in V$
    $U' \leftarrow U \cup \{e\}$
    $V' \leftarrow \{v \in V \setminus \{e\} \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{v\})\}$
    $S' \leftarrow S \cup \{v \in V \setminus \{e\} \mid \neg\mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{v\})\}$
    FENSTER$(U', V', \{s \in S' \mid \mathcal{C}_{\delta\text{-connected}}(U \cup \{e\} \cup \{s\})\})$
    FENSTER$(U, V \setminus \{e\}, S \cup \{e\})$
  **end if**
**end if**

Figure 44: The generalized FENSTER algorithm.

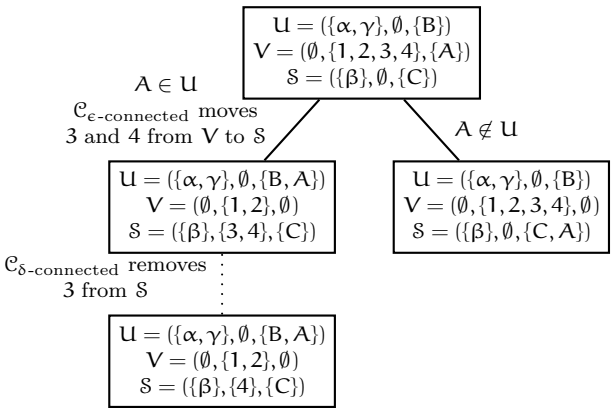



Figure 45: Illustration of Example 27.

*the element 4 is kept in $S^2$. Indeed $\mathcal{C}_{\delta\text{-connected}}(\{\alpha, \gamma\}, \{4\}, \{B, A\})$ is true. Figure 45 illustrates this example.*

## 6.3 *Global Quality Results*

The synthetic $32 \times 32 \times 32$ datasets, presented in Section 5, are reused to test the effect of a parametrization where $\epsilon \neq \delta$. Identical $\epsilon^i$ (resp. $\delta^i$) parameters are chosen for every domain. Figure 46 gives the global qualities obtained with minimal size constraints of four elements per attribute. Strengthening the closedness usually provides worse results than adopting a natural closedness, i. e., $\epsilon = \delta$. The same deterioration of the global quality occurs whatever the arity of the relation. In fact, the good global qualities obtained with $\epsilon = \delta$ are computed from collections of hundreds, or, for higher levels of noise, thousands of closed ET-$n$-sets that are multiple fragments of the four hidden patterns. These fragments are larger than with exact closed $n$-set mining but

*With a strong closedness, the quality of the closed ET-$n$-sets is worse.*

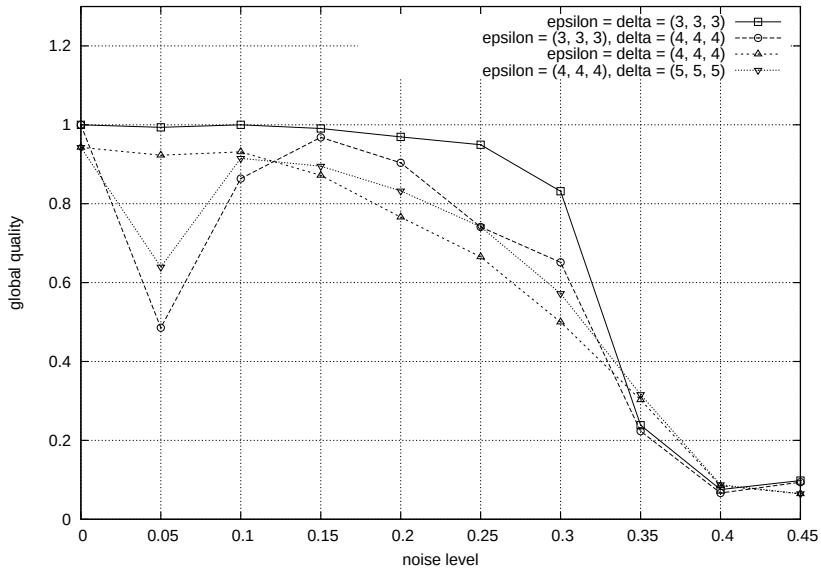



Figure 46: Global quality of the $\delta$-closed ET-3-sets extracted with at least four elements per attribute in a $32 \times 32 \times 32$ dataset.

they remain fragments, i. e., an $\epsilon$-closed ET-$n$-set is extensible, with one of the missing element from the hidden pattern, without introducing many $n$-tuples absent from the relation. In other terms, it does not verify a stronger closedness constraint.

With a strong closedness, a local minimum for the global quality is always observed at a low level of noise (around 0.05 in our experiments). Here is the reason for that: at low levels of noise, the tested noise tolerances $\epsilon$ allow, for every hidden pattern, the extraction of large fragments of it (but not the entire hidden pattern) that almost include the missing elements to form the hidden pattern (because the noise level is low). As a consequence, they are often filtered out by a stronger closedness constraint. For example, at a noise level of 0.05 and with $\epsilon = (4, 4, 4)$, the largest closed ET-3-sets are the $8 \times 8 \times 8$ hidden patterns with two elements missing out of the 24. With a greater tolerance to noise, i. e.greater $\epsilon^i$ ($i = 1..n$) parameters, the entire hidden patterns would be extracted and strengthening the closedness would, indeed, implement the notion of local pattern as defined by David J. Hand. Unfortunately, greater $\epsilon^i$ ($i = 1..n$) parameters mean longer extractions and compensating the whole false negative noise usually turns out to be intractable. Figure 47 gives the times it takes FENSTER to extract the closed ET-3-sets whose global qualities were plotted in Figure 33. A jump of almost one order of magnitude accompanies every increase of the $\epsilon^i$ parameters.

*Tolerating more noise costs much time.* FENSTER *can extract larger fragments of the hidden pattens; not* the *hidden patterns.*

Real-life datasets where the whole false negative noise can be compensated by FENSTER, while remaining tractable, are, at least, very rare. When this cannot be ensured, setting $\epsilon = \delta$ is safer. The tractability issue is inherent to the computational task FENSTER tackles. The closed itemset enumeration problem in binary relations is NP-hard (see, e. g., [94]). Generalizing the definition of a closed pattern to make it noise-tolerant widens the traversed search space because it has a severe impact on pruning. Consider exact closed $n$-set mining. When an $n$-tuple is, for sure, part of a candidate $n$-set but absent from the relation, neither this $n$-set nor the larger $n$-sets encompassing it is a

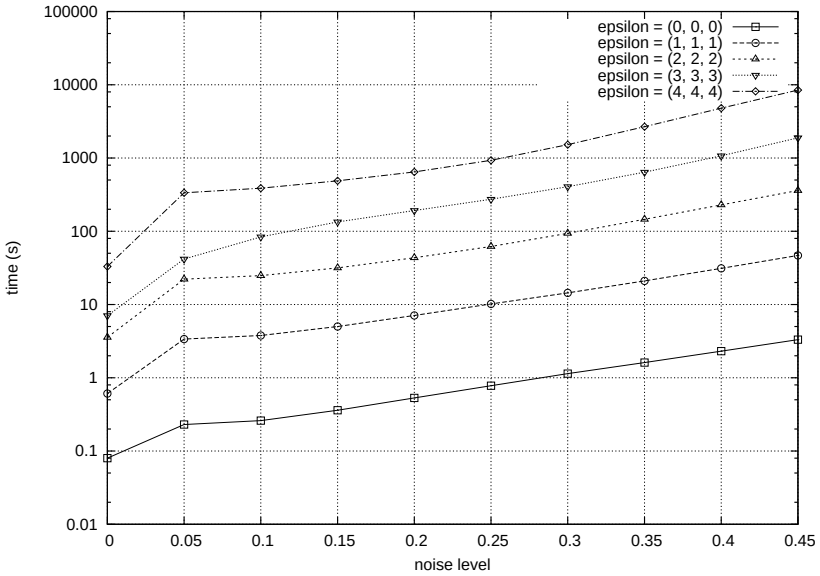



Figure 47: Times to extract every closed ET-3-sets with at least five elements per attribute in a $32 \times 32 \times 32$ dataset.

closed n-set and the related search space can be safely pruned. In the same situation, the search for noise-tolerant patterns cannot be aborted: either the current candidate n-set or a larger one[2] is a closed ET-n-set. To discover it, the search space must be traversed further.

## 7 CONCLUSION

FENSTER generalizes DATA-PEELER to make it tolerate false negative noise. Thus, n-tuples can be both absent from the relation and encompassed by closed ET-n-sets. Like DATA-PEELER, FENSTER performs complete extractions under piecewise (anti)-monotone constraints. How much noise is tolerated is parametrized by as many integers as there are attributes in the relation. In every closed ET-n-set, these integers are upper-bounds of the number of n-tuples involving an element from the related attribute domain, encompassed by the closed ET-n-set but absent from the relation. Although the generalization towards noise tolerance looks trivial from an abstract point of view, the implementation requires, to be efficient, to incrementally compute counters of absent n-tuples in many subspaces of the relation. As far as not too much noise is tolerated, FENSTER remains tractable on relatively large relations. The gain, in term of global quality (coverage of the hidden patterns and only them), is obvious. Nevertheless FENSTER usually cannot, in a reasonable time, tolerate as much noise as contained in the relation. As a consequence, the returned closed ET-n-sets are fragments (though larger fragments than without noise tolerance) of hidden patterns. Because of that, a strengthened closedness constraint does more harms than it helps in finding patterns that are isolated from the others.

2. Even *several* larger n-sets may be closed since the closure is not always unique (see Section 2).

# 5 AGGLOMERATING LOCAL PATTERNS HIERARCHICALLY WITH ALPHA

## 1 AGGLOMERATING CLOSED ET-n-SETS

### 1.1 *A Pattern Clustering Scheme*

Let us recall the objectives and difficulties raised in Section 6 of the previous chapter. The relevant patterns the analyst is in quest for should comply with David Hand's definition of a *local pattern* [39]:

> A local pattern is a data vector serving to describe an anomalously high local density of data points.

Furthermore, because of time complexity issues, a complete extractor such as FENSTER usually cannot tolerate as much errors as necessary to directly list the hidden patterns. Instead, it returns multiple fragments of these patterns. In such situations, the analyst is either forced to (a) interpret complete lists of insufficiently error-tolerant (hence much-overlapping) closed $n$-sets, or (b) revise his/her completeness demand downwards.

Anybody who has tried to *manually* interpret long lists of poorly relevant local patterns knows how counter-productive it can be, and this is definitively not an option. Another possibility can be to mine the data by means of incomplete (i. e., missing some solution patterns w.r.t. the specified constraints) but tractable approaches. Nevertheless, it is also possible to compute, under constraints, all closed and (as much as possible) noise-tolerant patterns before a post-processing phase that aims at deriving another collection that would be shorter and that would contain more relevant patterns. The latter approach appears appealing because the lossy heuristics are delayed as far as possible in the knowledge discovery process, hence trustier than a purely heuristic one. It is about inserting an *automatic* intermediary task between the complete extraction of patterns and the needed interpretation by analysts, postponed hence made easier.

Even though the fragments extracted by FENSTER are, individually, far from the hidden patterns, the returned collections *globally* have good qualities (see Section 5.2). This means that the agglomeration of the patterns matches the relation deprived of noise. That is why a pattern agglomeration task was investigated as a way to heuristically reconstruct the hidden patterns from the fragments listed in a complete manner. This task can be compared with solving an $n$-dimensional jigsaw puzzle: every piece is a pattern returned during the complete extraction phase and the image to produce is a perfect version of the one given on the box (the dataset), which is, contrary to classical jigsaw puzzle, altered by some noise. The perfect image must be composed of large (possibly) overlapping hyper-rectangles (modulo any permutation of the hyper-plans of any dimension) of '1' values "embedded" in a '0' valued hyper-space. It looks like a tough game and an automatic clustering can help.

*Following the philosophy "completeness as far as tractable", ALPHA post-processes FENSTER to tolerate more noise.*

Unlike classical clustering approaches, our goal is not to partition the original dataset but a set of small local patterns so that every cluster

represents a larger local pattern. Global (i. e., at every iteration, a global clustering refines the previous one) and divisive (the complete collection of local patterns, considered as a whole, is successively divided into smaller clusters) clustering approaches are not suitable because of the difference in nature between the elements to cluster (the complete collection of local patterns that tolerate a few errors) and the resulting clusters (large local patterns tolerating much noise). Using again the analogy with a jigsaw puzzle, the quality of the constructed image should not be reduced to how well every piece interlocks with its neighbors. This image should also match the noisy one on the box. To take into account this objective, the global and divisive approaches are, by nature, not much suitable. On the contrary, a bottom-up agglomeration successively merges previously established clusters (the previously assembled pieces) into larger ones, hence allowing to test the partial results against the original dataset (the noisy image on the box). Therefore, we need for a hierarchical agglomerative clustering scheme. The fact it does not require to fix, a priori, the number of clusters is also extremely useful. Indeed, the number of relevant local patterns to discover usually is unknown.

*ALPHA hierarchically agglomerates the patterns.*

It should be noticed that, even though the hidden patterns can overlap, there is no need for a fuzzy clustering method. Indeed, the closedness constraint on every ET-n-set extracted from the noisy relation makes it usually be a fragment of one hidden pattern only. If this is not the case, such a pattern can be associated to any of the overlapping pattern without much consequence since the n-dimensional space it occupies, at the intersection of several hidden patterns, must be covered by many other slightly different closed ET-n-sets.

## 1.2 *Hierarchical Agglomeration*

Like any hierarchical agglomeration scheme, ALPHA requires an agglomeration operator and a metric. The extracted closed ET-n-sets being fragments of the searched patterns, the merging operator essentially is a union. It is $\sqcup$ (see Definition 22). Defining a metric on n-sets is not trivial. It should not only depend on the n-sets but also on the relation they were extracted from. In this way, before agglomerating two n-sets, the information about the subspaces of the relation outside them, but inside the minimal envelope enclosing them, can be taken into account. The importance of using the initial relation in the agglomeration process is illustrated by the two relations in Figure 48. When mining closed 2-sets having at least three elements per attribute (bold '**1**' in the tables), the two settings are identical if restricted to the elements covered by these patterns. Nevertheless the closed 2-sets of Figure 48a are obviously better candidates for an agglomeration than those of Figure 48b. In fact, the reasoning (see, e. g., [52]) to define how error-tolerant a local pattern is can be applied to the n-set whose outline is the envelope of the two patterns. Thus, such a definition must be based on the quantity of n-tuples absent from the relation in the worst hyper-plan of this n-set. Contrary to FENSTER whose time performance could not be reached with a relative tolerance to noise, ALPHA uses a metric based on *proportions* of n-tuples absent from $\mathcal{R}$.

*The agglomeration of patterns is their minimal envelope.*

*The distance between patterns takes into account the relation they locally describe.*

| | | | | | |
|---|---|---|---|---|---|
| **1** | **1** | **1** | 0 | 1 | 1 |
| **1** | **1** | **1** | 1 | 0 | 1 |
| **1** | **1** | **1** | 1 | 1 | 0 |
| 0 | 1 | 1 | **1** | **1** | **1** |
| 1 | 0 | 1 | **1** | **1** | **1** |
| 1 | 1 | 0 | **1** | **1** | **1** |

(a)

| | | | | | |
|---|---|---|---|---|---|
| **1** | **1** | **1** | 0 | 0 | 0 |
| **1** | **1** | **1** | 0 | 0 | 0 |
| **1** | **1** | **1** | 0 | 0 | 0 |
| 0 | 0 | 0 | **1** | **1** | **1** |
| 0 | 0 | 0 | **1** | **1** | **1** |
| 0 | 0 | 0 | **1** | **1** | **1** |

(b)

Figure 48: Two toy binary relations

**Definition 32 (Unweighted metric)** *Given an $n$-set $X = (X^1, \ldots, X^n)$, $d(X)$ denotes its (unweighted) intrinsic distance measure and is computed as follows:*

$$d(X) = \max_{i=1}^{n} \left( \max_{x \in X^i} \left( \frac{|K \setminus \mathcal{R}|}{|K|} \right) \right) ,$$

*where $K = X^1 \times \cdots \times X^{i-1} \times \{x\} \times X^{i+1} \times \cdots \times X^n$.*

**Example 28** *It was shown in Example 20 $X = (\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ is a closed ET-3-set (with $\epsilon = (1, 1, 1)$) in $\mathcal{R}_E$ (represented in Table 5). This pattern would be in the initial collection* ALPHA *agglomerates. Its intrinsic distance measure is $\frac{1}{2}$. Indeed, its worst hyper-plan, in terms of proportion of 3-tuples absent from $\mathcal{R}_E$, is the one related to the element 3, i. e., $\{\alpha, \gamma\} \times \{3\} \times \{B\}$. It contains two 3-tuples, $(\alpha, 3, B)$ and $(\gamma, 3, B)$, among which one, $(\gamma, 3, B)$, is absent from $\mathcal{R}_E$.*

The intrinsic distance is easily generalized to specify to what extent noise is tolerated in every element. In some specific contexts (e. g., when the noise distribution is known and not uniform), such a parametrization may be useful:

*Background knowledge about the noise distribution can parametrized the metric.*

**Definition 33 (Weighted metric)** *Given an $n$-set $X = (X^1, \ldots, X^n)$ and a weight function $w : \cup_{i=1}^{n} D^i \to \mathbb{R}^+$ (a greater weight meaning less tolerance to noise for the related element), $d(X)$ denotes its weighted intrinsic distance measure and is computed as follows:*

$$d(X) = \max_{i=1}^{n} \left( \max_{x \in X^i} \left( w(x) \frac{|K \setminus \mathcal{R}|}{|K|} \right) \right) ,$$

*where $K = X^1 \times \cdots \times X^{i-1} \times \{x\} \times X^{i+1} \times \cdots \times X^n$.*

The distance between two $n$-sets is the (weighted or unweighted) intrinsic distance of their agglomeration.

*The distance between two patterns is the proportion of absent $n$-tuples in the worst hyper-plan of their agglomeration.*

**Definition 34 (Distance between $n$-sets)** *Given two $n$-sets $X$ and $Y$, the distance between them is $d(X \sqcup Y)$.*

**Example 29** ALPHA *initially computes the distance between every pair of closed ET-$n$-sets extracted with* FENSTER. *Going on with Example 28,* ALPHA *computes, in particular, the distance between $X = (\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ and another closed ET-3-set, $Y = (\{\gamma\}, \{1, 2, 3, 4\}, \{A, B\})$. Their agglomeration would provide $X \sqcup Y = (\{\alpha, \gamma\}, \{1, 2, 3, 4\}, \{A, B\})$. Its worst hyper-plan, in terms of proportion of 3-tuples absent from $\mathcal{R}_E$, is the one related to the element 3, i. e., $\{\alpha, \gamma\} \times \{3\} \times \{A, B\}$. It contains four 3-tuples. Among them, two, $(\alpha, 3, A)$ and $(\gamma, 3, B)$ are absent of $\mathcal{R}_E$. Thus, by definition, its intrinsic distance is $\frac{2}{4}$ and so is the (unweighted) distance between $X$ and $Y$.*

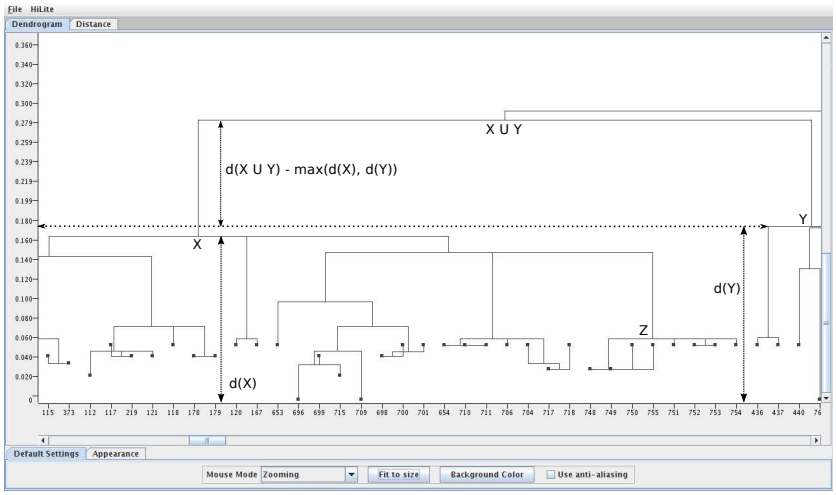



Figure 49: KNIME dendrogram representing the hierarchical agglomeration.

Given the agglomeration operator and the metric, the hierarchical agglomeration performed by ALPHA is now well defined. The constructed clusters, i. e., the agglomerated $n$-sets, can be organized into a binary tree called a dendrogram. For a visual interpretation of the result, the height of a node, representing a cluster, is advantageously set to its intrinsic distance measure. Figure 49 depicts a part of such a tree. Notice that most of the leaves of the dendrogram have an intrinsic distance that is not 0. Indeed, following the idea "completeness as far as tractable", this dendrogram depicts the hierarchical agglomeration of closed ET-$n$-sets defined with the greatest possible tolerance to noise and extracted with FENSTER. Thus, $n$-tuples covered by some closed ET-$n$-sets may be absent from the mined relation and their intrinsic distance is strictly positive. Notice also that the intrinsic distance is not always increasing along the clustering, i. e., an agglomerated $n$-set may have an intrinsic distance that is less than those of the two $n$-sets that were agglomerated. Indeed, this distance is a proportion of $n$-tuples that is not necessarily increasing when the $n$-set is enlarged.

## 2 RETURNING THE FEW RELEVANT PATTERNS

The computed dendrogram contains more local patterns than the initial complete collection of patterns that are not enough noise-tolerant [1]. Nevertheless, some of them now are relevant because they tolerate enough noise. To support the search for these relevant patterns, ALPHA ranks them and automatically selects only the best, being confident that this process preserves every relevant cluster.

### 2.1 *Cluster Relevancy Measure*

David J. Hand's definition of a local pattern (see Section 1.1) is a guideline to assess the relevancy of the clusters. In the context of an $n$-ary relation $\mathcal{R}$, a good cluster $X$ describes an "anomalously high local density" of $n$-tuples present in $\mathcal{R}$ when, *simultaneously*,:

1. $2N - 1$ clusters for $N$ closed ET-$n$-sets.

– it is apart from the rest of the data ("anomalously"), i.e., it maximizes its distance with the other clusters in the tree (but its ancestors and descendants);
– it minimizes the proportion of $n$-tuples absent from $\mathcal{R}$ on its worst hyper-plan ("high local density").

Both information can be easily quantified from the constructed dendrogram:

– the minimal distance between a parent cluster and its two children $X$ and $Y$, i.e. $d(X \sqcup Y) - \max(d(X), d(Y))$, is how distant $X$ and $Y$ are from each other and, even more, from the other clusters. Indeed, these two clusters were agglomerated because they were the closest at that time of the clustering;
– the intrinsic distance measure of $X$, i.e. $d(X)$, is the proportion of $n$-tuples absent from $\mathcal{R}$ on its worst hyper-plan.

Both quantities being proportions of $n$-tuples absent from $\mathcal{R}$, the relevancy of $X$ can now be computed by difference.

*The definition of a local pattern translates to a relevancy measure.*

**Definition 35 (Relevancy of an $n$-set)** *Given an $n$-set $X$ and its parent $X \sqcup Y$ in the binary tree obtained by hierarchical agglomeration, $r(X)$ denotes the relevancy of $X$ and is computed as follows:*

$$r(X) = d(X \sqcup Y) - \max(d(X), d(Y)) - d(X) \ .$$

Figure 49 depicts this computation.

**Example 30** *Going on with Example 29, assume $X = (\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ and $Y = (\{\gamma\}, \{1, 2, 3, 4\}, \{A, B\})$ were actually agglomerated, i. e., at a certain iteration of the hierarchical clustering, their distance $d(X \sqcup Y) = \frac{2}{4}$ was the smallest among all pairs of (previously agglomerated or not) 3-sets. The relevancy of $X$ is $\frac{2}{4} - \max(\frac{1}{2}, \frac{1}{2}) - \frac{1}{2} = \frac{-1}{2}$. This negative value for $r(X)$ means $X$ is irrelevant. Indeed, $X \sqcup Y$ having the same intrinsic distance as $X$, $X$ is a fragment of larger pattern ($X \sqcup Y$ or maybe a larger one). Instead of studying $X$, the analyst had better take a look at this larger pattern (that should receive a higher relevancy value).*

## 2.2 *Selecting the Relevant Clusters*

Ranking the clusters from the dendrogram w.r.t. to their relevancy values allows the analyst to start the interpretation with the most promising ones. However, the list of patterns he/she has to interpret is very long and its tail contains poorly relevant clusters. For example, it contains the initially extracted collection of small patterns (leaves of dendrogram), which usually do not tolerate enough noise. We explain how ALPHA automatically selects a small collection while keeping every relevant pattern. It assumes that all the initially extracted closed ET-$n$-sets are fragments of some relevant local pattern, i. e., that this complete extraction has been performed under constraints, like minimal size constraints, that prevent the false positive noise from being caught. This ensures that every closed ET-$n$-set satisfying them is a subset of a relevant local pattern the analyst is interested in. Thus, ALPHA reads, by decreasing relevancy order, the list of clusters. It outputs the cluster it reads and removes from the list its sub-patterns (i. e., the patterns beneath it in dendrogram). Once every initially extracted closed ET-$n$-set (leaf of the dendrogram) was removed (i. e., was covered by at least one previously read cluster), the procedure stops. In this way, the

ALPHA *selects relevant patterns that cover the initial complete collection of closed ET-$n$-sets.*

completeness of the first extraction is, somehow, preserved. Indeed, every pattern of this initial collection is part of at least one output cluster.

The selection procedure, which has just been exposed, presents this property: "a cluster with a lower relevancy than at least one of its ancestors is not to be kept". This makes sense because it must be a fragment of such a larger ancestor cluster. Interestingly, like in hierarchical tiling [32], it remains possible that both a large cluster and one of its sub-clusters are considered relevant. Whenever it happens the latter has a greater relevancy than the former, i. e., it describes an "anomalously high local density" of present $n$-tuples inside another anomalously high, but lower, local density of present $n$-tuples. Another interesting point relates to the assumption stated in the previous paragraph: "all the initially extracted closed ET-$n$-sets are fragments of some relevant local pattern". If the initial complete extraction is performed under too loose constraints, parts of some closed ET-$n$-sets cover regions of the dataset that actually are out of any relevant local pattern (but contain some $n$-tuples present in the relation because of noise). In other terms the assumption is not satisfied. However, this does not matter much. Indeed, such a closed ET-$n$-set $X$, that covers positive noise, receives a high relevancy value because:

- its distance to the closest cluster $Y$ is high (the worst hyper-plan of $X \sqcup Y$ is out of any relevant pattern, hence it contains many tuples absent from the relation);
- its intrinsic distance measure is low (the noise it contains was sufficiently low not to prevent its extraction by a complete algorithm).

As a consequence, $X$ is high in the list of ranked clusters and is browsed before any of its "super-patterns", which will probably not be selected (their intrinsic distances being very high). In this way, the list of kept clusters is not seriously lengthened. Furthermore, $X$ being small (it is the only pattern in the cluster) it can easily be identified by the analyst or automatically filtered out by size constraints in a final post-processing step.

## 3 EMPIRICAL STUDY

### 3.1 *Quality Measures*

Since the global quality, introduced in the previous chapter, does not reflect the quality as perceived by the analyst, it is finally time, to replace it with another measure, or, more precisely, with two complementary measures. Given a small collection of hidden local patterns (the local patterns in the relation deprived of noise) and another collection of extracted local patterns, these measures rate how useful the latter is for discovering the former. The first measure is the size of the output collection of patterns: if it is too large, interpreting it may be too costly. The second measure is the average similarity between every hidden pattern and its most similar counterpart in the extracted collection. This measure is named *best-ones quality*. It is mathematically expressed as follows.

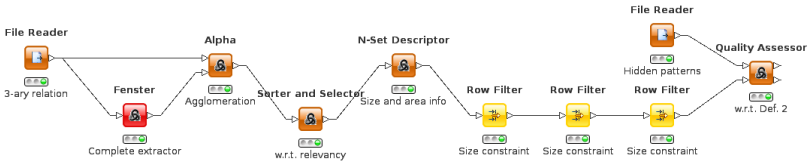



Figure 50: KNIME workflow when experimenting ALPHA on the synthetic ternary relations.

**Definition 36 (Best-ones quality)** *Given $\mathcal{H}$ a set of hidden patterns, $\mathcal{P}$ a set of extracted patterns and $s : \mathcal{H} \times \mathcal{P} \to [0, 1]$ a similarity measure, the best-ones quality of $\mathcal{P}$, denoted $q(\mathcal{P}, \mathcal{H})$ and ranging in $[0, 1]$, is:*

$$q(\mathcal{P}, \mathcal{H}) = \frac{1}{|\mathcal{H}|} \sum_{H \in \mathcal{H}} \left( \max_{P \in \mathcal{P}} (s(H, P)) \right) .$$

To define a similarity measure $s$ between two $n$-sets, several sensible options exist. We chose the average of a classical distance between the sets of elements in every attribute domain.

**Definition 37 (Similarity between $n$-sets)** *Given $X = (X^1, \ldots, X^n)$ and $Y = (Y^1, \ldots, Y^n)$ two $n$-sets, $s(X, Y)$ denotes the similarity between $X$ and $Y$ and is computed as follows:*

$$s(X, Y) = \frac{1}{n} \sum_{i=1}^{n} \frac{|X^i \cap Y^i|}{|X^i \cup Y^i|} .$$

### 3.1.1 *Experimental Protocol*

This empirical study follows that of the $32 \times 32 \times 32$ datasets presented in Section 5 of Chapter 4. The experimental protocol is the same except that, at every noise level, ten relations are synthesized and the results presented in this section always are averages on these ten relations. In this way, they are statistically more relevant. We recall that, in every relation, four hidden 3-sets, with eight elements per attribute, are randomly placed (they may overlap). The noise distribution is uniform. Given one of the relations, an analyst wants to find the four hidden patterns it contains and, if possible, only them. However he/she does not know, a priori, their exact geometries. To avoid missing some of them, he/she constrains the output patterns to gather at least five elements per attribute.

Figure 50 presents the experimental protocol in the form of a KNIME workflow. `Fenster` extracts, under constraints, every closed ET-3-set in the noisy relation read by `File Reader`. `Alpha` agglomerates them (see Section 1.2). `Sorter and Selector` ranks and selects the most relevant clusters (see Section 2). `N-Set Descriptor` appends size information to the output so that the next three `Row Filters` keep only those having at least five elements per attribute. Finally, `Quality Assessor` computes the best-ones quality measure (see Definition 36) of the remaining local patterns by comparison with the hidden patterns, i. e., those in the initial relation deprived of noise.

The experiments have been performed with a computer running the GNU/Linux™ operating system on an AMD Sempron™ 2600+ processor with 1.25 GBytes of RAM. ALPHA was developed within KNIME [19] using Java 6.

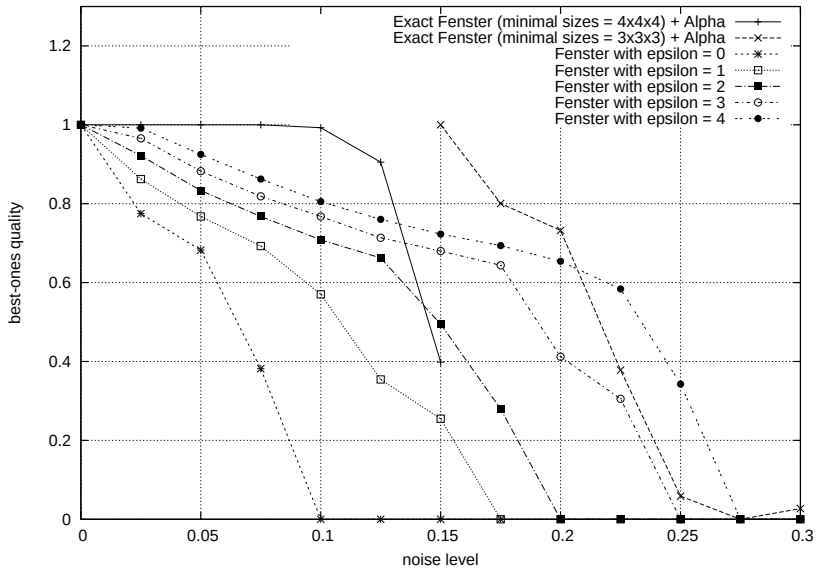



Figure 51: Best-ones qualities of the collections output by FENSTER only and by FENSTER + ALPHA.

## 3.2 *Assessing the Agglomeration*

Extracting, with FENSTER, the closed ET-3-sets having at least five elements per attribute and directly interpreting them is, at least, tedious. Figures 51, and 52 compare the results of FENSTER alone to those of FENSTER + ALPHA (following the experimental protocol detailed in the previous section). When run alone, FENSTER is tested with five different tolerances to noise (the $\epsilon^i$ parameters of Definition 26 and 27). When used as a pattern collection provider to ALPHA, FENSTER is parametrized to extract *exact* closed 3-sets, i.e., $(\epsilon^1, \epsilon^2, \epsilon^3) = (0, 0, 0)$. The principle "completeness as far as tractable", stated in Section 1.1, is not respected here to support a clear assessment of ALPHA's added-value. Indeed, this intermediary task is, in this way, forced to deal with all the noise in the relations. Even in this disadvantageous setting, FENSTER + ALPHA significantly outperforms FENSTER alone. As expected, the best-ones quality of the closed ET-3-sets is very poor if little noise is tolerated. Nevertheless, even with the greatest tested tolerance to noise, which implies very long extraction times (see Figure 47), the quality of the collections computed by FENSTER clearly is below that of ALPHA + FENSTER, which is almost 1 until a noise level of 0.15 (see Figure 51). Furthermore, if there is little noise in the relations, the number of closed ET-3-sets explodes when FENSTER tolerates more noise. For example, with a noise level of 0.075 and an error-tolerance $(\epsilon^1, \epsilon^2, \epsilon^3)$ of $(4, 4, 4)$, FENSTER returns after about 23 minutes of computation, in average, 14475 closed ET-3-sets, against 4.8 patterns for FENSTER + ALPHA (see Figure 52). Among these 4.8 patterns, 4 are those the analyst is in quest for, i. e., $q(\mathcal{P}, \mathcal{H}) = 1$.

*The quality of the individual patterns significantly increases with* ALPHA *w.r.t.* FENSTER *alone. Their number significantly decreases.*

The minimal size constraints on the closed 3-sets that ALPHA processes have not been discussed yet. These constraints are meant to provide enough *fragments* to enable the construction of the relevant patterns by agglomeration. Thus, they are chosen by merely looking at the number of completely extracted closed 3-sets. At least four elements per attribute is chosen when the level of noise is strictly below 0.15.

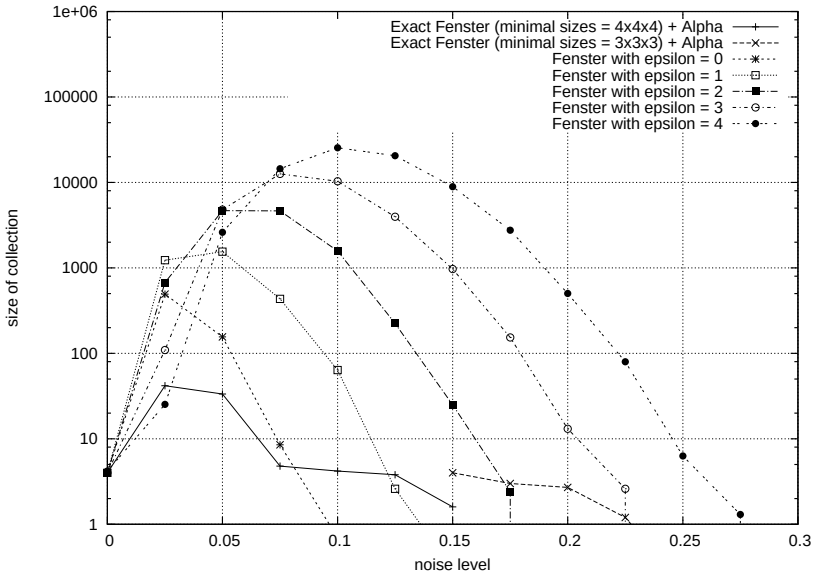



Figure 52: Sizes of the collections output by FENSTER only and by FENSTER + ALPHA.

At this point, keeping the same constraints provides, in average, 16.1 closed 3-sets (see Figure 52). This is obviously too few to construct, by agglomeration, the hidden patterns. That is why, in the relations with levels of noise above 0.15, the closed 3-sets are only constrained to gather at least three elements per attribute. With these looser size constraints, FENSTER extracts more (much overlapping) closed 3-sets. Once agglomerated, the hidden patterns in the relations with a 0.15 level of noise are always perfectly found, i. e., $q(\mathcal{P}, \mathcal{H}) = 1$. Furthermore the selection step (see Section 2) retains only these four patterns (see Figure 52).

### 3.3 *Assessing the Selection*

Stated in the terms of Section 3.1, the selection step aims at decreasing the size of the output collections while keeping the best-ones quality (see Definition 36) as high as possible. To empirically test it, the `Sorter and Selector` node is dropped from the KNIME workflow depicted in Figure 50. The results, plotted in Figure 53 and 54, are compared to those obtained *with* the selection. Until a noise level of 0.15, the size of the selected collection remains below 50, i. e., is between one and two orders of magnitude smaller than its superset obtained without the selection. At the same time, the best-ones quality almost remains identical. It can be written that the effect of the selection is, in those settings, very positive. With noise levels beyond 0.15, the hierarchical agglomeration constructs patterns that are very similar to the hidden ones. Unfortunately, the relevancy measure (see Definition 35) gives higher scores to the sub-patterns of the ones the analyst would like to be presented. That is why less than four patterns are, at the same time, selected and with at least five elements per attribute. With such very noisy relations, the selection step does not help and the analyst had better take a look at the long list of clusters. Indeed, the relevant ones are almost perfectly constructed up to a noise level of 0.25.

*In very noisy relations, the hidden patterns are found by agglomeration but the relevancy measure does not score them as well as their sub-patterns.*

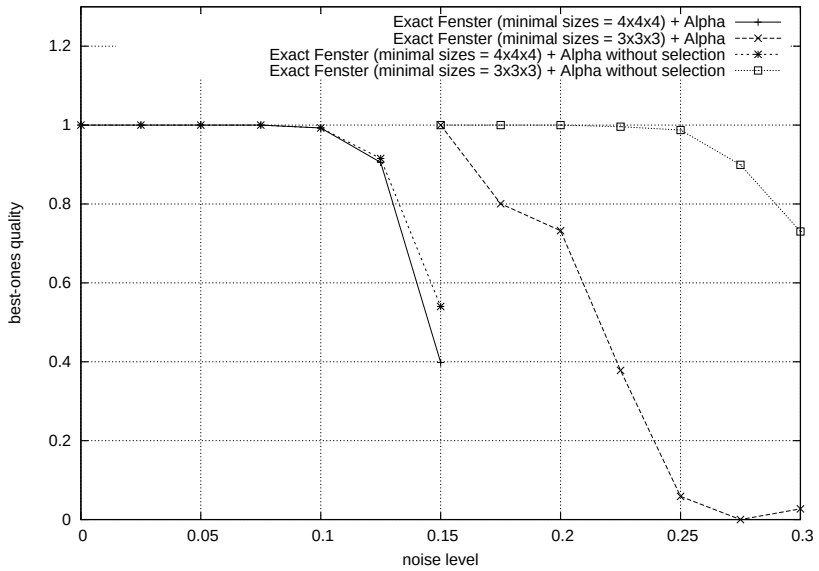



Figure 53: Best-ones qualities of the output collections with and without the selection of the relevant patterns.

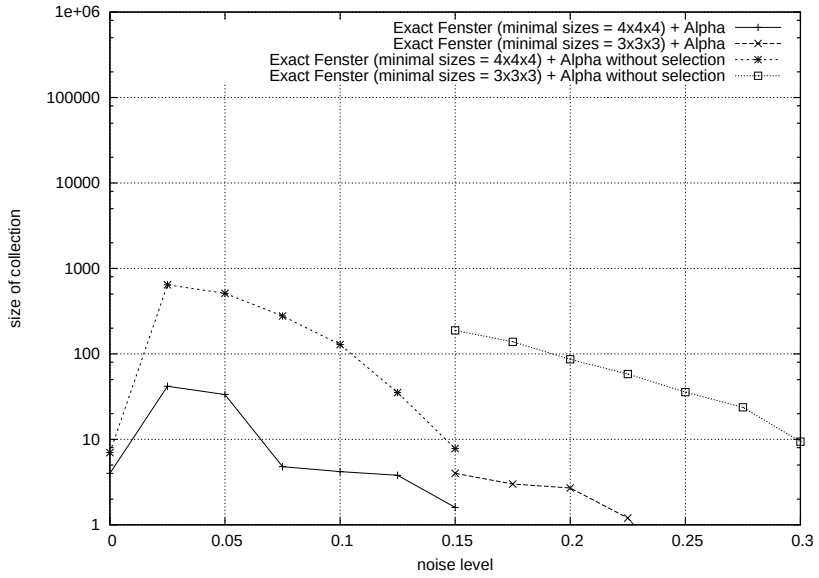



Figure 54: Sizes of the output collections with and without the selection of the relevant patterns.

## 4 CONCLUSION

Closed ET-n-sets globally well cover the hidden patterns. However, unless there is very little noise, FENSTER can only extract, in a reasonable time, fragments of these patterns. ALPHA hierarchically agglomerates these fragments. The involved metric takes into account not only the patterns but also the relation they were extracted from. Contrary to FENSTER, ALPHA uses a (more natural) relative tolerance to noise because it does not pretend to provide a condensed representation of the ET-n-set (it is heuristic) and because that does not increase its running times (quadratic in the number of closed ET-n-sets). The hidden patterns are supposed to be nodes in the constructed dendrogram. Nevertheless, this dendrogram contains about twice more patterns than the number of closed ET-n-sets FENSTER returned. To ease the interpretation, the output collection needs to be small. To do so, ALPHA ranks the agglomerated patterns by relevancy. This measure is the difference between the distance to the other agglomerated patterns (the greater the better) and the quantity of noise in the pattern (the smaller the better). In this respect, ALPHA complies with David Hand's definition of a *local pattern*. Finally, a simple cover test cuts off the least relevant patterns. Experimented on synthetic ternary relations, ALPHA presents small lists of top-quality patterns even when the noise level reaches 15%.

# Part V

# APPLICATION TO DYNAMIC GRAPH MINING

# OUTLINE

Until now, the content of this thesis has been consisting in generalizing closed itemset mining to make it applicable to $n$-ary relations on one hand and to noisy contexts on the other hand. The resulting more generic patterns would be useless if, in addition, the extractor, namely Fenster, could not deal with the particular aspects of a given dataset. Fortunately, it has been shown that its enumeration principles allow the analyst to specify any relevancy constraint as long as it is piecewise (anti)-monotone. Such constraints are particularly useful on datasets that are not only ternary relations but dynamic (directed) graphs too. The beginning of Chapter 6 explains the specialization between ternary relations and dynamic directed graphs. Then, two constraints, namely the almost-contiguity and the symmetry constraints, are proved piecewise (anti)-monotone. As a consequence, Fenster can use them to guide the extraction of the related patterns. These results were initially presented in [CNB09]. To further fasten the extraction of the closed ET-$n$-sets under these two constraints, modifications of the algorithm (rather than the generic enforcement of the constraints) are welcome. Chapter 6 details them.

Every method presented in this thesis was applied to real-life datasets. In particular, ternary and 4-ary relations derived from the logs of `DistroWatch.com` (a comprehensive presentation of the Free, as in freedom, operating systems) were mined. This even led to the publication of results in a two-part article ([Cer08a] and [Cer08b]) designed for a general audience. However, we decided to present, in Chapter 7, results on other data: the usage logs of the Vélo'v network. This network consists of 327 stations, spread over Lyon and its nearby. A rider rents a bicycle and returns it to any other station. Understanding how Vélo'v is used is valuable, for example to improve the fulfilled service. To study it, the routes, between every pair of stations, are tagged frequent (or not) for some days of the week and some time periods. In the difficult context of this application (four attributes, more than 100000 tuples, weak minimal size constraints, etc.), the algorithms presented in this thesis remain tractable. The knowledge discovery process, which takes advantage of them, returns meaningful patterns. In particular, both the symmetry and the almost-contiguity constraints significantly lower the running times without much limiting the discovery of unexpected patterns.

# 6 MINING DYNAMIC GRAPHS

## 1 SPECIALIZING $n$-ARY RELATION MINING

### 1.1 *Dynamic Graph*

Graph mining is a popular topic. Many researchers focus on graph pattern discovery from one large graph, while others consider large collections of graphs. Dynamic graphs (e. g., dynamic interaction networks or dynamic co-interest graphs) belong to this second category. Indeed, a dynamic graph is a set of graphs labelled with timestamps. Two complementary directions of research are observed. On one hand, the global properties of graphs, like the power-law distribution of node degrees or diameters, are studied (see, e. g., [49]). On the other hand, local pattern discovery techniques are used to identify local properties of the graphs. These local techniques benefit from the huge research effort on 0/1 data analysis. Indeed a graph can be seen as a particular bi-partite graph or as an adjacency matrix. This chapter exploits this analogy and the generic closed ET-$n$-set extractor FENSTER is specialized in the extraction of relevant local patterns in dynamic graphs.

*Dynamic graphs are useful across many applicative domains.*

Let $\mathcal{T} \in \mathbb{R}^{|\mathcal{T}|}$ a finite set of timestamps. Let $\mathcal{N}$ a set of nodes. A (possibly directed) graph is uniquely defined by its adjacency matrix $A \in \{0,1\}^{\mathcal{N}\times\mathcal{N}}$. A dynamic graph involving the nodes of $\mathcal{N}$ along $\mathcal{T}$ is uniquely defined by the $|\mathcal{T}|$-tuple $(A_t)_{t\in\mathcal{T}}$ gathering the adjacency matrices of the graph at every timestamp $t \in \mathcal{T}$. Visually, such a stack of adjacency matrices can be seen as a $|\mathcal{T}| \times |\mathcal{N}| \times |\mathcal{N}|$ cube of 0/1 values. We write $a_{t,n^{\text{tail}},n^{\text{head}}} = 1$ (resp. $a_{t,n^{\text{tail}},n^{\text{head}}} = 0$) when, at the timestamp $t$, a link from $n^{\text{tail}}$ to $n^{\text{head}}$ is present (resp. absent).

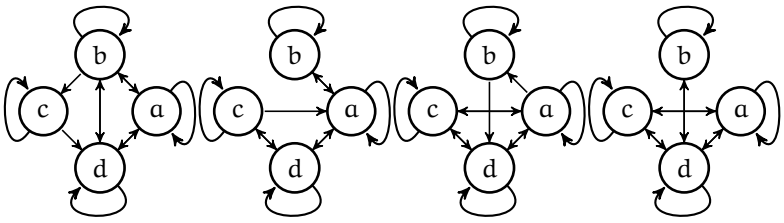
*Dynamic graphs are particular ternary relations.*

**Example 31** *Figure 55 depicts a dynamic directed graph. It involves four nodes:* $a$, $b$, $c$ *and* $d$. *Four snapshots of this graph are available at timestamps* 0, 0.5, 2 *and* 3. *Table 9 gives the related* 4*-tuple* $(A_0, A_{0.5}, A_2, A_3)$.

### 1.2 *A Closed ET-3-Set Under Constraints*

#### 1.2.1 *Specializing Closed ET-3-Set Mining*

In [50], the authors noticed that the problem of enumerating all maximal complete bipartite sub-graphs of a graph is equivalent to




(a) $A_0$ (b) $A_{0.5}$ (c) $A_2$ (d) $A_3$

Figure 55: A dynamic (directed) graph ($\mathcal{N} = \{a, b, c, d\}$, $\mathcal{T} = \{0, 0.5, 2, 3\}$).

| | a | b | c | d | a | b | c | d | a | b | c | d | a | b | c | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| b | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| c | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| d | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| | $A_0$ | | | | $A_{0.5}$ | | | | $A_2$ | | | | $A_3$ | | | |

Table 9: $(A_0, A_{0.5}, A_2, A_3)$ related to the dynamic graph depicted Figure 55.

extracting the closed patterns from its adjacency matrix. This chapter exploits a more general observation: enumerating all maximal cross-graph quasi-complete bipartite sub-graphs is equivalent to extracting the closed ET-3-sets (see Definition 28) in the "stack" of their adjacency matrices (such as that of Table 55). Thus, FENSTER can discover this type of patterns from dynamic graphs.

Nevertheless, the maximal cross-graph quasi-complete bipartite sub-graphs usually are too generic to be relevant for knowledge discovery purposes. In particular, the analyst often is specifically interested in quasi-cliques, i. e., among the quasi-complete bipartite sub-graphs, he/she wants to focus on those that involve the same set of nodes, both as tails and heads of the arcs in the pattern. Furthermore, a dynamic graph is not a generic set of graphs. It is naturally associated with a metric that gives how many seconds separate any two graphs, i. e., the absolute value of the difference between their timestamps. As a consequence, the analyst usually is specifically interested in patterns that involves contiguous (or almost-contiguous) timestamps. Indeed, such patterns described phenomena that are persistent along well localized time intervals. These two points can be translated into two constraints on every closed ET-3-set $(T, N^{tail}, N^{head}) \in 2^{\mathcal{T}} \times 2^{\mathcal{N}} \times 2^{\mathcal{N}}$ FENSTER extracts from the ternary relation associated with the dynamic graph. To ease the understanding of this section, every example uses no tolerance to noise, i. e., $\epsilon = (0, 0, 0)$.

*Cross-graph closed quasi-cliques can be defined as particular closed ET-3-sets. The relevant ones usually involve timestamps that are close to each other.*

### 1.2.2 $\tau$-*Contiguous Closed ET-3-Set*

It has just been argued that closed ET-3-sets that involve timestamps that are close makes more sense than those that involves timestamps that are very far from each other. This qualitative constraint now is quantified: given $\tau \in \mathbb{R}_+$, a $\tau$-contiguous pattern is such that it is possible to browse the whole subset of timestamps by jumps from one timestamp to another without exceeding a delay of $\tau$ for each of these jumps.

**Definition 38 ($\tau$-contiguity)** *A pattern* $(T, N^{tail}, N^{head})$ *is* $\tau$*-contiguous, denoted* $\mathcal{C}_{\tau\text{-contiguous}}(T, N^{tail}, N^{head})$*, iff* $\forall t \in [\min(T), \max(T)], \exists t' \in T$ *s.t.* $|t - t'| \leqslant \tau$.

Notice, in Definition 38, that $t$ does not necessarily belong to $\mathcal{T}$ (if $|T| \geqslant 2$, $[\min(T), \max(T)]$ is infinite). $\mathcal{C}_{\epsilon\text{-connected}} \wedge \mathcal{C}_{\tau\text{-contiguous}}$ being stronger than $\mathcal{C}_{\epsilon\text{-connected}}$ alone, a related and weaker closedness constraint can be defined. Intuitively, a local-closed pattern is closed w.r.t. both $\mathcal{N}$ sets and to the timestamps of $\mathcal{T}$ in the vicinity of those involved in the pattern. In other words, a timestamp that is too far away (delay exceeding $\tau$) from any timestamp inside the pattern, cannot prevent

its local-closedness. When required, the term $\tau$-local-$\epsilon$-closedness, that includes the parametrization $\tau \in \mathbb{R}_+$ of the locality and $\epsilon \in \mathbb{N}^3$ of the noise tolerance, is used.

*Mining almost-contiguous ET-3-sets, the closedness in time is limited to the timestamps close to the pattern.*

**Definition 39 ($\tau$-local-$\epsilon$-closedness)** *A pattern* $(T, N^{tail}, N^{head})$ *is* $\tau$*-local-*$\epsilon$*-closed, denoted* $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}(T, N^{tail}, N^{head})$*, iff*

$$\begin{cases} \forall t \in \mathcal{T} \setminus T, \left(\exists t' \in T \text{ s. t. } |t - t'| \leqslant \tau \Rightarrow \neg \mathcal{C}_{\epsilon\text{-connected}}(\{t\}, N^{tail}, N^{head})\right) \\ \forall n^{tail} \in \mathcal{N} \setminus N^{tail}, \neg \mathcal{C}_{\epsilon\text{-connected}}(T, \{n^{tail}\}, N^{head}) \\ \forall n^{head} \in \mathcal{N} \setminus N^{head}, \neg \mathcal{C}_{\epsilon\text{-connected}}(T, N^{tail}, \{n^{head}\}) \end{cases}$$

**Definition 40 ($\tau$-contiguous closed ET-3-set)** *A pattern* $(T, N^{tail}, N^{head})$ *is a* $\tau$*-contiguous closed ET-3-set iff it satisfies the conjunction* $\mathcal{C}_{\epsilon\text{-connected}} \wedge \mathcal{C}_{\tau\text{-contiguous}} \wedge \mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$.

**Example 32** $(\{2, 3\}, \{a, b, c, d\}, \{d\})$ *is a* 1.75*-contiguous closed* 3*-set in the toy dataset represented in Table 9. However, it is neither* 0.5*-contiguous (the timestamps* 2 *and* 3 *are not close enough) nor* 2*-closed (*0 *can extend the set of timestamps). This illustrates the fact that the number of* $\tau$*-contiguous closed* 3*-sets is not monotone in* $\tau$.

A $\tau$-contiguous closed ET-3-set is an obvious generalization of a closed ET-3-set. Indeed, $\forall \tau \geqslant \max(\mathcal{T}) - \min(\mathcal{T})$, $\mathcal{C}_{\tau\text{-contiguous}} \equiv \mathtt{true} \wedge \mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}} \equiv \mathcal{C}_{\epsilon\text{-closed}}$.

### 1.2.3 $\tau$-*Contiguous Closed ET-3-Clique*

A relevant pattern should involve *one* set of nodes, i. e., every graph should be considered as such and not as a bipartite graph where the nodes are duplicated in two disjoint sets (one set that includes the nodes as heads of the arcs, another where they are tails of the arcs). In other terms, a pattern $(T, N^{tail}, N^{head})$ where $N^{tail} \neq N^{head}$ is irrelevant. That is why a symmetry constraint is added.

**Definition 41 (Symmetry)** *A pattern* $(T, N^{tail}, N^{head})$ *is symmetric, denoted* $\mathcal{C}_{symmetric}(T, N^{tail}, N^{head})$*, iff* $N^{tail} = N^{head}$.

Again, $\mathcal{C}_{\epsilon\text{-connected}} \wedge \mathcal{C}_{\tau\text{-contiguous}} \wedge \mathcal{C}_{symmetric}$ being stronger than $\mathcal{C}_{\epsilon\text{-connected}} \wedge \mathcal{C}_{\tau\text{-contiguous}}$, a related and weaker closedness constraint can be defined. To be said unclosed, a pattern need to be extensible, without breaking $\mathcal{C}_{\epsilon\text{-connected}}$, by a node that would be *both* a tail and a head in the extended pattern.

*Mining ET-3-cliques, the closedness on the nodes is limited to those extending the pattern in both directions of the edges.*

**Definition 42 (Symmetric $\tau$-local-$\epsilon$-closedness)** $(T, N^{tail}, N^{head})$ *is symmetric* $\tau$*-local-*$\epsilon$*-closed, denoted* $\mathcal{C}_{sym\text{-}\delta\text{-closed}}(T, N^{tail}, N^{head})$*, iff*

$$\begin{cases} \forall t \in \mathcal{T} \setminus T, \left(\exists t' \in T \text{ s. t. } |t - t'| \leqslant \tau \Rightarrow \neg \mathcal{C}_{\epsilon\text{-connected}}(\{t\}, N^{tail}, N^{head})\right) \\ \forall n \in \mathcal{N} \setminus (N^{tail} \cap N^{head}), \neg \mathcal{C}_{\epsilon\text{-connected}}(T, N^{tail} \cup \{n\}, N^{head} \cup \{n\}) \end{cases}$$

Symmetric $\tau$-contiguous closed ET-3-sets are called $\tau$*-contiguous closed ET-3-cliques* to shorten a little its denomination. Here is its definition:

**Definition 43 ($\tau$-contiguous closed ET-3-clique)** $(T, N^{tail}, N^{head})$ *is a* $\tau$*-contiguous closed ET-3-clique iff it satisfies the conjunction* $\mathcal{C}_{\epsilon\text{-connected}} \wedge \mathcal{C}_{\tau\text{-contiguous}} \wedge \mathcal{C}_{symmetric} \wedge \mathcal{C}_{sym\text{-}\delta\text{-closed}}$.

**Example 33** *Consider again the dynamic graph represented in Table 9. Both* $(\{2,3\},\{a,c,d\},\{a,c,d\})$ *and* $(\{0,3\},\{b,d\},\{b,d\})$ *are symmetric. Among them,* $(\{0.5,2,3\},\{c,d\},\{c,d\})$ *is not closed w.r.t.* $\mathcal{C}_{closed}$ *because its third component can be extended with* $a$*, i. e.,* $\mathcal{C}_{connected}(\{0.5,2,3\},\{c,d\},\{a\})$*. However, it is symmetric* 1.75*-closed. Indeed, the node* $a$ *cannot simultaneously extend its second and third components without violating* $\mathcal{C}_{connected}$*.*

### 1.3 *Problem Setting*

Let $(A_t)_{t\in\mathcal{T}} \in \{0,1\}^{\mathcal{T}\times\mathcal{N}\times\mathcal{N}}$ and $\tau \in \mathbb{R}_+$. This chapter deals with computing every $\tau$-contiguous closed ET-3-clique that hold in this dataset. In other terms, every pattern satisfying $\mathcal{C}_{\epsilon\text{-connected}} \wedge \mathcal{C}_{\tau\text{-contiguous}} \wedge \mathcal{C}_{\text{symmetric}} \wedge \mathcal{C}_{\text{sym-}\delta\text{-closed}}$ must be listed. In practical settings, such a collection is huge. It makes sense to further constrain the extraction task by taking into account an application-dependent relevancy constraint $\mathcal{C}$. Thus, the problem becomes the complete extraction of the $\tau$-contiguous closed ET-3-cliques verifying $\mathcal{C}$.

## 2 RELATED WORK

### 2.1 *Cross-Graph Quasi-Clique Mining*

Collections of large graphs were built to help in understanding genetics. These graphs commonly have tens of thousands of nodes and are much noisy. For about five years, extracting knowledge by crossing such graphs has been a hot topic. For example, there is a need to extract patterns that remain valid across several co-expression graphs obtained from microarray data or to cross the data pertaining to physical interactions between molecules (e. g., protein-protein, protein-gene) with more conceptual data (e. g., co-expression of genes, co-occurrence of proteins in the literature). One of the most promising pattern helping in these tasks is the closed 3-clique or, better, the closed quasi-3-clique. CLAN [90] is able to extract closed 3-cliques from collections of large and dense graphs. Crochet+ [42], Cocain* [97] and Quick [51] are the state-of-the-art extractors of closed quasi-3-cliques. They all use the same definition of noise tolerance: every node implied in a pattern must have, in every graph *independently* from the others, a degree exceeding a user-defined proportion of the maximal degree it would reach if the clique was exact. Thus, a pattern involving a subset $T$ of the graphs and a subset $N$ of the nodes needs to satisfy $|T \times N|$ constraints to be a quasi-3-clique, i. e., one constraint *per couple* (timestamp, node).

*In the literature, cross-graph quasi-cliques tolerate noise in a more constrained way: one constraint per couple (node,graph). Moreover, the graphs are undirected.*

This definition of noise tolerance is different from the one involved in the definition of the closed ET-$n$-sets FENSTER extracts. Indeed, in Definition 26, an upper-bounded number of absent $n$-tuples (rather than a proportion) is tolerated *per element* involved in the pattern, i. e., $(T, N^{tail}, N^{head})$ is, by definition, an ET-3-set iff:

- $\forall t \in T$, the dynamic graph contains all 3-tuples in $\{t\} \times N^{tail} \times N^{head}$ but $\epsilon^{timestamps}$ or less.
- $\forall n^{tail} \in N^{tail}$, the dynamic graph contains all 3-tuples in $T \times \{n^{tail}\} \times N^{head}$ but $\epsilon^{tail}$ or less.
- $\forall n^{head} \in N^{head}$, the dynamic graph contains all 3-tuples in $T \times N^{tail} \times \{n^{head}\}$ but $\epsilon^{head}$ or less.

In this way, $(T, N^{tail}, N^{head})$ needs to satisfy $|T| + |N^{tail}| + |N^{head}|$ constraints to be an ET-3-set. If only symmetric patterns are considered,

i. e., $N^{tail} = N^{head} = N$, this number becomes $|T| + 2|N|$. In the specific context of undirected graphs (contrary to FENSTER, none of the previously cited approaches can deal with directed graphs), the constraints FENSTER applies on the tails and on the heads are identical. As a consequence, only $|T| + |N|$ constraints defines an ET-3-clique. By comparing this number to $|T \times N|$, it can be written that it is easier for a pattern to be an ET-3-set than a quasi-clique in the sense of Crochet+, Cocain* or Quick (the patterns involving only one timestamp or one node are exceptions to this assertion but they are not much interesting). As a consequence our approach does not scale well to graphs connecting thousands of nodes. Nevertheless, because FENSTER indifferently enumerates timestamps and nodes (no attribute is favored), it can extract closed ET-3-cliques in large collections of smaller graphs, whereas the other algorithms cannot (or they must be used with a very strong minimal size constraint on the number of involved graphs). The use of the $\tau$-contiguity constraint further increases this difference.

### 2.2 *Contiguity*

*The almost-contiguity constraint comes from sequence mining.*

The $\tau$-contiguity stems from an analogous constraint, called *max-gap* constraint, initially applied to sequence mining. It was introduced in the GSP approach [80]. The way the $\tau$-contiguity is enforced in our approach (see Section 3) is similar to that of this seminal article. The *min-gap* and the *window size* constraints [80] uses could as well be enforced in our approach. Nevertheless, in [80], these constraints modify the enumeration order, whereas, in our approach, they reduce the search space and let the enumeration strategy unaltered. Furthermore, the nature of the mined patterns is much different. In the context of [80], the considered datasets are multiple sequences of itemsets and the extracted patterns are sub-sequences of itemsets whose order (but not position in time) is to be respected in all (1-dimensional) supporting sequences. In our approach, the supporting domain contains (2-dimensional) graphs and their position in time must be aligned.

Notice that the max-gap constraint was used in other contexts too. For example, [17] and [57] enforce it to extract episodes (repetition of sub-sequences in one sequence) and [23] somehow combines sequence and episode mining by extracting, under a max-gap constraint, frequent sub-sequences whose support is the sum of the number of repetitions in all sequences of the dataset.

## 3 MINING $\tau$-CONTIGUOUS CLOSED ET-3-SET

### 3.1 *A Piecewise (Anti)-Monotone Constraint...*

*The almost-contiguity constraint is piecewise (anti)-monotone.*

FENSTER can enforce the constraint $\mathcal{C}_{\tau\text{-contiguous}}$ (see Definition 38) at extraction time. Indeed, it is piecewise (anti)-monotone.

**Proof 1** *Let $\mathcal{C}'_{\tau\text{-contiguous}}$ the following constraint:*

$$\begin{aligned} & \mathcal{C}'_{\tau\text{-contiguous}}(T_1, T_2, T_3, N^{tail}, N^{head}) \\ \equiv \quad & \forall t \in [\min(T_1), \max(T_2)], \exists t' \in T_3 \ s.t. \ |t - t'| \leqslant \tau \ . \end{aligned}$$

*The three arguments $T_1$, $T_2$ and $T_3$ substitute the three occurrences of $T$ (in the definition of $\mathcal{C}_{\tau\text{-contiguous}}$). $\mathcal{C}'_{\tau\text{-contiguous}}$ is monotone in on its third argument and anti-monotone on its first and second arguments ($T \subseteq T_1 \Rightarrow$*

$\min(T) \geqslant \min(T_1)$ *and* $T \subseteq T_2 \Rightarrow \max(T) \leqslant \max(T_2)$*). Moreover, since the two last arguments of* $\mathcal{C}'_{\tau\text{-contiguous}}$ *do not appear in its expression, this constraint is both monotone and anti-monotone on them. Therefore, by definition,* $\mathcal{C}_{\tau\text{-contiguous}}$ *is piecewise (anti)-monotone.*

### 3.2 ... *Partially Handled in Another Way*

Given the 3-sets $U = (U^{\text{times}}, U^{\text{tail}}, U^{\text{head}})$ and $V = (V^{\text{times}}, V^{\text{tail}}, V^{\text{head}})$ attached to the current enumeration node, the proof in Section 3.1 suggests to check whether it is possible to browse all elements in $[\min(U^{\text{times}}), \max(U^{\text{times}})] \cap (U^{\text{times}} \cup V^{\text{times}})$ by jumps of, at most, $\tau$.

By also taking a look "around" $[\min(U^{\text{times}}, \max(U^{\text{times}})] \cap (U^{\text{times}} \cup V^{\text{times}})$, FENSTER can do better than just telling whether there is no hope in extracting $\tau$-contiguous ET-3-sets from the current enumeration node. Indeed, it can prevent the traversal of some of such nodes. More precisely, FENSTER removes from $V^{\text{times}}$ the elements that would, if enumerated, generate left children violating $\mathcal{C}_{\tau\text{-contiguous}}$. To do so, the delay between $t = \min(U^{\text{times}})$ and $\text{before}(t) = \max(\{t' \in V^{\text{times}} \mid t' < t\})$ is considered. If it is strictly greater than $\tau$ then every element in $\{t' \in V^{\text{times}} \mid t' < t\}$ can be removed from $V^{\text{times}}$. Otherwise, the process goes on with $t = \text{before}(t)$ until a delay greater than $\tau$ is found or until $t = \min(V^{\text{times}})$ (in this case no element from $V^{\text{times}}$ lesser than $\min(U^{\text{times}})$ is removed). In a reversed way, the elements in $V^{\text{times}}$ that are too great to be moved to $U^{\text{times}}$ without violating $\mathcal{C}_{\tau\text{-contiguous}}$ are removed as well. Figure 56 gives a more technical definition of FENSTER's way to purge $V^{\text{times}}$ thanks to $\mathcal{C}_{\tau\text{-contiguous}}$.

*Given an enumeration sub-tree, the almost-contiguity constraint removes from the search space the timestamps that are too far away from the largest ET-3-set.*

In the same way, some elements of $\mathcal{S}^{\text{times}}$ may be too far away from the extrema of $U^{\text{times}} \cup V^{\text{times}}$ to prevent the local-closedness of any descending ET-3-set. These elements are those that cannot be added to $U^{\text{times}}$ without making the current enumeration node violate $\mathcal{C}_{\tau\text{-contiguous}}$. FENSTER removes them by applying a procedure PURGE_$\mathcal{S}^{\text{TIMES}}$ to every enumeration node. It is very similar to PURGE_$V^{\text{TIMES}}$ (see Algorithm 56) except that it is $\mathcal{S}^{\text{times}}$ which is browsed backward from $\text{before}(\min(U^{\text{times}} \cup V^{\text{times}}))$ and forward from $\text{after}(\max(U^{\text{times}} \cup V^{\text{times}}))$.

**Example 34** *Considering the extraction of* 1*-contiguous* 3*-sets from the example dataset represented in Table 9, if the first enumerated element is* 0.5*, Figure 57 depicts the root enumeration node and its two children. In the left child,* PURGE_$V^{\text{TIMES}}$ *removes* 2 *and* 3 *from its attached* $V^{times}$ *set because* $2 - 0.5 > 1$.

These purges of $V$ and $\mathcal{S}$ remind the way FENSTER handles $\mathcal{C}_{\epsilon\text{-connected}}$. Nevertheless $\mathcal{C}_{\epsilon\text{-connected}}$ is anti-monotone on all its arguments, whereas $\mathcal{C}_{\tau\text{-contiguous}}$ is *only* piecewise (anti)-monotone. That is why some enumeration nodes violating $\mathcal{C}_{\tau\text{-contiguous}}$ may be generated despite the calls of PURGE_$V^{\text{TIMES}}$ (whereas a generated enumeration node always complies with $\mathcal{C}_{\epsilon\text{-connected}}$). As a consequence, checking, at every enumeration node, whether $\mathcal{C}_{\tau\text{-contiguous}}$ holds remains necessary. For the same reason, some elements in the 3-sets $V$ and/or $\mathcal{S}$ attached to both left and right children may be purged thanks to $\mathcal{C}_{\tau\text{-contiguous}}$ (whereas $\mathcal{C}_{\epsilon\text{-connected}}$ cannot reduce the search space of a right child).

**Input:** $U^{times}, V^{times}$
**if** $U^{times} \neq \emptyset$ **then**
  $V^{times} \leftarrow$ **sort**$(V^{times})$
  $t \leftarrow \min(U^{times})$
  **if** $t > \min(V^{times})$ **then**
    $before(t) \leftarrow \max(\{t' \in V^{times} \mid t' < t\})$ {Binary search in $V^{times}$}
    **while** $before(t) \neq \min(V^{times}) \wedge t - before(t) \leqslant \tau$ **do**
      $t \leftarrow before(t)$
      $before(t) \leftarrow$ **previous**$(V^{times}, t)$ {$V^{times}$ is browsed backward}
    **end while**
    **if** $t - before(t) > \tau$ **then**
      $V^{times} \leftarrow V^{times} \setminus [\min(V^{times}), before(t)]$
    **end if**
  **end if**
  $t \leftarrow \max(U^{times})$
  **if** $t < \max(V^{times})$ **then**
    $after(t) \leftarrow \min(\{t' \in V^{times} \mid t' > t\})$ {Binary search in $V^{times}$}
    **while** $after(t) \neq \max(V^{times}) \wedge after(t) - t \leqslant \tau$ **do**
      $t \leftarrow after(t)$
      $after(t) \leftarrow$ **next**$(V^{times}, t)$ {$V^{times}$ is browsed forward}
    **end while**
    **if** $after(t) - t > \tau$ **then**
      $V^{times} \leftarrow V^{times} \setminus [after(t), \max(V^{times})]$
    **end if**
  **end if**
**end if**

Figure 56: The PURGE_$V^{TIMES}$ procedure.

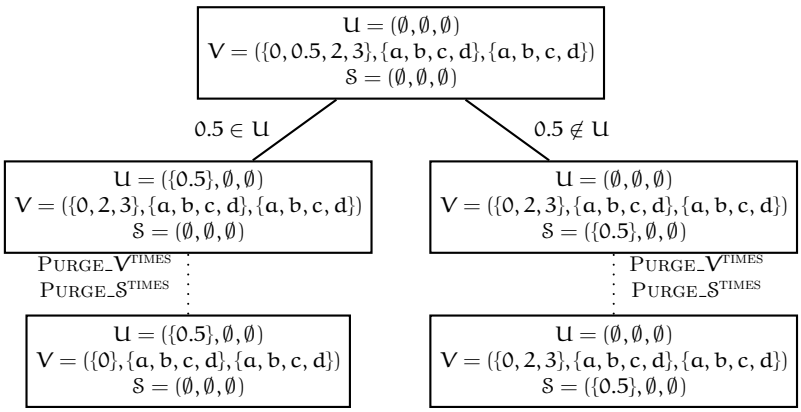



Figure 57: Enumeration of $0.5 \in V$ during the extraction of 1-contiguous 3-sets from the dataset represented in Table 9.

### 3.3 *Enforcing the τ-Closedness*

*The closedness constraint remains piecewise (anti)-monotone when limited in time.*

The constraint $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$ (see Definition 39) is piecewise (anti)-monotone.

**Proof 2** *Let $\mathcal{C}'_{\tau\text{-local-}\epsilon\text{-closed}}$ the following constraint:*

$$\mathcal{C}'_{\tau\text{-local-}\epsilon\text{-closed}}(T_1, T_2, T_3, T_4, N_1^{tail}, N_2^{tail}, N_3^{tail}, N_1^{head}, N_2^{head}, N_3^{head}) \equiv \begin{cases} \forall t \in \mathcal{T} \setminus T_1, \\ \left(\exists t' \in T_2 \text{ s.t. } |t - t'| \leqslant \tau \Rightarrow \neg\mathcal{C}_{\epsilon\text{-connected}}(\{t\}, N_1^{tail}, N_1^{head})\right) \\ \forall n^{tail} \in \mathcal{N} \setminus N_2^{tail}, \neg\mathcal{C}_{\epsilon\text{-connected}}(T_3, \{n^{tail}\}, N_2^{head}) \\ \forall n^{head} \in \mathcal{N} \setminus N_3^{head}, \neg\mathcal{C}_{\epsilon\text{-connected}}(T_4, N_3^{tail}, \{n^{head}\}) \end{cases}$$

*$\mathcal{C}'_{\tau\text{-local-}\epsilon\text{-closed}}$ is anti-monotone on its second argument and monotone on all its other arguments. Therefore, by definition, $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$ is piecewise (anti)-monotone.*

A way to enforce $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$ follows from the proof of its piecewise (anti)-monotonicity: an enumeration node, i. e., its attached 3-sets $U = (U^{times}, U^{tail}, U^{head})$ and $V = (V^{times}, V^{tail}, V^{head})$, may lead to some local-closed ET-3-set if $(U^{times} \cup V^{times}, U^{tail} \cup V^{tail}, U^{head} \cup V^{head})$:

- cannot be extended by any element in $\mathcal{T} \setminus (U^{times} \cup V^{times})$ distant, by at most $\tau$, from an element in $U^{times}$;
- cannot be extended by any element in $\mathcal{N} \setminus (U^{tail} \cup V^{tail})$;
- cannot be extended by any element in $\mathcal{N} \setminus (U^{head} \cup V^{head})$.

As done for $\mathcal{C}_{\epsilon\text{-closed}}$, to avoid useless (and costly) tests, FENSTER maintains the 3-set $\mathcal{S} = (\mathcal{S}^{times}, \mathcal{S}^{tail}, \mathcal{S}^{head})$ containing only the elements that may prevent the closure of the ET-3-sets descending from the current enumeration node, i. e., the previously enumerated elements and not those that were removed from $V$ thanks to $\mathcal{C}_{\epsilon\text{-connected}} \wedge \mathcal{C}_{\tau\text{-contiguous}}$. Moreover, as explained in Section 3.2, FENSTER purges $\mathcal{S}$ before checking $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$. Since it is used in conjunction with $\mathcal{C}_{\tau\text{-contiguous}}$, $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$ can be more strongly enforced: no element in $\mathcal{S}^{times} \cap [\min(U^{times}) - \tau, \max(U^{times}) + \tau]$ is allowed to extend $(U^{times} \cup V^{times}, U^{tail} \cup V^{tail}, U^{head} \cup V^{head})$. Indeed, an element in $\mathcal{S}^{times} \cap [\min(U^{times}) - \tau, \max(U^{times}) + \tau]$ may be distant, by strictly more than $\tau$, from any element in $U^{times}$ but this will never be the case at the leaves descending from the current enumeration since $U^{times}$ must then be $\tau$-contiguous. All in all, FENSTER prunes the sub-tree descending from the current enumeration node if $(U^{times} \cup V^{times}, U^{tail} \cup V^{tail}, U^{head} \cup V^{head})$ can be extended by any element in $\mathcal{S}^{times} \cap [\min(U^{times}) - \tau, \max(U^{times}) + \tau]$, $\mathcal{S}^{tail}$ or $\mathcal{S}^{head}$.

## 4 MINING τ-CONTIGUOUS CLOSED ET-3-CLIQUES

### 4.1 *A Piecewise (Anti)-Monotone Constraint...*

*"Being a clique" is piecewise (anti)-monotone.*

In an ET-3-clique, both subsets of $\mathcal{N}$ are identical. An equivalent definition to the symmetry constraint (Definition 41) would be as follows: $\mathcal{C}_{\text{symmetric}}(T, N^{tail}, N^{head}) \equiv N^{tail} \subseteq N^{head} \wedge N^{head} \subseteq N^{tail}$. In this form, a piecewise (anti)-monotone constraint is identified.

**Proof 3** *Let $\mathcal{C}'_{symmetric}$ the following constraint:*

$$\mathcal{C}'_{symmetric}(T, N_1^{tail}, N_2^{tail}, N_1^{head}, N_2^{head}) \equiv N_1^{tail} \subseteq N_1^{head} \wedge N_2^{head} \subseteq N_2^{tail} \ .$$

*$N_1^{tail}$ and $N_2^{tail}$ substitute the two occurrences of $N^{tail}$ (in the alternative definition of $\mathcal{C}_{symmetric}$). In the same way, $N_1^{head}$ and $N_2^{head}$ substitute the two occurrences of $N^{head}$. $\mathcal{C}'_{symmetric}$ is monotone on its third and fourth arguments ($N_2^{tail}$ and $N_1^{head}$) and anti-monotone on its second and fifth arguments ($N_1^{tail}$ and $N_2^{head}$). Moreover, since the first argument ($T$) does not appear in the expression of $\mathcal{C}'_{symmetric}$, this constraint is both monotone and anti-monotone on this argument. Therefore, by definition, $\mathcal{C}_{symmetric}$ is piecewise (anti)-monotone.*

Being piecewise (anti)-monotone, the symmetry constraint can be efficiently exploited by FENSTER. However, the enumeration tree can be further reduced if this constraint is enforced when choosing the element to be enumerated.

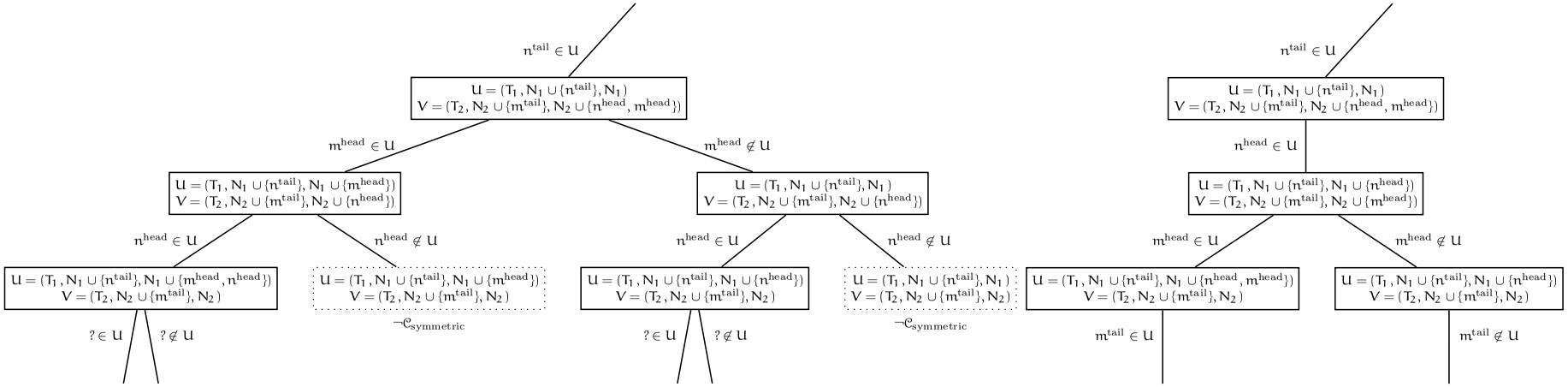
## 4.2 ... Better Handled in Another Way

In this section, a distinction between the nodes considered as tails (i. e., the rows of the adjacency matrices) and the nodes considered as heads (i. e., the columns of the adjacency matrices) must be made. They are respectively named $\mathcal{N}^{tail}$ and $\mathcal{N}^{head}$. Intuitively, when an element $n^{tail}$ in $V^{tail} \subseteq \mathcal{N}^{tail}$ is chosen to be present (resp. absent) in any ET-3-clique extracted from the current enumeration node (see Section 1.2 in Chapter 3), the element $n^{head}$ in $V^{head} \subseteq \mathcal{N}^{head}$ standing for the same node should be enumerated just after and only to be present (resp. absent) too. Thus, the enumeration tree is not a binary tree anymore (some enumeration nodes only have one child).

*The enumeration of a node as a head (resp. tail) follows that of the same node as a tail (resp. head) and either both are present or both are absent.*

When handled as a piecewise (anti)-monotone constraint, the symmetry constraint leads to many more enumeration nodes. When $n^{head}$ is chosen to be enumerated, the left (resp. right) child where $n^{head}$ is present (resp. absent) is generated even if its counterpart $n^{tail}$ in the other set was previously set absent (resp. present). Then the symmetry constraint prunes the sub-tree rooted by this node. Since there is no reason for $n^{head}$ to be enumerated just after $n^{tail}$, the intuition tells us that the number of such nodes, whose generation could be avoided by modifying the enumeration (as explained in the previous paragraph), increases exponentially with the average number of enumeration nodes between the enumeration of $n^{tail}$ and that of $n^{head}$. This is actually not a theorem because $\mathcal{C}_{\text{sym-}\delta\text{-closed}}$ or $\mathcal{C}$ may prune some descendant sub-trees before $n^{head}$ is enumerated. Anyway, in practical settings, handling the symmetry constraint via a modification of the enumeration usually is much more efficient than via the general framework for piecewise (anti)-monotone constraints.

Figure 58 informally depicts these two approaches (the probable diminutions of the $V$ sets in the left children and the possible pruning due to $\mathcal{C}_{\epsilon\text{-closed}}$ or $\mathcal{C}$ are ignored). $T_1$ and $T_2$ are subsets of $\mathcal{T}$. $N_1$ and $N_2$ are subsets of $\mathcal{N}$. In both examples, the elements $m^{head}$ and $n^{head}$ of $\mathcal{N}^{head}$ are enumerated. The resulting nodes are, of course, the same (the dotted nodes being pruned). However this result is straightforward when the enumeration constraint is handled through a modification of the enumeration (Figure 58b), whereas it usually requires more nodes when it is handled as an ordinary piecewise (anti)-monotone constraint (Figure 58a). The number of additional nodes in the latter case grows exponentially with the number of elements enumerated between $n^{tail}$ and $n^{head}$ (e. g., $m^{tail}$ could be enumerated in between).




(a) Symmetry handled as an ordinary piecewise (anti)-monotone constraint.

(b) Symmetry handled by a modified enumeration.

Figure 58: Handling the symmetry constraint.

### 4.3 *Constraining the Enumeration*

Let $\mathcal{N}^{\text{tail}} = (n_i^{\text{tail}})_{i=1..|\mathcal{N}|}$ and $\mathcal{N}^{\text{head}} = (n_i^{\text{head}})_{i=1..|\mathcal{N}|}$ its counterpart, i.e., $\forall i = 1..|\mathcal{N}|$, $n_i^{\text{tail}}$ and $n_i^{\text{head}}$ stand for the same node. $(T, N^{\text{tail}}, N^{\text{head}})$ being symmetric is a constraint that can be expressed as this list of, so called, *enumeration constraints*:

$$
\begin{array}{ll}
n_1^{\text{tail}} \in N^{\text{tail}} \Rightarrow n_1^{\text{head}} \in N^{\text{head}} & n_1^{\text{head}} \in N^{\text{head}} \Rightarrow n_1^{\text{tail}} \in N^{\text{tail}} \\
n_2^{\text{tail}} \in N^{\text{tail}} \Rightarrow n_2^{\text{head}} \in N^{\text{head}} & n_2^{\text{head}} \in N^{\text{head}} \Rightarrow n_2^{\text{tail}} \in N^{\text{tail}} \\
\vdots & \vdots \\
n_i^{\text{tail}} \in N^{\text{tail}} \Rightarrow n_i^{\text{head}} \in N^{\text{head}} & n_i^{\text{head}} \in N^{\text{head}} \Rightarrow n_i^{\text{tail}} \in N^{\text{tail}} \\
\vdots & \vdots \\
n_{|\mathcal{N}|}^{\text{tail}} \in N^{\text{tail}} \Rightarrow n_{|\mathcal{N}|}^{\text{head}} \in N^{\text{head}} & n_{|\mathcal{N}|}^{\text{head}} \in N^{\text{head}} \Rightarrow n_{|\mathcal{N}|}^{\text{tail}} \in N^{\text{tail}}
\end{array}
$$

We actually made FENSTER handle a more general class of constraints:

*A set of constraints defines "being a clique". They belong to a larger class of constraints efficiently handled via occasional modifications of* FENSTER*'s enumeration.*

**Definition 44 (Enumeration constraint)** *$\mathcal{C}_{enum}$ is said to be an enumeration constraint iff, given an ET-3-set $(T, N^{tail}, N^{head})$, it is of the form:*

$$\mathcal{C}_{enum}(T, N^{tail}, N^{head}) \equiv \wedge_{i=1..k} a_i \Rightarrow a_{k+1},$$

*where $k \in \mathbb{N}$ and $\forall i = 1..(k+1)$, $a_i$ is of the form $e \in A$ or $e \notin A$, $e$ being an arbitrary element from an arbitrary attribute domain $A \in \{T, N^{tail}, N^{head}\}$.*

**Example 35** *Here are three examples of enumeration constraints that can be enforced on any ET-3-set $(T, N^{tail}, N^{head})$:*
- $t_1 \in T \Rightarrow t_8 \notin T$
- $t_1 \notin T \wedge n_1^{tail} \in N^{tail} \Rightarrow t_2 \in T$
- *true* $\Rightarrow t_1 \notin T$ *($k = 0$ in Definition 44)*

*Notice that the last constraint is not equivalent to removing the element $t_1$ from the data. Indeed, a closed ET-3-set in the dataset deprived of $t_1$ may not be closed in the dataset containing $t_1$. In the latter case, it must not be extracted (and it is not extracted when the enumeration constraint is enforced).*

Before choosing the element to enumerate (see Figure 32), FENSTER browses the set of enumeration constraint, and tests whether the left parts of them are true or not. Considered as constraints, these left parts are, again, piecewise (anti)-monotone. Indeed, when there is a term of the form $e \in A$ (resp. $e \notin A$), the left part of the constraint is anti-monotone (resp. monotone) in this occurrence of $A$. Given the 3-sets $U$ and $V$ attached to the current enumeration node, three cases may arise:

1. The left part will never be fulfilled in the sub-tree rooted by the current enumeration node:
   - if an element in the left part is to be present but it is neither in $U$ not in $V$.
   - if an element in the right part is to be absent but it is in $U$.
2. The left part is fulfilled by at least one (but not every) node descending from the current enumeration node.
3. The left part is fulfilled by every node descending from the current enumeration node:
   - if an element in the left part is to be present, it is in $U$.

– if an element in the left part is to be absent, it is neither in $U$ nor in $V$.

FENSTER reacts differently at each of these cases:

1. This enumeration constraint is removed from the set of enumeration constraints when traversing the sub-tree rooted by the current enumeration node. Indeed, it never applies in this sub-tree. Uselessly checking it for every descendant enumeration node would only decrease FENSTER's performance.
2. This enumeration constraint is kept.
3. The right part of this enumeration constraint is considered.

When the right part of an enumeration constraint is considered, three new cases may arise:

3.1 The right part is already fulfilled:
– if its element is to be present and is already in $U$.
– if its element is to be absent and is already neither in $U$ nor in $V$.

3.2 The right part can be fulfilled: if its element is in $V$.

3.3 The right part cannot be fulfilled:
– if its element is to be present and is neither in $U$ nor in $V$.
– if its element is to be absent and is in $U$.

FENSTER differently reacts at each of these cases:

3.1 This enumeration constraint is removed from the set of enumeration constraints when traversing the sub-tree rooted by the current enumeration node. Indeed, it is satisfied for all ET-3-sets in this sub-tree. Uselessly checking it for every descendant enumeration node would only decrease FENSTER's performance.

3.2 The element on the right part of the constraint can be enumerated as specified (one child only).

3.3 The sub-tree rooted by the current enumeration node is pruned. Indeed, none of the ET-3-sets in this sub-tree verifies the constraint.

In Case 3.2, we write "the element *can* be enumerated" because, at a given enumeration node, several enumeration constraint may be in this case but only one can be applied.

## 4.4 *Contraposition of Enumeration Constraints*

If an enumeration constraint holds, its contraposition, logically, holds too. In the general case (conjunction of terms in the left part), the contraposition of an enumeration constraint is not an enumeration constraint (disjunction of terms in the right part). In the particular case of enumeration constraints of the form $a_1 \Rightarrow a_2$ (see Definition 44), e. g., those generated from $\mathcal{C}_{\text{symmetric}}$ (see Section 4.3), their contrapositions are enumeration constraints too. Thus, FENSTER enforces a larger set of enumeration constraints (the original set of enumeration constraints and the contrapositions of those of the form $a_1 \Rightarrow a_2$) for even faster extractions. Figure 59 gives a more technical definition of how this larger set is computed.

**Example 36** *Among the enumeration constraints of Example 35, only the first one* ($t_1 \in T \Rightarrow t_8 \notin T$) *admits a contraposition* ($t_8 \in T \Rightarrow t_1 \notin T$) *that is, itself, an enumeration constraint.*

**Input:** Set $E$ of enumeration constraints
**Input:** Set $E$ enlarged with contrapositions
$E' \leftarrow E$
**for** $a_1 \wedge a_2 \wedge \cdots \wedge a_k \Rightarrow a_{k+1} \in E$ **do**
  **if** $k = 1$ **then**
    $E' \leftarrow E' \cup \{\neg a_2 \Rightarrow \neg a_1\}$
  **end if**
**end for**
**return** $E'$

Figure 59: APPEND_CONTRAPOSITION.

## 4.5 *Enforcing the Symmetric τ-Closedness*

FENSTER can enforce the constraint $\mathcal{C}_{\text{sym-}\delta\text{-closed}}$ (see Definition 42) at extraction time. Indeed, it is piecewise (anti)-monotone.

*The closedness constraint remains piecewise (anti)-monotone for almost-contiguous cross-graph quasi-cliques.*

**Proof 4** *Let $\mathcal{C}'_{sym\text{-}\delta\text{-}closed}$ the following constraint:*

$$\mathcal{C}'_{sym\text{-}\delta\text{-}closed}(T_1, T_2, T_3, N_1^{tail}, N_2^{tail}, N_3^{tail}, N_1^{head}, N_2^{head}, N_3^{head}) \equiv \begin{cases} \forall t \in \mathcal{T} \setminus T_1, \\ \qquad \left(\exists t' \in T_2 \text{ s.t. } |t - t'| \leqslant \tau \Rightarrow \neg \mathcal{C}_{\epsilon\text{-}connected}(\{t\}, N_1^{tail}, N_1^{head})\right) \\ \forall n \in \mathcal{N} \setminus (N_2^{tail} \cap N_2^{head}), \neg \mathcal{C}_{\epsilon\text{-}connected}(T, N_3^{tail} \cup \{n\}, N_3^{head} \cup \{n\}) \end{cases}$$

*$\mathcal{C}'_{sym\text{-}\delta\text{-}closed}$ is anti-monotone on its second argument ($T_2$) and monotone on all its other arguments. Therefore, by definition, $\mathcal{C}_{sym\text{-}\delta\text{-}closed}$ is piecewise (anti)-monotone.*

A way to enforce $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$ follows from the proof of its piecewise (anti)-monotonicity: an enumeration node, i. e., its attached 3-sets $U = (U^{\text{times}}, U^{\mathcal{N}^{\text{tail}}}, U^{\mathcal{N}^{\text{head}}})$ and $V = (V^{\text{times}}, V^{\mathcal{N}^{\text{tail}}}, V^{\mathcal{N}^{\text{head}}})$, may lead to some local-closed ET-3-set if $(U^{\text{times}} \cup V^{\text{times}}, U^{\mathcal{N}^{\text{tail}}} \cup V^{\mathcal{N}^{\text{tail}}}, U^{\mathcal{N}^{\text{head}}} \cup V^{\mathcal{N}^{\text{head}}})$:

- cannot be extended by any element in $\mathcal{T} \setminus (U^{\text{times}} \cup V^{\text{times}})$ distant, by at most $\tau$, from an element in $U^{\text{times}}$;
- cannot be simultaneously extended by any element in $\mathcal{N} \setminus (U^{\mathcal{N}^{\text{tail}}} \cup V^{\mathcal{N}^{\text{tail}}})$ (row of the adjacency matrices) and its related element in $\mathcal{N} \setminus (U^{\mathcal{N}^{\text{head}}} \cup V^{\mathcal{N}^{\text{head}}})$ (column of the adjacency matrices).

In a similar way to what was done with $\mathcal{C}_{\tau\text{-local-}\epsilon\text{-closed}}$ (see Section 3.3), FENSTER maintains the 3-set $\mathcal{S} = (\mathcal{S}^{\text{times}}, \mathcal{S}^{\mathcal{N}^{\text{tail}}}, \mathcal{S}^{\mathcal{N}^{\text{head}}})$ containing only the elements that may prevent the closure of the ET-3-sets descending from the current enumeration node and prunes the sub-tree descending from it if $(U^{\text{times}} \cup V^{\text{times}}, U^{\mathcal{N}^{\text{tail}}} \cup V^{\mathcal{N}^{\text{tail}}}, U^{\mathcal{N}^{\text{head}}} \cup V^{\mathcal{N}^{\text{head}}})$ can be extended by any element in $\mathcal{S}^{\text{times}} \cap [\min(U^{\text{times}}) - \tau, \max(U^{\text{times}}) + \tau]$ or by any element in $\mathcal{S}^{\mathcal{N}^{\text{tail}}}$ *and* its related element in $\mathcal{S}^{\mathcal{N}^{\text{head}}}$. Thus, when $\mathcal{S}^{\mathcal{N}^{\text{tail}}}$ (respectively $\mathcal{S}^{\mathcal{N}^{\text{head}}}$) is purged from an element (because it cannot extend $(U^{\text{times}} \cup V^{\text{times}}, U^{\mathcal{N}^{\text{tail}}} \cup V^{\mathcal{N}^{\text{tail}}}, U^{\mathcal{N}^{\text{head}}} \cup V^{\mathcal{N}^{\text{head}}})$ without violating $\mathcal{C}_{\epsilon\text{-connected}}$), the related element in $\mathcal{S}^{\mathcal{N}^{\text{head}}}$ (respectively $\mathcal{S}^{\mathcal{N}^{\text{tail}}}$) is removed as well.

An overall view of the complete extraction of the τ-contiguous closed ET-3-cliques under constraint can now be presented. The details and

**Input:** $(A_t)_{t\in\mathcal{T}} \in \{0,1\}^{\mathcal{T}\times\mathcal{N}\times\mathcal{N}}$, $\epsilon \in \mathbb{N}^3$, $\tau \in \mathbb{R}_+$ and a user-defined piecewise (anti)-monotone constraint $\mathcal{C}$
**Output:** Every $\tau$-contiguous closed ET-3-clique in $(A_t)_{t\in\mathcal{T}}$ satisfying $\mathcal{C}$
$E \leftarrow$ Enumeration constraints pertaining to $\mathcal{C}_{\text{symmetric}}$ (see Section 4.3)

$E' \leftarrow$ APPEND_CONTRAPOSITION($E$)
FENSTER($(\emptyset,\emptyset,\emptyset)$, $(\mathcal{T},\mathcal{N},\mathcal{N})$, $(\emptyset,\emptyset,\emptyset)$)

Figure 60: MAIN.

justifications of how every identified constraint is handled are present within the two previous sections, hence proving its correctness. Figure 60 shows the main procedure solving the problem stated in Section 1.3. It calls the algorithm in Figure 61 which can be regarded as a specialization of that of FENSTER (see Figure 32).

## 5 CONCLUSION

Together a symmetry and a contiguity constraints specialize FENSTER to make it extract every $\tau$-contiguous closed ET-3-clique in a dynamic graph. Because these constraints are piecewise (anti)-monotone, the expressive power of this class of constraints is emphasized and they could be enforced on the top of a "plain-vanilla" FENSTER. However, to scale up to very large dynamic graphs, these constraints must be enforced more cleverly. Interestingly, the idea, which is carried out, is the same for the two constraints (and for the connection constraint too): they must be used as soon as possible in the enumeration tree. The symmetry constraint has even been split into many small enumeration constraints that are individually exploited as soon as possible. Enumeration constraints are particular since they change the structure of the enumeration, which is not binary anymore. This chapter focuses on extracting $\tau$-contiguous closed ET-3-cliques. However, FENSTER is not restricted to ternary relations. Thus, it can mine graphs that are parametrized with more than one attribute. The next chapter studies the patterns in such a graph. Notice also that several $\tau$-contiguity constraints can be enforced if there are several real-valued attributes.

**Input:** $U, V, S$
**Output:** Every $\tau$-contiguous closed ET-3-clique containing every element in $U$, possibly some elements in $V$, and satisfying $\mathcal{C}$
PURGE_$V^{\text{TIMES}}$
PURGE_$S^{\text{TIMES}}$
**if** $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V)$ **then**
  $U \leftarrow U \sqcup V$
  $V \leftarrow (\emptyset, \ldots, \emptyset)$
**end if**
**if** $\mathcal{C} \wedge \mathcal{C}_{\tau\text{-contiguous}} \wedge \mathcal{C}_{\text{sym-}\delta\text{-closed}}$ may be satisfied by an ET-3-set descending from this node **then**
  Process $E'$ as detailed in Section 4.3
  **if** Case 3.3 was never encountered **then**
    **if** $V = (\emptyset, \emptyset, \emptyset)$ **then**
      **output**$(U)$
    **else**
      **if** Case 3.2 was encountered with an enumeration constraint concluding on $a_{k+1}$ (see Definition 44) **then**
        **if** $a_{k+1}$ is of the form $e \in A$ **then**
          FENSTER$(U \cup \{e\}, \{v \in V \setminus \{e\} \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{v\})\}, \{s \in S \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{s\})\})$
        **else**
          $a_{k+1}$ is of the form $e \notin A$
          FENSTER$(U, V \setminus \{e\}, S \cup \{e\})$
        **end if**
      **else**
        Choose $e \in V$
        FENSTER$(U \cup \{e\}, \{v \in V \setminus \{e\} \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{v\})\}\}, \{s \in S \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{s\})\})$
        FENSTER$(U, V \setminus \{e\}, S \cup \{e\})$
      **end if**
    **end if**
  **end if**
**end if**

Figure 61: FENSTER specialization for $\tau$-contiguous closed ET-3-clique mining.

# 7 MINING THE VÉLO'V USAGE NETWORK

## 1 DATASET

Vélo'v is a bicycle rental service run by the urban community of Lyon, France. Vélo'v stations are spread over Lyon and its nearby. One of them is depicted in Figure 62 [1]. At any of these stations, the users can take a bicycle and return it to any other station. Whenever a bicycle is rented or returned, this event is logged. Our research group obtained parts of these logs (e. g., no user identification to preserve privacy) recorded between May 27th 2005 (when the system was opened to the public) and December 17th 2007. They represent more than 13.1 million rides. The earliest records relate to the users discovering Vélo'v and how useful it may be in their daily mobility. To study the network usage in "normal" conditions, these earliest records were ignored. The chosen date, after which the considered dataset starts, is December 17th 2005. In this way, two full years are kept and aggregations do not favor any part of the year (along which the network usage evolves). Many records stand for rides from a station to itself. These rides usually last a few seconds. They can be mainly explained by users who are not satisfied by the quality of the bicycle they have just rent (e. g., a flat tire) or who have changed their mind (e. g., a bus arrives). Because, from a given station, the most frequent rides are to itself, keeping these records influence a lot any normalization procedure. That is why these records are removed but, after the post-processing steps, the related routes are all claimed frequent, i. e., appended to the relation. A few more records were removed. They relate to abnormal rides (incoherent dates) or rides implying stations that are not opened to the public (e. g., where bicycles are repaired). About 10.2 million records remain after these first steps.

*Vélo'v logs are timestamped rides between 327 stations and during two years.*

*The dataset is cleaned. Rides from/to a same station are removed before normalizing but considered frequent in the end.*

To discover patterns that depend on both the time and the day of the week, the data are aggregated w.r.t. these two scales. More precisely, one directed graph is built per period of time (a one-hour period was chosen) and per day of the week. For instance, one of these graphs presents the rides between nine o'clock and ten o'clock on Mondays. The vertices correspond to the Vélo'v stations. The edges are labeled with the total number of rides from the head vertex (departure station) to the tail vertex (arrival station) during the considered period of time and day of the week. The global activity of the Vélo'v network varies a lot between the different days of the week. For instance, there are 51.3% more rides on Fridays than on Sundays. This difference is even greater between the time periods. For instance, there are about 22 times more rides between 6 pm and 7 pm than between 5 am and 6 am. This global behavior is known. To ignore it, the data are normalized so that the sum of the labels is the same whatever the graph. In this way, when the data are binarized, the Boolean predicate decides whether routes that are frequent w.r.t. the period of time and the day

*The data are normalized such that every pair (day,time period) has the same importance.*

1. © 2005 Frédéric Bonifas (from Wikimedia Commons)
This picture is licensed under the Creative Commons Attribution ShareAlike 3.0 License.

Figure 62: A Vélo'v station.

of the week. The distribution of the rides w.r.t. the stations is far from being constant too. One reason is structural. Some stations can contain/receive many more bicycles than others. Because no bicycle can be rented from an empty station and no bicycle can be returned to a full station, the largest stations imply more rides. Furthermore they are better known by the users (who want to minimize the risk of finding an empty or a full station). Another reason is the progressive installation of the stations. In December 17th 2005, there were 172 stations in activity. In December 17th 2007, they were 315. Because some stations were closed, there are 327 different stations involved in the dataset. Obviously a station that opened little before December 17th 2007 cannot be implied in as many records as another one that has been in activity since the beginning. A local binarization partially handles these differences. The computation of a p-value inspires the details of this technique. It considers the vertices one by one, computes the sum S of the labels of both its incoming and outgoing edges, and claims frequent the routes related to the edges with the greatest values and whose sum is just beyond $0.1 \times S$. By definition, this procedure keeps at least one edge involving each station. In average, 191 edges per station are kept (still excluding the reflexive routes). The resulting 4-ary relations contains 117411 4-tuples (including the reflexive routes, which were previously put to one side), hence a $\frac{117411}{7 \times 24 \times 327 \times 327} = 0.7\%$ density. This relation is named $\mathcal{R}_{\text{Vélo'v}}$. In the following, it is always mined under this conjunction of minimal size constraints:

*The binarization is local, i.e., the relevant routes are frequent for the departure or the arrival station.*

- at least two days of the week;
- at least three time periods;
- at least three departure stations;
- at least three arrival stations.

The results of every experiment in this chapter are obtained on an Intel® processor cadenced at 2.8GHz, 3 Gb of RAM and running a GNU/Linux™ operating system.

| $\epsilon$ | Number of patterns | | Symmetry |
|---|---|---|---|
| | w/o $\mathcal{C}_{\text{symmetric}}$ | with $\mathcal{C}_{\text{symmetric}}$ | w/o $\mathcal{C}_{\text{symmetric}}$ |
| $(0,0,0,0)$ | 13 | 11 | 84.62% |
| $(1,1,1,1)$ | 111 | 63 | 54.05% |
| $(3,2,2,2)$ | 743 | 342 | 41.05% |
| $(4,3,3,3)$ | - | 1163 | - |

Table 10: Number of patterns in $\mathcal{R}_{\text{Vélo'v}}$.

## 2 SYMMETRY BETWEEN DEPARTURE AND ARRIVAL STATIONS

### 2.1 *Avoiding False Positive Noise*

Table 10 lists, with and without the symmetry constraint (see Section 1.2.3 in Chapter 6), the number of patterns (verifying the minimal size constraints) in $\mathcal{R}_{\text{Vélo'v}}$. The parameters for noise tolerance, given in the first column, follow the order $(\epsilon^{\text{day}}, \epsilon^{\text{time}}, \epsilon^{\text{dep}}, \epsilon^{\text{arr}})$. The last column gives the proportion of closed 4-sets that actually are symmetric (when $\mathcal{C}_{\text{symmetric}}$ is not enforced). In other terms, it is the proportion of closed 4-sets that are discovered as well when $\mathcal{C}_{\text{symmetric}}$ is enforced. With a low tolerance to noise, this proportion is very large: the closed 4-sets, in $\mathcal{R}_{\text{Vélo'v}}$, naturally are symmetric. When tolerating more noise, the proportion of symmetric closed 4-sets decreases. Nevertheless:

*The patterns in the Vélo'v usage network naturally are symmetric.*

- it remains much larger than what would be obtained with a random distribution of the frequent routes;
- most of the closed 4-sets remain "almost" symmetric (i. e., most of the departure stations are arrival stations and reciprocally);
- part of the tendency is due to closed 4-sets that false positive noise enlarges (more departure *or* arrival stations).

Given the number of Vélo'v stations (327), the first point is quite obvious. A formal test could be: randomize $\mathcal{R}_{\text{Vélo'v}}$ (see [33] for such a method designed for binary relations), extract the closed 4-sets, compute the proportion of symmetric ones and compare it to the value obtained with the non-randomized version of $\mathcal{R}_{\text{Vélo'v}}$. The "almost" symmetry, the second point mentions, is easy to quantify. E. g., the Jaccard index, $(X^{\text{dep}}, X^{\text{arr}}) \rightarrow \frac{|X^{\text{dep}} \cap X^{\text{arr}}|}{|X^{\text{dep}} \cup X^{\text{arr}}|}$, measures, for every extracted closed 4-set, the similarity between its departure and arrival stations. With $\epsilon = (3,2,2,2)$, its average, over all extracted patterns, is 0.63, hence an "almost" symmetry. The last point will be granted more attention because it relates to a topic this thesis discusses. It states that the proportion of symmetric closed 4-sets decreases because some "naturally" symmetric patterns are extended with additional departure *or* arrival stations that false positive noise affects. Section 1.1.1 in Chapter 2 explains how minimal size constraints prevent this phenomenon. When mining $\mathcal{R}_{\text{Vélo'v}}$, these constraints are quite weak and the positive noise becomes problematic when much noise is tolerated. Strengthening the minimal size constraints is a solution. Enforcing the symmetry constraint is another one. Indeed, in a closed ET-4-clique, every station must be *both* a departure and an arrival station. On the contrary, without $\mathcal{C}_{\text{symmetric}}$, a station can be only a departure (resp. arrival) station. That makes twice less 4-tuples involved in the process

*The symmetry constraints fights against false positive noise.*

| $\epsilon$ | Running time | | Variation |
|---|---|---|---|
| | w/o $\mathcal{C}_{\text{symmetric}}$ | with $\mathcal{C}_{\text{symmetric}}$ | |
| $(0,0,0,0)$ | 3′59s | 3′51s | -3.6% |
| $(1,1,1,1)$ | 1:16′15s | 33′32s | -56.02% |
| $(3,2,2,2)$ | 19:29′06s | 3:09′00s | -79.52% |
| $(4,3,3,3)$ | - | 3:59′29s | - |

Table 11: Running times of FENSTER on $\mathcal{R}_{\text{Vélo'v}}$.

of extending a pattern. That also means twice less 4-tuples the positive noise needs to affect so that the station erroneously extends the pattern.

## 2.2 *Decreasing the Running Times*

*The symmetry constraint significantly decreases the running times.*

Table 11 gives the time it takes FENSTER to extract the patterns Table 10 counts. When much noise is tolerated, $\mathcal{C}_{\text{symmetric}}$ greatly reduces the running times. The effect of $\mathcal{C}_{\text{symmetric}}$ on the false positive noise explains, again, the relatively weak gain observed with little noise tolerance. Indeed, in this setting, the chosen size constraints are enough to fight against positive noise and, even without enforcing $\mathcal{C}_{\text{symmetric}}$, a station usually is declared irrelevant both as a departure and an arrival station. With more noise tolerance, it may be kept for one of these two roles and the size constraints may not prune the search space (or prune it later in the enumeration).

It has been shown that $\mathcal{R}_{\text{Vélo'v}}$ naturally contains patterns that are symmetric w.r.t. the departure and arrival stations. This justifies, in the remaining of this chapter, the enforcement of $\mathcal{C}_{\text{symmetric}}$. This constraint guides the search for the relevant patterns, helps in fighting against positive noise and allows to tolerate quite a lot of noise while remaining tractable.

# 3 EFFECT OF A $\tau$-CONTIGUITY CONSTRAINT

The effect of a $\tau$-contiguity constraint on the time attribute is tested. To do so, a different real value must substitute every time period. However, in $\mathcal{R}_{\text{Vélo'v}}$, the time is cyclic. Replacing every time period by its starting hour (for instance) and enforcing a 1-contiguity would not allow the discovery of a pattern that holds between 22 pm and 1 am because $|22-0| > 1$ and $|23-0| > 1$. Here is a workaround: clone the 4-tuples such that $(x^{\text{day}}, x^{\text{time}}, x^{\text{dep}}, x^{\text{arr}}) \in \mathcal{R} \Leftrightarrow (x^{\text{day}}, x^{\text{time}} + 24, x^{\text{dep}}, x^{\text{arr}}) \in \mathcal{R}_{\text{clone}}$ and mine $\mathcal{R} \cup \mathcal{R}_{\text{clone}}$ with enumeration constraints that specify that whatever the time element $x^{\text{time}}$ in $U$, every time element lower than $x^{\text{time}} - 24$ or greater than $x^{\text{time}} + 24$ must not be in $U$. In this experiment, the time was simply kept acyclic but the "cutting" date was set between the 4am-5am and 5am-6am time periods. This choice is justified by the expected absence of closed 4-sets running across 5 am. Indeed, the users behaves very differently before and after the public transportation services restart (at about 5 am).

The chosen noise tolerance is $(3,2,2,2)$. To test the effect of the $\tau$-contiguity constraint, $\tau$ varies between 0 and 8. Figure 63 gives the number of closed 4-cliques and the times to extract them. Notice that

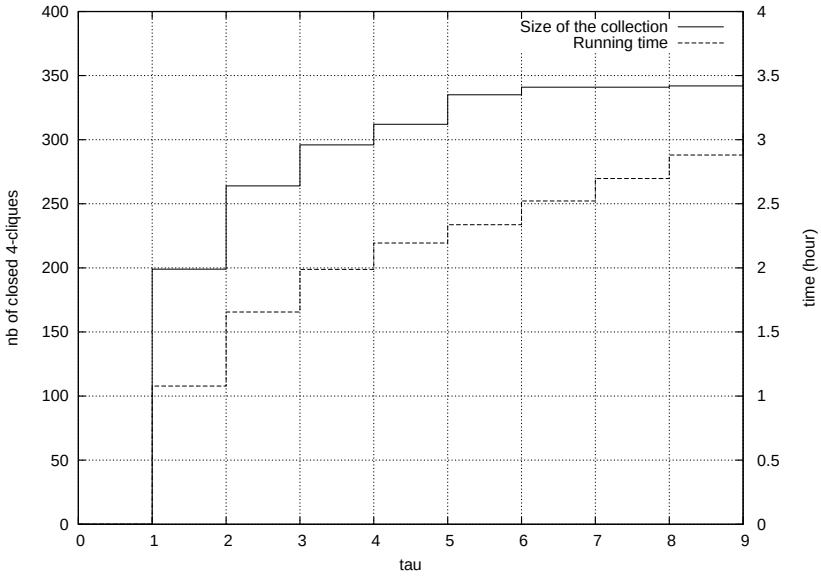



Figure 63: Effect of a $\tau$-contiguity on the number of closed 4-cliques and the time to extract them.

the number of closed 4-cliques satisfying the 8-contiguity constraint is 342, i. e., the same number as without any contiguity constraint (see Table 10). As a consequence, the assumption that there is no closed 4-set running across 5 am is true. Of course, a normal discovery process, under a $\tau$-contiguity constraint, could not ensure that. Notice also that even with $\tau = 8$ (and the same output collection), forcing the closed 4-sets to be almost-contiguous decreases the extraction time of 8.6%. Of course a smaller $\tau$ provides a higher reduction and filters out some poorly relevant closed 4-sets. For example, with a 3-contiguity, the extraction lasts less than two hours. This is to be compared with the three hours that are required without this constraint and the 19 hours and a half with the minimal size constraints only. The 296 3-contiguous closed 4-cliques are chosen for the next step: ALPHA (see Chapter 5).

*The almost-contiguity constraint significantly decreases the running times.*

## 4 AGGLOMERATION, SELECTION AND INTERPRETATION

### 4.1 *Agglomeration and Selection*

Within a few seconds, ALPHA agglomerates the 296 3-contiguous closed 4-cliques in $\mathcal{R}_{\text{Vélo'v}}$. After its selection procedure (see Section 2 in Chapter 5), 125 patterns remain. Notice each of these patterns remains symmetric, i. e., its departure stations and arrival stations are the same. Indeed, an interesting property of the $n$-clique agglomeration is the preservation of $\mathcal{C}_{\text{symmetric}}$. It trivially derives from the definition of this constraint (Definition 41) and that of that of the merging operator (Definition 22).

*Agglomerates of symmetric patterns are symmetric.*

The selection procedure of ALPHA is rather conservative. Indeed, it assumes that "all the initially extracted closed ET-$n$-sets are fragments of some relevant local pattern" (see Section 2.2 in Chapter 5). Because $\mathcal{R}_{\text{Vélo'v}}$ contains many small patterns, the number of patterns is reduced by "only" 58%. This remains a welcome gain and the larger patterns that ALPHA outputs are more relevant than those at its input. To discuss

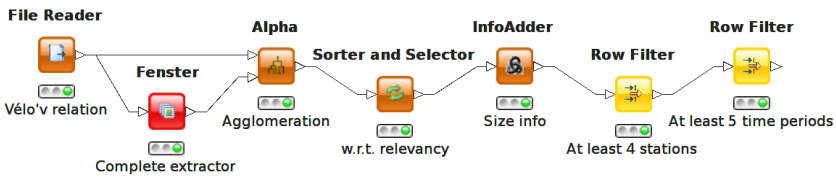



Figure 64: Mining $\mathcal{R}_{\text{Vélo'v}}$.

some of them, the patterns that involve at least four stations and five time periods were selected in a final post-processing step. Figure 64 presents the whole process in the form of a KNIME workflow.

## 4.2 *Seven Patterns*

*The largest discovered patterns make sense.*

The final post-processing step keeps seven patterns. Red dots in Figures 65 to 71 [2] indicate the geographic positions of the stations involved in the patterns. The captions give the related days of the week and time periods. Remember that every pattern stands for frequent rides between every pair of stations and in both directions. The figures are ordered by decreasing relevancy (see Definition 35) of the patterns they depict. In this way, Figure 65 represents the most relevant pattern. It clearly stands for pleasure trips during the week-end afternoons, inside the "Parc de la Tête d'Or" (the main square of Lyon), on the Rhône river side and up to the historical center of the city. In Figure 66, the stations that are involved are among the largest ones in and around the historical center. This pattern holds during the afternoons on Fridays, Saturdays, Sundays and Mondays. Like the most relevant pattern, the one Figure 67 describes clearly relates to pleasure trips during the week-end afternoons. This times the rides takes place more to the south of of the Rhône river side. Figure 68 depicts, gain, pleasure trips during the week-end afternoon. It indicates that the Vélo'v users also like riding in the very touristic "Vieux Lyon" (the oldest district of Lyon) and up to (or from) the closest entrance to the "Parc de la Tête d'Or". On the contrary, the pattern depicted in Figure 69 only stands during the working days. The related rides are earlier in the afternoon too. The involved stations are almost that of second pattern (Figure 66) but the new station is the closest to the universities Lyon 2 and Lyon 3, hence the working days only. The red dots in Figure 70 are more to the east of Lyon, i. e., in the newer part of the city. Three of them follow one of the main axises of the city: the "Cours Gambetta". The outlying station is close to the largest inner-city shopping center in Europe, which is, in average, visited more than 80000 times a day. Its reduced activity on Sundays probably explains that the patterns holds every afternoon but on this day of rest. Finally, Figure 70 represents a pattern that is observable every day from midday to 9 pm. It describes frequent rides between the largest stations in and at the close periphery of the historical center.

2. These figures were created from OpenStreetMap project data.
© 2004-2010 OpenStreetMap contributors
These maps are licensed under Creative Commons Attribution ShareAlike 2.0 License.

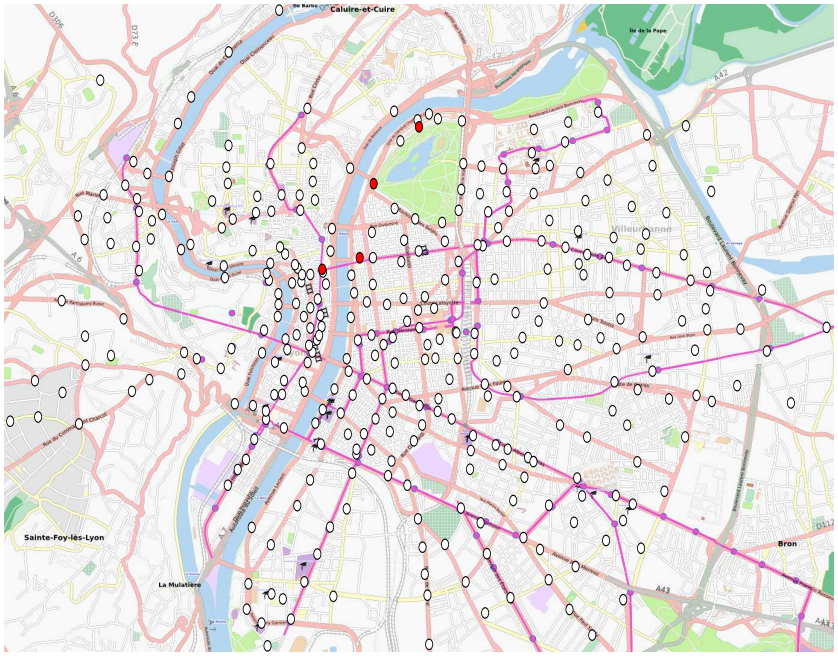



Figure 65: During the week-end from 3 pm to 8 pm.

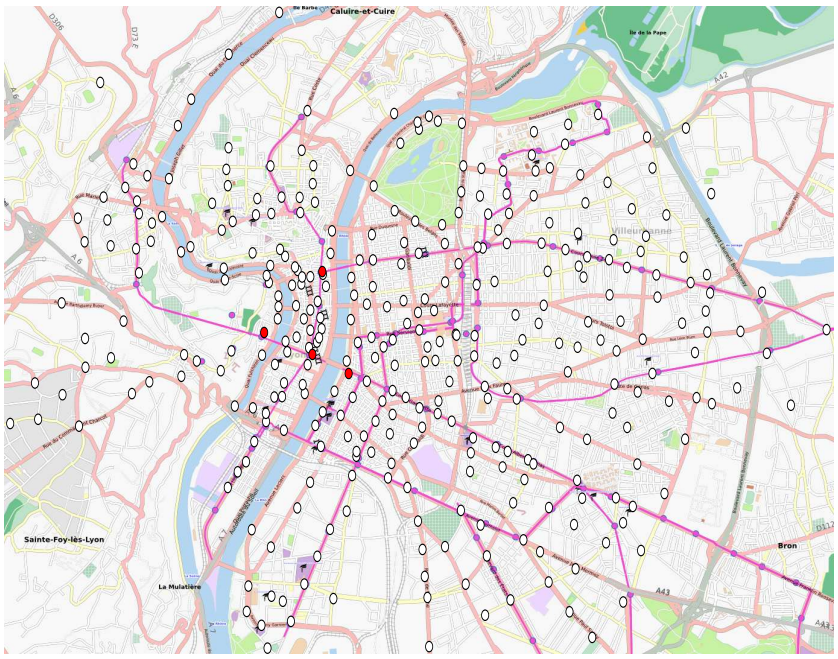



Figure 66: On Fridays, Saturdays, Sundays and Mondays from 3 pm to 8 pm.

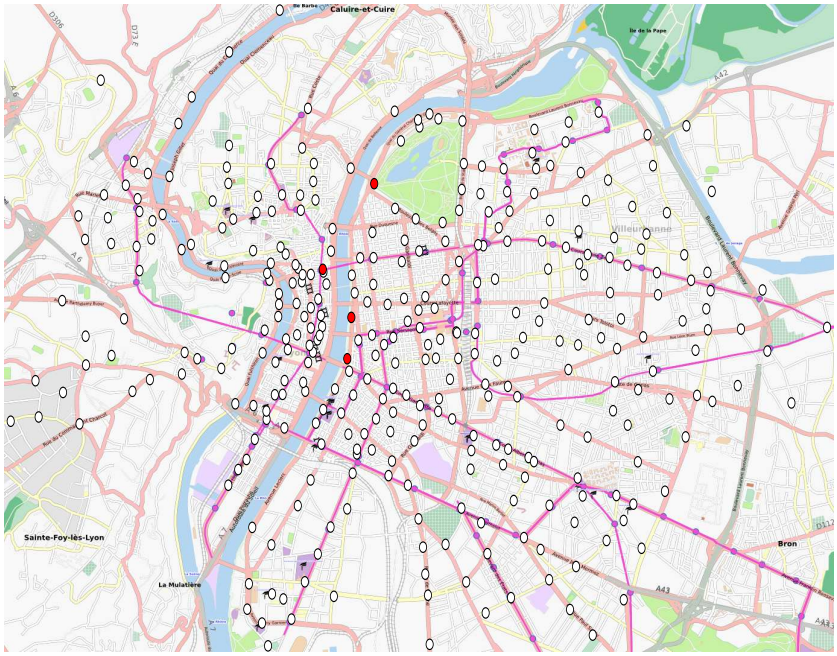



Figure 67: During the week-end from 3 pm to 8 pm.

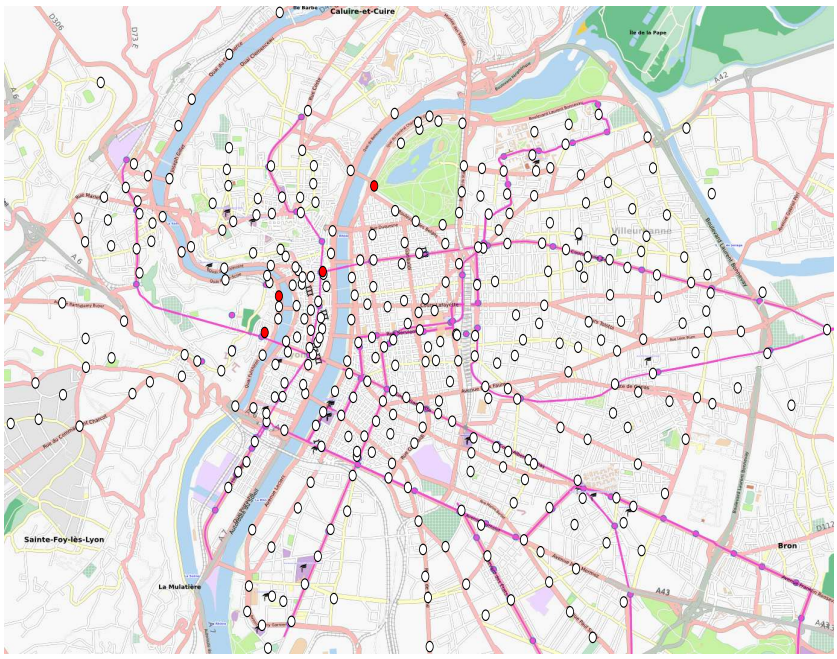



Figure 68: During the week-end from 3 pm to 8 pm.

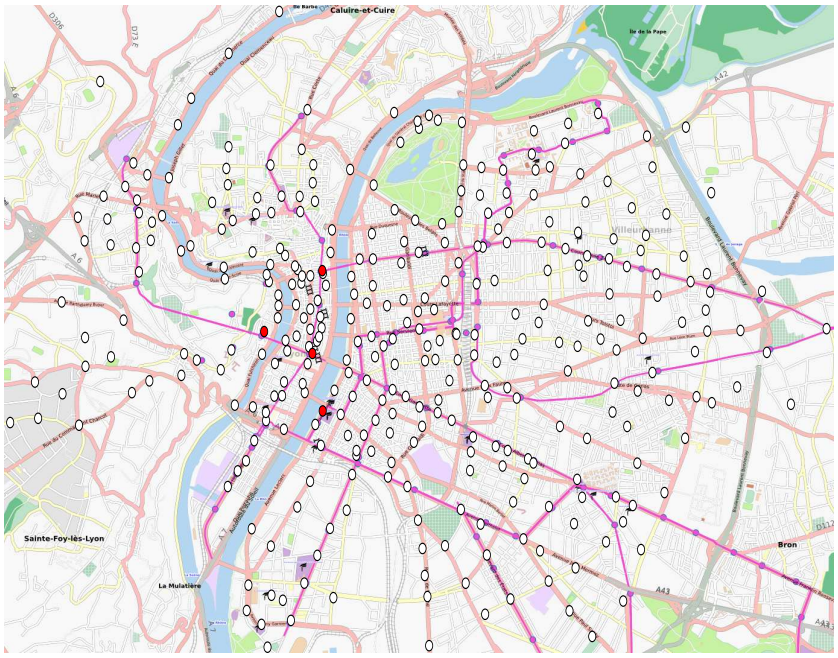



Figure 69: On Mondays, Tuesdays, Wednesdays, Thursdays and Fridays from 12 noon to 5 pm.

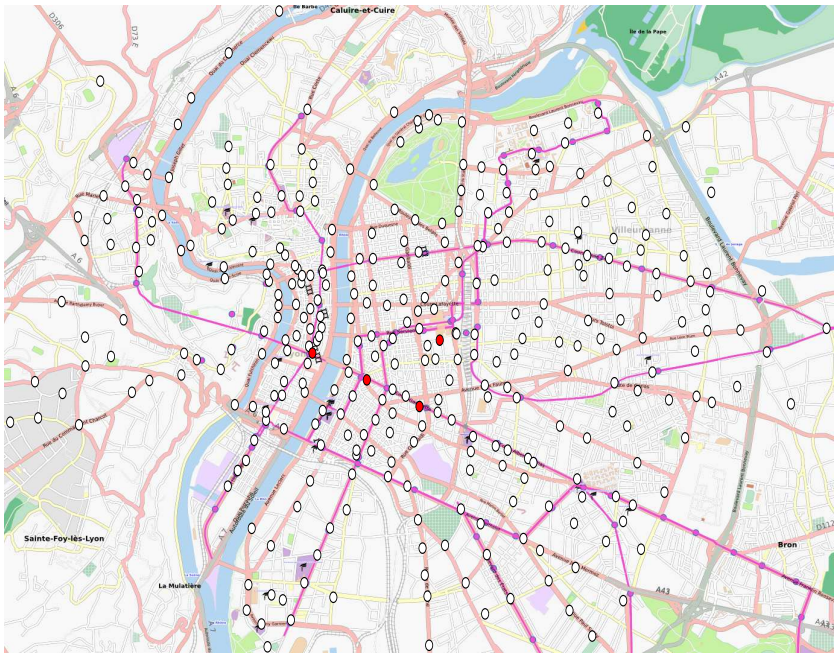



Figure 70: Everyday but Sunday from 3 pm to 8 pm.

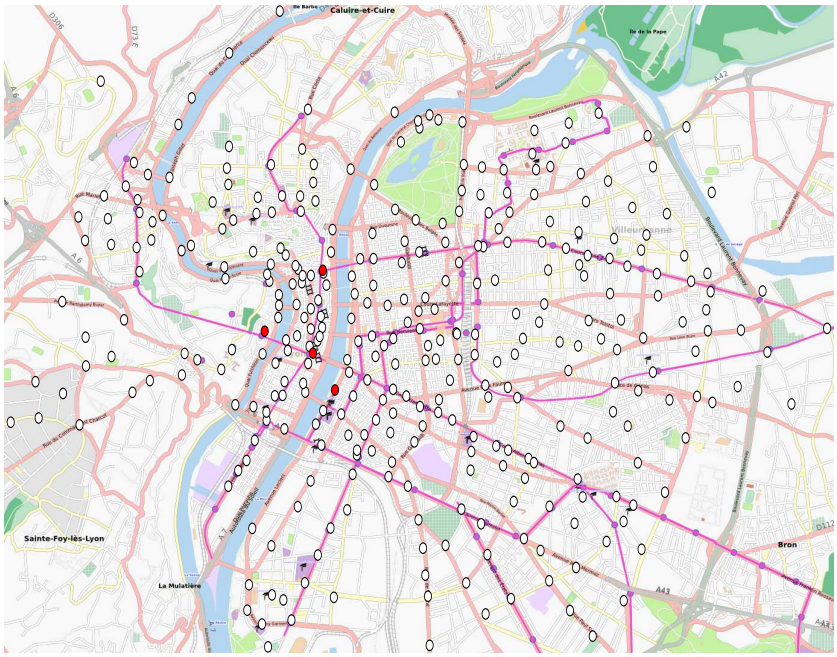



Figure 71: All week long from 12 noon to 9 pm.

## 4.3 *General Observations*

### 4.3.1 *Predominance of Large Stations*

*The largest stations often are in patterns.*

In the computed patterns, the largest stations clearly occur more often than the smaller ones. Several reasons explain their popularity. First of all, they are geographically positioned at key places. E. g., the two stations that occur the most, in the patterns discussed in the previous section, are at intersections of subway lines (for multimodal mobility). As a consequence, many users want to go there. Then, the largest stations are better known. That is why a user who wishes to rent (resp. return) a bicycle usually goes to such a station. Furthermore, even if the user is aware of the presence of a small station in the nearby, he/she may not take the higher risk of finding it empty (resp. full) and directly heads to a farther but larger station. The local binarization (see Section 1) does not favor the largest stations when it comes to deciding the significant rides from/to them. Nevertheless, when the same procedure is applied to the other stations, the routes from/to the large stations around usually are claimed frequent.

### 4.3.2 *Predominance of Day-Time*

*Almost all patterns take place at day-time.*

Almost all closed ET-4-cliques, which FENSTER extracts, take place at day-time. The reason for that is the absence of key places at night. Users rent bicycles to go home and residences are spread all over Lyon and its nearby. That is why, even though the time periods are normalized (see Section 1), frequent route occur at day-time. To a lesser extent, the same phenomenon applies to the morning rides. Indeed, they usually are from the residence to the working place.

### 4.3.3 *Short Trips*

The stations involved in a pattern are, in terms of trip duration, quite close to each other. In particular many patterns have stations aligned on roads that were developed (e. g., the Rhône river side) or redeveloped (e. g., the "Cours Gambetta") taking into consideration the bicycle riders. Therefore, the land settlement obviously plays an essential role in the Vélo'v usage. A query on the raw data constitutes an easy verification of the Vélo'v users preferring short trips. Indeed, every bicycle was equipped with a milometer. Among the rides that were used to obtain $\mathcal{R}_{\text{Vélo'v}}$ (excluding the reflexive routes), the median distance is below two kilometers. That explains why some agglomerations are less relevant that what we could expect. E. g., the patterns depicted in Figure 65 and 67 share the same days of the week, the same time periods and two stations. However their agglomeration would group two stations that are at about four kilometers from each other. For the slower riders (e. g., who stop for a drink on the river side), that may even mean a trip that is not free (Vélo'v is free for rides below half an hour).

*The stations in a pattern often are aligned and quite close to each other.*

### 4.3.4 *A Natural Symmetry?*

The natural symmetry of the patterns in $\mathcal{R}_{\text{Vélo'v}}$ (see Section 2) indicates that, whatever the day of the week or the time period, there exists groups of stations, in which bicycles flow between any pair of stations and in both directions. It may look surprising. The preference for short rides, which has just been mentioned, may be the main reason. Consider, for example, the rides from/to the square (during the weekend afternoons) or the rides in the historical center (in the evening). These short rides are more pleasure trips than daily migrations from/to work/residence. For a specific time period, the former take place in both directions, whereas the latter are one-way. The limited number of bicycles a station can contain/receive also favors the presence of symmetric patterns. Indeed, at every station involved in such patterns, the bicycles are, the same day and during the same time period, both rented and returned. In this way, the stations rarely are empty or full and more bicycles can be rented or returned. In other terms, the Vélo'v is used as much as wanted from/to these stations and both incoming and outgoing edges are claimed frequent in $\mathcal{R}_{\text{Vélo'v}}$. In the opposite, if, a given day and during a given time period, every user wants to rent (resp. return) a bicycle at a given station, this station is soon empty (resp. full), the flow stops and the number of rented (resp. returned) bicycles, limited by the number of bicycles the station can contain (resp. receive), may not be great enough to claim the related outgoing (resp. ingoing) edges frequent in $\mathcal{R}_{\text{Vélo'v}}$. To study the *desired* (rather than *actual*) Vélo'v mobility, the number of rides from (resp. to) a station, a given day and during a given time period, could be divided by the total time this station was not empty (resp. full). Unfortunately this information was not logged.

*The natural symmetry of the patterns is partly explained by the limited capacity of every station.*

### 4.3.5 *A Natural Contiguity*

The same arguments as those given in favor of a natural symmetry (see Section 2) could have been used to claim, right after the complete extraction, a natural contiguity of the patterns in $\mathcal{R}_{\text{Vélo'v}}$. However, this

*The patterns in the Vélo'v usage network naturally are contiguous.*

fact is even more obvious after ALPHA proceeds. All seven patterns, presented in Section 4.2, are 1-contiguous. Without the final post-processing step, 74% of the 125 patterns, at ALPHA's outputs, are 1-contiguous (against 67% at its input). When it is expected, beforehand, that the patterns are naturally contiguous, the $\tau$-contiguity, with a higher $\tau$ than what a hidden pattern should tolerate, can be seen as another way to tolerate false negative noise. By agglomeration, ALPHA may, then, recover the hidden, and "more contiguous", patterns.

## 5 CONCLUSION

The methods, presented in this thesis, support the discovery of relevant patterns. They remain tractable even in a difficult context (four attributes, more than 100000 tuples, weak minimal size constraints, etc.), which no other complete extractor can handle. The symmetry and the almost-contiguity constraints significantly reduce the running times without much limiting the discovery of unexpected patterns. Indeed, many datasets, in the manner of $\mathcal{R}_{\text{Vélo'v}}$, naturally contains symmetric and/or contiguous patterns and enforcing the related constraints simply guides the search towards those patterns.

# Part VI

# CONCLUSION

## FROM CLOSED ITEMSETS TO CLOSED ET-$n$-SETS

### *Summary*

The broad applicability of closed itemset mining is often praised. Indeed, binary relations can represent whether customers buy some products (and an itemset is a group of customers buying a same subset of products) as well as whether genes are over-expressed in different biological samples (and an itemset is a synexpression group, i. e., genes that are involved together in some biological processes). The complete extractors presented in this thesis further extend the scope of itemset mining. Generalizing it towards $n$-ary relation ($n \geqslant 2$) makes it possible to take into consideration $n$ orthogonal dimensions of analysis altogether. It enables, for example, a localized analysis of buying behaviors (ternary relations binding customers, products and places) or a kinetic analysis of gene expressions (ternary relation binding genes, biological samples and timestamps). To list every closed itemset in a binary relation, the state-of-the-art extractors enumerate subsets of properties and derive the supporting subsets of objects. This is possible because the subsets of the two attribute domains (partially ordered by $\subseteq$) form a Galois connection. Closed patterns in $n$-ary relations, i. e., closed $n$-sets, do not have this convenient property. That is why DATA-PEELER relies on original enumeration principles that do not favor any attribute. When enlarging the current candidate pattern, the freedom to choose any element in any attribute domain allows to make a choice that heuristically maximizes search space pruning. Together with other procedural innovations (e. g., to fasten the enforcement of the closedness constraint), this original enumeration strategy explains the excellent performance of DATA-PEELER. Indeed, all experiments show it runs orders of magnitude faster than its competitors, which were specifically designed for closed 3-set mining.

Generalizing DATA-PEELER towards noise tolerance is fighting against a plague that affects most datasets. Indeed, by simply defining noise as an unwanted perturbation of the data, many phenomena are sources of noise. E. g., sold out products may have an undesired effect on transactional data that are aimed at understanding buying behaviors on a grand scale. Genetic datasets are, somehow, even worse because they often represent intrinsically stochastic processes. The perfectibility of knowledge discovery processes is source of noise too. In particular, when relations are derived from numerical datasets there is a cumbersome need to fix thresholds beneath/beyond which a Boolean property is claimed satisfied. State-of-the-art approaches, which compute noise tolerant itemsets (aka ET-itemsets), show a wide range of definitions. Indeed, the mere declarative specification of noise tolerance raises discussions. FENSTER tolerates, in every hyper-plan of a pattern, an upper-bounded number of $n$-tuples that are absent from the relation. The choice of upper-bounds for *every element in every attribute domain* avoids matching patterns in which some elements are much disconnected. The choice of an *absolute* tolerance to noise allows a closedness constraint to restrict the output pattern collection to a lossless condensation of all ET-$n$-sets. Furthermore, from a procedural point of view, absolute parameters enable far more search space pruning than relative ones. Although DATA-PEELER's enumeration principles are largely reused, FENSTER's time performance fundamentally

depends on efficient enforcements of the generalized constraints beneath. That is why an incremental computation of the quantity of noise, which candidate patterns tolerate, is implemented. Nevertheless, it usually remains intractable to tolerate enough noise so that FENSTER recovers the *real* patterns. That is ALPHA's *raison d'être*. It heuristically complements FENSTER by hierarchically agglomerating the fragments a complete extraction returns. Then, among the agglomerated patterns, the most relevant ones are selected. They are those covering the seminal collection of closed ET-$n$-sets and showing the best trade-offs between a small proportion of (supposedly) noise inside them and a great distance to the outside patterns.

*Perspective*

Applications often rely on numerical data. Unfortunately, FENSTER and ALPHA only work on $n$-ary relations. A significant source of noise directly relates to the pre-process converting real numbers into the satisfaction or the violation of the encoded Boolean property. Indeed, by definition, Boolean properties are "all or nothing" (the tuple is in the relation or not), i. e., any numerical values $x \in \mathbb{R}$ must go through an Heaviside step function comparing it with a threshold $\alpha \in \mathbb{R}$:

$$\forall \alpha \in \mathbb{R}, x \rightarrow \begin{cases} 0 \text{ if } x < \alpha \\ 1 \text{ if } x \geqslant \alpha \end{cases} .$$

Because of the discontinuity at $\alpha$, this conversion numerical/Boolean is prone to errors. The Heaviside step function is depicted in Figure 72 along with two logistic functions. The logistic functions are smooth approximation of the Heaviside step function, i. e., the "threshold effect" is avoided ($k \in \mathbb{R}$ controls how sharp the transition around $\alpha$):

$$\forall (\alpha, k) \in \mathbb{R}^2, x \rightarrow \frac{1}{1 + e^{k(\alpha - x)}} .$$

Using such a function would return to what extent, a property is satisfied, i. e., the related $n$-tuples would be "member" of the relation to a certain degree $m \in [0, 1]$. Such a dataset is said *probabilistic*. Extending the scope of FENSTER and ALPHA to probabilistic datasets is a timely challenge. Sums of $1 - m$ values would be used to quantify the false negative noise inside a pattern. In this way, the connection constraint remains anti-monotone w.r.t. each of its variables, the closedness constraint still provides a lossless condensation of the patterns, and a similar performance is expected despite the generalization.

### FROM CLOSED ET-$n$-SETS TO SPECIFIC PATTERNS

*Summary*

In front of an applicative problem, no data mining algorithm can, blindly, take the data as input and directly return *actionable patterns*, i. e., patterns that directly translate to actions solving the problem. Instead, the extraction of such patterns requires whole knowledge discovery processes that are specifically designed for the considered applications. This thesis has presented a pre-processing which takes advantage of

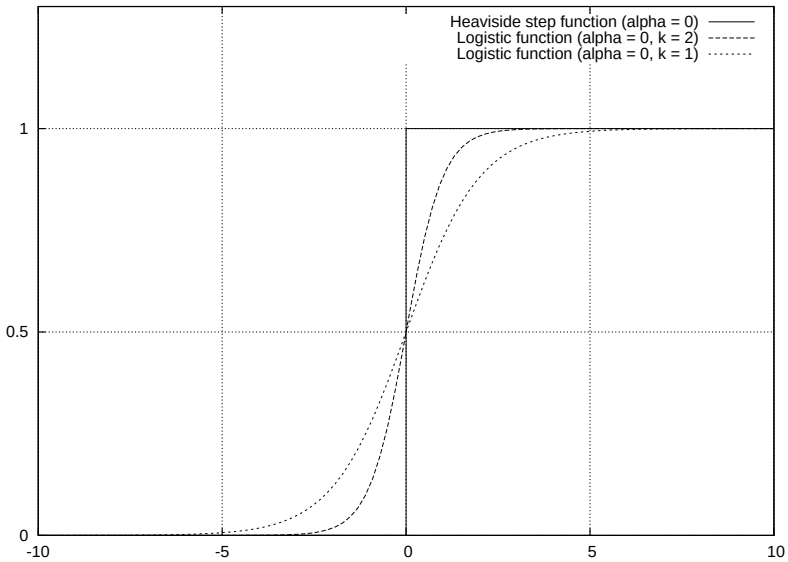



Figure 72: The Heaviside step function and two logistic functions.

an additional analysis dimension so that DATA-PEELER/FENSTER mines patterns that are robust w.r.t. binarization. It also has shown that post-processing the closed $n$-sets is a way to efficiently minimize multi-valued logic functions. The obtained compression rates are even better than state-of-the-art approaches focusing on this problem. Nevertheless, in the quest for actionable patterns, the most powerful leverage is the very expressive constraints DATA-PEELER/FENSTER efficiently handles. Not only the piecewise (anti)-monotone constraints can finely outline the relevant closed ET-$n$-sets but they also lower the running times. E. g., listing every closed ET-4-set formed of frequent Vélo'v routes is about ten times faster if, in addition to minimal size constraints, two piecewise (anti)-monotone constraints are enforced. These constraints, namely the symmetry and the almost-contiguity constraints, specialize FENSTER. They translate the relevancy of a pattern in a dynamic graph (rather than in any $n$-ary relation). More generally, the background knowledge of the applicative context can come into the picture as far as it can be expressed as piecewise (anti)-monotone constraints. Because this class of constraints is very broad, it can be written that FENSTER both generalizes itemset mining (see the previous section) and makes it applicable to more specific problems. In fact, the class of constraint is so broad that there is no need to distinguish the relevancy constraints from the definition of a basal pattern. Indeed, this definition is a conjunction of two constraints that were proved piecewise (anti)-monotone.

*Perspectives*

Dynamic graphs being $n$-ary relations, FENSTER was specialized to handle this interesting particular case. In the same vein, additional piecewise (anti)-monotone constraints could be designed to specifically mine trees, sequences, strings, etc. Like with the symmetry and the almost-contiguity constraints, ad-hoc implementations may provide better performance than the ones directly deriving from the proofs of piecewise (anti)-monotonicity. Anyway, as far as the basic enumera-

tion principles are left unchanged it can be written that the resulting extractors specialize FENSTER. Future interactions with experts in specific applicative domains (e. g., in genetics) may lead the design of specific knowledge discovery processes. Part of the background knowledge would probably be translated into piecewise (anti)-monotone constraints and, hopefully, FENSTER and ALPHA would help the discovery of new pieces of knowledge. It may also be useful to pre or post-process FENSTER in new ways. Ongoing developments [NCB10] deal with generalizing association rules and extracting them from the closed $n$-sets DATA-PEELER computes. In this attempt, the hardest issue lies in the mere definition of a descriptive semantics for generalized association rules. More precisely, if the consequent of a rule contains elements from attributes that the antecedent does not involve, the definition of a confidence measure is not clear.

# Part VII

# BIBLIOGRAPHY

The bibliography directly related to the work presented in this thesis is listed on page vii.


[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994. (Cited on pages 15 and 16.)

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. Technical report, IBM Research Division, Almaden Reasearch Center, 650 Harry Road, San Jose, CA 95120-6099, 1995. Introduction of the Quest data generator. (Cited on page 67.)

[3] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1–3):21–46, 1996. (Cited on page 26.)

[4] Sir Francis Bacon. *Instauratio Magna*, chapter Novum Organum - Liber Primus - APHORISMUS LI. 1620. (Cited on page 3.)

[5] Jérémy Besson, Céline Robardet, Jean-François Boulicaut, and Sophie Rome. Constraint-based formal concept mining and its application to microarray data analysis. *Intelligent Data Analysis*, 9(1):59–82, 2005. (Cited on page 46.)

[6] Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *ICCS '06: Proceedings of the 14th International Conference on Conceptual Structures*, pages 144–157. Springer, 2006. (Cited on page 42.)

[7] Jérémy Besson, Céline Robardet, and Jean-Francois Boulicaut. Un algorithme générique d'extraction de bi-ensembles sous contraintes dans des données booléennes. *Information - Interaction - Intelligence*, Hors Série:141–160, 2007. (Cited on page 15.)

[8] Sylvain Blachon, Ruggero G. Pensa, Jérémy Besson, Céline Robardet, Jean-Francois Boulicaut, and Olivier Gandrillon. Clustering formal concepts to discover biologically relevant knowledge from gene expression data. *In Silico Biology*, 7(0033):1–15, 2007. (Cited on page 43.)

[9] Mario Boley, Tamás Horváth, and Stefan Wrobel. Efficient discovery of interesting patterns based on strong closedness. In *SDM '09: Proceedings of the 9th SIAM International Conference on Data Mining*, page 2009. SIAM, 1002–1013. (Cited on page 35.)

[10] Francesco Bonchi. *Frequent Pattern Queries: Language and Optimizations*. PhD thesis, Dipartimento di Informatica, Università di Pisa, December 2003. (Cited on page 36.)

[11] Francesco Bonchi and Claudio Lucchese. On closed constrained frequent pattern mining. In *ICDM '04: Proceedings of the 4th IEEE International Conference on Data Mining*, pages 35–42. IEEE Computer Society, 2004. (Cited on page 34.)

[12] Francesco Bonchi and Claudio Lucchese. Pushing tougher constraints in frequent pattern mining. In *PAKDD '05: Proceedings*

*of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 114–124. Springer, 2005. (Cited on page 24.)

[13] Jean-François Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In *PAKDD '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 62–73. Springer, 2000. (Cited on page 14.)

[14] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984. (Cited on pages 47 and 75.)

[15] Robert Bringhurst. *The Elements of Typographic Style*. Version 2.5. Hartley & Marks, Publishers, 2002. (Cited on page 169.)

[16] Cristian Bucila, Johannes Gehrke, Daniel Kifer, and Walker M. White. DualMiner: a dual-pruning algorithm for itemsets with constraints. In *KDD '02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 42–51. ACM Press, 2002. (Cited on page 22.)

[17] Gemma Casas-Garriga. Discovering unbounded episodes in sequential data. In *PKDD '03: Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 83–94. Springer, 2003. (Cited on page 127.)

[18] Gemma Casas-Garriga, Roni Khardon, and Luc De Raedt. On mining closed sets in multi-relational data. In *IJCAI '07: Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 804–809. AAAI Press, 2007. (Cited on page 46.)

[19] Chair for Bioinformatics and Information Mining at the University of Konstanz, Germany. KNIME, 2004–2010. http://www.knime.org. (Cited on page 113.)

[20] Michel Chein. Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin Mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie (Nouvelle Série)*, 13(61)(1):21–25, 1969. (Cited on page 14.)

[21] Hong Cheng, Philip S. Yu, and Jiawei Han. AC-Close: Efficiently mining approximate closed itemsets by core pattern recovery. In *ICDM '06: Proceedings of the 6th IEEE International Conference on Data Mining*, pages 839–844. IEEE Computer Society, 2006. (Cited on pages 42, 94, and 99.)

[22] James Cheng, Yiping Ke, and Wilfred Ng. $\delta$-tolerance closed frequent itemsets. In *ICDM '06: Proceedings of the 6th IEEE International Conference on Data Mining*, pages 139–148. IEEE Computer Society, 2006. (Cited on page 35.)

[23] Bolin Ding, David Lo, Jiawei Han, and Siau-Cheng Khoo. Efficient mining of closed repetitive gapped subsequences from a sequence database. In *ICDE '09: Proceedings of the 25th International Conference on Data Engineering*, pages 1024–1035. IEEE Computer Society, 2009. (Cited on page 127.)

[24] Elena Dubrova. Multiple-valued logic in VLSI: Challenges and opportunities. In *NORCHIP '99: Proceedings of the 17th IEEE Nordic Microelectronics event*, pages 340–350. IEEE Computer Society, 1999. (Cited on page 47.)

[25] Arianna Gallo, Tijl De Bie, and Nello Cristianini. MINI: Mining informative non-redundant itemsets. In *PKDD '07: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 438–445. Springer, 2007. (Cited on page 14.)

[26] Arianna Gallo, Alessia Mammone, Tijl De Bie, Marco Turchi, and Nello Cristianini. From frequent itemsets to informative patterns. Technical Report 123936, University of Bristol, Senate House, Tyndall Avenue, Bristol BS8 1TH, UK, December 2009. (Cited on page 14.)

[27] Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors. *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*, 2005. Springer. (Cited on page 15.)

[28] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. CACTUS—clustering categorical data using summaries. In *KDD '99: Proceedings of the 5th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 73–83. ACM Press, 1999. (Cited on page 49.)

[29] Minxi Gao, Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Subarna Sinha, and Robert Brayton. MVSIS. In *Notes of the IEEE International Workshop on Logic Synthesis*. IEEE Computer Society, 2000. (Cited on page 75.)

[30] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In *DS '04: Proceedings of the 7th International Conference Discovery Science*, pages 278–289. Springer, 2004. (Cited on page 47.)

[31] Elisabeth Georgii, Koji Tsuda, and Bernhard Schölkopf. Multi-way set enumeration in real-valued tensors. In *DMMT '09: Proceedings of the 2nd ACM SIGKDD Workshop on Data Mining using Matrices and Tensors*, pages 32–41. ACM Press, 2009. (Cited on page 48.)

[32] Aristides Gionis, Heikki Mannila, and Jouni K. Seppänen. Geometric and combinatorial tiles in 0-1 data. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 173–184. Springer, 2004. (Cited on page 112.)

[33] Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data*, 1(3), 2007. (Cited on page 141.)

[34] Gösta Grahne, Laks V. S. Lakshmanan, and Xiaohong Wang. Efficient mining of constrained correlated sets. In *ICDE '00: Proceedings of the 16th International Conference on Data Engineering*, pages 512–521. IEEE Computer Society, 2000. (Cited on page 23.)

[35] Peter D. Grünwald. *The Minimum Description Length Principle*. Adaptive Computation and Machine Learning. MIT Press, 2007. (Cited on page 73.)

[36] Gunjan K. Gupta, Alexander Strehl, and Joydeep Ghosh. Distance based clustering of association rules. In *ANNIE '99: Proceedings of the 9th Intelligent Engineering Systems Through Artificial Neural Networks*, pages 759–764. ASME Press, 1999. (Cited on page 43.)

[37] Rohit Gupta, Gang Fang, Blayne Field, Michael Steinbach, and Vipin Kumar. Quantitative evaluation of approximate frequent pattern mining algorithms. In *KDD '08: Proceedings of the 14th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 301–309. ACM Press, 2008. (Cited on page 41.)

[38] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004. (Cited on pages 15 and 66.)

[39] David J. Hand. Pattern detection and discovery. In *Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery*, pages 1–12. Springer, 2002. (Cited on pages 34, 100, and 107.)

[40] Robert Jaschke, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Gerd Stumme. TRIAS–an algorithm for mining iceberg tri-lattices. In *ICDM '06: Proceedings of the 6th IEEE International Conference on Data Mining*, pages 907–911. IEEE Computer Society, 2006. (Cited on pages 47, 68, 94, and 97.)

[41] Liping Ji, Kian-Lee Tan, and Anthony K. H. Tung. Mining frequent closed cubes in 3D data sets. In *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 811–822. VLDB Endowment, 2006. (Cited on pages 46, 68, 94, and 97.)

[42] Daxin Jiang and Jian Pei. Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data*, 2(4): 1–42, 2009. (Cited on page 126.)

[43] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 85–93. ACM Press, 1998. (Cited on page 24.)

[44] Maurice Karnaugh. The map method for synthesis of combinational logic circuits. *Transactions of American Institute of Electrical Engineers part I*, 72(9):593–599, 1953. (Cited on page 47.)

[45] Arno Knobbe, Bruno Crémilleux, Johannes Fürnkranz, and Martin Scholz. From local patterns to global models: The LeGo approach to data mining. In *LeGo '08: Proceedings of the International Workshop From Local Patterns to Global Models*, pages 1–16. Springer, 2008. (Cited on page 73.)

[46] Jia-Ling Koh and Pei-Wy Yo. An efficient approach for mining fault-tolerant frequent patterns based on bit vector representations. In *DASFAA '05: Proceedings of the 10th International Conference on Database Systems for Advanced Applications*, pages 568–575. Springer, 2005. (Cited on page 42.)

[47] Sergei O. Kuznetsov. On stability of a formal concept. *Annals of Mathematics and Artificial Intelligence*, 49(1–4):101–115, 2007. (Cited on page 35.)

[48] Sau Dan Lee. *Constrained mining of patterns in large databases*. PhD thesis, Institut für Informatik, Alber-Ludwig-Universität Freiburg, February 2006. (Cited on page 36.)

[49] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007. (Cited on page 123.)

[50] Jinyan Li, Haiquan Li, Donny Soh, and Limsoon Wong. A correspondence between maximal complete bipartite subgraphs and closed patterns. In *PKDD '05: Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 146–156. Springer, 2005. (Cited on page 123.)

[51] Guimei Liu and Limsoon Wong. Effective pruning techniques for mining quasi-cliques. In *ECML PKDD '08: Proceedings of the 12th European Conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 33–49. Springer, 2008. (Cited on page 126.)

[52] Jinze Liu, Susan Paulsen, Xing Sun, Wei Wang, Andrew B. Nobel, and Jan Prins. Mining approximate frequent itemsets in the presence of noise: Algorithm and analysis. In *SDM '06: Proceedings of the 6th SIAM International Conference on Data Mining*, pages 405–416. SIAM, 2006. (Cited on pages 41, 99, and 108.)

[53] Heikki Mannila and Hannu Toivonen. Multiple uses of frequent sets and condensed representations. In *KDD '96: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 189–194. AAAI Press, 1996. (Cited on page 14.)

[54] Edward J. McCluskey. Minimization of boolean functions. *Bell System Technical Journal*, 35(5):1417–1444, 1956. (Cited on page 47.)

[55] Ieva Mitašiūnaitė and Jean-François Boulicaut. Looking for monotonicity properties of a similarity constraint on sequences. In *SAC '06: Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 546–552. ACM Press, 2006. (Cited on page 22.)

[56] Basil Montagu, editor. *The Works of Francis Bacon, Lord Chancellor of England*, chapter Novum Organum - Book I - APHORISM 51. Pickering, 1831. Translation by William Wood. (Cited on page 3.)

[57] Nicolas Méger and Christophe Rigotti. Constraint-based mining of episode rules and optimal window sizes. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 313–324. Springer, 2004. (Cited on page 127.)

[58] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained association rules. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 13–24. ACM Press, 1998. (Cited on page 23.)

[59] Feng Pan, Gao Cong, Anthony K.H. Tung, Joing Yang, and Mohammed J. Zaki. CARPENTER: Finding closed patterns in long biological datasets. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 637–642. ACM Press, 2003. (Cited on pages 15 and 71.)

[60] Feng Pan, Anthony K. H. Tung, Gao Cong, and Xin Xu. COBBLER: Combining column and row enumeration for closed pattern discovery. In *SSDBM '04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 21–30. IEEE Computer Society, 2004. (Cited on page 19.)

[61] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999. (Cited on pages 15 and 16.)

[62] Jian Pei and Jiawei Han. Can we push more constraints into frequent pattern mining? In *KDD '00: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 350–354. ACM Press, 2000. (Cited on page 23.)

[63] Jian Pei, Anthony K. H. Tung, and Jiawei Han. Fault-tolerant frequent pattern mining: Problems and challenges. In *DMKD '01: Proceedings of the 6th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. ACM Press, 2001. (Cited on page 42.)

[64] Ruggero G. Pensa and Jean-François Boulicaut. *Local Pattern Detection*, volume 3539/2005, chapter Boolean Property Encoding for Local Set Pattern Discovery: An Application to Gene Expression Data Analysis, pages 115–134. Springer, 2005. (Cited on page 71.)

[65] Wim Pijls and Jan C. Bioch. Mining frequent itemsets in memory-resident databases. In *BNAIC '99: Proceedings of the 11th Belgium-Netherlands Conference on Artificial Intelligence*, pages 75–82. Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten, 1999. (Cited on page 16.)

[66] Ardian K. Poernomo and Vivekanand Gopalkrishnan. Mining statistical information of frequent fault-tolerant patterns in transactional databases. In *ICDM '07: Proceedings of the 7th IEEE International Conference on Data Mining*, pages 272–281. IEEE Computer Society, 2007. (Cited on page 42.)

[67] Ardian K. Poernomo and Vivekanand Gopalkrishnan. Efficient computation of partial-support for mining interesting itemsets. In *SDM '09: Proceedings of the 9th SIAM International Conference on Data Mining*, pages 1014–1025. SIAM, 2009. (Cited on page 42.)

[68] Ardian K. Poernomo and Vivekanand Gopalkrishnan. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 697–706. ACM Press, 2009. (Cited on page 41.)

[69] Luc De Raedt and Jan Ramon. Condensed representations for inductive logic programming. In *KR '04: Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning*, pages 438–446. AAAI Press, 2004. (Cited on page 46.)

[70] Luc De Raedt, Manfred Jaeger, Sau Dan Lee, and Heikki Mannila. A theory of inductive query answering. In *ICDM '02: Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 123–130. IEEE Computer Society, 2002. (Cited on page 22.)

[71] François Rioult, Jean-François Boulicaut, Bruno Crémilleux, and Jérémy Besson. Using transposition for pattern discovery from microarray data. In *DMKD '03: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 73–79. ACM Press, 2003. (Cited on page 15.)

[72] Jorma Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978. (Cited on page 73.)

[73] Richard Rudell and Alberto Sangiovanni-Vincentelli. Espresso-MV: Algorithms for multiple valued logic minimization. In *Proceedings of the 1985 IEEE Custom International Circuit Conference*, pages 230–234. IEEE Computer Society, 1985. (Cited on pages 47 and 75.)

[74] Jouni K. Seppänen and Heikki Mannila. Dense itemsets. In *KDD '04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 683–688. ACM Press, 2004. (Cited on page 41.)

[75] Arno Siebes, Jilles Vreeken, and Matthijs van Leeuwen. Item sets that compress. In *SDM '06: Proceedings of the 6th SIAM International Conference on Data Mining*, pages 393–404. SIAM, 2006. (Cited on page 48.)

[76] James R. Slagle, Chin Liang Chang, and R. C. T. Lee. A new algorithm for generating prime implicants. *IEEE Transactions on Computers*, 19(4):304–310, 1970. (Cited on page 24.)

[77] Arnaud Soulet. *Un cadre générique de découverte de motifs sous contraintes fondées sur des primitives*. PhD thesis, Université de Caen, November 2006. (Cited on page 36.)

[78] Arnaud Soulet and Bruno Crémilleux. An efficient framework for mining flexible constraints. In *PAKDD '05: Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 661–671. Springer, 2005. (Cited on page 27.)

[79] Arnaud Soulet and Bruno Crémilleux. Exploiting virtual patterns for automatically pruning the search space. In *Knowledge Discovery in Inductive Databases, 4th International Workshop KDID '05, Revised Selected and Invited Papers*, pages 202–221. Springer, 2005. (Cited on page 28.)

[80] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT '96: Proceedings of the 5th International Conference on Extending Database Technology*, pages 3–17. Springer, 1996. (Cited on page 127.)

[81] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with TITANIC. *Data & Knowledge Engineering*, 42(2):189–222, 2002. (Cited on page 15.)

[82] Xing Sun and Andrew B. Nobel. Significance and recovery of block structures in binary matrices with noise. In *COLT '06: Proceedings of the 19th Annual Conference on Learning Theory*, pages 109–122. Springer, 2006. (Cited on page 37.)

[83] Hannu Toivonen. *Discovery of frequent patterns in large data collections*. PhD thesis, Tietojenkäsittelytieteen laitos, Helsingin Yliopisto, November 1996. (Cited on page 36.)

[84] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen, Kimmo Hätönen, and Heikki Mannila. Pruning and grouping discovered association rules. In *Proceedings of the ECML '95 Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, pages 47–52, 1995. (Cited on page 43.)

[85] Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes. Compression picks item sets that matter. In *PKDD '06: Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 585–592. Springer, 2006. (Cited on page 48.)

[86] Matthijs van Leeuwen, Francesco Bonchi, Börkur Sigurbjörnsson, and Arno Siebes. Compressing tags to find interesting media groups. In *CIKM '09: Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1147–1156. ACM Press, 2009. (Cited on page 48.)

[87] Jilles Vreeken and Arno Siebes. Filling in the blanks — krimp minimisation for missing data. In *ICDM '08: Proceedings of the 8th IEEE International Conference on Data Mining*, pages 1067–1072. IEEE Computer Society, 2008. (Cited on page 48.)

[88] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. Characterising the difference. In *KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 765–774. ACM Press, 2007. (Cited on page 48.)

[89] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. Preserving privacy through data generation. In *ICDM '07: Proceedings of the 7th IEEE International Conference on Data Mining*, pages 685–690. IEEE Computer Society, 2007. (Cited on page 48.)

[90] Jianyong Wang, Zhiping Zeng, and Lizhu Zhou. CLAN: An algorithm for mining closed cliques from large dense graph databases. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, pages 73–82. IEEE Computer Society, 2006. (Cited on page 126.)

[91] Rudolf Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered Sets*, pages 445–470. D. Reidel Publishing Company, 1982. (Cited on page 15.)

[92] Andrew K. C. Wong and Gary C. L. Li. Simultaneous pattern and data clustering for pattern cluster analysis. *IEEE Transations on Knowledge and Data Engineering*, 20(7):911–923, 2008. (Cited on page 43.)

[93] Cheng Yang, Usama Fayyad, and Paul S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. Technical Report 2000-20, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, February 2000. (Cited on pages 38 and 41.)

[94] Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD '04: Proceedings of the 10th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 344–353. ACM Press, 2004. (Cited on pages 20 and 104.)

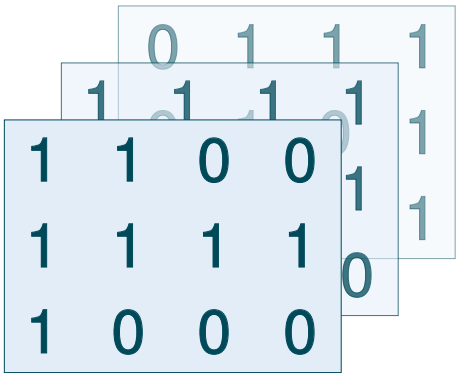
[95] Mohammed J. Zaki and Ching-Jui Hsiao. CHARM: An efficient algorithm for closed association rule mining. Technical Report 99-10, Computer Science Department, Rensselaer Polytechnic Institute, Troy NY 12180, October 1999. (Cited on page 16.)

[96] Mohammed J. Zaki, Markus Peters, Ira Assent, and Thomas Seidl. CLICKS: An effective algorithm for mining subspace clusters in categorical datasets. *Data & Knowledge Engineering*, 60(1):51–70, 2007. (Cited on page 49.)

[97] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. Out-of-core coherent closed quasi-clique mining from large dense graph databases. *ACM Transactions on Database Systems*, 32(2): 13–42, 2007. (Cited on page 126.)

[98] Lizhuang Zhao and Mohammed J. Zaki. MicroCluster: Efficient deterministic biclustering of microarray data. *IEEE Intelligent Systems*, 20(6):40–49, 2005. (Cited on page 43.)

[99] Lizhuang Zhao and Mohammed J. Zaki. TRICLUSTER: An effective algorithm for mining coherent clusters in 3D microarray data. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 694–705. ACM Press, 2005. (Cited on pages 49 and 71.)




## COLOPHON

This thesis was typeset with LaTeX using Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL* were used). The listings are typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

The typographic style was inspired by Bringhurst's genius as presented in *The Elements of Typographic Style* [15]. It is available for LaTeX via CTAN as "`classicthesis`".