Carnegie Mellon University
Tepper School of Business

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
OPERATIONS RESEARCH

Titled

"Interpretable Learning and Pattern Mining: Scalable Algorithms and Data-Driven Applications"

Presented by

Amin Hosseininasab

Accepted by

Willem van Hoeve	**4/30/2020**
Chair: Prof. Willem van Hoeve	**Date**

Approved by The Dean

Robert M. Dammon	**5/15/2020**
Dean Robert M. Dammon	**Date**

Interpretable Learning and Pattern Mining: Scalable Algorithms and Data-Driven Applications

Amin Hosseininasab

April 2020

Tepper School of Business,
Carnegie Mellon University

Submitted to the Tepper School of Business in Partial Fulfillment of the Requirements for the Degree of Doctor in Operations Research

Dissertation Committee:
Willem-Jan van Hoeve (Chair)
Andre A. Cire
Pascal van Hentenryck
Andrew A. Li
Sridhar Tayur

Abstract

In recent years, data-driven methodologies have enjoyed great success due in large part to the increasing accessibility of highly accurate machine learning tools. The challenge, however, is that such tools generally offer little interpretability in their learning tasks. This is a significant concern in many applications, such as scenarios with ethical implications or high risk. On the other hand, current interpretable learning algorithms are typically less accurate for prediction tasks, and comparably not scalable to accommodate large size datasets. This motivates the theme of this dissertation, which is to develop interpretable, yet accurate and scalable learning algorithms and apply them to real-life data-driven applications in management science.

As our first problem, we investigate the Multiple Sequence Alignment (MSA) problem. Our aim is to learn the optimal alignment between sequences of data. Applications of MSA include bioinformatics, high frequency trading, speech recognition, and computer vision. Although higher quality alignments offer significantly more insight for practitioners, most MSA algorithms are heuristic, and have been shown to often generate alignments far below the accuracy of an optimal solution on benchmark instances. In fact, only one viable exact algorithm has been developed for MSA, which is limited to solving small sized instances with five sequences and a total of 600 data entries. Using tools from dynamic programming, mathematical programming, constraint programming, and decomposition techniques, we develop a novel exact alignment algorithm capable of aligning up to ten sequences and 1,600 total data entries. Our method is able to close 37 out of 51 real-life benchmark instances to optimality for the first time, and considerably improves the alignment quality on the remaining instances.

In the next chapter, we develop novel techniques for constraint-based sequential pattern mining (SPM). SPM is an unsupervised learning algorithm that involves finding frequent patterns in data. Frequent patterns are used, e.g., to extract knowledge from data, and to develop novel association rules. Constraint satisfaction is often critical in the practice of SPM, to prevent overwhelming the practitioner with millions of uninteresting patterns. Unfortunately, the literature accommodates only simple constraints, and is further limited to databases with up to a million data entries. Using novel modelling techniques, we increase the scalability of our SPM algorithm by an order of magnitude, and further design and prove constraint-specific information for a number of complex constraints common in practice. Our algorithm is the first to handle complex constraints such as

average and median, but is also competitive or more efficient when compared to a state-of-the-art SPM algorithms with simple constraints.

We next study data-driven sequential decision making with an emphasis on interpretability and sequential structure in SPM. Using a novel data tree model of the database, we are able to increase the capability of SPM algorithms from databases with ten million to three billion data entries. We leverage data trees to design a pattern mining algorithm capable of extracting novel sequential patterns from large datasets, and design an interpretable knowledge tree equipped with statistical hypothesis tests, to increase reliability in data-driven decision making. Using our approach, we investigate two large-size real-world applications in marketing and finance. In marketing, we consider reducing the skip rate of users in an online music streaming platform. We find that almost all one billion user skips in the database can be explained using an average of 6,400 sequential patterns, with an average likelihood of 83%. In finance, we assess using historical sequential patterns of price change to aid investment decision making in the stock market. We find that, at best, 80% of nine hundred thousand monthly price change events can be explained using approximately 7,000 sequential patterns, with a low average likelihood of 53%.

In the final chapter, we study the provider network selection and insurance design problem faced by a major healthcare insurance provider. The provider network consists of physicians and hospitals that are under contract by the insurer and offer a wide range of healthcare services to insured patients. The problem involves choosing/changing the physicians and hospitals to contract, and designing insurance plans to target patients under competition with a rival firm. We provide a novel methodology that incorporates the literature's interpretable utility approach for patient behavior into a simultaneous multi-column-and-row generation optimization framework. By optimizing the provider network as a whole, we compose higher quality insurance plans that increase the profit of the insurer by an average of 548% on test instances, while decreasing the overall patient healthcare costs by 36% and the overall premiums payed by patient by 21%. We showed how interpretable information can be extracted from the optimization framework to aid decision making, and lastly analyzed the impact of inaccurate patient predictions on the insurer. Our results show that a more accurate yet interpretable predictor of patient choice can greatly enhance insurance plan design.

Acknowledgments

During my PhD, I have met and worked with a number of amazing people. Without a doubt, I have been most fortunate to be advised by Professor Willem-Jan van Hoeve. Willem is a rockstar of an advisor and an inspiration to follow. I am grateful not only for his mentorship, integrity, and wisdom, but also for his truly amazing personality. Willem, thank you for everything. I have learned many valuable lessons from you both in academia and in life, and I will cherish them forever.

The work of this dissertation has been made possible through collaboration with many great minds. Thank you Willem for being a constant contributor and an integral part of my research. My gratitude goes to Professor Andre Cire for his contributions towards my work in pattern mining. Many thanks to Professor Sridhar Tayur for his valuable feedback on presenting my work, and his guidance and contributions to the last chapter of this dissertation. Lastly, I would like to thank the remaining members of my dissertation committee, Professors Pascal van Hentenryck and Andrew Li for their helpful comments.

I have benefited from the help of many other faculty, and would like to specially thank Professors John Hooker, David Bergman, Benjamin Moseley, Jeff Galak, John Kececioglu, and Gunther Friedl.

Thank you also to Lawrence Rapp and Laila Lee for their kind help in the administrative needs of PhD students at Tepper.

My PhD journey would not have been as enjoyable if not for the great friends I made along the way. Thank you to Aleks, Arash, Bo, Christian, Gerdus, Majid, Mohammad, Nam, Neda, Ozgun, Sagnik, Thiago, Thomas, and Ziye. It was a pleasure getting to know every one of you. I am particularly thankful to my close friends and office mates, Mehmet and Franco. It was a lot of fun spending time with you guys in and outside of work, and truly made my PhD a unique experience.

My warmest gratitude goes to my family, especially my parents, brother and sister. Thank you to my father for being the pillar and inspiration in my life. Thank you to my mother for her sacrifices and prayers. Thank you to my brother and sister for their love and affection. I would not be where I am without the unconditional love and support of you all.

Words cannot express my gratitude for my amazing wife Elnaz. Thank you for your understanding and patience in all the long nights, weekends, and holidays that I had to work. Thank you for your love, affection, and companionship. You have shared with me the joys and struggles of this journey, and I dedicate this dissertation to you.

Table of Contents

Chapter 1

Introduction

Predictive models have significantly impacted the way information is organized and leveraged in the practice of management, including areas such as retail (Akter and Wamba, 2016), accounting (Appelbaum et al., 2017), sports (Coleman, 2012), and finance (Corea, 2016). Of notably prevalence are *sequential* applications, i.e., in which predictions (and subsequent managerial actions) are made consecutively over time based on past observed events. Examples include buying/selling decisions in financial markets (Pang et al., 2018) and recommender systems for products and services that account for observed consumer behavior (Cheng and Shen, 2016).

The growth of data-driven sequential approaches is due in large part to the increasing accessibility of highly accurate machine learning tools, specifically deep-learning and ensemble frameworks (Wu et al., 2019; Sagi and Rokach, 2018). The challenge, however, is that such tools generally offer little interpretability in their learning tasks and predictions. This is a significant concern in scenarios with ethical implications or higher risk. For instance, Vayena et al. (2018) argue that the lack of transparency is one of the primary barriers to the use of artificial intelligence in healthcare, with half of the United States decision makers at healthcare organizations believing it would produce fatal errors and not meet expectations. Similar issues are also raised, e.g., in marketing and finance (Drew et al., 2019; Hajek and Henriques, 2017; Cui et al., 2006).

In such cases, practitioners may be more inclined to use simpler and more interpretable machine learning models, such as rule-based learning and decision trees. While such algorithms are comparably less accurate for predictive tasks, they provide descriptive results and often expose insightful relationships in data. Interpretability further allows practitioners to explain the outcome of machine learning algorithms, resulting in higher reliability and accountability in decision-making tasks. These benefits have led to a growing research in interpretable machine learning (see, e.g., Došilović et al. 2018; Murdoch et al. 2019; Molnar et al. 2018).

Despite recent advancements in interpretable machine learning, unsupervised algorithms are still faced with major challenges when used for management science applications. In particular, interpretable learning algorithms are limited by design to simple procedures and methodologies.

Consequently, interpretable algorithms are generally less accurate than their supervised counterparts for prediction tasks, as they cannot capture complex relationships in data. Lower prediction accuracy may be critical in many applications, e.g., in healthcare and finance, where decision making depends highly on the predicted outcome. Lower accuracy can thus be a barrier to the application of interpretable learners in management science.

Another challenge faced by interpretable learners is limited scalability in the size of the input dataset. Unlike supervised algorithms which can recursively split and train on batches of the dataset, unsupervised algorithms generally require to fit the entire dataset in memory. For large-size datasets, e.g., ones commonly found in real-world application, unsupervised algorithms are thus either not a viable option, or are executed on a subset of the dataset. The latter entails generating suboptimal results and the discovery of only a subset of the knowledge hidden in data.

The above challenges motivate the work of this dissertation. In particular, we focus on the design of interpretable, yet accurate and scalable learning algorithms, and show their benefit when used to solve real-life applications in management science. Using a range of techniques in optimization and statistics, such as mathematical programming, constraint satisfaction, decision diagrams, dynamic programming, decomposition techniques, and hypothesis tests, we design novel methods that increase the scalability and accuracy of results generated by a number of unsupervised algorithms. We show the benefits of our developed algorithms and solution methods for applications in bioinformatics, marketing, finance, and healthcare. We produce novel results that show orders of magnitude improvement and solve benchmark instances to optimality for the first time.

1.1 Outline of Dissertation

The outline of this dissertation may be categorized into two themes:

(i) Developing interpretable, accurate, and scalable learning algorithms with the objective of accommodating management science applications.

(ii) Using the developed algorithms as a basis to design interpretable data-driven approaches for decision making tasks in marketing, finance, and healthcare.

Interpretable, Accurate, and Scalable Learning Algorithms

In the second chapter of the dissertation we develop an exact solution algorithm for the Multiple Sequence Alignment (MSA) problem. The MSA problem involves comparing the similarity between a number of sequences, which are strings of characters with a meaningful order. The objective is to extract interpretable insights into the relationships of the sequences, with better alignments exposing richer information. To solve the problem we initially design a dynamic programming model and use it to construct a novel Multi-valued Decision Diagrams (MDD) representation of all

pairwise sequence alignments (PSA). PSA MDDs are then synchronized using side constraints to model the MSA problem as a Mixed-Integer Program (MIP), for the first time, in polynomial space complexity. Two bound-based filtering procedures are developed to reduce the size of the MDDs, and the resulting MIP is solved using logic-based Benders decomposition. For a more effective algorithm, we develop a two-phase solution approach. In the first phase we use optimistic filtering to quickly obtain a near-optimal bound, which we then use for exact filtering in the second phase to prove or obtain an optimal solution. Numerical results on benchmark instances show that our algorithm solves several instances to optimality for the first time, and in case optimality cannot be proven, considerably improves upon a state-of-the-art heuristic MSA solver. Comparison to an existing state-of-the-art exact MSA algorithm shows that our approach is more time efficient and yields significantly smaller optimality gaps. This work is accepted for publication in INFORMS Journal on Computing (Hosseininasab and van Hoeve, 2019).

In the third chapter of the dissertation, we focus on Constraint-based sequential pattern mining. Constrained sequential pattern mining aims at identifying frequent patterns on a sequential database of items while observing constraints defined over the item attributes. Frequent patterns are used, e.g., to expose interpretable knowledge hidden in data, which are then used to aid decision making tasks. We introduce novel techniques for constraint-based sequential pattern mining that rely on a multi-valued decision diagram representation of the database. Specifically, our representation can accommodate multiple item attributes and various constraint types, including a number of non-monotone constraints. To evaluate the applicability of our approach, we develop an MDD-based prefix-projection algorithm and compare its performance against a typical generate-and-check variant, as well as a state-of-the-art constraint-based sequential pattern mining algorithm. Results show that our approach is competitive with or superior to these other methods in terms of its scalability and efficiency. This work is published in the conference proceedings of AAAI 2019 (Hosseininasab et al., 2019).

Data-Driven Decision Making and Applications

In the fourth chapter of the dissertation, we study data-driven sequential decision making with an emphasis on interpretability and sequential structure. We first develop a novel data tree model of the database that is able to fit in memory orders of magnitude larger datasets compared to traditional tabular encodings. We leverage data trees to design a pattern mining algorithm capable of extracting novel sequential patterns from large datasets. We then assess sequential patterns on the likelihood that they are associated to an outcome of interest, and use patterns that satisfy a minimum likelihood constraint as interpretable explanations on why those outcomes occurred. Using statistical hypothesis tests, we provide a reliability measure for such explanations, and analyze the trade-off between the imposed minimum likelihood and the percentage of outcomes that can be explained by feasible patterns. Lastly, we show how patterns can be aggregated into a knowledge

tree to provide a structural view of the decision making process. Using our approach, we investigate two large-size real-world applications in marketing and finance. In marketing, we consider reducing the skip rate of users in an online music streaming platform. We find that almost all one billion user skips in the database can be explained using an average of 6,400 sequential patterns, with an average likelihood of 83%. In finance, we assess using historical sequential patterns of price change to aid investment decision making in the stock market. We find that, at best, 80% of nine hundred thousand monthly price change events can be explained using approximately 7,000 sequential patterns, with a low average likelihood of 53%. This work is submitted for publication, and under review.

In the final chapter of the dissertation, we consider the provider network selection problem in healthcare insurance plan design. Provider network selection takes the perspective of an insurance firm that is tasked with finding the optimal subset of healthcare providers to contract. The set of contracted providers are used to compose the insurer's health insurance plans, which are then offered to patients for selection. Patients select to enroll in an insurance plan among the ones offered by the insurer and its competitor, possibly generating revenue for the insurer. To solve this problem, we develop a novel methodology based on optimization. We incorporate the interpretable utility function approach of the literature for predicting patient behavior into our optimization framework, and develop a simultaneous multi-column-and-row generation algorithm. The algorithm involves recursively generating new insurance plans that improve on the current set of plans offered by the insurer. This is challenging, as the benefit of adding a newly composed insurance plan to the optimization model depends on the dual information of constraints that are simultaneously added to the model. We provide and prove a sufficient and necessary condition for newly composed insurance plans to be profitable, and discuss how the optimization model can be used to extract interpretable knowledge for decision making. Our numerical tests show that our approach improves over the literature in all considered instances. We lastly evaluate the effects of inaccuracies in the utility function in predicting patient choice. Our results show that inaccurate prediction of patient utilities can be detrimental in the provider selection framework, motivating the need for an interpretable yet accurate prediction methodology.

Chapter 2

Exact Multiple Sequence Alignment

2.1 Introduction

Sequence alignment is a fundamental problem in computational biology, but also finds application in other fields such as high frequency trading, speech recognition, and computer vision. The problem involves comparing a number of *sequences*, with the objective of finding their best *alignment*. A sequence s is defined as a string of characters $s := \langle c^1, \ldots, c^{|s|} \rangle$ with a meaningful order, for example, representing a DNA, RNA, or protein molecule in the context of computational biology. The optimal alignment of a set of sequences $\mathbb{S}$ is obtained by inserting gaps "$-$" into each sequence $s \in \mathbb{S}$, such that all sequences achieve the same length and their similar regions overlap.

The Pairwise Sequence Alignment (PSA) problem consists of finding the best alignment between a pair of sequences (s, s'), and as a generalization, the Multiple Sequence Alignment (MSA) problem involves aligning a set $\mathbb{S} \geq 3$ of sequences. The output of the sequence alignment problem is a matrix with $|\mathbb{S}|$ rows, where each row corresponds to a sequence $s \in \mathbb{S}$, and each column corresponds to a set of aligned characters. Two characters $c_s^i, c_{s'}^j$, or a character c_s^i and a gap symbol "$-$", are aligned if they are placed in the same column of the MSA matrix. An alignment is feasible if and only if it satisfies the following two requirements:

1. Order requirement: Ignoring the gap characters "$-$", the row associated with a sequence $s \in \mathbb{S}$ must be identical to sequence s, and preserve the order of its characters $\left\langle c_s^1, \ldots, c_s^{|s|} \right\rangle$.
2. Column alignment requirement: Each set of aligned characters belongs to exactly one column, and any column must contain at least one character $c_s^i \in s, s \in \mathbb{S}$.

A feasible alignment is optimal if it gives the highest objective value with respect to the *scoring function*. The scoring function assigns a reward for matching two characters, and penalizes alignments of characters to gaps. The reward of aligning different characters is determined by a predefined *substitution matrix*. Different substitution matrices result in different optimal solutions, and are used to accommodate various user objectives.

The penalty of aligning characters to gaps is primarily dependent on the user objective, and may be a constant, or a function of the number of consecutive gap alignments (Vingron and Waterman, 1994). In affine or convex gap penalty functions, an opening gap alignment, i.e., a gap alignment immediately after an alignment of characters, is penalized by the *opening* penalty. An extension of a gap alignment is thereafter penalized by the *extension* penalty, which is a linear function in affine, or a convex function in convex gap penalties.

To obtain the overall alignment score for MSA, a popular technique is the *Sum-of-Pairs* objective template. In this template, the objective value is the sum of the rewards or penalties assigned to all pairs of aligned characters and gaps. Note that a pair of aligned gaps is not assigned any reward or penalty. Regardless of the scoring template, solving MSA is proven to be NP-complete (Wang and Jiang, 1994), and is also challenging in practice. Thus, the majority of research in solving MSA is dedicated to heuristic algorithms (Katoh et al., 2017; Hung et al., 2015; Sievers et al., 2011; Chakraborty and Bandyopadhyay, 2013; Magis et al., 2014). Although heuristic methods are fast and accommodate large-size problems, they fail to give any guarantee on the solution quality. In fact, the quality of heuristic solutions has been shown to be often far below the optimal solution on benchmark instances (Thompson et al., 2011; Nuin et al., 2006). As higher quality alignments offer significantly more insight into relationship of sequences, it is desirable to develop algorithms with higher accuracy. This motivates the development of algorithms that attempt to solve MSA exactly and guarantee optimality.

To the best of our knowledge, three exact solution approaches exist for MSA. Namely, the Dynamic Programming (DP) approach introduced by Needleman and Wunsch (1970); the Integer Programming (IP) approach of the Maximum Weight Trace (MWT) problem introduced by Kececioglu (1993) and solved exactly by Althaus et al. (2006); and the convex optimization approach of Yen et al. (2016). Although exact approaches may be slow compared to heuristics, they are guaranteed to reach the optimal solution eventually, even for large-size problems. However, a major bottleneck of existing exact MSA algorithms is their exponential space requirement, which is generally too high to allow the solution algorithm to fit in memory for large-size problems.

The DP algorithm models the MSA problem as a $|\mathbb{S}|$ dimensional cube of size $n = \max_{s \in \mathbb{S}} \{|s|\}$, with a memory requirement of $O(n^{|S|})$ which is too space consuming to even model small sized instances. The convex optimization approach also has exponential space requirements, and is limited to solving small-sized sequences of length below 50 in practice. Similarly, the MWT approach has exponential space requirements, with a worst-case $O(|\mathbb{S}|n^{|\mathbb{S}|})$ space complexity.

The most successful exact approach for MSA is the branch-and-cut algorithm developed for the MWT problem (Althaus et al., 2006). Althaus et al. (2006) consider small to medium sized problems with 4 to 6 sequences and a total number of 300 to 600 characters. The branch-and-cut algorithm works well on benchmark instances with high degrees of similarity, and is able to solve those to optimality within a short amount of time. However, the algorithm does not converge for instances

with lower degrees of similarity within 10 hours of computation, and terminates with 16-580% gaps to optimality. For such instances, heuristic solutions with good quality are generated by optimizing over a carefully chosen subset of the feasible solutions.

In this work, we follow the Sum-of-Pairs template and develop an algorithm for global sequence alignment (Needleman and Wunsch, 1970). As in most MSA algorithms, our algorithm may accommodate local alignment (Smith and Waterman, 1981) or any objective template using straightforward modifications.

In the first step, we develop a DP model and use it to construct a novel representation of the MSA problem using Multi-valued Decision Diagrams (MDD). Decision diagrams were originally used for circuit design problems as a compact graphical representations of Boolean functions (Lee, 1959; Bryant, 1986). In recent years, decision diagrams have been successfully used to represent the set of feasible solutions of discrete optimization problems (Becker et al., 2005; Bergman et al., 2014, 2016b; Cire and van Hoeve, 2013; Darwiche and Marquis, 2004; Bouquet and Jégou, 1995; Cayrol et al., 1998).

An MDD approach to MSA possesses a number of advantages not available for existing DP, IP, or convex optimization approaches. Using the order preserved in MDDs, the structure of sequences may be exploited to remove a set of infeasible alignments prior to optimization. This is not case in the MWT algorithm for example, which identifies and removes such solutions iteratively by branch and cut on a linear program (Reinert et al., 1997; Althaus et al., 2006; Kececioglu et al., 2000). Using the MDD representation of PSA of all pairs $(s, s') : s, s' \in \mathbb{S}$, we model the MSA problem in worst-case $O(|\mathbb{S}|^2 n^3)$ space for affine gap penalty functions and $O(|\mathbb{S}|^2 n^4)$ for convex gap penalty functions, while existing exact approaches require a worst-case exponential space. This is a significant improvement, as it allows to model and potentially solve larger sized problems (in particular for affine penalty functions), even though they may not be solved to optimality in a reasonable amount of time.

We consider the MSA problem as the synchronized PSA of all pairs $(s, s') : s, s' \in \mathbb{S}$. More specifically, PSA MDDs are used as the underlying network structure to formulate the MSA problem as a Mixed-Integer Program (MIP), and their solution is synchronized using side constraints. Transforming DP models into an equivalent MIP formulation was introduced by Martin (1987); Martin et al. (1990), and shown to be an effective modeling approach. The resulting MIP in our approach is a collection of longest path problems with side constraints, and hard to solve in general.

To lower the solution difficulty, we first attempt to reduce the size of the feasible solution set by removing non-optimal solutions embedded in the PSA MDDs. We develop two bound-based filtering procedures using the Carrillo and Lipman (1988) lower bound, and an upper bound based on linear programming strengthened through additive bounding (Fischetti and Toth, 1989). To the best of our knowledge, this is the first upper bounding procedure used to prune infeasible or non-optimal solutions in the MSA problem. We show that these filtering procedures complement each other, and in particular, additive bounding filtering is able to prune a significant number of

solutions even after the Carrillo-Lipman filtering procedure is used. The MSA MIP is defined on the filtered PSA MDDs, and solved using a logic-based Benders decomposition algorithm (Hooker and Ottosson, 2003).

For a more effective solution algorithm, we develop a two phase approach. In the first phase, we aggressively filter the PSA MDDs using an optimistic guess for the optimal solution objective. This allows us to optimize over a considerably smaller solution set and quickly find quality solutions (in most considered instances, optimal or near-optimal). Using the bound obtained by the optimal solution of the first phase, we exactly filter the PSA MDDs in the second phase to prove optimality of the first phase solution, or find an optimal solution.

We show that our algorithm is able to solve several benchmark instances to optimality (with respect to our considered objective functions). For instances not solved withing the imposed time limit, a primal heuristic embedded in the Benders decomposition algorithm generates feasible MSA solutions, which significantly improve alignment accuracy compared to a state-of-the-art MSA heuristic solver. The solution algorithm is compared to the best exact approach of the literature, and shown to optimally solve or reach lower gaps to optimality on almost all considered instances.

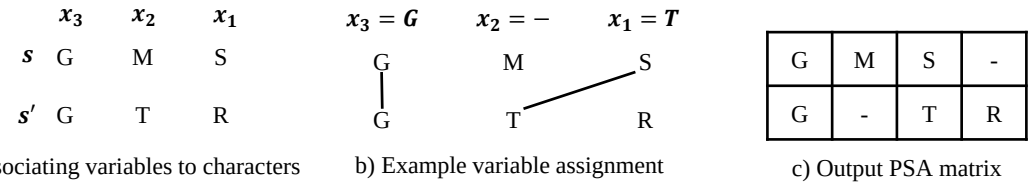
The rest of the chapter is structured as follows. §2.2 models the PSA problem as a DP model, and uses that model to represent the PSA problem as an MDD. In §2.3 we show how to synchronize PSA MDDs to model the MSA problem as an MIP, discusses the filtering procedures used to prune non-optimal solutions, and designs the two phase solution algorithm. In §2.4, we develop the logic-based Benders decomposition and primal heuristic algorithms. We give the numerical results on benchmark instances in §2.5, and the chapter is concluded in §2.6.

2.2 A Dynamic Programming Model and MDD Representation for the PSA Problem

In this section, we develop a DP model for the PSA of sequences $s, s' \in \mathbb{S}$. Our model is similar to the matrix-format DP model of Needleman and Wunsch (1970) (in particular when gap penalties are linear); however, our DP model is better suited to develop an MDD representation of PSA. For clarity, we first define the DP model for the case of linear gap penalty functions, and thereafter expand the model to affine and convex settings.

2.2.1 Dynamic Programming Model for the PSA Problem

For a PSA alignment of sequences (s, s'), we define x as a tuple of discrete variables $(x_1, \ldots, x_i, \ldots, x_{|s|})$, one for each character $c_s^i : 1 \leq i \leq |s|$. Variables x_i are associated with finite domains D_i, defined as the possible alignment decisions for c_s^i, i.e., $D_i = \left\{c_{s'}^1, \ldots, c_{s'}^{|s'|}, -\right\}$. Assigning x_i to a value in D_i represents aligning character c_s^i to some character $c_{s'}^j, 1 \leq j \leq |s'|$, or to the gap symbol "$-$". For example, for the PSA of the sequences in Figure 2.1.a, we define three variables $x_1, \ldots, x_3$



a) Associating variables to characters b) Example variable assignment c) Output PSA matrix

Figure 2.1: Example variable association and assignment, and corresponding output PSA matrix.

with the domains $D_i = \{G, T, R, -\}, i = 1, \ldots, 3$. A PSA solution is thus represented by a feasible assignment to tuple x. For example, $\bar{x} = (\bar{x}_1, \bar{x}_2, \bar{x}_3) = (T, -, G)$ represents the alignment made in Figure 2.1.b, with the output alignment matrix shown in Figure 2.1.c.

To determine the optimal assignment for x, we solve the PSA problem using a DP with $i = 1, \ldots, |s|$ stages, where at each stage i, a decision is made for the value of x_i. The model is formulated by backwards induction, i.e., stages $i = 1, \ldots, |s|$ (consequently variables $x_1, \ldots, x_{|s|}$) correspond to the alignment decision for characters $c_s^{|s|}, \ldots, c_s^1$, respectively. The PSA DP model has the following elements:

- state spaces $\mathcal{S}_i$, for $i = 0, \ldots, |s| + 1$;
- transition functions $\mathcal{T}_i : \mathcal{S}_{i-1} \times D_i \to \mathcal{S}_i$, for $i = 1, \ldots, |s|$;
- transition reward functions $\mathcal{R}_i : \mathcal{S}_{i-1} \times \mathcal{S}_i \to \mathbb{R}$, $i = 1, \ldots, |s| + 1$;

and is written as:

$$\max_{z,x} \quad \sum_{i=1,\ldots,|s|+1} \mathcal{R}_i(z_{i-1}, z_i) \tag{2.1}$$

$$\text{s.t.} \quad z_i = \mathcal{T}_i(z_{i-1}, x_i) \qquad i = 1, \ldots, |s|, \tag{2.2}$$

$$z_0 = |s'| + 1, z_{|s|+1} = 0 \tag{2.3}$$

$$z_i \in \mathcal{S}_i, x_i \in D_i \qquad i = 1, \ldots, |s|. \tag{2.4}$$

The alignment variables $x_1, \ldots, x_{|s|}$ are regarded as controls, where a control x_i takes the DP system from state $z_{i-1} \in \mathcal{S}_{i-1}$ to state $z_i = \mathcal{T}_i(z_{i-1}, x_i)$ and accumulates a reward/penalty $\mathcal{R}_i(z_{i-1}, z_i)$. We define states z_i as the position j of the last non-gap alignment made to a character $c_{s'}^j$. That is, $z_i = j$, where $c_{s'}^j$ is the last non-gap alignment decision for a character $c_s^{i'}, i' \leq i$. Following this definition, the transition functions are defined as:

$$\mathcal{T}_i(z_{i-1}, x_i) = \begin{cases} j & \text{if } x_i = c_{s'}^j, \text{ and} \\ z_{i-1} & \text{if } x_i = -. \end{cases}$$

The DP model is initiated from state $z_0 = |s'| + 1$ and terminated at state $z_{|s|+1} = 0$. For example, Figure 2.2 shows the transition values for the example PSA of Figure 2.1.b.

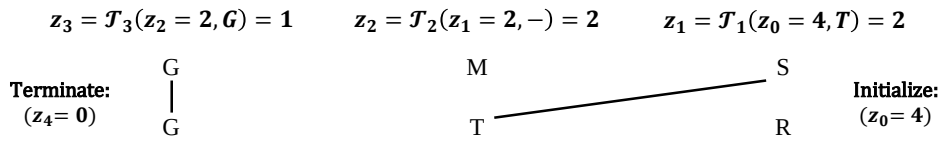


Figure 2.2: Example DP transition function values.

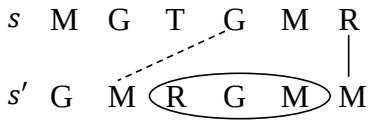


Figure 2.3: The alignment shown by the dashed arc implies gap alignments of characters $c^j_{s'}, j = 3, 4, 5$ (circled).

PSA feasibility is satisfied by ensuring that $z_i \leq z_{i-1}$ for any transition $z_i = \mathcal{T}_i(z_{i-1}, x_i)$, as proved in Lemma 2.1. State spaces $\mathcal{S}_i$ are defined as the set of possible state values for z_i at stage i of the DP model. Therefore, $\mathcal{S}_0 = \{|s'| + 1\}$, $\mathcal{S}_{|s|+1} = \{0\}$, and $\mathcal{S}_i = \{|s'| + 1, \ldots, 1\}$ for all $1 \leq i \leq |s|$.

Lemma 2.1. *A transition from a state z_{i-1} to a state z_i is feasible if and only if $z_i \leq z_{i-1}$.*

Proof. The "only if" direction holds due to the well-known Markovian property of the DP solution for PSA. The "if" direction holds as any alignment such that $j = z_{i-1} > z_i = j'$ implies an alignment of character $c^{i'}_s, i' \leq i - 1$ to $c^j_{s'}$, and c^i_s to $c^{j'}_{s'}, j > j'$, and violates the order requirement. □

The transition reward function $\mathcal{R}(z_{i-1}, z_i)$ is designed to accumulate a gap penalty if $z_i = z_{i-1}$. To model character alignment rewards, we first point out that the proposed DP model explicitly determines gap alignment decisions for characters c^i_s only, and it does not explicitly determine the gap alignment decisions for characters $c^j_{s'}$. Gap alignments for characters $c^j_{s'}$ are however implied by non-gap alignment decisions for characters c^i_s. For example, in Figure 2.3 the alignment of $c^4_s = M$ implies that characters $c^3_{s'}, c^4_{s'}, c^5_{s'}$ are aligned with gap symbols. To capture the penalty of such implied gap alignments, $\mathcal{R}(z_{i-1}, z_i)$ is designed to accumulate a gap penalty of a length $z_i - z_{i-1} - 1$ gap, in addition to the explicit character alignment reward for characters c^i_s.

2.2.2 Extension to Affine and Convex Gap Penalty Functions

The state definition z_i of the previous section is only sufficient for a linear gap penalty function. To accommodate more complex penalty functions such as affine or convex, we are required to define additional state values for variables z_i. In particular gap opening and extension penalties are unequal in such functions. Therefore, to correctly penalize gap alignments at stage i of the DP model, we are required to distinguish whether a gap or a non-gap alignment is made at stage $i - 1$. In case a

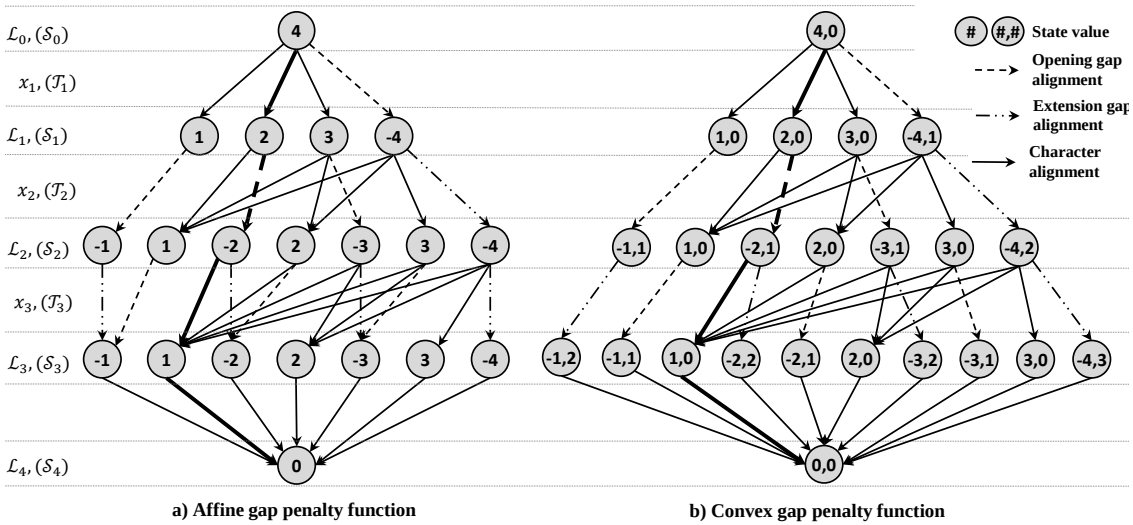


Figure 2.4: A PSA MDD with layers $\mathcal{L}_i$ corresponding to state spaces $\mathcal{S}_i$, and variables x_i (transitions $\mathcal{T}_i$) corresponding to arcs $(u, v) \in A^p$. The highlighted path corresponds to an alignment $(\bar{x}_1, \bar{x}_2, \bar{x}_3) = (c^1_{s'}, -, c^2_{s'})$

.

non-gap alignment is made at stage $i-1$, a gap alignment at stage i is penalized by the opening gap penalty. Similarly, if a gap alignment is made at stage $i-1$, an extension gap penalty is incurred for gap alignments at stage i.

To distinguish gap and non-gap alignments at each stage, we introduce negative state values $z_i < 0$, and associate them to gap alignments. That is, a negative state $z_i < 0$ denotes that a gap alignment is made at stage $i-1$. In this setting, a character alignment is modeled by a transition from a positive or negative state value z_{i-1}, to a positive state value $z_i > 0$. Opening gap alignments are modeled by a transition from a positive state value $z_{i-1} > 0$ to a negative state value $z_i < 0$; and gap extension alignments are modeled by transition between negative state values $z_i < 0, z_{i-1} < 0$. Similar to before, a transition $z_i = \mathcal{T}_i(z_{i-1}, x_i)$ is feasible (permitted) if and only if $0 \leq z_i < |z_{i-1}|$ for character alignments, or $z_i = -|z_{i-1}|$ for gap alignments.

The state spaces $\mathcal{S}_i$ for the DP model with affine penalty functions are defined as follows. For stage $i = 1$, we define $\mathcal{S}_1 = \{-(|s'|+1), 1, \ldots, |s'|\}$, where a state $1 \leq z_1 \leq |s'|$ indicates that $c^{|s|}_s$ is aligned to character $c^{z_1}_{s'}$; and state $z_1 = -(|s'|+1)$ indicates an opening gap alignment for $c^{|s|}_s$. Note that gap extension alignments are not possible at stage $i = 1$. For stages $i = 2, \ldots, |s|$, we define $\mathcal{S}_i = \{-(|s'|+1), -|s'|, \ldots, -1, 1, \ldots, |s'|\}$, with a negative state value $-|s'| \leq z_i \leq -1$ representing a gap alignment at stage $i-1$. Figure 2.4.a. shows example transitions (represented by arcs) between different state values z_i (represented by nodes) for affine penalty functions.

Unlike an affine penalty function where gap extension penalties incurred at stage i are equal regardless of the number of consecutive gap alignments prior to stage i, a convex penalty function gives different penalties when extending gaps of different lengths. To accommodate a convex penalty

function we must thus keep track of the number of consecutive gap alignments prior to stage i, in addition to representing gap states by negative values. This is done by redefining state variables z_i from single values j, to size-two tuples (j, g). The definition of values j is consistent with the case of affine penalty functions, and the newly introduced state gap length value g denotes the number of consecutive gap alignments made prior to state z_i. Using values (j, g) we can correctly calculate the cost of convex gap penalties in $\mathcal{R}_i(z_{i-1}, z_i)$.

To define the state spaces $\mathcal{S}_i$ for convex gap penalties, we first define $\mathcal{S}_i = \{(-|s'|-1, i), (1, 0), \ldots, (|s'|, 0), \}, 1 \leq i \leq |s'|$, which denote character alignments by $z_i = (j, 0) : j > 0$, and the possibility of no character alignment (i consecutive gap alignments) before stage i by $z_i = (-|s'|-1, i)$. To represent gap alignment states, we add state values $\{(-1, i'), \ldots, (-|s'|, i'), \}$ to $\mathcal{S}_i$ for every value of $i' \in \{1, \ldots, i\}$, as there are a possible i' number of possible values for g at stage i. Figure 2.4.b shows example transitions (represented by arcs) between different state values z_i (represented by nodes) for a convex penalty functions.

This concludes the definition of the proposed DP model for PSA. The next section uses the designed DP model to represent PSA as a MDD.

2.2.3 MDD Representation for the PSA Problem

This section uses the PSA DP model to represent the PSA of a pair of sequences $p = (s, s')$ as an MDD. For succinctness, we mainly discuss the MDD representation for the DP model with affine penalty functions. The extension to convex penalty functions is straightforward.

A PSA MDD $\mathcal{M}^p = (U^p, A^p)$ is a layered directed acyclic graph, where U^p is its set of nodes, and A^p is its set of arcs. Set U^p is partitioned into layers $(\mathcal{L}_0, \ldots, \mathcal{L}_{|s|+1})$, such that layers $\mathcal{L}_i : 1 \leq i \leq |s|$ correspond to stages $1 \leq i \leq |s|$ of the DP model, and contain one node per value of state spaces $\mathcal{S}_i$. For example, Layers $\mathcal{L}_0$ and $\mathcal{L}_{|s|+1}$ consist of single nodes, namely the root node $r \in \mathcal{L}_0$ ($z_0 = |s'|+1$), and the terminal node $t \in \mathcal{L}_{|s|+1}$ ($z_{|s|+1} = 0$), which are used to initialize and terminate the PSA, respectively.

Nodes $u \in \mathcal{L}_i : 1 \leq i \leq |s|$, represent the possible values that z_i can take at stage i of the DP formulation. For example, for the DP formulation with affine penalty function, we create $|\mathcal{S}_1| = |s'| + 1$ nodes for layer $\mathcal{L}_1$, and $|\mathcal{S}_i| = 2|s'| + 1$ nodes for layers $\mathcal{L}_i : 2 \leq i \leq |s'|$. A node $u \in \mathcal{L}_i : 0 \leq i \leq |s| + 1$ thus represents a specific state value in $\mathcal{S}_i$, denote as $state(u)$. Transitions from states in $\mathcal{S}_{i-1}$, i.e., nodes $u \in \mathcal{L}_{i-1}$, to states in $\mathcal{S}_i$, i.e., nodes $v \in \mathcal{L}_i$, are represented by arcs $(u, v) \in A^p$. Arcs $(u, v) \in A^p$ model the assignment of a value in D_i to control variables x_i, and take the system from $state(u)$ to $state(v) = \mathcal{T}_i(state(u), x_i)$. As in the DP model, transitioning to a node $v : state(v) > 0$ at stage i, implies an alignment of c_s^i to character $c_{s'}^{state(v)}$. Transitioning from node u to a node $v : state(v) < 0$ implies an opening gap alignment if $state(u) > 0$, or an extension gap alignment if $state(u) < 0$.

Following the DP model, PSA feasibility is ensured by creating an arc (u, v) only if $state(v) <$

$|state(u)|$ for character alignments, or $state(v) = -|state(u)|$ for gap alignments. By this representation, any path from r to t corresponds to a feasible PSA solution, as the satisfaction of the order and column alignment requirements are guaranteed by the definition of transition functions $state(v) = \mathcal{T}_i(state(u), x_i)$. Moreover, the MDD models all possible feasible PSA solutions, as its arcs model all feasible transitions of the DP model.

Figures 2.4.a, 2.4.b show the MDD representation of the example PSA in Figure 2.1, with the correspondence of layers $\mathcal{L}_i$ to state spaces $\mathcal{S}_i$, arcs (u, v) to transitions $state(v) = \mathcal{T}_i(state(u), x_i)$, and nodes $u \in \mathcal{L}_i$ to state values z_i, for affine and convex penalty functions respectively. The example alignment $x = (G, -, T)$ corresponds to the highlighted path from the root node r to the terminal node t in both MDDs. Note that the structure of any two PSA MDDs $\mathcal{M}^p, \mathcal{M}^{p'}$, respectively modeling the alignment of sequence pairs (s_1, s_2), (s_3, s_4), is identical if $|s_1| = |s_3|, |s_2| = |s_4|$. In particular, Figure 2.4 shows the MDD structure for any two sequences (s, s') such that $|s| = 3, |s'| = 3$.

To implement the reward/penalty of alignments we use weighted MDDs, where arcs $(u, v) \in A^p$ are associated to values $w_{uv} = \mathcal{R}_i(state(u), state(v))$. The length of an r-t path is thus the objective value $\sum_{i=1,\dots,|s|+1} \mathcal{R}_i(z_{i-1}, z_i)$. Consequently, the longest r-t path corresponds to optimal PSA solution $\hat{x}$. Proposition 2.2 proves that the PSA MDD correctly models the PSA problem.

Proposition 2.2. *The PSA MDD is a valid representation for the PSA problem.*

Proof. The PSA MDD mimics the DP model for the PSA problem by modeling all states by nodes $u \in A^p$, and all feasible transitions by arcs (u, v). The transition reward functions are also captured by weights w_{uv} on arcs $(u, v) \in A^p$. As the DP model is valid for the PSA problem, the PSA MDD is a valid representation of the PSA problem. □

We next prove in Proposition 2.3 that our PSA MDDs are *reduced*. An MDD is reduced if there are no equivalent nodes in any of its layers $\mathcal{L}_i$ (Wegener, 2000). Two nodes u, u' in a layer $\mathcal{L}_i$ are equivalent if there is a one to one equivalence between the set of u-t paths, and the set of u'-t paths. For equivalent nodes u, u', we can delete node u' and all its outgoing arcs $(u', v) : v \in \mathcal{L}_{i+1}$, and redirect its incoming arcs $(v, u') : v \in \mathcal{L}_{i-1}$ to node u, without altering the decision structure of the MDD.

Proposition 2.3. *The PSA MDD representation above is reduced by construction.*

Proof. By construction, $state(u) \neq state(u')$ for any pair of nodes $u, u' \in \mathcal{L}_i, i = 0, \dots, |s| + 1$. As arcs $(u, v) \in A^p$ are constructed based on $state(u)$, no two single nodes $u, u' \in \mathcal{L}_i$ have a common set of outgoing arcs (i.e., arcs representing the same decision and the same weights w_{uv}). Therefore, no two paths u-t, u'-t are equivalent for any nodes $u', u \in U^p$. □

The current definition of a PSA MDD models all feasible PSA solutions in $O(n^3)$, $O(n^4)$ space complexity for affine and convex penalty functions, respectively. Although the entire PSA MDDs

are required in order to model an exact solution approach for MSA (discussed in the next section), the space complexity of the algorithm may be reduced if we are interested in only finding an optimal PSA. To that end, we prove in Lemma 2.4, that for PSA the only required information at stage i of the DP model is the longest path values leading to nodes $u \in \mathcal{L}_{i-1}$.

Lemma 2.4. *Let $I^{\downarrow}(u)$ denote the value of a longest r-u path. Values $I^{\downarrow}(u) \in \mathcal{L}_i$ are the only information required to solve the PSA problem.*

Proof. Proof is by induction on stages i of the DP model. The statement is trivial for stage $i = 1$. Assume the statement holds for stage $i > 1$. For any node $v \in \mathcal{L}_{i+1}$ we have by definition of the longest path, $I^{\downarrow}(v) = \max\limits_{u:(u,v)\in A^p} \left\{I^{\downarrow}(u) + w_{uv}\right\}$. By the principal of mathematical induction, the statement holds for any stage i. □

The DP model is thus only required to keep in memory $|\mathcal{L}_{i-1}| + |\mathcal{L}_i|$ number of values at each stage i, and update the partial optimal PSA at every step similar to the algorithm of Hirschberg (1975). This gives a space complexity of $O(n), O(n^2)$ for the PSA problem with affine or convex penalty gaps, which is consistent with the best complexity proven for the PSA problem with linear penalty costs in Hirschberg (1975).

2.3 Solving the MSA Problem using Synchronized PSA MDDs

2.3.1 Modeling the MSA Problem as an MIP

The MSA problem may be viewed as the simultaneous alignment of all possible PSAs. In this section, we show how to synchronize all PSA MDDs for an exact solution to the MSA problem. We consider $1 \leq s \leq |\mathbb{S}| - 1$ and $s + 1 \leq s' \leq |\mathbb{S}|$, and let set $\mathbb{P}$ be the union set of all pairs $p = (s, s')$. We construct all pairwise MDDs $\mathcal{M}^p : p \in \mathbb{P}$, and define $\mathbb{M} = (U^c, A^c)$ as the union of all pairwise MDDs $\mathcal{M}^p$, where $U^c = \bigcup\limits_{p\in\mathbb{P}} U^p, A^c = \bigcup\limits_{p\in\mathbb{P}} A^p$. Observe that the worst-case space complexity of $\mathbb{M}$ is $O(|\mathbb{S}|^2 n^3), O(|\mathbb{S}|^2 n^4)$ for affine and convex penalty functions, respectively. To simultaneously solve the PSA MDDs, we first formulate the MIP model below, defined on set $\mathbb{M}$:

$$\max \quad \sum_{(u,v)\in A^c} w_{uv} y_{uv} \tag{2.5}$$

$$\text{s.t.} \quad \sum_{u:(r,u)\in A^p} y_{ru} = 1 \qquad \forall p \in \mathbb{P}, \tag{2.6}$$

$$\sum_{u:(u,t)\in A^p} y_{ut} = 1 \qquad \forall p \in \mathbb{P}, \tag{2.7}$$

$$\sum_{u:(u,v)\in A^p} y_{uv} - \sum_{u:(v,u)\in A^p} y_{vu} = 0 \qquad \forall p \in \mathbb{P}, \forall v \in U^p \setminus \{r, t\}, \tag{2.8}$$

$$y_{uv} \in \{0, 1\} \qquad \forall (u, v) \in A^c. \tag{2.9}$$

Here, y_{uv} is a binary variable defined for arcs $(u,v) \in A^c$, determining whether arc (u,v) is part of the optimal solution or not. The objective function (2.5) maximizes the reward of all pairs of character alignments (sum-of-pair score), and constraints (2.6)-(2.8) preserve the required flow in the longest path problems defined on $\mathbb{M}$, the union of disjoint directed acyclic graphs $\mathcal{M}^p$.

Solving the above MIP corresponds to finding the shortest paths in all PSA problems independently, and is likely infeasible with respect to the order requirement in MSA. To enforce the order requirement, we add additional continuous *column* variables $\pi_s^i \geq 0$, which denote the column placement of character c_s^i in the output MSA matrix. We enforce constraints (2.10) and (2.11), which ensure that any aligned characters $c_s^i, c_{s'}^j$ are placed into the same column of the MSA matrix (i.e., $\pi_s^i = \pi_{s'}^j$). Here, M denotes a big number, defined as $M = \sum_{s \in \mathbb{S}} |s|$ in our computational experiments.

$$\pi_s^i \leq \pi_{s'}^j + M \left(1 - \sum_{\substack{(u,v) \in A^p : u \in \mathcal{L}_i^p, \\ state(v) = j}} y_{uv} \right) \qquad \forall p = (s,s'), \forall (i,j) \in (s,s'), \tag{2.10}$$

$$\pi_{s'}^j \leq \pi_s^i + M \left(1 - \sum_{\substack{(u,v) \in A^p : u \in \mathcal{L}_i^p, \\ state(v) = j}} y_{uv} \right) \qquad \forall p = (s,s'), \forall (i,j) \in (s,s'). \tag{2.11}$$

Finally, we add constraints (2.12), which enforce the order requirement between columns of the MSA matrix:

$$\pi_s^i \leq \pi_s^{i+1} - 1 \qquad \forall k, \forall i = \{1, \ldots, |s| - 1\}. \tag{2.12}$$

This gives a constrained longest path formulation:

$$\mathbf{P:} \quad \{(2.5) | (2.6) - (2.12), \pi \geq 0\},$$

to solve MSA. Note that although variables π_s^i are continuous, in a solution for $\mathbf{P}$ they take integer values due to their integer upper and lower bounds. The worst-case space complexity of model $\mathbf{P}$ is consistent with the size of $\mathbb{M}$, which is lower than the worst-case exponential space requirements of all current exact MSA algorithms for $|\mathbb{S}| > 3$, in particular for linear or affine gap penalty functions. To prove correctness, Proposition 2.5 proves that constraints (2.6)-(2.12) are sufficient to enforce the feasibility requirements of MSA, and Corollary 2.6 proves that the solution of $\mathbf{P}$ is an exact MSA solution.

Proposition 2.5. *An MSA matrix satisfies the MSA feasibility requirements, if and only if constraints (2.6)-(2.12) are satisfied.*

Proof. If the MSA matrix satisfies the MSA feasibility requirement, we can construct solution vector $\bar{y}$ by setting $\bar{\pi}_s^i = j, \forall s \in \mathbb{S}, \forall i \in s$, where character c_s^i is placed in column j of the MSA matrix. Solutions $\bar{y}, \bar{\pi}$ satisfy constraints (2.6)-(2.12), as all aligned characters are placed in the same column of the matrix, and the order of columns satisfy constraint (2.12). For the converse, we can construct a feasible MSA matrix using the ordering given by solution $\bar{\pi}$. All aligned characters have equal $\bar{\pi}$ and thus fall into the same column, and all columns satisfy the order requirement enforced by constraint (2.12). □

Corollary 2.6. *The solution of* $\mathbf{P}$ *is an exact MSA solution.*

Proof. By Theorem 2.5, $\mathbf{P}$ optimizes over the set of all feasible MSA solutions. Thus, the optimal solution to $\mathbf{P}$ is an optimal MSA solution. □

Despite its polynomial model size, solving $\mathbf{P}$ to optimality is only practically feasible for small-sized instances when using a generic MIP solver. To increase the solution efficiency, we next introduce filtering algorithms to prune and reduce the size of $\mathbb{M}$ by removing arcs $(u, v) \in A^c$ (consequently variables y_{uv}) which cannot be part of an optimal MSA solution.

2.3.2 MDD Filtering Procedures for the MSA Problem

Bound-based filtering is an effective method in reducing the size of large-scale problems (Detienne et al., 2016, 2015). For decision diagrams, filtering refers to the process of identifying and deleting arcs (u, v) which do not contribute to a feasible or optimal solution (Hoda et al., 2010; Andersen et al., 2007; Cayrol et al., 1998). Any r-t path using such arcs is either infeasible or not optimal. Therefore, we may delete arc (u, v) without removing any feasible or optimal solution. By construction, all arcs $(u, v) \in A^p$ may be part of a feasible MSA solution. Therefore, the the PSA MDDs are already filtered in that sense. However, not all arcs can participate in an optimal MSA solution. We use two bounds to filter $\mathbb{M}$ based on optimality, namely the Carrillo and Lipman (1988) lower bound, and the additive bounding Fischetti and Toth (1989) upper bound.

Carrillo-Lipman Lower Bound

As the first method, we use the well-known Carrillo and Lipman (1988) lower bound to reduce the size of each pairwise MDD $\mathcal{M}^p$. The procedure generates a lower bound on the objective value of any $\mathcal{M}^p$, and removes any solution that violates that lower bound.

Let $\bar{y}$ be a feasible MSA solution, which may be obtained using any fast heuristic method, e.g., MUSCLE (Edgar, 2004). Further, let $\hat{y}^p$ be the optimal PSA solution of any $\mathcal{M}^p$, and let $f(\bar{y})$ denote the objective value of solution $\bar{y}$. Carrillo and Lipman (1988) prove that $f(\bar{y}) - \sum_{p':p' \neq p} f(\bar{y}^{p'})$ is a valid lower bound for the PSA of pair p, and any PSA solution that violates this bound cannot

be part of an optimal MSA solution. Formally,

$$f(\bar{y}^p) \geq f(\bar{y}) - \sum_{p':p' \neq p} f(\hat{y}^{p'}), \tag{2.13}$$

holds for all $p \in \mathbb{P}$. Now, let $I^{\downarrow}(u)$ denote the value of a longest path from root r to node $u \in U^p$. Similarly, let $I^{\uparrow}(u)$ denote the value of a longest path from a node $u \in U^p$ to terminal t. Using lower bound (2.13), we filter PSA MDDs $\mathcal{M}^p$ by deleting arcs (u, v) such that:

$$I^{\downarrow}(u) + w_{uv} + I^{\uparrow}(v) < f(\bar{y}) - \sum_{p':p' \neq p} f(\hat{y}^{p'}).$$

Lastly, we delete any node $u \in U^c$, which does not have any incoming arc $(v, u) \in \mathbb{M}$ or any outgoing arc $(u, v) \in \mathbb{M}$.

Additive Bounding Upper Bound

Although solving $\mathbf{P}$ is time-consuming, solving its Linear Programming (LP) relaxation is efficient. The LP relaxation of $\mathbf{P}$ is obtained by replacing the binary constraints (2.9), with $0 \leq y_{uv} \leq 1$. The solution of this LP relaxation may be used in filtering procedures for decision diagrams, using the additive bounding procedure (Fischetti and Toth, 1989; Kinable et al., 2017).

Given an optimization problem $\mathbf{P}$, additive bounding solves a series of relaxations $\tilde{P}_1, \ldots \tilde{P}_k$ of P, such that $\tilde{P}_1$ is defined with respect to the original objective coefficients and $\tilde{P}_{k'}$ receives as objective coefficients the reduced costs obtained from $\tilde{P}_{k'-1}$, for $k' = 2, \ldots, k$. Fischetti and Toth (1989) show that the sum of the bounds from $\tilde{P}_1$ to $\tilde{P}_k$ is a valid bound for $\mathbf{P}$.

In the context of our problem, we combine the LP relaxation of $\mathbf{P}$ with the discrete relaxation defined by (2.5)-(2.9), i.e., the longest path formulation on the pairwise PSA MDDs. First, we solve the LP relaxation of $\mathbf{P}$, and denote the resulting upper bound by $\tilde{f}(\mathbf{P})$, its optimal solution by $\tilde{y}$, and the reduced cost of variable y_{uv} by μ_{uv}. We then associate with each arc $(u, v) \in A$ a weight $\tilde{w}_{uv}$ defined as

$$\tilde{w}_{uv} = \begin{cases} \mu_{uv} & \text{if } \tilde{y}_{uv} = 0, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Observe that $\tilde{w}_{uv} \leq 0$ for all $(u, v) \in A$ since $\mathbf{P}$ is a maximization problem. Given a feasible solution $\bar{y}$ for $\mathbf{P}$ with objective value $f(\bar{y})$, the additive bound

$$\tilde{f}(P) + \sum_{(u,v) \in A} \tilde{w}_{uv} y_{uv}, \tag{2.14}$$

is a valid upper bound for $\mathbf{P}$, i.e. $f(\bar{y}) \leq \tilde{f}(P) + \sum_{(u,v) \in A} \tilde{w}_{uv} y_{uv}$.

We can now define an MDD arc filtering rule as follows. Let $\tilde{I}^{\downarrow}(u)$ denote the value of a of a

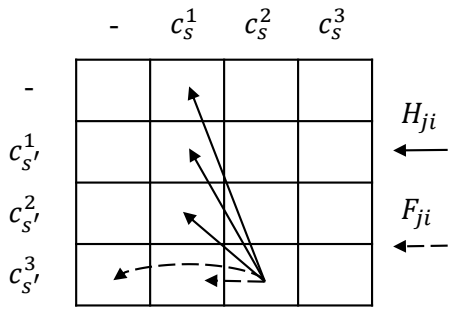


Figure 2.5: Example matrix format for PSA, and update procedure for for cell $\mathcal{C}_{4,3}$.

longest path, with respect to weights $\tilde{w}_{uv}$, from root node $r \in U^P$ to a node $u \in U^p$. Similarly, let $\tilde{I}^{\uparrow}(v)$ denote the value of a longest path, with respect to weights $\tilde{w}_{uv}$, from a node $v \in U^p$ to terminal node $t \in U^p$. Using the upper bound (2.14), we filter the MDD M^p by deleting any arcs (u, v) such that:

$$\tilde{I}^{\downarrow}(u) + \tilde{w}_{uv} + \tilde{I}^{\uparrow}(v) < f(\bar{y}) - \tilde{f}(P).$$

As before, we delete any node $u \in U^c$ that does not have any incoming arc $(v, u) \in \mathbb{M}$ or any outgoing arc $(u, v) \in \mathbb{M}$.

Filtering the MDD During Construction

It is more efficient (in both time and memory) to filter $\mathbb{M}$ during its construction, rather than build $\mathbb{M}$ and filter it thereafter. To that end, we can use the Carrillo-Lipman filtering procedure to filter the PSA MDDs during their construction, as its bound can be calculated before construction of the PSA MDDs. In particular, we can calculate the optimal PSA of all pairs $p = (s, s')$, and generate a heuristic solution $\bar{y}$ (e.g. using MUSCLE) to obtain the Carrilo-Lipman bound (2.13), without constructing the PSA MDDs. The only MDD-dependant step is calculating values $I^{\downarrow}(u), I^{\uparrow}(v)$ that are used to determine whether an arc (u, v) is constructed or not (filtered).

Values $I^{\downarrow}(u)$ can be calculated during construction, as the PSA MDDs are built incrementally from the root node r to the terminal node t. However, values $I^{\uparrow}(v)$ cannot be calculated from the MDDs before they are fully constructed. Fortunately, it is possible to calculate these values using a matrix-format PSA DP method, similar to that of Mott (1999) and Needleman and Wunsch (1970). The matrix-format DP method is suitable to calculate values $I^{\uparrow}(v)$ due to its forward induction on aligning characters $c_s^1, c_s^2, \ldots, c_s^{|s|}$ (recall that the PSA MDD follows a backwards induction). In the interest of space, we refer the interested reader to Mott (1999) and Needleman and Wunsch (1970) for thorough explanations of matrix-format PSA procedure, and briefly discuss the modified algorithm to calculate values $I^{\uparrow}(v)$ in our approach.

We create a $(|s| + 1) \times (|s'| + 1)$ alignment matrix, such as the example given in Figure 2.5 for $|s| = |s'| = 3$. In the matrix-format PSA, the value of cell $\mathcal{C}_{ji}$ at row j and column i of the matrix

gives the objective of the best PSA alignment of the first $i-1$ characters $c_s^i \in s$ to the first $j-1$ characters $c_{s'}^j \in s'$. To accommodate affine and convex penalty functions, we additionally define g_{ji} as the number of consecutive gap alignments contributing to the value of cell $\mathcal{C}_{ji}$. Note that g_{ji} may be reduced to a binary value in affine penalty functions, as the penalty of extending a gap does not depend on the length of that gap. Lastly, we define function $\mathcal{G}(g, x)$ which gives a g length gap penalty cost plus an alignment reward based on the decision x.

The value of $\mathcal{C}_{00}$ is initialized to zero, with $g_{00} = 0$, and the values of the first row and column cells are calculated as consecutive gap alignments: $\mathcal{C}_{0i} = \mathcal{G}(i, -), g_{0i} = i, \mathcal{C}_{j0} = \mathcal{G}(j, -), g_{j0} = j$. The rest of the cells are calculated incrementally by a DP procedure as $\mathcal{C}_{ji} = \max\{F_{ji}, H_{ji}\}$, where $H_{ji} = \max_{j'<j}\left\{\mathcal{C}_{j',i-1} + \mathcal{G}(j-j'-1, c_{s'}^{j'})\right\}, g_{ji} = g_{j',i-1} + j - j' - 1$, and $F_{ji} = \max_{i'<i}\left\{\mathcal{C}_{j,i'} + \mathcal{G}(i-i'-1, -)\right\}$, $g_{ji} = g_{ji'} + i - i' - 1$. Values g_{ji} are updated according to whichever value F_{ji} or H_{ji} gives $\mathcal{C}_{ji}$. Figure 2.5 shows an example schema of determining these values for cell $\mathcal{C}_{43}$. Finally, cell $\mathcal{C}_{|s'|+1,|s|+1}$ gives the optimal PSA objective.

The calculated values of cells $\mathcal{C}_{ji}$ can be used to exactly determine values $I^{\uparrow}(v)$. By definition, $I^{\uparrow}(v), v \in \mathcal{L}_i, state(v) = j > 0$ is the value of the longest path from node v to terminal t. This path corresponds to the best PSA alignment of the first $i-1$ characters $c_s^i \in s$ to the first $j-1$ characters $c_{s'}^j \in s'$, i.e. $\mathcal{C}_{ji}$. However, observe that the above procedure assumes no prior gaps leading to a cell $\mathcal{C}_{ji}$ when calculating its value. For example, when calculating C_{01}, the gap penalty is calculated according to a gap of length 1 in $\mathcal{G}(1, -)$, which gives an opening gap penalty. Although this is correct for $I^{\uparrow}(v), v \in \mathcal{L}_i, state(v) = j > 0$, it is an incorrect value for $I^{\uparrow}(v), v \in \mathcal{L}_i, state(v) = j < 0$. This is because extending a gap leading to node v by g additional gaps is not equal to opening a gap of length g in affine or convex penalty functions. In order to calculate values $I^{\uparrow}(v), v \in \mathcal{L}_i, state(v) = j < 0$, we need to build other matrices.

For affine penalty functions, it is sufficient to build an extra PSA matrix with the assumption that all cells C_{ji} proceed a gap of length one. The function $\mathcal{G}(g, x)$ thus calculates an extension gap penalty when calculating the value of cells $\mathcal{C}_{ji}$. The values calculated in this new matrix are used to determine values $I^{\uparrow}(v), v \in \mathcal{L}_i, state(v) = j < 0$.

For PSA with a convex penalty function, the extension gap penalty additionally depends on the number of consecutive gaps leading to nodes $v : state(v) < 0$. To exactly calculate values $I^{\downarrow}(u), u \in \mathcal{L}_i, state(u) = (j, g), j < 0$ we must solve $|s|$ number of PSA matrices, one per possible value of gap lengths g, which takes a relatively long time. A compromise between time and quality of our estimate for values $I^{\downarrow}(u)$ is to solve 2 PSA matrices. The original matrix assumes that $i > 0, g = 0$ to exactly calculate $I^{\downarrow}(u), u \in \mathcal{L}_i, state(u) = (j, 0), j > 0$, and for the other matrix we assume $i < 0, g = |s| - i - 1$ to give an upper bound for the values of $I^{\downarrow}(u), u \in \mathcal{L}_i, state(u) = (j, g), j < 0$. That is, as $|s| - i - 1$ is the maximum possible value of g at stage i, all values of the second matrix are an upper bound to the exact values of $I^{\downarrow}(u), u \in \mathcal{L}_i, state(u) = (j, g'), j < 0, g' < g$ due to the convex property of the penalty function. Using the values of this second matrix, we estimate all

values $I^{\uparrow}(v), v \in \mathcal{L}_i, state(v) = j < 0$ and filter the PSA MDD during construction. After a PSA MDD is constructed using the estimated values $I^{\uparrow}(v)$, we exactly calculate all values $I^{\downarrow}(u)$, and re-filter the MDD.

This concludes the procedure to filter the PSA MDDs during construction. The next section discussed how to use the filtering procedures for a more effective two phase solution for MSA.

Two Phase Solution Algorithm for MSA Based on Optimistic Filtering

Initial numerical results showed that the filtering algorithms do not prune a significant number of arcs $(u, v) \in \mathbb{M}$, when using (commercial) heuristic solvers to provide the bound $f(\bar{y})$. Unfortunately, obtaining solutions with higher quality quickly is not possible for large-size instances using any other method in the literature.

On the bright side, the discussed filtering procedures do not require the explicit alignments of an MSA solution $\bar{y}$, and operate only using its objective value $f(\bar{y})$. Therefore, it is not required to explicitly generate a solution to filter $\mathbb{M}$, and it is possible to use a guess for the value $f(\bar{y})$. A guess f^g is valid if $f^g \leq f(\hat{y})$, where $\hat{y}$ is an optimal MSA solution. A valid guess close to the optimal objective value $f(\hat{y})$ leads to considerable pruning of $\mathbb{M}$, and consequently a smaller problem to solve. On the other hand, if the guess is invalid, i.e., $f^g > f(\hat{y})$, the filtering procedures may delete the optimal solution $\hat{y}$ from $\mathbb{M}$. In case of an invalid guess, the solution of $\mathbf{P}$ may turn out to be a local optimal solution, or in the worst-case $\mathbf{P}$ becomes infeasible and does not contain any feasible MSA solutions.

The validity of f^g cannot be determined unless the optimal value $f(\hat{y})$ is known. However, it is possible to ensure that $\mathbf{P}$ is feasible after optimistic filtering, by ensuring that it contains at least one feasible MSA solution. A straightforward, yet time-consuming, approach to determine the feasibility of $\mathbf{P}$ after optimistic filtering is to directly solve $\mathbf{P}$ using an MIP solver, and observe the result. A much faster approach is to check whether the filtered $\mathbb{M}$ contains some feasible solution $\bar{y}$, which guarantees that $\mathbf{P}$ is feasible. This is done by ensuring all arcs (u, v) corresponding to variables $y_{uv} \in \bar{y} : \bar{y}_{uv} = 1$ remain in the filtered $\mathbb{M}$.

We consider checking whether $\mathbb{M}$ contains the best found feasible solution $\bar{y}$ (e.g., the solution generated by a heuristic solver). If $\mathbb{M}$ does not contain $\bar{y}$, we reduce the optimistic guess f^g by a step size stp (e.g., one which is chosen heuristically), and re-run the filtering procedures using the updated guess $f^g - stp$. This procedure is repeated until $\mathbb{M}$ contains solution $\bar{y}$, in which case the feasibility of $\mathbf{P}$ is guaranteed.

The optimal solution $\hat{y}^o$ of $\mathbf{P}$ on the resulting optimistically filtered $\mathbb{M}$, may be a local optimal if $f^g > f(\hat{y})$. To ensure global optimality, we solve the MSA problem in two phases. In the first phase, we perform optimistic filtering on $\mathbb{M}$ and solve $\mathbf{P}$ to generate a possibly local optimal solution $\hat{y}^o$. In the second phase, we check global optimality by using the first phase solution bound $f(\hat{y}^o)$ to filter $\mathbb{M}$. As $\hat{y}^o$ is a feasible MSA solution, the resulting filtered $\mathbb{M}$ is guaranteed to contain an

optimal MSA solution $\hat{y}$. The solution of **P** in the second phase either proves optimality of $\hat{y}^o$, or generates an optimal solution $\hat{y}$ for the MSA problem.

The first phase optimal solution $\hat{y}^o$ is guaranteed to have a higher objective value $f(\hat{y}^o) \geq f(\bar{y})$, where $\bar{y}$ is generated by any heuristic MSA solver, proven in Lemma 2.7. In fact, it is noteworthy to say that our numerical results show that either $\hat{y} = \hat{y}^o$ or $f(\hat{y}^o)$ is in worst-case within 97% of the global optimal objective $f(\hat{y})$.

Lemma 2.7. *The solution $\hat{y}^o$ generated by solving **P** on an optimistically filtered $\mathbb{M}$ satisfies $f(\hat{y}^o) \geq f(\bar{y})$, where $\bar{y}$ is any feasible MSA solution.*

Proof. In the first phase $\mathbb{M}$ is guaranteed to contain the best found feasible solution, e.g., $\bar{y}$. Therefore, solving **P** on $\mathbb{M}$ generates a solution $\hat{y}^o$ which is either $\hat{y}^o = \bar{y}$, or satisfies $f(\hat{y}^o) > f(\bar{y})$ due to the maximization objective (2.5). □

This concludes the filtering procedures for $\mathbb{M}$. The filtered $\mathbb{M}$ is expected to be of smaller size and consequently leads to an easier problem to solve. However, solving **P** on the filtered $\mathbb{M}$ is still hard and time consuming using a commercial MIP solver. Our next step to reduce solution difficulty is to use Benders decomposition.

2.4 Logic-based Benders Decomposition for Solving Synchronized PSA MDDs

2.4.1 Logic-based Benders Decomposition

In this section, we design a decomposition algorithm to solve model **P** more effectively than a generic MIP solver. Based on the structure of model **P**, we propose using logic-based Benders decomposition (Hooker, 2002). We refer the interested reader to Hooker and Ottosson (2003) for a thorough explanation of logic-based Benders decomposition.

As the first step, **P** is reformulated such that it only contains binary variables y_{uv}. This is done by relaxing constraints (2.10)-(2.12), and moving them to subproblem:

$$\textbf{SP:} \quad \max \left\{ 0^T \pi \mid (2.10) - (2.12), \pi \geq 0 \right\}.$$

This gives the relaxed master problem:

$$\textbf{RMP:} \quad \max \left\{ (2.5) \mid (2.6) - (2.9) \right\}.$$

The solution $\bar{y}$ of **RMP** may or may not satisfy the MSA order requirement, i.e., constraints

(2.10)-(2.12). To check this, **DSP** the dual of **SP**, is solved at $\bar{y}$:

$$\min \quad M\left(1-\sum_{\substack{(u,v)\in A^p: u\in\mathcal{L}_i^p,\\ state(v)=j}} \bar{y}_{uv}\right)\left(\sum_{(s,s')}\sum_{(i,j)\in(s,s')} \alpha^{ij}_{(s,s')}-\gamma^{ij}_{(s,s')}\right)+\sum_{s\in\mathbb{S},i\in s}\beta^i_s \tag{2.15}$$

$$\text{s.t.} \quad \sum_{s'\in\mathbb{S}:s'>s}\sum_{j\in s'}\alpha^{1j}_{(s,s')}-\sum_{s'\in\mathbb{S}:s'>s}\sum_{j\in s'}\gamma^{1j}_{(s,s')}+\beta^1_s\geq 0 \qquad \forall s\in\mathbb{S}, \tag{2.16}$$

$$\sum_{s'\in\mathbb{S}:s'>s}\sum_{j\in s'}\alpha^{ij}_{(s,s')}-\sum_{s'\in\mathbb{S}:s'>s}\sum_{j\in s'}\gamma^{ij}_{(s,s')}-\beta^i_s+\beta^{i+1}_s\geq 0$$

$$\forall s\in\mathbb{S},\forall i\in s: 2\leq i\leq |s|-1, \tag{2.17}$$

$$\sum_{s'\in\mathbb{S}:s'>s}\sum_{j\in s'}\alpha^{|s|j}_{(s,s')}-\sum_{s'\in\mathbb{S}:s'>s}\sum_{j\in s'}\gamma^{|s|j}_{(s,s')}-\beta^{|s|}_s\geq 0 \qquad \forall s\in\mathbb{S}, \tag{2.18}$$

$$\alpha^{ij}_{(s,s')},\gamma^{ij}_{(s,s')}\geq 0 \qquad \forall (s,s'),\forall (i,j)\in(s,s'), \tag{2.19}$$

$$\beta^i_s\geq 0 \qquad \forall s,\forall i\in s. \tag{2.20}$$

where variables $\alpha^{ij}_{(s,s')},\gamma^{ij}_{(s,s')},\beta^i_s$ correspond to the dual variables associated with constraints (2.10), (2.11), (2.12), respectively. If **DSP** is feasible, then $\bar{y}$ is a feasible solution for **P**. Otherwise, if **DSP** is unbounded, a Benders feasibility cut (2.21) is generated and added to **RMP** to remove the infeasible solution.

$$M\left(\sum_{(s,s')}\sum_{(i,j)\in(s,s')}\bar{\alpha}^{ij}_{(s,s')}-\bar{\gamma}^{ij}_{(s,s')}\right)\left(1-\sum_{\substack{(u,v)\in A^p: u\in\mathcal{L}_i^p,\\ state(v)=j}} y_{uv}\right)\geq -\sum_{s\in\mathbb{S},i\in s}\bar{\beta}^i_s \tag{2.21}$$

We consider any feasible solution generated by the MIP solver when solving **RMP**, and add a corresponding cut (2.21) to **RMP** if that solution is an infeasible MSA solution. The Benders algorithm iterates until the solution of **RMP** leads to a feasible (not unbounded) solution for **DSP**, which indicates an optimal solution for **P**.

Solving **RMP** is significantly easier than solving **P**. However, the rate of convergence to the optimal solution is slow, and it takes many feasibility cuts (2.21) to reach the optimal solution. Cuts (2.21) tend to remove one infeasible solution per iteration of the algorithm, and given the difficulty of solving **RMP** at each iteration, this approach is not effective. To generate stronger cuts, we develop logic-based *column alignment* cuts, which are added to **RMP** together with constraints (2.21). Our column-alignment cuts are in fact a projection of the"generalized transitivity inequalities" used in Althaus et al. (2006) onto the MDD variable space.

Column alignment cuts are designed to enforce the alignment of a pair of characters $c^i_s, c^j_{s'}$, if those characters are both aligned with a third character $c^k_{s''}$. Although this alignment is guaranteed

for **P** due to constraints (2.10), (2.11), it is not enforced when constraints (2.10), (2.11) are relaxed, i.e., in **RMP**. Cuts enforcing this alignment can thus be used to increase the solution quality of **RMP** and aid in improving convergence. We model such cuts by considering a triple of characters $\left(c^i_s, c^j_{s'}, c^k_{s''}\right)$ of sequence pairs $p = (s, s'), p' = (s, s''), p'' = (s', s'')$, and imposing the following constraints:

$$\sum_{u:u\in\mathcal{L}_i^{p'}} \sum_{\substack{v:(u,v)\in A^{p'},\\ state(v)=k}} y_{uv} + \sum_{u:u\in\mathcal{L}_j^{p''}} \sum_{\substack{v:(u,v)\in A^{p''},\\ state(v)=k}} y_{uv} \leq 1 + \sum_{u:u\in\mathcal{L}_i^{p}} \sum_{\substack{v:(u,v)\in A^{p},\\ state(v)=j}} y_{uv} \tag{2.22}$$

Proposition 2.8 proves that the triple consideration of sequences (s, s', s'') is sufficient to ensure correct column alignment for any number of sequences above three.

Theorem 2.8. *It suffices to consider all* $\binom{|\mathbb{S}|}{3}$ *triples of sequences, to ensure correct column alignment in MSA, for any number of sequences.*

Proof. Consider a graph with nodes denoting characters c and arcs denoting alignment of characters. A column in the MSA matrix corresponds to a set of connected nodes in the graph. Observe that aligned characters respect the column alignment requirement if the induced subgraph of their nodes forms a clique. If constraints (2.22) are satisfied, then for any connected component, all induced sub-components of size three are cliques. This implies that the component is itself a clique. □

It is possible to generate all possible cuts (2.22) a priori and add them to **RMP**; however, doing so makes **RMP** much harder to solve. In fact, adding all such cuts makes the LP relaxation of **RMP** unsolvable within 10 hours of computation, even for small instances. On the other hand, not all cuts (2.22) contribute to the solution of **RMP**, and most are non-binding for its optimal solution. We can thus only add a subset of these constraints to **RMP**, and achieve the same results in much lower computation time. The problem is that we do not know beforehand which cuts (2.22) are binding, and this is only observed given a solution to **RMP**.

We thus add cuts (2.22) to **RMP** as logic-based cuts within the Benders decomposition algorithm. At each iteration, in addition to Benders feasibility cuts (2.21), we generate violated cuts (2.22) and add them to **RMP**. The separation procedure to generate cuts (2.22) involves modeling the alignment determined by the solution of **RMP** as a graph (following the same design as in the proof of Theorem 2.8), and finding any triple of nodes which do not form a clique for any connected component. As proven in Althaus et al. (2006), the time complexity of finding such cuts is bounded by $O(n^4)$. Figure 2.6 shows the overall Benders decomposition algorithm. Note that the heuristic procedure shown in Figure 2.6 is discussed in §2.4.3.

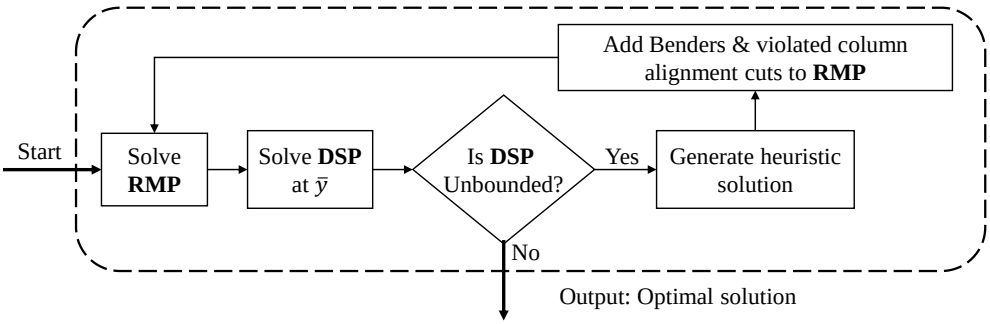


Figure 2.6: Logic-based Benders decomposition algorithm.

2.4.2 Warm-starting the Logic-based Benders Decomposition Algorithm

The Benders decomposition algorithm solves **RMP** once per iteration. Repeatedly solving **RMP** is time consuming, and lowers the effectiveness of the algorithm. Moreover, cuts generated in early iterations of Benders decomposition tend to be of low quality, and not worth the computational effort of solving **RMP**. One approach to increase the effectiveness of Benders decomposition is to warm-start the relaxed master problem (McDaniel and Devine, 1977). The warm-start algorithm is an iterative process that consists of solving the LP relaxation of **RMP** at each iteration, and adding to it violated column alignments cuts (2.22), and Benders feasibility cuts (2.21). The warm-start algorithm iterates until there is no improvement in the LP bound $\tilde{f}(\mathbf{P})$. Warm-starting enables fast generation of valid cuts, without the need to solve an MIP problem at each iteration. The resulting master problem has a tighter linear relaxation, and is used to start the Benders decomposition.

Warm-starting has additional benefits for our problem, as it provides opportunities to further reduce the size of $\mathbb{M}$ using the additive bounding filtering procedure. At each iteration of the warm-start algorithm, the LP relaxation of **RMP** gives an updated objective bound $\tilde{f}(\mathbf{P})$ and reduced costs μ_{uv}. These solutions can be used in the additive bounding filtering procedure to further filter $\mathbb{M}$. At each iteration, **RMP** is redefined to optimize over the newly filtered $\mathbb{M}$, which is smaller in size and easier to solve. Figure 2.7 gives the overall warm-start algorithm.

2.4.3 Primal Heuristic Procedure to Generate Feasible MSA Solutions

As the last component of our solution framework, we develop a heuristic procedure that modifies an infeasible MSA solution $\tilde{y}$ derived from **RMP**, to generate a feasible MSA solution $\bar{y}$. The heuristic is executed within the Benders decomposition, and used to possibly generate solutions $\bar{y}$ with higher quality than those generated by heuristic MSA solvers. Such solutions $\bar{y}$ are used to update the incumbent, and provide better bounds $f(\bar{y})$ for the MDD filtering algorithms.

The heuristic algorithm procedure consists of two steps. In the first step, the algorithm removes a set of alignments from the infeasible solution $\tilde{y}$, such that the resulting solution does not violate MSA requirements. To determine which set of alignments should be removed, we sort all made

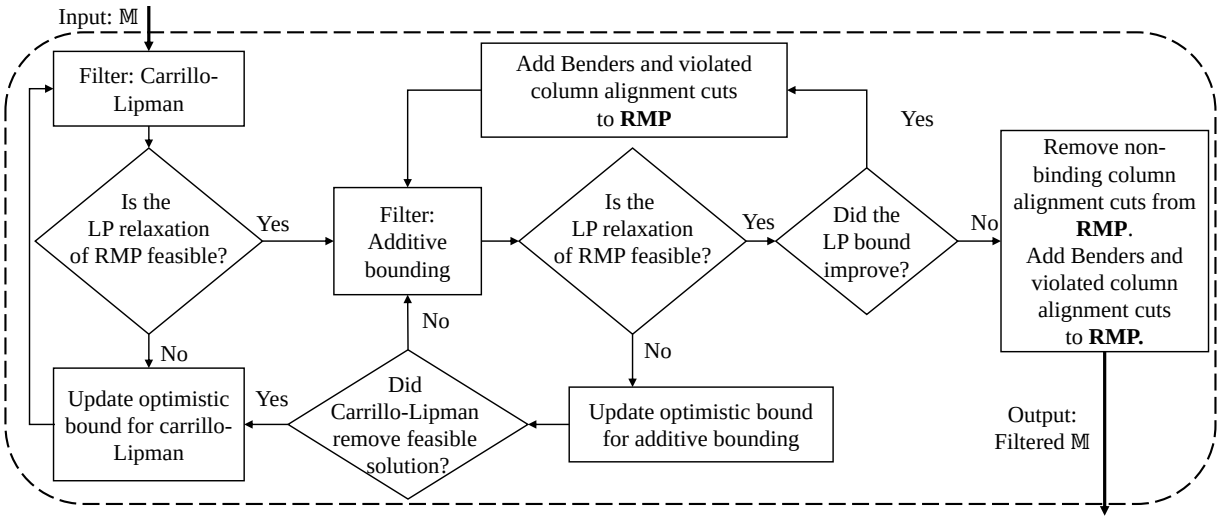


Figure 2.7: Warm-start algorithm.

alignments $\tilde{y}_{uv} = 1$ in ascending order of their objective coefficient w_{uv}. We then remove alignments in the sorted order until the resulting solution does not violate MSA requirements.

In the second step, we use the feasible solution generated in the first step, and add to it any possible feasible alignments that improve the overall objective. This is done by feeding the solution to a heuristic solver, e.g MUSCLE (Edgar, 2004), to refine the alignment.

The heuristic procedure is executed for every feasible solution generated by the MIP solver when solving **RMP**, in each iteration of the Benders algorithm. At any point, if the heuristic solution improves the bound $f(\bar{x})$ (or f^g in optimistic filtering) used in the filtering procedures, we re-filter $\mathbb{M}$ using the new bound.

This concludes all procedures for the complete MSA solution algorithm, summarized in Figure 2.8. The second phase to check global optimality is executed after the first phase converges to an optimal solution. The second phase is identical to the first phase algorithm, with the exception of using the generated solution of the first phase for exact filtering, instead of the guessed bounds used to optimistically filter the PSA MDDs.

2.5 Numerical Experiments

We describe two sets of numerical experiments on our MDD solution approach for MSA. In our first set of experiments, we consider an affine penalty function and analyze the performance of solving instances from BaliBASE benchmark version 4 (BaliBASE4, 2017). We evaluate the effects of the filtering algorithms in reducing the size of $\mathbb{M}^c$, and compare the quality of our solutions to a state-of-the-art heuristic MSA solver. In our second set of numerical tests, we consider both convex and affine penalty functions and compare our algorithm to the best exact approach in the literature on

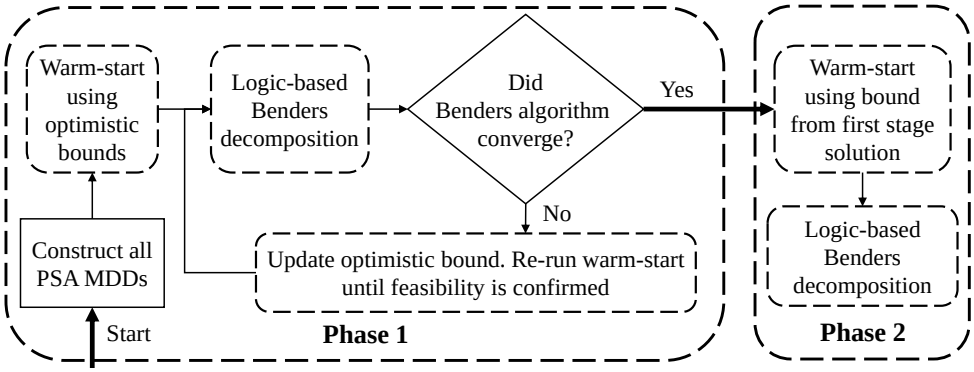


Figure 2.8: Overall algorithm for solving MSA by synchronized PSA MDDs.

instances from BaliBASE benchmark version 1.

2.5.1 Experimental Setup

For alignment rewards we use the BLOSUM62 substitution matrix, and for our convex penalty function we use $8 + 2g + 2\sqrt{g}$, where g is the length of a gap in the alignment. This function is adopted from Althaus et al. (2006), and shown to provide accurate alignments. For our tests using an affine penalty function, we use $12 + 2.24g$, where the opening and extension penalties are obtained from a least squares regression approximation of function $8 + 2g + 2\sqrt{g}$, for gap lengths $g = 1, ..., 50$.

For optimistic filtering, we heuristically choose a step size of $stp = \left(\sum_p f(\hat{y}^p) - f(\bar{y})\right) / (|S| * 75/4 - 50)$, where $\bar{y}$ is the solution obtained from a heuristic MSA solver. We use two independent optimistic guesses for the Carrillo-Lipman (f^g_{CL}) and additive bounding (f^g_{AB}) filtering procedures. An initial value of $\sum_p f(\hat{y}^p) - stp$ is used for both guesses. At any time during the warm-start algorithm, if f^g_{AB} turns out to be greater than the value of the LP objective $\tilde{f}(\mathbf{P})$, we set $f^g_{AB} = \tilde{f}(\mathbf{P}) - stp$.

All algorithms are coded in C++, CPLEX 12.7.1 is used as the commercial MIP solver, and MUSCLE 3.8.31 (Edgar, 2004) is used as the commercial MSA heuristic solver. MUSCLE was chosen as it consistently provided the best alignment for our objective setting compared to other heuristic solvers. All tests are executed on a PC with Intel Xeon E5345 2.33 GHz processor, with 24GB of memory on an Ubuntu 14.04.6 platform. Tests are limited to one thread of the CPU, and a 10 hour time limit is chosen for the overall computation time for each instance. Our algorithm is open-source and available to download[1].

[1] https://github.com/aminhn/MSAMDD

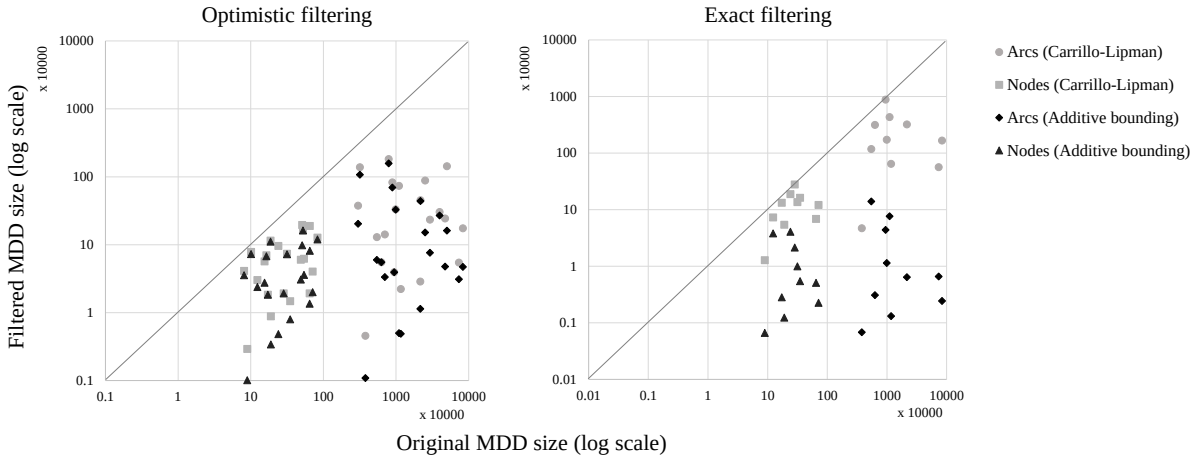


Figure 2.9: Effects of filtering algorithms on the size of $\mathbb{M}^c$.

2.5.2 Experiments on Balibase Benchmark 4 and Comparison to Muscle

For our first set of experiments, we use the latest BaliBASE benchmark, i.e. version 4 (BaliBASE4, 2017), and initially limit our tests to instances with at most 1000 total number of characters, i.e., $\sum_{s \in S} |s| \leq 1000$. We then attempt to solve number of the largest size instance that can fit in our system memory, to investigate the limits of our algorithm in solving larger sized problems. All test in this section use an affine gap penalty function, as to the best of our knowledge no commercial heuristic solver accommodates convex penalty functions.

Effects of Filtering Algorithms

To evaluate the effects of the filtering algorithm, we report in Figure 2.9 the relative size of the filtered $\mathbb{M}^c$ compared to its original size, in terms of number of arcs $|A^c|$ and nodes $|N^c|$. Results are reported both for optimistic filtering in the first phase, and exact filtering in the second phase (for instances that did not reach the time limit in the first phase).

Results show that the filtering algorithms filter a considerable number of nodes and arcs in most instances, in particular using the additive bounding filtering procedure. On average, the Carrillo-Lipman filtering procedure filters a considerable 94.5% of arcs, and 74.6% of nodes in $\mathbb{M}$ for optimistic filtering. The additive bounding filtering procedure is performed after the Carrillo-Lipman filtering, and filters on average 47.3% of the remaining arcs, and 31.1% of the remaining nodes in the filtered $\mathbb{M}$. In exact filtering for the second phase, the Carrillo-Lipman filtering procedure filters 75.4% of arcs, and 52.8% of nodes, with the additive bounding filtering procedure filtering 98.2%, 88.9% of the remaining arcs and nodes, respectively. It is noteworthy that in some instances the Carrillo-Lipman filtering procedure does not filter many nodes and arcs, but the additive bounding procedure is still able to considerably filter $\mathbb{M}$.

Comparison to MUSCLE

Table 2.1 reports results on the CPU time, and accuracy of the proposed algorithm compared to the heuristic MSA solver MUSCLE. Instances are divided into two groups, with the first group reporting results for any instance in the benchmark with less than 1000 total characters, and the second group reporting results for a number of larger-sized instances, up to 1500 total number of characters. The "Tot-CPU" column reports the total CPU time (phase one plus phase two). The "Conv-Gap" column reports the Benders convergence gap of the first phase, calculated as $\frac{100(f(\mathbf{RMP})-f(\bar{y}))}{f(\bar{y})}$, where $\bar{y}$ is the best found solution. Note that $f(\mathbf{RMP})$ may not be the global upper bound in phase one due to optimistic filtering. Column "Impr-Mus" reports the percentage of improvement $\frac{100\left(f(\bar{y})-f(\bar{y}^h)\right)}{f(\bar{y}^h)}$ made by our algorithm compared to the initial heuristic solution $\bar{y}^h$ generated by MUSCLE. The final columns report the CPU time, global optimality gap, and improvement of solution $\bar{y}$ in the second phase compared to the optimal solution of the first phase. For the latter, a 0% improvement indicates that the first phase solution was in fact the global optimal MSA solution, and a dash indicates that the second phase was not executed, as the first phase reached the imposed time limit.

The majority of CPU-time was spent in the warm-start algorithm and solving the MIP formulation of **RMP**. On average, 50.5% of the time was consumed in the warm-start algorithm, and 42.8% of the total CPU time was spent solving **RMP**. The primal heuristic procedure generally spends a few minutes, on average 3.9% of total CPU-time, and on average improves the solution generated by MUSCLE by 33.3%. Improvement is as high as 285.8% such as in instance BB11001, which was solved within 7 seconds.

Out of the 24 considered instances, the algorithm solves 12 instances to optimality. This is true for instances with as many as 9 sequences, or 1467 total characters. In 11 of the 12 instances solved, the solution found in phase one proved to be the global optimum, with the remaining instance within 99.16% of global optimality. To the best of our knowledge, our algorithm is the first to solve these Benchmarks to global optimality, for our considered affine objective function. The convergence gap is generally low for instances that did not converge within the 10 hour time limit, with the exception of instances BB12041. Our conjecture is that the upper bound in such instances is relatively tight, and the gap percentages are mainly due to lower quality incumbents solutions.

It is noteworthy that all instances with less than 1000 total characters used at most 4GB of memory, with the larger instances using up to 9GB of memory. Attempting to solve larger instances with 2000 total characters failed due to the algorithms exhausting the 24GB system memory.

2.5.3 Experiments on Balibase Benchmark 1 and Comparison to MWT

For a comparison to the best exact approach in the literature, we compare to the MWT algorithm of Althaus et al. (2006). We do not compare to the DP approach as it does not scale to the size of our considered instances, or the convex optimization approach of Yen et al. (2016) which is limited to aligning sequences of length at most 50. We also do not report results of solving the original

Table 2.1: Results on Balibase benchmark 4, and comparison to MUSCLE. CPU time is given for the overall algorithm, phase one, and phase two. The convergence gap (Cnv-Gap) in phase one, and improvement of the best found solution compared to MUSCLE (Imp-Mus). The last columns give the CPU time and global optimality gap in phase two, and improvement of the best found solution compared to phase one (Impr-Ph1).

Instance	Tot-CPU	Phase one			Phase two		
		CPU	Cnv-Gap%	Imp-Mus%	CPU	Gap%	Imp-Ph1%
BB11001 (4/345)	7	4	**0**	285.8	3	**0**	0
BB11002 (8/784)	36,993	36,993	24.4	17.5	-	-	-
BB11009 (4/864)	36,553	36,553	13.0	0	-	-	-
BB11013 (5/358)	36,140	36,140	13.2	19	-	-	-
BB11021 (4/469)	36,176	36,176	35.8	0	-	-	-
BB11022 (4/472)	36,234	36,234	11.3	4	-	-	-
BB11025 (4/331)	36,106	36,106	11.0	16.4	-	-	-
BB11029 (4/408)	13,030	7,998	**0**	22.2	5,033	**0**	0
BB11035 (5/487)	36,118	36,118	39.0	0	-	-	-
BB12003 (8/570)	1,474	327	**0**	12.3	1,147	**0**	0
BB12006 (4/928)	490	384	**0**	9.2	106	**0**	0
BB12009 (5/563)	2,302	669	**0**	25.7	1,633	**0**	0.84
BB12014 (9/875)	36,645	36,645	6.7	16.2	-	-	-
BB12020 (4/502)	104	66	**0**	28.6	38	**0**	0
BB12021 (6/454)	378	105	**0**	33	272	**0**	0
BB12024 (4/973)	649	493	**0**	17.7	156	**0**	0
BB12025 (4/808)	36,268	36,268	28.0	0	-	-	-
BB12032 (9/595)	12,419	9,561	**0**	165	2,858	**0**	0
BB12040 (5/661)	261	123	**0**	20.2	138	**0**	0
BB12041 (7/795)	36,114	36,114	684.8	76.9	-	-	-
BB40010 (9/981)	38,258	38,258	**0**	0.9	-	-	-
BB11012 (4/1456)	5,329	4,806	**0**	21.4	523	**0**	0
BB11028 (10/1588)	36,215	36,215	46.0	0	-	-	-
BB12012 (4/1494)	38596	38596	20.1	0	-	-	-
BB12036 (7/1467)	3,873	1,558	**0**	6.8	2,315	**0**	0

model **P** directly using a commercial solver (e.g. CPLEX), or a classical Benders decomposition algorithm without logic-based cuts 2.21. These algorithms performed poorly, and terminated with considerably high gaps to optimality within the imposed time limit, even smaller instances.

Unfortunately, Althaus et al. (2006) were not able to find and send us the MWT algorithm, and so we base our comparison on their reported results for the BaliBASE benchmark version 1. We remark that CPU time results of Althaus et al. (2006) were reported in 2006, and although their CPU has higher clock speed of 3.06GHz compared to our 2.33GHz (note that we limit our tests to one core of the CPU for fairer comparison), their results are still likely faster when executed on today's PCs. Moreover, for a fairer comparison, we analyzed the speedup from CPLEX 9.0 used in the MWT branch-and-cut algorithm to CPLEX 12.7.1 used in our algorithm.[2] This is measured by the speedup of solving the linear relaxation of our **RMP** for all considered instances. Note that the only relevant speedup involves the solution of linear programs, as "the running time for the branch-and-cut algorithm was dominated by the time required for solving the LPs" (Althaus et al., 2006), and no MIP is solved in any part of the MWT algorithm to the best of our understanding. Our results showed that the geometric mean of CPU time for CPLEX 12.7.1 is 15.4% to 24.0% faster than CPLEX 9. This range is obtained by considering all instances (15.4%), or only instances which had a solution time above 1 second (24%). We therefore adjust the CPU time of (Althaus et al., 2006) by a reporting a speedup range between 15.4% to 24% for their reported times.

Table 2.2 reports the comparison of the MDD and MWT approaches. Following the setting of Althaus et al. (2006), instances are divided into three groups, where the first group consists of similar sequences with identity $> 35\%$, group two sequences have $20 - 40\%$ identity, and group three consists of dissimilar sequences with $< 25\%$ identity. Results are evaluated based on the total CPU time, and gap to optimality (Gap%). We also report the convergence gap (Cnv-Gap) of our MDD approach in the first phase. We perform test under both an affine and convex penalty function. For the remainder of this section, we refer to the MDD algorithm under the convex or affine penalty function as the convex or affine MDD algorithm, respectively. Results of the convex MDD algorithm are used as comparison to the results of the MWT approach, and results of the affine MDD algorithm serve as an approximation to the optimal convex solution. To evaluate the quality of this approximation, the last column of Table 2.2 reports the improvement of the best found solution using the convex MDD algorithm $\bar{y}_{cnv}$, over the best found solution of the affine MDD algorithm $\bar{y}_{aff}$. This is done by calculating the objective of $\bar{y}_{aff}$ using the convex penalty function, and obtaining the percentage of improvement compared to $\bar{y}_{cnv}$.

Results show that our algorithm is competitive with the MWT approach in aligning the highly similar sequences of the first two group of instances. The MWT approach is effective for such instances, as the root node value of its branch-and-cut tree is near optimal (in some instances optimal), and not many cuts and branches are required to converge to the the optimal solution. For

[2]We thank Ed Klotz for sharing the CPLEX 9.0 library.

Table 2.2: Results on Balibase benchmark 1 and comparison to MWT. Results include the reported CPU time, adjusted CPU time due to CPLEX version (CPU-adj) and optimality gap for MWT, CPU time, convergence gap (phase one gap), and global optimality gap (phase two gap) for our MDD approach under affine and convex penalty functions. The last column (Imp-Aff) gives the improvement of the best found solution under the convex penalty function, over the best found solution under the affine penalty function evaluated by the convex function.

Instance	MWT (convex)			MDD (affine)			MDD (convex)			
	CPU	CPU-adj	Gap%	CPU	Cnv-Gap%	Gap%	CPU	Cnv-Gap%	Gap%	Imp-Aff%
1aho (5/320)	194	[147, 165]	0	17	0	0	44	0	0	0.02
1csp (5/339)	2	[2, 2]	0	14	0	0	28	0	0	0
1dox (4/374)	428	[325, 364]	0	29	0	0	43	0	0	1.35
1fkj (5/517)	315	[239, 268]	0	64	0	0	186	0	0	0
1fmb (4/400)	2	[2, 2]	0	20	0	0	26	0	0	0
1krn (5/390)	17	[13, 14]	0	35	0	0	44	0	0	0
1plc (5/470)	34	[26, 29]	0	49	0	0	91	0	0	0
2fxb (5/287)	4	[3, 3]	0	5	0	0	15	0	0	0
2mhr (5/572)	30	[23, 26]	0	76	0	0	215	0	0	0
9rnt (5/499)	20	[15, 17]	0	40	0	0	78	0	0	0
1aab (4/291)	16	[12, 14]	0	2	0	0	8	0	0	0
1fjlA (6/398)	471	[358, 400]	0	67	0	0	91	0	0	0.32
1hfh (5/606)	1,782	[1,354, 1,515]	0	188	0	0	1,352	0	0	0.16
1hpi (4/293)	469	[356, 399]	0	9	0	0	19	0	0	0.94
1csy (5/510)	564	[429, 479]	0	120	0	0	430	0	0	0
1pfc (5/560)	2,438	[1,853, 2,072]	0	287	0	0	1,106	0	0	0.92
1tgxA (4/239)	128	[97, 109]	0	17	0	0	27	0	0	0.38
1ycc (4/426)	36,124	[27,454, 30,705]	14.1	505	**0**	**0**	1,584	**0**	**0**	0.59
3cyr (4/414)	415	[315, 353]	0	37	0	0	102	0	0	0.83
451c (5/400)	45,186	[34,341, 38,408]	582.7	21,482	**0**	**0**	37,184	88.3	-	-11.02
1aboA (5/297)	36,494	[27,735, 31,020]	38.1	36,002	18.8	-	36,003	39.3	-	-21.01
1idy (5/269)	37,402	[28,426, 31,792]	24.2	22,776	**0**	**0**	36,003	4.1	-	-2.01
1r69 (4/277)	36,375	[27,645, 30,919]	17.2	5,554	**0**	**0**	12,647	**0**	**0**	0
1tvxA (4/242)	37,713	[28,662, 32,056]	60.06	36,005	0	1.9	36,006	12.3	-	-10.06
1ubi (4/327)	37,851	[28,767, 32,173]	37.7	19,541	**0**	**0**	36,001	7.0	-	-7.52
1wit (5/484)	38,264	[29,081, 32,524]	16.2	1,098	**0**	**0**	12,281	**0**	**0**	1.27
2trx (4/362)	36,006	[27,365, 30,605]	50.05	158	**0**	**0**	479	**0**	**0**	0

such instances, MUSCLE is also able to produce high quality solutions which are either optimal or near optimal. Using the initial heuristic solutions, The convex MDD approach is able to significantly filter $\mathbb{M}^c$, and also solves all instances within a few minutes.

The MWT approach is not effective for instances with low similarity, and does not solve such instances withing 10 hours of computation. The convex MDD algorithm solves 22 out of the 27 instances to optimality, and reaches relatively low optimality gaps in the remaining instances, except for instances 451c and 1aboA. In general, the MWT approach requires many cuts and branches to reach the optimal solution if its root node objective is not near optimal. For this reason, it may be the case that the results of the MWT algorithm do not improve significantly when solving harder instances, even using today's faster PCs.

The MDD approach performs well under an affine penalty function setting. The affine MDD algorithm is able to solve all but 2 of the instances, and reaches low gaps in the unsolved instances. The solution time is also considerably lower compared to the convex MDD algorithm, while solution quality remains high. The solutions obtained from the affine MDD algorithm are in most cases within 99% of the optimal convex solution. In fact, for harder instances not solved to optimality by the convex MDD algorithm, the affine MDD algorithm generates better results evaluated under a convex penalty function.

In terms of memory requirements, the affine MDD algorithm uses at most 4GB for any instance solved using the affine penalty function. The memory requirement is much higher for the convex MDD algorithm, with the 451c instance consuming close to 22GB of memory. This shows that although our MDD approach has a worst-case polynomial space complexity, its initial memory consumption is much higher compared to the MWT algorithm.

To summarize the numerical experiments, our MDD approach to MSA gives favorable results compared to the best exact approach in the literature, and closes a number of benchmark instances for the first time under our considered convex and affine penalty functions. For instances not solved to optimality, our algorithm is able to generate solutions with much higher quality compared to a state-of-the-art heuristic MSA solver.

2.6 Conclusion

This chapter developed an exact solution approach for the Multiple Sequence Alignment (MSA) problem. We considered MSA as the simultaneous solution to all Pairwise Sequence Alignments (PSA), which are modeled by dynamic programming and represented using Multi-valued Decision Diagrams (MDD). We used the collection of all PSA MDDs as the underlying graph structure to represent the MSA problem as a Mixed-Integer Program (MIP). The MIP problem synchronizes the solution to all PSA MDDs using side constraints, for the first time, in polynomial space complexity.

We designed two filtering procedures to reduce the size of the feasible set and lead to easier problems to solve. Using the filtering procedures, we increased the effectiveness of our algorithm

by developing a two phase solution approach. In the first phase, the PSA MDDs are filtered aggressively using an optimistic guess for the bound used in the filtering procedures, resulting in a heuristic optimization. The advantage of optimistic filtering is that it filters a considerable size of the feasible solution set and enables a relatively fast algorithm to obtain near optimal solutions. On average, 96.3% of arcs, and 81.1% of nodes in our PSA MDDs were pruned using optimistic filtering, while not pruning the optimal solution in almost all solved instances. Global optimality is ensured by re-running the algorithm in a second phase with exact filtering using the bound from the first phase optimal solution.

Lastly, we designed a logic-based Benders decomposition algorithm to solve **P**. We developed a fast primal heuristic and integrated it into the Benders decomposition algorithm. On average, our heuristic solver improved the solution quality of a state-of-the-art MSA heuristic solver by 33.3%, and was able to improve the solution of 19 out of the 25 instances considered. The logic-based Benders algorithm was able to solve 37 out of 51 Benchmark instances for an affine penalty function, and close them for the first time.

Chapter 3

Constraint-based Sequential Pattern Mining

3.1 Introduction

Sequential Pattern Mining (SPM) is a fundamental data mining task with a large array of applications in marketing, health care, finance, and bioinformatics, to name a few. Frequent patterns are used, e.g., to extract knowledge from data within decision support tools, to develop novel association rules, and to design more effective recommender systems. We refer the reader to Fournier-Viger et al. (2017) for a recent and thorough review of SPM and its applications.

In practice, mining the entire set of frequent patterns in a database is not of interest, as the resulting number of items is typically large and may provide no significant insight to the user. It is hence desirable to restrict the mining algorithm search to smaller subsets of patterns that satisfy problem-specific constraints. For example, in online retail click-stream analysis, we may seek frequent browsing patterns from sessions where users spend at least a minimum amount of time on certain items that have specific price ranges. Such constraints limit the output of SPM and are much more effective in knowledge discovery, as compared to an arbitrary large set of frequent click-streams.

A naïve approach to impose constraints in SPM is to first collect all unconstrained frequent patterns, and then to apply a post-processing step to retain patterns that satisfy the desired constraints. This approach, however, may be expensive in terms of memory requirements and computational time, in particular when the resulting subset of constrained patterns is small in comparison to the full unconstrained set. Constraint-based sequential pattern mining (CSPM) aims at providing more efficient methods by embedding constraint reasoning within existing mining algorithms Pei et al. (2007); Negrevergne and Guns (2015). Nonetheless, while certain constraint types are relatively easy to incorporate in a mining algorithm, others of practical use are still challenging to handle in a general and effective way. This is particularly the case of non-monotone constraints representing,

e.g., sums and averages of attributes.

Contributions. In this chapter, we propose a novel representation of sequential database using a multi-valued decision diagram (MDD), a graphical model that compactly encodes the sequence of items and their attributes by leveraging symmetry. The MDD representation can be augmented with constraint-specific information, so that constraint satisfaction is either guaranteed or enforced during the mining algorithm. Finally, as a proof of concept, we implement a general prefix-projection algorithm equipped with an MDD to enforce several constraint types, including complex constraints such as average ("avg") and median ("md"). To the best of our knowledge, this work is the first to consider the "sum," "avg," and "md" constraints with arbitrary item-attribute association within the pattern mining algorithm. Lastly, we provide an experimental comparison on real-world benchmark databases, and show that our approach is competitive with or superior to a state-of-the-art CSPM algorithm.

3.2 Related Work

Research in CSPM has primarily focused on exploiting special properties of constraints, such as monotonicity or anti-monotonicity, to guarantee the feasibility of pattern extensions in the mining algorithm Garofalakis et al. (1999); Zaki (2000); Lin and Lee (2005); Bonchi and Lucchese (2005); Chen and Hu (2006); Pei et al. (2007); Nijssen and Zimmermann (2014); Mallick et al. (2014); Aoga et al. (2017). Constraint types that do not possess such properties remain a challenge for CSPM algorithms, although some of these have been successfully incorporated in more general item-set mining on databases where events have no specific order Soulet and Crémilleux (2005); Bistarelli and Bonchi (2007); Bonchi and Lucchese (2007); Le Bras et al. (2009); Leung et al. (2012), as well as in CSPM when items and attributes are interchangeable Pei et al. (2007).

Recently, constraint programming (CP) has emerged as a successful tool for CSPM Negrevergne and Guns (2015); Kemmar et al. (2016, 2017); Aoga et al. (2017); Guns et al. (2017). CP search techniques, albeit general, can potentially be more efficient when compared to specialized CSPM algorithms. Nonetheless, they still rely on constraint-specific properties to effectively prune undesired patterns.For example, Aoga et al. (2017) show how to effectively implement a number of prefix anti-monotone constraints into CP, but indicate that post-processing is still required to handle monotone constraints such as the minimum span.

Graphical representations of a database have been shown to be effective in item-set mining Han et al. (2004); Pyun et al. (2014); Borah and Nath (2018) and SPM Masseglia et al. (2009). Previous works have also applied binary decision diagrams as a database modeling tool Loekito and Bailey (2006, 2007); Loekito et al. (2010); Cambazard et al. (2010), which are effective when the sequences of the database are similar, but typically do not scale otherwise. We show that our MDD representation retains its size regardless of the similarity between sequences, and provides a more concise representation in the context of SPM.

Table 3.1: Example $\mathcal{SD}$, with attributes of time and price.

S_{ID}	Sequence: $\{(item, time, price)\}$
1	$\langle (B,1,5),(B,3,3)\rangle$
2	$\langle (B,3,3),(A,8,1),(B,9,3)\rangle$
3	$\langle (C,2,1),(C,5,2),(A,8,3)\rangle$

3.3 Problem Definition

We next formally describe the SPM problem and then discuss the handling of constraints.

3.3.1 The SPM Database and Mining Algorithm

The SPM database consists of a set of events, which are modeled by a set of literals I denoted by *items*. Items $i \in I$ are associated with a set of *attributes* $\mathbb{A} = \{\mathcal{A}_1, ..., \mathcal{A}_{|\mathbb{A}|}\}$; for example, attributes can be price, quality, or time. A *sequence database* $\mathcal{SD}$ is defined as a collection of N item sequences $\{S_1, S_2, \ldots, S_N\}$, where all sequences are ordered with respect to the same attribute $\mathcal{A} \in \mathbb{A}$; e.g., occurrence in time. Table 3.1 illustrates an example $\mathcal{SD}$ with $N := 3$, $|I| := 3$, and $M := \max_{n \in \{1,\ldots,N\}} \{|S_n|\} = 3$, where items $i \in I$ are associated with time and price attributes.

The SPM task asks for the set of frequent *patterns* within $\mathcal{SD}$. A pattern $P = \langle i_1, i_2, \ldots, i_{|P|}\rangle$ is a subsequence of some $S \in \mathcal{SD}$. Let $S[j]$ denote the j^{th} position (i.e., item) of sequence S. A subsequence relation $P \preceq S$ holds if and only if there exists an embedding $e : e_1 \leq e_2 \leq ... \leq e_{|P|}$ such that $S[e_j] = i_j, i_j \in P$. For example, $P = \langle A, B\rangle$ is a subsequence of $S = \langle A, B, C, B\rangle$ with two possible embeddings $(1, 2)$ or $(1, 4)$. We define a super-sequence relation $S \succeq P$ analogously, with "$\leq$" replaced by "$\geq$". A pattern is frequent if it is a subsequence of at least θ number of sequences in $\mathcal{SD}$, where θ is a given frequency threshold.

The two best-known mining algorithms for SPM are the Apriori algorithm introduced by Agrawal et al. (1994), and the prefix-projection algorithm introduced by Han et al. (2001). Both are iterative procedures and operate by extending frequent patterns one item at a time. In Apriori, candidate patterns are generated by expanding a pattern with all available items, and thereafter checking the frequency of generated candidates. As candidates may or may not be frequent, the Apriori algorithm suffers from the exponential explosion of the number of generated candidates and redundancy. The prefix-projection algorithm, in turn, operates by projecting each sequence $S \in \mathcal{SD}$ onto a smallest subsequence $\bar{S} = \langle i_1, i_2, \ldots, i_j\rangle$, denoted by *prefix*, and searching for frequent items in this reduced database. Any sequence that is obtained by extending a frequent prefix is guaranteed to be frequent in the original database. Prefix-projection is more efficient than the Apriori algorithm as it rules out infrequent patterns more effectively, but it requires the full database to be in memory Han et al. (2001).

Table 3.2: Characterization of constraints as monotone (M), anti-monotone (AM), or non-monotone (NM) for SPM.

Name	**Constraint := definition**	**M**	**AM**	**NM**
Maximal	$C_{mxl}(P) := \nexists P' \in \mathcal{SD} : P \prec P'$	•		
Sup-Patt	$C_{spt}(P) := \exists P' \in \mathcal{SD} : P' \prec P$	•		
Length	$C_{len}(P) \geq c := \lvert P \rvert \geq c$	•		
	$C_{len}(P) \leq c$		•	
Reg Expr	$C_{reg}(P) := P[i] \in \bar{I} \subset I$		$\star$	
Gap	$C_{gap}(\mathcal{A}) \leq c := \alpha_j - \alpha_{j-1} \leq c,$ $\alpha_j \in \mathcal{A}, 2 \leq j \leq \lvert P \rvert$		$\star$	
	$C_{gap}(\mathcal{A}) \geq c$		•	
Span	$C_{spn}(\mathcal{A}) \leq c := \max\{\mathcal{A}\} - \min\{\mathcal{A}\} \leq c$		•	
	$C_{spn}(\mathcal{A}) \geq c$	•		
Max/Min	$C_{max}(\mathcal{A}) \geq c, C_{min}(\mathcal{A}) \leq c$	•		
	$C_{max}(\mathcal{A}) \leq c, C_{min}(\mathcal{A}) \geq c$		•	
Stats	$C_{sum}(\mathcal{A}), C_{avg}(\mathcal{A}), C_{var}(\mathcal{A}), C_{med}(\mathcal{A})$			•

*Not anti-monotone, but prefix anti-monotone.

3.3.2 Constraint Satisfaction in CSPM

A constraint $C_{type}(\cdot)$ is a Boolean function imposed on either the patterns or their attributes. A pattern P satisfies a constraint if and only if $C_{type}(P) = true$. The objective of CSPM is to find all frequent patterns that satisfy a set of user-defined constraints. In particular, the challenge of CSPM is to impose constraints during the mining algorithm, rather than post-processing mined patterns for constraint satisfaction.

The standard framework for CSPM is to classify constraints as being monotone or anti-monotone, as such constraint are easy to handle within the mining algorithm Pei et al. (2007).[1] A constraint is *monotone* if its violation by a sequence S implies that all subsequences $\bar{S} \preceq S$ also violate the constraint.

A constraint is *anti-monotone* if its violation by a sequence S implies violation by all super-sequences $\hat{S} \succeq S$. Table 3.2 lists common constraint types with their characterization. The concepts of monotonicity, anti-monotonicity, and violation are analogously extended to prefixes.

Constraints that are neither monotone nor anti-monotone are called *non-monotone* and are the

[1] A third classification is succinctness, which allows immediate pattern generation using a formula rather than an algorithm.

most challenging to enforce during mining. While dedicated approaches have been developed for certain non-monotone constraints Pei et al. (2007), they are otherwise handled by post-processing Aoga et al. (2017). Our goal is to develop a generic platform to handle non-monotone constraints effectively.

3.4 An MDD Representation of the Database

MDDs are widely applied as an efficient data structure in verification problems Wegener (2000) and were more recently introduced as a tool for discrete optimization and constraint programming Bergman et al. (2016a). Here, we use an MDD to fully encode the sequences from $\mathcal{SD}$; we refer to such data structure as an *MDD database*. We show how constraint satisfaction is achieved by storing constraint-specific information at the MDD nodes, thereby removing the need to impose constraint-specific rules in a mining algorithm.

3.4.1 MDD Construction for the SPM Problem

An MDD $M = (U, A)$ is a layered directed acyclic graph, where U is the set of nodes, and A is the set of arcs. Set U is partitioned into layers $(l_0, l_1, ..., l_{m+1})$, such that layers $l_i : 1 \leq i \leq m$ correspond to position (item) i of a sequence $S \in \mathcal{SD}$. Layers l_0, and l_{m+1} consist of single nodes, namely the root node $r \in l_0$, and the terminal node $t \in l_{m+1}$. The root and terminal node are used to model the start and end of all sequences, respectively. Figure 3.1.a shows the MDD database model for the $\mathcal{SD}$ of Table 3.1.

Layers $l_j, 1 \leq j \leq m$, contain one node per item $i \in I : \exists S \in \mathcal{SD}, S[j] = i$, and model the possible items at position j of all sequences $S \in \mathcal{SD}$. For example, layer 1 of the MDD database in Figure 3.1.a has two nodes corresponding to items B, C, and no node associated to item A. To distinguish which nodes are associated to which sequences $S \in \mathcal{SD}$, we define labels d_u for nodes $u \in U$, and store the associated sequence index S_{ID} in d_u. The first label of node B at layer 1 of Figure 3.1.a, indicates that sequences 1 and 2 contain item B at their first position. In addition, we store the attribute labels associated with the item, one per S_{ID} at each node. For example, in Figure 3.1.a we store the time and price attributes.

An arc $a = (u, v) \in A$, is directed from a node $u \in l_j$ to a node $v \in l_{j'} : j' > j$, and represents the next possible item after node u, for all sequences in $\mathcal{SD}$. Similar to nodes $u \in U$, labels d_a are defined for arcs $a \in A$ and store their associated sequences. A sequence S is thus represented by a path from r to t, following the nodes and arcs associated to S_{ID}. As we will search the MDD for patterns during the mining algorithm, we explicitly allow arcs to skip layers. That is, arc $(u, v) \in A$ can refer to any pair of nodes u, v on an r-t path P representing a sequence S. In Fig. 3.1 we only depict the arcs that represent the original sequences in $\mathcal{SD}$, for clarity. For example, the arc between node B at layer 1 and node B at layer 3 (following sequence $S_{\text{ID}} = 2$) is formally defined

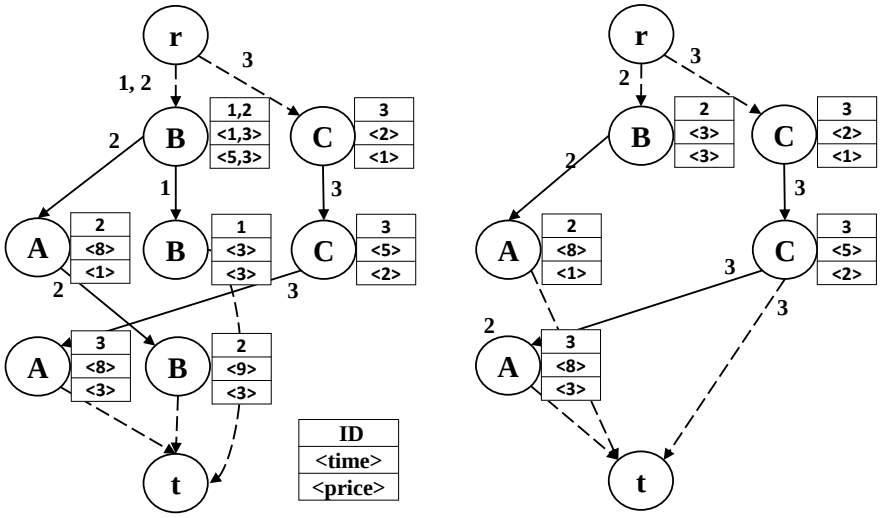


Figure 3.1: MDD database for the example SD in Table 3.1. Arcs skipping layers in Figure a) are not shown for clarity.

but omitted from the picture. Observe that any prefix or subsequence is represented by a partial path in the MDD, possibly using the arcs that skip layers. Lastly, we note that the MDD database (without imposed constraints) is built by a single scan of the database.

3.4.2 Imposing Constraints on the MDD Database

We use the MDD structure to enforce certain constraints on the MDD database itself. This has three main benefits, as follows. First, constraint satisfaction is performed only once, and not once per projected database as in the prefix-projection algorithm. Second, several constraints can be considered simultaneously, as opposed to iterative methods that consider each constraint individually and incur larger computational costs. Lastly, imposing constraints results in a smaller MDD, and consequently reduced computational requirements for the mining algorithm.

A constraint C_{type} can be imposed directly on the MDD if it is prefix monotone or prefix anti-monotone. That is, the feasibility of extending a pattern P ending at item i by an item i', is only dependent on the relationship between consecutive items i, i'. Examples of such constraints are the gap and regular expression constraints. An infeasible extension of such constraints is prevented by not creating an arc between their respective nodes. For example, if item i cannot be followed by item i', then no arc of the MDD database is constructed between their corresponding nodes.

Constraints on the MDD database are incorporated during its construction. In particular, the MDD database is built in increments using a backwards induction on the position j of a sequences $S \in \mathcal{SD}$. A backwards induction is chosen, as it allows us to gather constraint-specific information, used for constraint satisfaction later in the mining algorithm. For sequence S, the algorithm starts

from the node corresponding to the item at position $S[j] : j = |S|$, and checks whether this item may be used to extend a pattern ending in any of the sequence's previous items $i \in l_{j'} < l_j$. Whenever an extension is feasible, an arc (u, v) is created between the items' respective nodes in the MDD. The algorithm then increments and repeats the same procedure for the item in position $j - 1$.

By the construction above, a node connects to all nodes representing a feasible extension with respect to the imposed constraints. Thus, the mining algorithm needs only to search the children of a node $u \in U$ to extend any pattern ending at u. Figure 3.1.b shows an example of imposing constraint $C_{gap}(time) \geq 3$ on the MDD database of Figure 3.1.a.

Imposing constraints on the MDD database can be made more efficient by exploiting their properties such as anti-monotonicity. For example, given an anti-monotone constraint, if the extension of item i at $S[j]$ to an item at position $S[j']$ is infeasible, it is guaranteed that any extension of i to items $S[k] : k \geq j'$ is also infeasible. If a constraint is non-monotone, we are required to check its satisfaction for all possible extensions, which is done only if all monotone and anti-monotone constraints are satisfied.

Algorithm 3.1 gives the overall procedure to construct the MDD database.

Not all constraints can be imposed on the MDD database. The satisfaction of such constraints is performed during the mining algorithm, discussed in the next section.

3.5 Pattern Mining with MDD Databases

In this section, we discuss how to perform constraint reasoning by incorporating specific information into the MDD nodes. Such information is used to establish conditions to efficiently remove infeasible patterns from the database.

3.5.1 Information Exploitation for Effective Mining

By construction, an r-u path in the MDD database represents the prefix of a pattern ending at node u. Similarly, any extension of this prefix is modeled by a u-t path. Post-processing patterns for constraint satisfaction corresponds to checking the feasibility of all u-t paths. We can, however, exploit the MDD structure to determine whether it is possible to extend an infeasible pattern to a feasible one. This is achieved by augmenting the MDD nodes with constraint-specific information that allow us to perform such reasoning.

For instance, consider a constraint $C_{min}(price) \geq 5$ and the extension of an infeasible pattern ending at node u, as shown in Figure 3.2. Observe that only one u-t path results in a feasible pattern. Instead of explicitly searching all u-t paths, we can store the minimum price reachable from nodes $u \in U$, during the MDD construction, and then use it to guarantee that a feasible extension exists.

Algorithm 3.1 MDD construction with constraints

```
1: procedure BuildMddDatabase(Item attributes α, Constraint set C)
2:     for Sequences S ∈ SD do strp = |S| − 1; endp = |S|; antmon = 0
3:         while strp > 0 do
4:             while antmon == 0 do
5:                 if (strp, endp) satisfies anti-mon constraint then
6:                     antmon = 1;
7:                 else
8:                     endp = endp − 1
9:                     if strp == endp then
10:                        strp = strp − 1;
11:                    if strp == 0 then
12:                        break;
13:        if antmon == 1 then
14:            lastp = endp
15:            while endp! = strp do
16:                if (strp, endp) satisfies monotone constraint then
17:                    Add an arc from node at strp of S to node at endp of S
18:                    Associate nodes with labels and attributes
19:                    Add an arc from root node r to node at position strp of S;
20:                else
21:                    break;
22:            endp = endp − 1;
23:        strp = strp − 1
24:        antmon = 0
25:        endp = lastp;
```

3.5.2 Categories of Constraint-specific Information

We now describe constraint-specific information for a number of practical constraint classes. We only present the proof for lower bound constraints; upper bound conditions can be established analogously. We define $\alpha^u \in \mathcal{A}$ to be the attribute value of item i at node u of the MDD.

Span Constraint:

Let β_1^u and β_2^u denote the minimum and maximum values of α reachable from u, respectively. Values β_1^u are initially set to α^u. When adding an arc (u, v), we update $\beta_1^u \leftarrow \beta_1^v$ if $\beta_1^u > \beta_1^v$, and $\beta_2^u \leftarrow \beta_2^v$ if $\beta_2^u < \beta_2^v$ for node v. By this procedure, β_1^u, β_2^u give the minimum and maximum values of α reachable from u. Proposition 3.1 proves that by using these variables, we can guarantee the satisfaction of the span constraint.

Proposition 3.1. *An infeasible pattern P can be extended to a feasible pattern with respect to $C_{spn}(\alpha) \geq c$ if and only if* $\max\left\{\max_{\alpha \in P}\{\alpha\}, \beta_2^u\right\} - \min\left\{\min_{\alpha \in P}\{\alpha\}, \beta_1^u\right\} \geq c$.

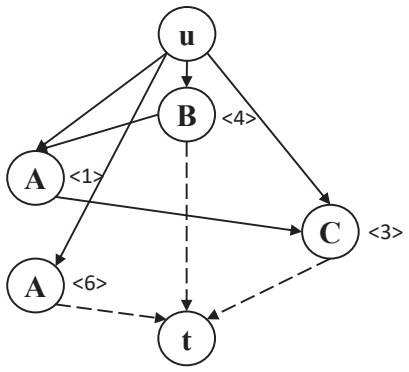


Figure 3.2: Extending a pattern ending at node u, with constraint $C_{min}(\mathcal{A}) \geq 5$. The label at each node represents the attribute of the item.

Proof. The necessity is straightforward. For the converse, assume $\alpha^{\max} - \alpha^{\min} < c$. Then no u-t path contains values of α such that P can become feasible. □

Sum Constraint:

Let β^u denote the maximum sum of values α reachable from u. We first initialize $\beta^u \leftarrow \alpha^u$. Next, when adding an arc (u, v), we update $\beta^u \leftarrow \beta^v + \alpha^u$ if $\beta^u < \beta^v + \alpha^u$, which results in the maximum sum possible to be stored for node u. Proposition 3.2 proves that this information is sufficient.

Proposition 3.2. *There exists a feasible extension from node u with respect to each individual constraint if and only if* $\sum_{\alpha \in P} \alpha + \beta^u \geq c$.

Proof. The necessity is straightforward. For the converse, assume $\sum_{\alpha \in P} \alpha + \beta^u < c$. By the construction of β^u, we can conclude $\sum_{\alpha \in P} \alpha + \sum_{\alpha \in (u,t)} \alpha < c$, for all u- paths. □

Average Constraint:

Let β_1^u denote a sum of values α on a u-t path, and β_2^u denote the number of attributes α contributing to the sum in β_1^u. For constraint $C_{avg}(\alpha) \geq c$, and any pattern P ending at node u, our objective is to generate values of β_1^u, β_2^u that give the maximum possible average $\frac{\sum_{\alpha \in P} \alpha + \beta_1^u}{|P| + \beta_2^u}$ above the threshold c.

The generation of β_1^u depends on the value c of constraint $C_{avg}(\alpha) \geq c$. Initially we set $\beta_1^u = \alpha^u$, and $\beta_2^u = 1$. When adding an arc (u, v) during the construction of the MDD, we update $\beta_1^u \leftarrow \alpha^u + \beta_1^v, \beta_2^u \leftarrow \beta_2^v + 1$ if $(\alpha^u + \beta_1^v) - c\,(1 + \beta_2^v) > \beta_1^u - c\beta_2^u$. This ensures that the best values to maximize $\frac{\sum_{\alpha \in P} \alpha + \beta_1^u}{|P| + \beta_2^u}$ are generated, proven in Lemma 3.3.

Lemma 3.3. *For constraint $C_{avg}(\alpha) \geq c$, the update procedure above generates values β_1^u, β_2^u that give the maximum average $\frac{\sum_{\alpha \in P} \alpha + \beta_1^u}{|P| + \beta_2^u}$ above threshold c, for a pattern p ending at node u.*

Proof. Proof by induction. By the initial definitions of β_1^u, β_2^u, the statement is true for any u in the last layer l_m of the MDD. Now assume the statement holds for all nodes in layer greater than l_j. For nodes u in layer l_j we choose the path giving the maximum average $\max_{u\text{-}t} \left\{ \frac{\sum_{\alpha \in P} \alpha + \beta_1^v + \alpha^u}{|P| + \beta_2^v + 1} - c \right\} = \max_{u\text{-}t} \{\beta_1^v + \alpha - c\,(\beta_2^v + 1)\}$. □

Proposition 3.4 shows that β_i^1, β_u^2 are the only required information to check satisfaction of the maximum average constraint. The proof for the minimum average constraint is similar, and omitted for brevity.

Proposition 3.4. *It suffices to record β_1^u, β_2^u as defined above, to check satisfaction for the minimum average constraint $C_{avg}(\alpha) \geq c$.*

Proof. The maximum average reachable from node u is $\frac{\beta_1^u}{\beta_2^u}$ by definition. Therefore, if $\frac{\sum_{\alpha \in P} \alpha + \beta_1^u}{|P| + \beta_2^u} < c$, then no (u, t) paths exists that satisfies $C_{avg}(\alpha) \geq c$ for a pattern ending at node u. □

Median Constraint:

Let β_1^u denote the maximum difference of the number of values $\alpha \geq c$ and the number of values $\alpha < c$, between all possible paths u-t, i.e. $\beta_1^u = \max_{u\text{-}t} \{ \big| \{\alpha \in u\text{-}t : \alpha \geq c\} \big| - \big| \{\alpha \in u\text{-}t : \alpha < c\} \big| \}$. Further, let β_2^u denote the maximum of values $\alpha < c$ contributing to the count in β_1^u, and β_3^u denote the minimum of values $\alpha \geq c$ contributing to the count in β_1^u. Observe that the satisfaction of $C_{med}(\alpha) \geq c$ can be determined using values β_1^u to β_3^u. Namely, if $\beta_1^u > 0$ then there exists more values α above c than below it, guaranteeing satisfaction. Similarly if $\beta_1^u < 0$ the median constraint is violated. If $\beta_1^u = 0$ then we calculate the average $\frac{\beta_2^u + \beta_3^u}{2}$ which gives the median.

The generation of β_1^u to β_3^u depends on the constant c. Initially, we set $\beta_1^u = 0, \beta_2^u = \min_{\alpha \in S} \{\alpha\} - 1, \beta_3^u = \alpha^u$ for all nodes $u : \alpha^u \geq c$, and $\beta_1^u = 0, \beta_2^u = \alpha^u, \beta_3^u = \max_{\alpha \in S} \{\alpha\} + 1$ for all remaining nodes. Next, during the construction of the MDD, for a node u, we find the path u-t that has the highest potential to extend an infeasible pattern P ending at u to a feasible one. The best path u-v-t, denote v-t, is a path that contains a feasible extension for P given any other feasible extensions available by the remaining u-v'-t paths, denote v'-t. We prove four dominance rules that when satisfied, guarantee this for v-t.

The first rule is if $\beta_1^v > \beta_1^{v'}$, proven valid in Lemma 3.5.

Lemma 3.5. *If $\beta_1^v > \beta_1^{v'}$ holds, and extension of a pattern P by path v'-t is feasible, so is the extension of P by path v-t.*

Proof. Let β_1^p denote the difference of the number of values $\alpha \in P : \alpha \geq c$ to the number of values $\alpha \in P : \alpha < c$. Then, $\beta_1^p + \beta_1^v > \beta_1^p + \beta_1^{v'}$, meaning there is a greater number of values $\alpha \geq c$ on path v-t, compared to path v'-t. □

All other conditions require $\beta_1^v = \beta_1^{v'}$. For these conditions, we first calculate $med_{v'} = \frac{\beta_2^{v'} + \beta_3^{v'}}{2}$, $med_v = \frac{\beta_2^v + \min\{\beta_3^v, \alpha^v\}}{2}$. Conditions two to four are proved in 3.6.

Lemma 3.6. *Given $\beta_1^v = \beta_1^{v'}$, any feasible extension of an infeasible pattern P by path v'-t is also feasible for path v-t, if one of the following three conditions hold: 1. $med_v \geq c, med_{v'} < c$, 2. $med_v \geq c, med_{v'} \geq c, \beta_2^v > \beta_2^{v'}$, 3. $med_v < c, med_{v'} < c, \beta_3^v > \beta_3^{v'}$.*

Proof. Let β_1^p to β_3^p be defined as before. For condition 1, as $med_{v'} < c$ and P is infeasible, any extension of P by u-t must have $\beta_1^p + \beta_1^{v'} > 0$, which is also satisfied by path v-t. For condition 2, if an infeasible pattern P can be extended to a feasible pattern by v'-t, then either $\beta_1^p + \beta_1^{v'} > 0$ which implies feasibility of v-t, or $\beta_1^p + \beta_1^{v'} = 0$. In this case, the only value of $\frac{\max\{\beta_2^p, \beta_2^{v'}\} + \min\{\beta_3^p, \beta_3^{v'}\}}{2}$ (i.e., the median of pattern P extended by v'-t), which is not guaranteed to be feasible or infeasible is $\frac{\beta_2^{v'} + \beta_3^p}{2}$. However, if $\frac{\beta_2^{v'} + \beta_3^p}{2} \geq c$, we also have $\frac{\beta_2^v + \beta_3^p}{2} \geq c$. The proof of the third rule is similar to the second rule, and omitted due to space limits. □

If any of the above rules are satisfied, we update $\beta_1^u \leftarrow \beta_1^v + 1, \beta_2^u \leftarrow \max\{\beta_2^v, \alpha\}, \beta_3^u \leftarrow \beta_3^v$ if $\alpha^u \geq c$, or $\beta_1^u \leftarrow \beta_1^v - 1, \beta_2^u \leftarrow \beta_2^v, \beta_3^u \leftarrow \max\{\beta_3^u, \beta_3^v\}$ otherwise. Proposition 3.7 shows that these values are sufficient to determine whether an infeasible pattern P can be extended to a feasible one.

Proposition 3.7. *Let $\beta_1^p - \beta_3^p$ be defined as before. There exists a feasible extension from node u with respect to $C_{med}(\alpha) \geq c$ if and only if $\beta_1^u + \beta_1^p > 0$, or $\beta_1^u + \beta_1^p = 0, \frac{\min\{\beta_3^p, \beta_3^p\} + \max\{\beta_2^p, \beta_2^u\}}{2} \geq c$.*

Proof. The necessity is straightforward. For the converse, first assume $\beta_1^p + \beta_1^v < 0$, then by Lemma 3.6, no u-t path contains enough values $\alpha \geq c$ to satisfy $C_{med}(\alpha) \geq c$. For the second condition, if $\beta_1^u + \beta_1^p = 0, \frac{\min\{\beta_3^p, \beta_3^u\} + \max\{\beta_2^p, \beta_2^u\}}{2} < c$, then by Lemma 3.6, the maximum median between all u-t paths is below threshold c. □

3.6 Mining the MDD Database with Prefix-projection

We now present our *MDD prefix-projection* (MPP) algorithm, which performs prefix-projection on the MDD database. The first step of the algorithm is to find all frequent items i, i.e. patterns of size one, using a depth-first-search. This is automatically done during the construction of the MDD database, and modeled by the children of root node r. In the next steps, the algorithm attempts to expand a frequent pattern generated in previous iterations. Using the stored information in the MDD, we prune extensions that cannot lead to a feasible pattern. In particular, for an infeasible pattern P ending at node $u \in U$, the algorithm uses the information stored at u to determine

whether P may be extended to a feasible pattern. If a feasible extension does not exist, the search is pruned. Otherwise, pattern P is extended and investigated in future iterations.

In contrast to searching the database rows in prefix-projection, the MPP algorithm follows feasible paths in the MDD database. This leads to a more efficient search, as some infeasible extensions have been removed when constructing the MDD database. The trade-off is that finding paths corresponding to a sequence S requires a search on labels d_u, a_u, thereby incurring additional computational cost. For efficient memory utilization, the MDD is not physically projected, but rather *pseudo projected* Han et al. (2001). In pseudo projection, only the initial $\mathcal{SD}$ is stored in memory, and search is initiated from "projection pointers" pointing to the MDD nodes.

In prefix-projection, all N sequences are searched in each iteration, and an item $i \in I$ is frequent if its final count is at least θ. As opposed to searching all N sequences, we propose to stop when it is guaranteed that an item i is not frequent. Let n denote the number of sequences searched so far when searching for i, and let $Sup(P)$ denote the number of sequences that contain pattern P. We use the following proposition to detect that item i cannot be frequent, given a frequent pattern P:

Proposition 3.8. *If $n - Sup(i) > Sup(P) - \theta$, item i cannot be frequent in the projected database.*

Proof. The left-hand-side is the number of searched sequences that do not contain i, and the right-hand-side is the maximum number of sequences that do not contain i while it remains frequent. □

Algorithm 3.2 Extending frequent patterns in MPP.

```
1:  procedure ExtendFreqPattern(Frequent pattern P.)
2:      for Sequences S ∈ 𝒮𝒟 do
3:          for Nodes u : P ends at u do
4:              for Nodes v : (u, v) ∈ A and v is associated to S do
5:                  if (u, v) gives a feasible extension for P then
6:                      Store v as end point for potential pattern P′;
7:                  else if Extension by v violates an anti-monotone constraint then
8:                      No other children of u satisfy constraints, Break;
9:                  else if A feasible extension exists from v then
10:                     Conditionally store v as starting point for potential pattern P′;
11:     for Potential patterns P′ do
12:         if More than θ sequences contain P′ without condition then
13:             Store P′ as new frequent pattern;
14:         else if More than θ sequences contain P′ with condition then
15:             Conditionally store P′ as new frequent pattern;
16:     if No potential pattern is stored, and P is not conditionally stored then
17:         P is a maximal frequent pattern;
```

Projecting the minimal prefix containing a pattern P (as done in SPM) is not sufficient for CSPM Aoga et al. (2017). Extensions from the minimal prefix may violate a constraint, while it

Table 3.3: Five real-life datasets and their features.

$\mathcal{SD}$	N	$\|I\|$	M	$avg(\|S\|)$*
Kosarak	837,206	41,001	2,498	9.3
MSNBC	989,818	19	29,591	10.5
Kosarak (small)	59,261	20,894	796	9.2
BMSWebView1	26,667	497	267	4.4
BMSWebView2	52,619	3,335	161	6.3

*Average length of sequences

may be the case that another larger prefix of the sequence satisfies such extensions. For example, the minimal prefix containing item C in sequence 3 of Table 3.1 cannot be extended by item A under a constraint $C_{gap}(time) \leq 3$. However, extending the larger prefix containing C is feasible. We are thus required to store all prefixes and their extension at each iteration of MPP.

A time-consuming task of the general prefix-projection algorithm is to determine whether a specific item i exists in sequences of the projected database. To avoid searching the entire sequence for every item, Aoga et al. (2017) store the last position of items $i \in I$ for sequences $S \in \mathcal{SD}$. An MDD database enables search for the extension of all items $i \in I$ simultaneously, resulting in more efficient search. That is, as opposed to searching for a specific item i, all children of node u are searched, and record the items which enable a feasible extension.

Algorithm 3.2 gives the procedure to extend a frequent pattern using the MDD database.

3.7 Numerical Results

For our numerical tests, we use real-life click-stream benchmark databases[2], listed in Table 3.3. We note that two of these databases, Kosarak and MSNBC, are considerably larger than those typically reported in the CSPM literature, with about 900,000 sequences of length up to 29,500, and containing up to 40,000 items. None of these standard benchmark datasets are annotated with attributes. To be able to evaluate our approach, we therefore generate three attributes of time, price, and quality, as follows. For the time attribute, we randomly generate a number between 0 and 600 seconds, to represent the time spent by users at each click. With a probability of 5%, we model the user leaving the session by setting the time between clicks to a value between 1 to 10 hours. For the price and quality attributes, we generate a number between 1 and 100 for each item $i \in S, \forall S \in \mathcal{SD}$.

All algorithms are coded in C++, with the exception of PPICt which is coded in Scala.[3] All experiments are executed on the same PC with an Intel Xeon 2.33 GHz processor, 24GB of memory,

[2] http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

[3] We thank the developers of PPICt for sharing their code.

using Ubuntu 12.04.5 as operating system. We limit all tests to use one core of the CPU. The MPP code is available and open source.[4]

3.7.1 Comparison with Prefix-projection and Constraint Checks

Our first goal is to evaluate the impact of the MDD database and the associated constraint reasoning, especially in presence of more complex constraints. However, no other CSPM system accommodates constraints such as average and median and multiple item attributes. Because simple generate-and-test (via post-processing) does not scale due to the size of the databases, we developed a prefix-projection algorithm for the original database, that can handle multiple item attributes and effectively prune the search space for anti-monotone constraints such as gap and maximum span. We name this algorithm Prefix-Projection with Constraint Checks (PPCC). PPCC operates by prefix-projection and extends a pattern P if it satisfies all anti-monotone constraints, and prunes the extension otherwise. For non-monotone constraints, PPCC extends infeasible patterns with the hope that a feasible super-pattern exists, and performs a constraint check at the end.

In Figure 3.3 we compare the performance of MPP and PPCC in terms of total CPU time (MDD construction plus mining algorithm), given minimum support (Min supp) as a percentage of the total number of sequences. The experiment uses three scenarios with constraints on one, two, and three attributes, respectively:

$$\begin{aligned}
&\text{time: } 30 \leq C_{gap}(time) \leq 900,\ 900 \leq C_{spn}(time) \leq 3600,\\
&\text{price: } 30 \leq C_{avg}(price) \leq 70,\ 40 \leq C_{med}(price) \leq 60,\\
&\text{quality: } 40 \leq C_{avg}(quality) \leq 60,\ 30 \leq C_{med}(quality) \leq 70.
\end{aligned}$$

Scenario one (PPCC1 and MPP1) only considers the time constraints. Scenario two (PPCC2 and MPP2) considers the time and price constraints. Scenario three (PPCC3 and MPP3) considers all time, price, and quality constraints. The results in Figure 3.3 show that mining more constrained patterns takes more time for both methods. However, MPP is always more efficient than PPCC, and often considerably. For example, finding all frequent patterns with minimum support of 4% with all constraints (scenario three) in the MSNBC database takes PPCC about 4,000s while MPP only needs about 2,000s. Moreover, Table 3.4 shows that the time required to construct the MDD database and generate constraint specific information is quite small. This indicates that our MDD database can be used to effectively and efficiently handle constraints such as average and median.

3.7.2 Comparison with PPICt

We next compare our approach to the state-of-the-art CSPM algorithm PPICt, which is implemented in the CP framework OscaR[5] Aoga et al. (2017). PPICt accommodates a wide range of

[4] https://github.com/aminhn/MPP
[5] https://bitbucket.org/oscarlib/oscar/wiki/Home

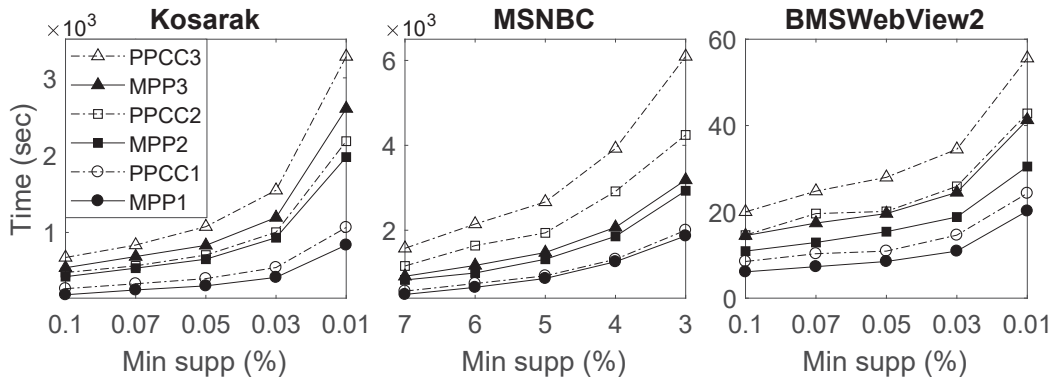


Figure 3.3: Mining with constraints $30 \leq C_{gap}(time) \leq 900, 900 \leq C_{spn}(time) \leq 3600, 30 \leq C_{avg}(price) \leq 70, 40 \leq C_{med}(price) \leq 60, 40 \leq C_{avg}(quality) \leq 60, 30 \leq C_{med}(quality) \leq 70$. Attributes and their corresponding constraints are added incrementally from 1 to 3.

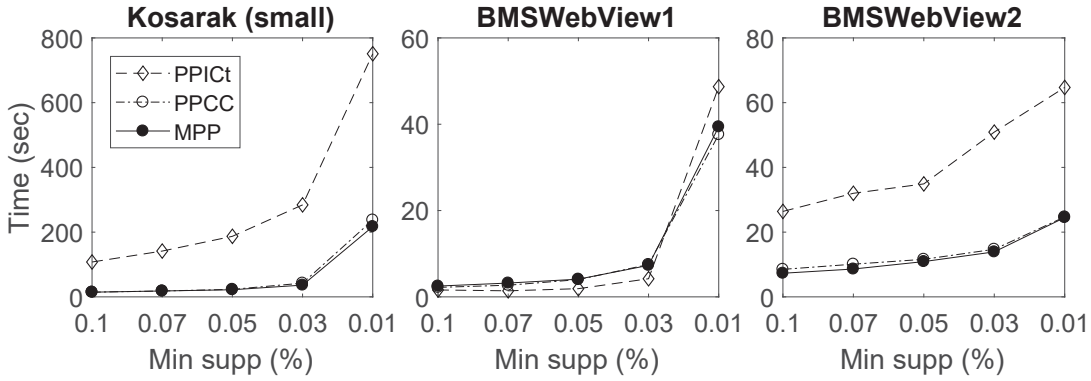


Figure 3.4: Mining with one item attribute (time) and constraints $30 \leq C_{gap}(time) \leq 900, C_{spn}(time) \leq 3600$.

constraints, including gap and maximum span constraints, but is restricted to a single attribute. We therefore evaluate MPP and PPICt for mining patterns with the following gap and maximum span constraints over the time attribute:

$$30 \leq C_{gap}(time) \leq 90, \quad 900 \leq C_{spn}(time) \leq 3600.$$

Initial tests indicated that the PPICt code is unstable when executed on the full databases Kosarak and MSNBC. We therefore executed the codes on the smaller benchmark variant of Kosarak (which is also used in Aoga et al. (2017)), BMSWebView1, and BMSWebView2. The results are presented in Figure 3.4, which follows the same format as Figure 3.3.

A first observation is that MPP and PPCC produce almost identical results, as they both benefit from the same pruning rules for anti-monotone constraints. The time required to build the MDD database, shown in Table 3.4, is made up by a faster prefix-projection algorithm due to implementing the gap constraints on the MDD itself. Both MPP and PPCC also outperform PPICt on Kosarak

Table 3.4: Time (in seconds) required for MDD construction and information generation.

Algorithm	Kosarak	MSNBC	BMS2	BMS1	Kosarak(small)
MPP1	47	45	2	-	-
MPP2	106	103	4	-	-
MPP3	151	158	5	-	-
MPP	-	-	2	0.5	4

(small) and BMSWebview2, but all three methods perform similarly on BMSWebView1. However, PPICt uses significantly more memory, up to 14Gb, while MPP uses up to 1Gb, and PPCC consumes the lowest with at most 0.5Gb. We conclude that on this benchmark our approach is competitive with or more efficient than PPICt.

3.8 Conclusion

In this chapter, we developed a novel MDD representation for CSPM. We prove how constraint satisfaction is achieved for a number of constraints, including sum, average, and median, by storing constraint-specific information at the MDD nodes. Moreover, our approach is able to accommodate several item attributes with constraints, which occur frequently in real-world problems.

We embedded our MDD representation within a prefix-projection algorithm, called MPP, and performed an experimental evaluation on real-life benchmark databases with up to 980,000 sequences and 40,000 items. The results showed that the MPP mining algorithm is always more efficient than a prefix-projection algorithm with constraint checks. The benefits of MPP become larger as we increase the size of the database, the number of constraints, or the number of attributes. Although MPP is primarily designed for efficient constraint satisfaction of rich constraints and multiple item attributes, it remains competitive with a CP-based state-of-the-art CSPM algorithm, for databases with only one item attribute and anti-monotone constraints.

Chapter 4

Pattern Mining for Interpretable Data-driven Sequential Decision Making

4.1 Introduction

Predictive models have significantly impacted the way information is organized and leveraged in the practice of management, including areas such as retail (Akter and Wamba, 2016), accounting (Appelbaum et al., 2017), sports (Coleman, 2012), and finance (Corea, 2016). Of notable prevalence are *sequential* applications, i.e., in which predictions (and subsequent managerial actions) are made consecutively over time based on past observed events. Examples include buying/selling decisions in financial markets (Pang et al., 2018) and recommender systems for products and services that account for observed consumer behavior (Cheng and Shen, 2016).

The growth of data-driven sequential approaches is due in large part to the increasing accessibility of highly accurate machine learning tools, specifically deep-learning and ensemble frameworks (Wu et al., 2019; Sagi and Rokach, 2018). The challenge, however, is that such tools generally offer little interpretability in their learning tasks and predictions. This is a significant concern in scenarios with ethical implications or high risk. For instance, Vayena et al. (2018) argue that the lack of transparency is one of the primary barriers to the use of artificial intelligence in healthcare, with half of the United States' decision makers at healthcare organizations believing it would produce fatal errors and not meet expectations. Similar issues are also raised, e.g., in marketing and finance (Drew et al., 2019; Hajek and Henriques, 2017; Cui et al., 2006).

In such cases, practitioners may be more inclined to use simpler and more interpretable machine learning models, such as rule-based learning and decision trees. While such algorithms are comparably less accurate for predictive tasks than state-of-the-art black-box tools, they provide descriptive results and expose simpler but often insightful relationships in data. Interpretability further allows practitioners to explain the outcome of machine learning algorithms, resulting in higher reliability and accountability in decision-making tasks. These benefits have led to a growing research in inter-

pretable machine learning (see, e.g., Došilović et al. 2018; Murdoch et al. 2019; Molnar et al. 2018). Nonetheless, the literature on interpretable learning for sequential tasks is still scarce.

In this paper, we study methodologies for identifying patterns in large-scale sequential datasets for interpretability and descriptive purposes. We focus on sequential decision-making tasks that aim at maximizing the probability of an outcome of interest, such as increasing product sales in a retail recommender system. The objective of our approach is to learn the association between the sequence of decisions (e.g., product recommendations) and the occurrence of the desired outcome (e.g., product sales). In particular, we build on the assumption that such associations are best described by a frequent sequence of decisions that leads to its occurrence (Cheng and Shen, 2016). Finding and extracting such patterns, in turn, fall under the umbrella of the extensive field of *sequential pattern mining*, an unsupervised learning task (see, e.g., Han et al. 2000).

Our first contribution is a *data tree model* for extracting frequent sequential patterns from large datasets. A data tree is a network embedding of the database that leverages the sequential structure for higher space compression. From a practical perspective, data trees address the size limitations of current state-of-the-art sequential pattern mining algorithms (Han et al., 2001). In particular, we show that the memory required for a data tree model can be orders of magnitude smaller than a traditional tabular encoding, and in the worst case never larger. From a managerial perspective, we exploit the network encoding of our model to reveal structured frequent patterns that, to the best of our knowledge, cannot be identified by existing algorithms. Thus, data trees may potentially strengthen explanations and insights associated with desired outcomes.

Our second contribution is to use sequential patterns (e.g., as identified through our methodology) to analyze the likelihood that a sequence of decisions is associated with an outcome of interest. Specifically, we propose that patterns satisfying a given minimum likelihood constraint are considered as interpretable explanations on why different outcomes occurred. To test the reliability of such explanations, we design a p-value hypothesis test that generates likelihood probabilities with a guaranteed statistical confidence. Furthermore, using regression hypothesis tests, we evaluate the trade-off between the minimum likelihood constraint imposed on patterns and the percentage of all possible outcomes explained by feasible patterns.

Finally, we show how mined patterns can be aggregated into a *knowledge tree* to provide an interpretable decision support tool for sequential decision making. Similar to a data tree model of the dataset, knowledge trees lead to higher space compression as well as more efficiency in analytical tasks performed over mined patterns. More importantly, knowledge trees provide a structural view of the decision making process by displaying the transition of likelihood probabilities between decisions (or events) and outcomes of interest. Using the path structure of knowledge trees, we calculate expectations of future sequence of likelihood probabilities to provide more concise and interpretable knowledge to practitioners. The result of our approach is a visual, interpretable, and statistically guaranteed data-driven decision support tool for sequential decision making tasks.

We investigate our approach on two real-world large-size applications in marketing and finance.

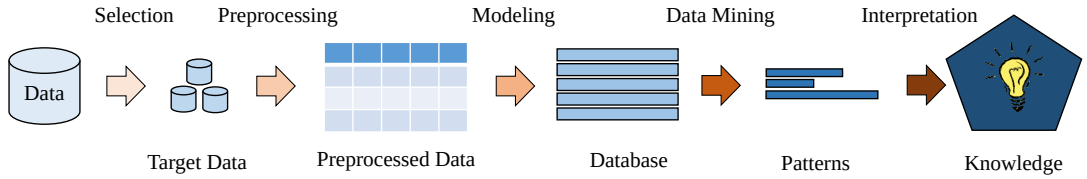


Figure 4.1: Path from data to knowledge.

In marketing, we study an online music streaming platform with the objective of reducing user skip rates. Using data trees, we model and fit 2.2 billion track recommendations in memory, while a tabular encoding of the database can fit at most a couple of million events. We perform pattern mining over the data tree and generate frequent patterns of musical track recommendations based on their audio features. We then analyze sequences of track audio features based on the likelihood that they lead to user skips, and use patterns with a high likelihood to explain skip outcomes. Lastly, we aggregate frequent patterns into a knowledge tree and show how the resulting interpretable view can be used to aid the recommendation process.

In finance, we identify novel patterns associated with price changes to derive insights for sequential investment decisions in the stock market. In particular, our model is based on a data tree that encodes a large-scale time series of stock prices. We analyze whether our generated patterns can indeed be used to explain price change events in the database through hypothesis tests with statistical guarantees. The resulting knowledge tree provides a decision support tool for technical analysis in investment decision making.

The remainder of the paper is organized as follows. We review related work in §4.2 and develop data trees and the pattern mining algorithm in §4.3. We discuss how patterns are used for interpretable explanations and knowledge discovery in §4.4, and apply the developed framework to our marketing and finance applications in §4.5 and §4.6. The paper is finally concluded in §4.7.

4.2 Background and Related Work

Our work contributes to the field of interpretable machine learning, which has been receiving a growing interest in management science in recent years. We refer to Murdoch et al. (2019) and Molnar et al. (2018) for a review of distinct interpretable tasks. In this study, we perceive interpretability as an explanation on why an outcome of interest has occurred. Specifically, we generate sequential patterns of events and analyze their association with the occurrence of a certain outcome. If the association between the sequential pattern and the desired outcome is strong, we consider the sequence of events in the pattern as an explanation for the outcome.

Pattern discovery and data mining have long been used for knowledge discovery tasks. For example, data mining is the key step prior to interpretation of patterns in the classical path from data to knowledge by Fayyad et al. (1996), as depicted in Figure 4.1. Pattern mining was first

introduced by Agrawal et al. (1993, 1994) and used to develop association rules for market basket analysis, i.e., statements of the form *if-then* that identify products that are purchased in bundles by consumers. The field has since grown significantly with over 50,000 papers to date. Our study refers to an area of data mining known as *sequential pattern mining*, in which the data is semi-structured, i.e., in our case represents a temporal or sequential set of events.

While the literature on sequential pattern mining mostly emphasizes computational efficiency (e.g., Lin and Lee 2005; Aoga et al. 2017; Hosseininasab et al. 2019; Han et al. 2000; Ayres et al. 2002), state-of-the-art sequential mining algorithms still face challenges in memory requirements and are limited to relatively small-sized datasets. In particular, the two dominant pattern mining methodologies, *Apriori* (Agrawal et al., 1994) and *prefix-projection* (Han et al., 2001), suffer from the explosion of candidate patterns or require an explicit representation of the entire database in memory, respectively. They are therefore limited in usage for knowledge discovery in applications with larger data requirements.

In this paper, we design a novel tree model of the database that is able to accommodate orders of magnitude larger datasets. Network models and graphical models of the database have been shown to be effective in item-set mining (Han et al., 2004; Pyun et al., 2014; Borah and Nath, 2018) and pattern mining (Masseglia et al., 2009; Hosseininasab et al., 2019; Loekito and Bailey, 2007). In particular, our pattern-mining methodology is motivated by the work of Hosseininasab et al. (2019), where a prefix-projection algorithm (Han et al., 2001) is applied over a decision-diagram network model of the database. Although data trees are different from the decision diagram model of the database by Hosseininasab et al. (2019), our proposed mining algorithm over the network structure is similar. The main focus of such models is, however, time efficiency and they do not show noticeable gains in memory requirements. Other network-based data mining frameworks, e.g., webpage ranking, anomaly detection, and time-stamped social networks (Lambiotte et al. (2018), Rozenshtein and Gionis (2019), and references therein) investigate how networks evolve with time, such as links between pairs in a social network. Our work, in turn, does not assume that the dataset has a priori network structure. Instead, our approach models any general sequential database as a tree model that exploits equivalence classes within the data for higher space compression.

From an application point of view, the pattern mining literature shows a large focus on the development of domain-specific association rules as an intermediate step in recommender systems (e.g., Lin et al. 2002, 2000; Najafabadi et al. 2017; Suchacka and Chodak 2017). Our work builds upon this literature by generating sequential association rules and developing interpretable explanations for the occurrence of outcomes, as well as pattern aggregation schemes to provide a structural and graphical view of the decision making process.

Lastly, this paper is related to supervised learning tasks for sequential prediction. The state-of-the-art machine learning models for such tasks are neural networks and deep learning, such as recurrent neural networks (Bengio et al., 2015; Lipton et al., 2015; Martinez et al., 2017). Recurrent neural networks are generally applied to extend a sequence of observed data with one or multiple

events (Rather et al., 2015; Bahdanau et al., 2016; Bengio et al., 2015). The main focus of such algorithms is prediction accuracy, which in turn decreases as the size of the desired predicted sequence increases. Specifically, an incorrect prediction early in the sequence negatively impacts future predictions and consequent decisions. In contrast, the focus of our proposed method is interpretability and knowledge discovery, where sequential patterns are used to anticipate and analyze the entire set of significant future events and outcomes. Any change in future events is therefore anticipated and analyzed to aid decision making.

4.3 Data Trees and Pattern Mining

In this section, we develop a novel data tree model of a sequential dataset and use it to mine frequent patterns from large datasets. Data trees form the basis for our pattern mining algorithm and are key to its scalability. We begin in §4.3.1 with the formalization of the sequential database and frequent patterns, and in §4.3.2 we define a data tree model for a given sequential database. Next, in §4.3.3 we characterize the minimum-sized data tree and conditions for which it will lead to memory savings, developing in §4.3.4 a procedure to build such a data tree efficiently. Finally, in §4.3.5 we discuss the pattern mining algorithm tailored to a data tree model of the database.

4.3.1 Sequential Database and Sequential Patterns

Let $\mathbb{E}$ be a finite set encoding the possible events or decisions within an application of interest. A sequential database is a set $\mathscr{S} := \{\mathcal{S}_1, \ldots, \mathcal{S}_N\}$ where each $\mathcal{S}_i := \langle e_{i1}, \ldots, e_{i|\mathcal{S}_i|}\rangle$ is a sequence of events $e_{i1}, \ldots, e_{i|\mathcal{S}_i|} \in \mathbb{E}$ for all $i = 1, \ldots, N$. Each j-th event of a sequence $\mathcal{S}_i$ is also associated with an observed *outcome* $u_{ij} \in \mathbb{U}$. Without loss of generality, we restrict our attention to a binary set $\mathbb{U} = \{0, 1\}$ indicating if an outcome occurred ($u_{ij} = 1$) or not ($u_{ij} = 0$) at event e_{ij}, focusing on the case $u_{ij} = 1$ as the desired outcome to investigate. We denote by $\mathcal{S}_i[j]$ and $\mathcal{S}_i\{j\}$ the j-th event and its associated outcome, respectively, of a sequence $\mathcal{S}_i$. Moreover, let $M := \max_{i=1,\ldots,N} |\mathcal{S}_i|$ be the size of the largest sequence stored in the database.

We wish to identify patterns that are frequently associated with the desired outcome in a sequential database $\mathscr{S}$. We define a pattern $\mathcal{P} := \langle e_1, \ldots, e_{|\mathcal{P}|}\rangle$ as an ordered contiguous subsequence of some sequence $\mathcal{S}_i \in \mathscr{S}$. That is, for any pattern $\mathcal{P}$, there exists some $\mathcal{S}_i \in \mathscr{S}$ and integer $k \in \mathbb{Z}_{\geq 0}$ such that $\mathcal{P}[j] = \mathcal{S}_i[k + j]$ for all $j = 1, \ldots, |\mathcal{P}|$, where $\mathcal{P}[j]$ denotes the j-th event of $\mathcal{P}$. The frequency $f(\mathcal{P}) \in \mathbb{Z}_{\geq 0}$ of a pattern $\mathcal{P}$ is the number of times it can be identified in $\mathscr{S}$, i.e., the number of distinct indices i and integers k for which the subsequence $\mathcal{P}$ is a pattern of $\mathcal{S}_i$. The *outcome* frequency $\hat{f}(\mathcal{P}) \in \mathbb{Z}_{\geq 0}$ is the number of such occurrences that result in the desired outcome, i.e., for which $\mathcal{S}_i\{k + |\mathcal{P}|\} = 1$. Finally, a pattern is considered frequent if $f(\mathcal{P}) \geq \theta$, where θ is chosen a priori based on domain knowledge.

Example 4.1 presents a hypotension prediction case motivated by the study in Ghosh et al.

Table 4.1: Sequential database consisting of ten sequences of blood pressure events (H:high, N:normal, L:low) and associated hypotension outcomes (1:hypotension occurred, 0:hypotension did not occur).

Seq#	Events				Seq#	Outcomes			
$\mathcal{S}_1$	H	L	L		$\mathcal{S}_1$	0	0	1	
$\mathcal{S}_2$	L	L	N	L	$\mathcal{S}_2$	0	0	0	1
$\mathcal{S}_3$	H	L			$\mathcal{S}_3$	0	1		
$\mathcal{S}_4$	L	L	N	L	$\mathcal{S}_4$	0	0	0	0
$\mathcal{S}_5$	L	H	N	H	$\mathcal{S}_5$	0	0	0	0
$\mathcal{S}_6$	H	L	L	L	$\mathcal{S}_6$	0	0	0	1
$\mathcal{S}_7$	L	L			$\mathcal{S}_7$	0	1		
$\mathcal{S}_8$	L	H	N		$\mathcal{S}_8$	0	0	0	
$\mathcal{S}_9$	H	L	L	L	$\mathcal{S}_9$	0	0	0	1
$\mathcal{S}_{10}$	L	H	N	H	$\mathcal{S}_{10}$	0	0	0	0

(2015). We use it as a running example throughout this paper.

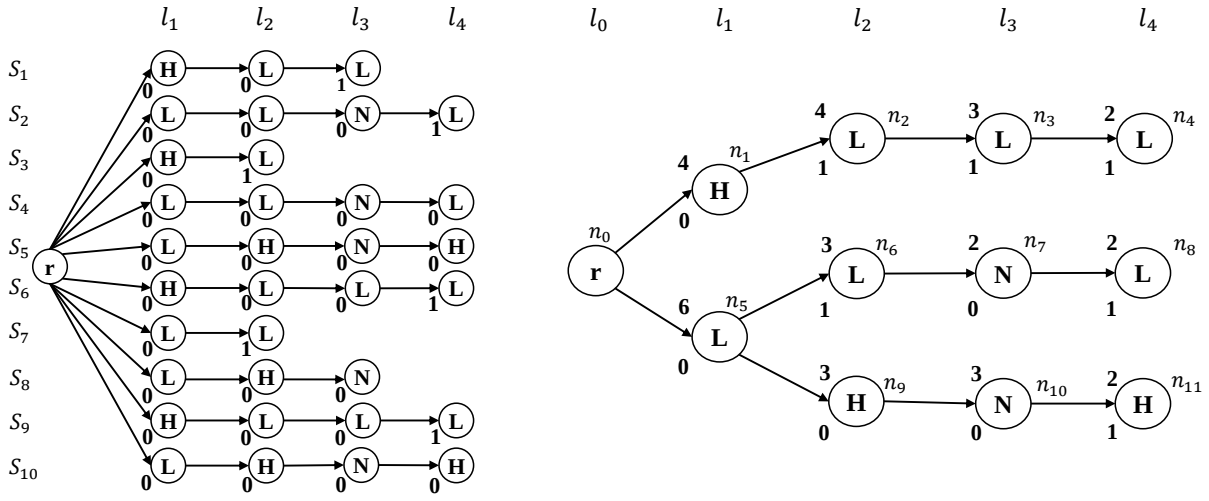
Example 4.1. Consider a sequential database which records measurements of blood pressure events and hypotension outcomes for 10 patients as depicted in Table 4.1. The sequential database $\mathscr{S}$ is composed of 10 sequences $\mathcal{S}_1, \ldots, \mathcal{S}_{10}$ of events within the space $\mathbb{E} := \{\text{L,N,H}\}$. Each event indicates whether the blood pressure was low (L), normal (N), or high (H). The pattern $\mathcal{P} = \langle L, L \rangle$ has a frequency of $f(\mathcal{P}) = 8$, occurring once in sequences $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_4, \mathcal{S}_7$ and twice in $\mathcal{S}_6$ and $\mathcal{S}_9$. The maximum length of any sequence is $M = 4$. The j-th outcome $\mathcal{S}_i\{j\}$ records whether the patient was diagnosed with hypotension ($\mathcal{S}_i\{j\} = 1$) or not ($\mathcal{S}_i\{j\} = 0$) at its j-th event. The outcome frequency of $\mathcal{P}$ is $\hat{f}(\mathcal{P}) = 4$ from $\mathcal{S}_1$, $\mathcal{S}_6$ ($k = 2$), $\mathcal{S}_7$, and $\mathcal{S}_9$ ($k = 2$). □

4.3.2 Data Tree Model of Sequential Databases

A data tree $\mathscr{D}$ is a compressed network model of a sequential database $\mathscr{S}$. Specifically, $\mathscr{D} := (\mathcal{N}, \mathcal{A}, \mathcal{H})$ is a directed tree with node set $\mathcal{N}$, arc set $\mathcal{A}$, and node label set $\mathcal{H}$. Given the maximum sequence length M of $\mathscr{S}$, the node set $\mathcal{N}$ is partitioned into $M + 1$ subsets $\mathcal{N}_0, \ldots, \mathcal{N}_m$ denoted by layers. In particular, $\mathcal{N}_0 := \{\mathbf{r}\}$ is a singleton containing an auxiliary root node $\mathbf{r}$. With each node $n \in \mathcal{N} \setminus \{\mathbf{r}\}$ we associate an event $e_n \in \mathbb{E}$, a frequency $f_n \in \mathbb{Z}_{\geq 0}$, and an outcome frequency $\hat{f}_n \in \mathbb{Z}_{\geq 0}$ that compose $\mathcal{H}$. Definition 4.2 formalizes the model $\mathscr{D}$ in terms of such node labels.

Definition 4.2. A data tree $\mathscr{D}$ models a sequential database $\mathscr{S}$ if conditions (1)-(3) are satisfied:

1. There is a one-to-one correspondence between patterns of $\mathscr{S}$ and paths in $\mathscr{D}$. That is, for any pattern $\mathcal{P}$ of $\mathscr{S}$, there exists a path $(n_1, \ldots, n_{|\mathcal{S}_i|})$ in $\mathscr{D}$ with $n_1 \neq \mathbf{r}$ that *encodes* the events of



a) Naïve data tree.

b) Minimum size data tree.

Figure 4.2: Example of a naïve and minimum size data tree model of the sequential database in Table 4.1.

$\mathcal{P}$, i.e., $e_{n_j} = \mathcal{P}[j]$ for all $j = 1, \ldots, |\mathcal{P}|$. Conversely, any sequence $\mathcal{S} = \langle e_{n_1}, \ldots, e_{n_k} \rangle$ encoded by a path $(n_1, \ldots, n_k)$ of $\mathscr{D}$ with $n_1 \neq \mathbf{r}$ and $k \leq M$ is a pattern of $\mathscr{S}$.

2. Let $\mathcal{N}(\mathcal{P}) := \{n_{|\mathcal{P}|} \in \mathcal{N} \colon (n_1, \ldots, n_{|\mathcal{P}|}) \in \mathscr{D}, e_{n_j} = \mathcal{P}[j], \quad \forall j = 1, \ldots, |\mathcal{P}|\}$ be the set of terminal nodes of the paths in $\mathscr{D}$ encoding a pattern $\mathcal{P}$. Then, $\sum_{n \in \mathcal{N}(\mathcal{P})} f_n$ is the frequency of $\mathcal{P}$ in $\mathscr{S}$,

$$\sum_{n \in \mathcal{N}(\mathcal{P})} f_n = f(\mathcal{P}).$$

3. Analogously, $\sum_{n \in \mathcal{N}(\mathcal{P})} \hat{f}_n$ is the outcome frequency of the pattern $\mathcal{P}$ in $\mathscr{S}$, i.e.,

$$\sum_{n \in \mathcal{N}(\mathcal{P})} \hat{f}_n = \hat{f}(\mathcal{P}).$$

□

Example 4.3. Figure 4.2(a) depicts a naïve data tree for the sequential database from Table 4.1. For each node $n \in \mathcal{N}$, the label within the circle represents the event $e_n \in \{L, N, H\}$, and the label below the circle indicates $\hat{f}_n$. As we have $f_n = 1$ for all $n \in \mathcal{N}$, node frequencies are omitted for clarity. There is a one-to-one correspondence between sequences in $\mathscr{S}$ and paths from $\mathbf{r}$ to leaf nodes of $\mathscr{D}$; the sequence $\mathcal{S}_i$ associated with each path is depicted on the left-hand side of the figure. Because each $\mathbf{r}$-leaf path maps to exactly some $\mathcal{S}_i \in \mathscr{S}$, condition (1)-(3) are satisfied. □

Figure 4.2(a) in Example 4.3 suggests a straightforward procedure to generate $\mathscr{D}$. Specifically, we start by constructing one path for each sequence $\mathcal{S}_i \in \mathscr{S}$, $i = 1, \ldots, N$, where the j-th node n_j of such a path satisfies $e_{n_j} = \mathcal{S}_i[j]$, $f_{n_j} = 1$, and $\hat{f}_{n_j} = u_{ij}$ for all $j = 1, \ldots, |\mathcal{S}_i|$. Finally, we add an artificial root node $\mathbf{r}$ and connect it to the first node of each path. This procedure yields a data tree with size polynomial in $\mathscr{S}$, since the number of nodes and arcs is bounded by $\mathcal{O}\left(\sum_{i=1}^{N} |\mathcal{S}_i|\right)$. However, such a tree is not necessarily of minimum size. Figure 4.2(b), for example, depicts a significantly more compact tree for $\mathscr{S}$, where the labels above each circle indicate node frequencies.

4.3.3 Minimum-sized Data Trees

We now characterize the structure of the minimum-sized data tree for $\mathscr{S}$. In particular, such a result also yields a simple procedure to transform any valid data tree into the smallest one. The size of a data tree is defined as the number of nodes $|\mathcal{N}|$. This follows from the fact that a tree has equal number of nodes and arcs, and its size is bounded by $\mathcal{O}(|\mathcal{N}|)$. Theorem 4.4 provides the necessary and sufficient conditions for a data tree to be of minimum size.

Theorem 4.4. *A data tree $\mathscr{D}$ that models $\mathscr{S}$ is of minimum size if and only if $e_n \neq e_{n'}$ for any two nodes n, $n' \in \mathcal{N}$ with the same parent node in $\mathscr{D}$. Moreover, the minimum-sized data tree for $\mathscr{S}$ is unique.*

Proof. Proof. Assume $\mathscr{D}$ is of minimum size, and assume by contradiction there exist two nodes $n, n' \in \mathscr{D} : e_n = e_{n'}$ with the same parent node $\hat{n}$. As data trees are acyclic (by definition of a tree), the path $(\mathbf{r}, \ldots, \hat{n})$ from root node $\mathbf{r}$ to parent node $\hat{n}$ is unique. Let $\mathcal{P}$ be any pattern of $\mathscr{S}$ that is encoded by a path $(n_1, \ldots, \hat{n}, n) : n_1 \neq \mathbf{r}$. As $e_n = e_{n'}$, $\mathcal{P}$ is also encoded by path $(n_1, \ldots, \hat{n}, n')$. Node n' and can thus be *merged* into node n by redirecting arcs from n' to n and updating frequencies $f_n = f_n + f_{n'}$ and $\hat{f}_n = \hat{f}_n + \hat{f}_{n'}$. This results in a smaller data tree that satisfies all conditions of Definition 4.2, a contradiction.

For the converse, we prove that nodes $n, n' \in \mathcal{N}$ that satisfy the statement cannot be merged into a single node, concluding that $\mathscr{D}$ is of minimum size. Merging any two nodes with a different parent leads to a cycle in $\mathscr{D}$ and violates its definition. On the other hand, any two nodes n, n' with the same parent and $e_n \neq e_{n'}$, encode different patterns $\mathcal{P} \neq \mathcal{P}'$ and cannot be merged while maintaining conditions of Definition 4.2.

The proof of uniqueness is by induction on the uniqueness of layers $\mathcal{N}_0, \ldots, \mathcal{N}_M$ for a minimum sized data tree $\mathscr{D}$. The statement trivially holds for singleton layer $\mathcal{N}_0 = \{\mathbf{r}\}$. Assume the statement holds for layers $\mathcal{N}_1, \ldots, \mathcal{N}_k : k < M$. As the data tree is of minimum size, any two nodes $n, n' \in \mathcal{N}_{k+1}$ are unique in their parent or their labels $e_n, e_{n'}$, giving a unique set of nodes in layer $\mathcal{N}_{k+1}$. By the principle of mathematical induction, the statement holds for all layers of the data tree. □

Theorem 4.4 indicates that, if $\mathscr{D}$ is not minimum, the tree contains two nodes n, n' that are redundant with respect to their encoded patterns. For example, in Figure 4.2(a) the sequence $\mathcal{S}_3$

Algorithm 4.1 Data Tree Compression Procedure

procedure CompressDataTree(Input: $\mathscr{D}$)
  for each layer $l = 1, \ldots, M+1$ **do**
    while there exists $n, n' \in \mathcal{N}_l$ with the same parent such that $e_n = e_{n'}$ **do**
      For each arc (n', n'') emanating from n', add an arc (n, n'') to $\mathcal{A}$;
      Update $f_n := f_n + f_{n'}$ and $\hat{f}_n := \hat{f}_n + \hat{f}_{n'}$.
      Remove node n' and its arcs, updating $\mathcal{N}$ and $\mathcal{A}$ accordingly.

could be incorporated into the first two nodes of the path encoding $\mathcal{S}_1$ by adjusting their frequency and outcome frequency accordingly, thereby obtaining a smaller valid data tree without the nodes encoding $\mathcal{S}_3$. This could be initially identified, e.g., because the first non-root nodes of $\mathcal{S}_1$ and $\mathcal{S}_3$ have the same parent $\mathbf{r}$ and event H.

To compress a given $\mathscr{D}$, we therefore must merge nodes that violate the minimality condition stated by Theorem 4.4. Merging a node n' into n consists of redirecting arcs from n' to n and updating the corresponding pattern frequencies. The procedure in Algorithm 4.1 applies such a node merging one layer at a time starting from $\mathcal{N}_1$, which we show in Proposition 4.5 always yields the minimum-sized data tree for $\mathscr{S}$. Example 4.6 exemplifies Algorithm 4.1 for our running example.

Proposition 4.5. *For any data tree $\mathscr{D}$ that models $\mathscr{S}$, the compressed data tree that results from Algorithm 4.1 also models $\mathscr{S}$ and is of minimum size. Moreover, Algorithm 4.1 can be implemented in linear-time complexity on the original data tree size, i.e., $\mathcal{O}(|\mathcal{N}|)$.*

Proof. Proof. The procedure of Algorithm 4.1 ensures that for any two nodes $n, n' \in \mathcal{N}_i, \forall i = 1, \ldots, M+1$ either have a different parent or satisfy $e_n \neq e_{n'}$. As nodes in different layers cannot have the same parent, the statement of Theorem 4.4 holds for all nodes, ensuring the minimum size of the resulting data tree. Checking the merge condition of the algorithm is performed once per node of the data tree and takes $\mathcal{O}(|\mathcal{N}|)$ time. A merge operation, in the worst case, redirects all arc in $\mathcal{A}$ using $\mathcal{O}(|\mathcal{A}|)$ time. As $|\mathcal{N}| = |\mathcal{A}|$ in a tree, the algorithm takes $\mathcal{O}(|\mathcal{N}|)$ total time. □

Example 4.6. Consider the naïve data tree model of Figure 4.2(a). In the first iteration of Algorithm 4.1, the set of nodes in layer $\mathcal{N}_1$ (i.e., all nodes n with $\mathbf{r}$ as parents) can be partitioned into two classes, one for event H and another for event L. Nodes in each class are then merged, resulting in layer $\mathcal{N}_1$ of Figure 4.2(b) with their updated frequency (shown above each circle) and outcome frequency labels (shown below each circle). This process is repeated one layer at a time until the data tree of Figure 4.2(b) is obtained. □

Lastly, we provide a characterization of the memory requirement of the minimum-sized tree $\mathscr{D}$ as compared to a traditional tabular (i.e., explicit) representation of $\mathscr{S}$. Theorem 4.7 below indicates that $\mathscr{D}$ is guaranteed to be more compact whenever the event space $\mathbb{E}$ is smaller than the number of sequences N in the database. Thus, as the number of events $|\mathbb{E}|$ is typically orders

of magnitudes smaller than the number of sequences N in a large databases, data trees are also expected to be orders of magnitude more memory efficient than a tabular encoding. We indeed observe such memory savings, e.g., in our studied large-scale applications in §4.5 and §4.6.

Theorem 4.7. *A minimum sized data tree model $\mathscr{D}$ of a sequential database $\mathscr{S}$ is more memory efficient than a tabular encoding of $\mathscr{S}$ whenever $|\mathbb{E}| < N$, and otherwise never larger.*

Proof. Proof. The worst case size of a data tree $\mathscr{D}$ is when all N sequences $\mathcal{S} \in \mathscr{S}$ are modelled by exactly N unique paths in $\mathscr{D}$, i.e., a naïve data tree model of $\mathscr{S}$. In such a case, a node $n \in \mathcal{N}$ uses two units of memory, one for node label e_n and the other for label $\hat{f}_n$. Note that in a naïve data tree $f_n = 1, \forall n \in \mathcal{N}$ and may be omitted. This space complexity is consistent with the memory used by a tabular encoding, one to store event e_{ij} and the other to store outcome u_{ij}. Arcs $(n, n') \in \mathcal{A}$ may be defined as linked lists between nodes $n \in \mathcal{N}$, to not consume extra memory. The exact space requirement of a naïve data tree is thus bounded by that of a tabular encoding.

We now prove that if $|\mathbb{E}| < N$, at least one merge operations can be performed the worst-case size of a data tree (naïve data tree) leading to a lower memory requirement. A merge operation is guaranteed to lower memory requirements, as it merges two nodes that use at least two units of memory into one node that uses three units of memory, one per labels $e_n, f_n, \hat{f}_n$. Now, assume by contradiction that no merge operation can be performed for any two nodes in $\mathscr{D}$. By Theorem 4.4, this implies that for any two nodes $n, n' \in \mathcal{N}_1$ (which have the same parent $\mathbf{r}$) we have $e_n \neq e_{n'}$. Moreover, as each sequence $\mathcal{S} \in \mathscr{S}$ is non-empty and has at least one event, we have $|\mathcal{N}_1| = N$. Given $|\mathbb{E}| < N$, and by assigning one event $e \in \mathbb{E}$ to each of the N nodes $n \in \mathcal{N}_1$, we are thus guaranteed to have $e_n = e_{n'}$ for at least one pair of nodes $n, n' \in \mathcal{N}_1$ according to the pigeon hole principle, a contradiction. □

4.3.4 Incremental Compilation of Minimum-sized Data Trees

While Algorithm 4.1 efficiently minimizes the size of any given data tree, the procedure still requires some initial $\mathscr{D}$ as input (e.g., the naïve representation of Figure 4.2(a)), which may not be computationally feasible for large-scale databases due to memory considerations. In this section, we provide an alternative procedure that iteratively builds a minimum-sized data tree by incorporating one sequence at a time from $\mathscr{S}$. Such a construction methodology is also more appropriate for dynamic databases that are often updated with new sequences.

The incremental procedure is depicted in Algorithm 4.2. Given a sequence $\mathcal{S}_i = \langle e_{i1}, \ldots, e_{i|\mathcal{P}|} \rangle$, the procedure iterates over each e_{ij} to check if a subpath of $\mathscr{D}$ matches all the events up to j. If such a path exists, it suffices to update the associated frequencies at each node. Otherwise, new nodes are added to incorporate $\mathcal{S}_i$ into $\mathscr{D}$. Proposition 4.8 states the validity of Algorithm 4.2 by showing that the conditions of Definition 4.2 and Theorem 4.4 hold at every iteration of the procedure. Example 4.9 exemplifies Algorithm 4.2 for our running example.

Algorithm 4.2 Incremental Data Tree Compilation

1: **procedure** IncrementalCompilation(Input: $\mathscr{S}$)
2: Initialize $\mathscr{D}$ with $\mathcal{N} := \{\mathbf{r}\}$, $\mathcal{A} := \emptyset$
3: Let $n^* := \mathbf{r}$
4: **for** each $\mathcal{S}_i \in \mathscr{S}$ and $j = 1, \ldots, |\mathcal{S}_i|$ **do**
5: **if** there exists $(n^*, n) \in \mathcal{A}$ for some $n \in \mathcal{N}$ such that $e_n = \mathcal{S}_i[j]$ **then**
6: Update $f_n := f_n + 1$ and $\hat{f}_n := \hat{f}_n + \mathcal{S}_i\{j\}$
7: Update $n^* := n$
8: **else**
9: Add new node n to $\mathcal{N}_j$ with $e_n = \mathcal{S}_i[j]$, $f_n = 1$, and $\hat{f}_n = \mathcal{S}_i\{j\}$
10: Update $n^* := n$
11: **return** $\mathscr{D}$.

Proposition 4.8. *The incremental compilation of Algorithm 4.2 outputs a valid minimum-sized data tree model of $\mathscr{S}$. Moreover, its run-time complexity is linear in the size of $\mathscr{S}$, i.e., $\mathcal{O}(\sum_{i=1}^{N} |\mathcal{S}_i|)$.*

Proof. Proof. The validity of Definition 4.2 is proved by induction over the iterations of Algorithm 4.2. All conditions trivially hold at iteration 1, as the algorithm outputs a single path identical to the first sequence $\mathcal{S}_1 \in \mathscr{S}$. Assume that the data tree built up to iteration $i < N$ is valid with respect to the conditions of definition 4.2. Let $(n_1, \ldots, n_j, \ldots, n_{|\mathcal{S}_{i+1}|})$ be the path constructed in iteration $i+1$, where nodes $n_1, \ldots, n_j$ were built in previous iterations, and nodes $n_{j+1}, \ldots, n_{|\mathcal{S}_{i+1}|}$ were built in iteration $i+1$. Condition 4.2-(1) holds for every node of path $(n_1, \ldots, n_j, \ldots, n_{|\mathcal{S}_{i+1}|})$ due to Step 4.2-9. Conditions 4.2-(2) and 4.2-(3), hold for any pattern $\mathcal{P}$ with terminal node $n_{j'} : j' \leq j$ due to Step 4.2-6, and for any pattern with terminal node $n_{j'} : j' > j$ due to Step 4.2-9. By the principle of mathematical induction, the statement holds for every iteration of Algorithm 4.2.

Algorithm 4.2 ensures that a node $n \in \mathscr{D}$ has no two children n, n' such that $e_n = e_{n'}$, thus satisfying the conditions of Theorem 4.4. The algorithm constructs the minimum size data tree by one scan of the sequential database, i.e., in $\mathcal{O}(\sum_{i=1}^{N} |\mathcal{S}_i|)$ time. □

Example 4.9. Using Algorithm 4.2, we model the sequential database $\mathscr{S}$ of Table 4.1 in ten iterations, one per sequence $\mathcal{S}_i \in \mathscr{S}$, $i = 1, \ldots, 10$. In the first iteration, sequence $\mathcal{S}_1$ is modelled by creating path $(\mathbf{r}, n_1, n_2, n_3)$, such that $e_{n_1} = H$; $e_{n_2} = L$; $e_{n_3} = L$; $f_{n_j} = 1$ for all $j = 1, 2, 3$; $\hat{f}_{n_j} = 0$ for all $j = 1, 2$ and $\hat{f}_{n_3} = 1$. This path is depicted in Figure 4.2(b). The second iteration is similar to the first iteration and creates path $(\mathbf{r}, n_5, n_6, n_7, n_8)$ to model sequence $\mathcal{S}_2$. In the third iteration, sequence $\mathcal{S}_3$ is modelled by updating the frequency labels $f_{n_1} = 2$, $f_{n_2} = 2$, and outcome frequency $\hat{f}_{n_1} = 0$, $\hat{f}(n_2) = 1$ of the path built in the first iteration. The process is repeated up to sequence $\mathcal{S}_{10}$, resulting in the minimum sized data tree of Figure 4.2(b). □

Algorithm 4.3 Sequential Pattern Mining over $\mathscr{D}$

procedure SEQUENTIALFREQUENTPATTERN(Input: $\mathscr{D}$)
  Initialize $\mathscr{F} := \left\{(\mathcal{P}, \mathcal{N}(\mathcal{P})) : \mathcal{P} = \langle e \rangle,\ \forall e \in \mathbb{E}, \sum_{n \in \mathcal{N}(\mathcal{P})} f(\mathcal{P}) \geq \theta\right\}$
  for each pair $(\mathcal{P}, \mathcal{N}(\mathcal{P})) \in \mathscr{F}$ not yet verified **do**
    for each event $e' \in \mathbb{E}$ **do**
      Let $\mathcal{N}' := \{n' \in \mathcal{N} \colon e_{n'} = e',\ \exists (n, n') \in \mathcal{A} \text{ for some } n \in \mathcal{N}(\mathcal{P})\}$
      if $\sum_{n \in \mathcal{N}'} f(\mathcal{P}) \geq \theta$ **then**
        Add $(\langle \mathcal{P}, e' \rangle, \mathcal{N}')$ to $\mathscr{F}$
  return $\mathscr{F}$.

4.3.5 Mining Sequential Patterns from Data Trees

We now present a methodology to exploit the compression provided by a data tree $\mathscr{D}$ to extract frequent patterns more efficiently than using a tabular encoding of $\mathscr{S}$. We restrict our attention to patterns $\mathcal{P}$ that satisfy $f(\mathcal{P}) \geq \theta$ for a given threshold θ as discussed in §4.3.1. For an application that defines the frequency threshold with respect to outcomes, i.e., $\hat{f}(\mathcal{P}) \geq \theta$, the procedure below can be adapted directly by replacing the functional $f(\cdot)$ by $\hat{f}(\cdot)$ throughout.

Our procedure is a special case of a prefix-projection algorithm applied to the data tree model of the database. Specifically, it follows from Definition 4.2-(2) that a pattern $\mathcal{P}$ is frequent if

$$\sum_{n \in \mathcal{N}(\mathcal{P})} f(\mathcal{P}) \geq \theta,$$

where $\mathcal{N}(\mathcal{P})$ is the set of terminal nodes of the subpaths in $\mathscr{D}$ that encode $\mathcal{P}$. To identify these patterns, the procedure maintains a set of pattern-node pairs $(\mathcal{P}, \mathcal{N}(\mathcal{P}))$ that are already known to be frequent. For each such pair $(\mathcal{P}, \mathcal{N}(\mathcal{P}))$, it then verifies if it is possible to extend $\mathcal{P}$ into a new frequent pattern $\mathcal{P}' = \langle \mathcal{P}, e' \rangle$ for each possible event $e' \in \mathbb{E}$. To that end, and due to Definition 4.2-(1), it suffices to verify the frequency condition over the nodes n' that are children of $\mathcal{N}(\mathcal{P})$ and such that $e_{n'} = e'$.

The full procedure is depicted in Algorithm 4.3. For the initial set of frequent patterns, we start with single-pattern events $\langle e \rangle$ for all $e \in \mathbb{E}$ that are frequent. Such sets are readily available during the construction of the data tree by Algorithm 4.2. Each iteration of the mining algorithm consists of extending such patterns with one additional event by checking the children of the nodes in $\mathcal{N}(\mathcal{P})$. The validity of the procedure follows from the conditions of Definition 4.2, and the validity of the prefix-projection algorithm (Han et al., 2001). We provide a lower bound on the efficiency of Algorithm 4.3 in comparison to existing techniques that operate directly on $\mathscr{S}$ in Proposition 4.10.

Proposition 4.10. *The run-time complexity of Algorithm 4.3 is at least* $\sum_{i=1}^{N} |\mathcal{S}_i|/|\mathcal{N}|$ *times more efficient than a prefix-projection algorithm performed over a tabular encoding of* $\mathscr{S}$.

Proof. Proof. Extending a pattern $\mathcal{P}$ by mining a data tree $\mathscr{D}$, involves scanning the children of

Table 4.2: Frequent patterns (with $\theta = 2$) mined from data tree of Figure 4.2(b) (or sequential database of Table 4.1).

Pattern	Events				$f(\mathcal{P})$	$\hat{f}(\mathcal{P})/f(\mathcal{P})$	$\mathbb{P}^{H_0}(\mathcal{P})$	Adj. $f(\mathcal{P})$	Adj. $\mathbb{P}^{H_0}(\mathcal{P})$
$\mathcal{P}_1$	H	L			4	0.25	0.05	400	0.21
$\mathcal{P}_2$	H	L	L		3	0.33	0.07	300	0.29
$\mathcal{P}_3$	H	L	L	L	2	1	0.42	200	0.98
$\mathcal{P}_4$	H	N			3	0	0	300	0
$\mathcal{P}_5$	H	N	H		2	0	0	200	0
$\mathcal{P}_6$	L	L			8	0.5	0.24	800	0.47
$\mathcal{P}_7$	L	L	N		2	0	0	200	0
$\mathcal{P}_8$	L	L	N	L	2	0	0	200	0
$\mathcal{P}_9$	L	L	L		2	1	0.42	200	0.98
$\mathcal{P}_{10}$	L	N			2	0	0	200	0
$\mathcal{P}_{11}$	L	N	L		2	0	0	200	0
$\mathcal{P}_{12}$	L	H			3	0	0	300	0
$\mathcal{P}_{13}$	L	H	N		3	0	0	300	0
$\mathcal{P}_{14}$	L	H	N	H	2	0	0	200	0
$\mathcal{P}_{15}$	N	H			2	0	0	200	0
$\mathcal{P}_{16}$	N	L			2	0.5	0.12	200	0.44

nodes in $\mathcal{N}(\mathcal{P})$. A scan of a node n in $\mathscr{D}$ is equivalent to scanning f_n sequences in a sequential database $\mathscr{S}$. The prefix-projection over a sequential database $\mathscr{S}$ scans each event $e_{ij} \in \mathscr{S}$ at least once (Han et al., 2001). Similarly, the algorithm scans each node of the data tree at least once. Therefore, the minimum $|\mathcal{N}|$ scans of all nodes $n \in \mathcal{N}$ is equivalent to $\sum_{n \in \mathcal{N}} f_n = \sum_{i \in \{1,\dots,N\}} |S_i|$ number of scans on the sequential database $\mathscr{S}$. Pattern mining over a data tree is thus at least $\sum_{i \in \{1,\dots,N\}} |S_i|/|\mathcal{N}|$ times more efficient than pattern mining over a sequential database $\mathscr{S}$. □

Example 4.11. Consider a minimum frequency $\theta = 2$. When applying Algorithm 4.3 to $\mathscr{D}$ from Figure 4.2(b), we initialize $\mathscr{F}$ with three pairs $(H, \{n_1, n_9, n_{11}\})$, $(L, \{n_2, n_3, n_4, n_5, n_6, n_8\})$, and $(N, \{n_7, n_{10}\})$. Next, the algorithm takes, e.g., the pair $(H, \{n_1, n_9, n_{11}\})$, which has a frequency $f_{n_1} + f_{n_9} + f_{n_{11}} = 9 \geq \theta$, and verifies the children of nodes $\{n_1, n_9, n_{11}\}$ for three possible extensions of event H to patterns $\langle H, H\rangle$, $\langle H, L\rangle$, or $\langle H, N\rangle$. The frequent extensions are $\langle H, L\rangle$ with frequency 4 ($f_{n_2} = 4$), and $\langle H, N\rangle$ with frequency 3 ($f_{n_{10}} = 3$). The pairs $(\langle H, L\rangle, \{n_2\})$, and $(\langle H, N\rangle, \{n_{10}\})$ are thus added to $\mathscr{F}$. The procedure is repeated until all patterns in Table 4.2 are found. □

4.4 Interpretable Explanations and Knowledge Trees

In this section, we leverage the same principles of the network encoding of §4.3 to extract knowledge from frequent sequential patterns. We begin in §4.4.1 with a generalization of association rules for interpreting sequential outcomes and develop a p-value test to estimate likelihood probabilities. Next, we formally define the concept of explanatory patterns in §4.4.2 and present a regression test to establish statistical guarantees of its validity. Based on these concepts, we develop in §4.4.3 a structural view of frequent patterns and their resulting association rules in the form of *knowledge trees*. A knowledge tree provides a graphical and compact tool of the (possibly many) frequent patterns and can be used either for deriving insights or directly as a decision support tool. We note that the concepts presented in this section only require a set of frequent patterns that are generated, e.g., using the data tree procedure from §4.3.

4.4.1 Sequential Association Rules and Likelihood Probabilities

We recall from §4.3.1 that we wish to identify patterns that typically lead to a desired outcome $1 \in \mathbb{U}$. Such an association is represented here through a *rule* of the form $\mathcal{P} \rightarrow 1$, which indicates that a pattern $\mathcal{P}$ of $\mathscr{S}$ frequently leads to the outcome 1. A straightforward measure of the strength of this association could be obtained, e.g., by the ratio between the number of desired outcomes following $\mathcal{P}$ and the frequency of $\mathcal{P}$, that is, $\frac{\hat{f}(\mathcal{P})}{f(\mathcal{P})}$.

Nonetheless, since a sequential database $\mathscr{S}$ encodes a sample of observations, such a ratio is only an estimate of the true association probability between $\mathcal{P}$ and the desired outcome. In order to generate more accurate estimates, we propose to use p-value hypothesis tests to derive a *likelihood probability* of the desired outcome given the pattern $\mathcal{P}$. Hypothesis tests have been previously used to determine whether generated association rules are statistically significant (Klayman and Ha, 1989; Riondato and Upfal, 2012; Hämäläinen and Nykänen, 2008). Statistically significant rules are then approved and considered valid, while non-significant rules are disregarded.

Specifically, let $\mathbb{P}(\mathcal{P})$ be the true probability of the association between a pattern $\mathcal{P}$ and the desired outcome. For a frequent pattern $\mathcal{P}$ (i.e., one with frequency larger than θ), we state the null hypothesis as $H_0 : \mathbb{P}(\mathcal{P}) < \mathbb{P}^{H_0}(\mathcal{P})$, where we denote by $\mathbb{P}^{H_0}(\mathcal{P})$ the desired likelihood probability of the outcome given $\mathcal{P}$. The task is to find the minimum $\mathbb{P}^{H_0}(\mathcal{P})$ such that the hypothesis test gives a 95% or higher confidence to reject H_0. Thus, to obtain $\mathbb{P}^{H_0}(\mathcal{P})$, we initialize $\mathbb{P}^{H_0}(\mathcal{P}) = \frac{\hat{f}(\mathcal{P})}{f(\mathcal{P})}$, and incrementally decrease $\mathbb{P}^{H_0}(\mathcal{P})$ until the p-value test gives a confidence of at least 95% for rejecting H_0. Example 4.12 exemplifies this calculation for our running example.

Example 4.12. Consider the output patterns in Table 4.2. Each pattern is a sequential association rule leading to hypotension with a ratio of $\hat{f}(\mathcal{P})/f(\mathcal{P})$, given in the fourth column, and a likelihood of $\mathbb{P}^{H_0}(\mathcal{P})$, given in the fifth column. Observe that the likelihood probabilities are considerably lower than ratios $\hat{f}(\mathcal{P})/f(\mathcal{P})$ due to the low frequencies $f(\mathcal{P})$. For example, although pattern

$\langle L, L, L\rangle$ leads to hypotension in 100% of its occurrences, its likelihood probability is only 42% with a statistical confidence of 95%.

For illustration purposes, we multiply the frequency (and outcome frequency) of all patterns in Table 4.2 by a 100, and re-run the p-value hypothesis test to observe the change in pattern likelihoods. We observe that while ratios $\hat{f}(\mathcal{P})/f(\mathcal{P})$ remain unchanged, the adjusted likelihood probabilities increase as shown in column seven. For example, adjusting the frequency and outcome frequency of pattern $\langle L, L, L\rangle$ from 2 to 200, increases its likelihood probability from 42% to 98%. □

4.4.2 Explanation of Outcomes and Regression Testing

A high likelihood probability $\mathbb{P}^{H_0}(\mathcal{P})$ suggests a possible explanation of the outcome in the form of the pattern $\mathcal{P}$. We formalize this concept in Definition 4.13 below.

Definition 4.13. A pattern $\mathcal{P}$ *explains* the outcome $\mathcal{S}_i\{j^*\} = 1$ for a sequence $\mathcal{S}_i \in \mathscr{S}$, $j^* \leq |\mathcal{S}_i|$, if

1. $\mathcal{P}$ is a pattern of $\mathcal{S}_i$ that ends in j^*, i.e., there exists some integer k such that $\mathcal{P}[j] = \mathcal{S}_i[k+j]$ for $j = 1, \ldots, |\mathcal{P}|$ and $k + |\mathcal{P}| = j^*$.
2. $\mathbb{P}^{H_0}(\mathcal{P}) \geq \mathbb{P}(1) + \alpha$, where $\mathbb{P}(1)$ is the probability of the outcome 1 (i.e., number of events e_{ij} in $\mathscr{S}$ with $\mathcal{S}_i\{j\} = 1$ divided by the total number of events e_{ij} in $\mathscr{S}$) and $\alpha > 0$ is a constant probability chosen a priori based on domain knowledge. □

Condition (2) from Definition 4.13 above allows us to evaluate outcomes that rarely occur in $\mathscr{S}$, since we consider a high marginal change in outcome probabilities (in the form of α) to regard a pattern as a possible explanation of an outcome. We show its application in Example 4.14.

Example 4.14. Observe that the overall probability of hypotension is $\mathbb{P}(1) = 6/34 = 17.6\%$ in the database of Table 4.1. Using $\alpha = 5\%$ and the original pattern frequencies, only pattern $\langle L, L\rangle$ is feasible. In other words, pattern $\langle L, L\rangle$ increases the probability of hypotension by at least 5%. However, 5% may be too low to consider pattern $\langle L, L\rangle$ as an explanation for hypotension.

Given the adjusted pattern frequencies and a higher value $\alpha = 80\%$, patterns $\langle H, L, L, L\rangle$ and $\langle L, L, L\rangle$ are feasible with a high likelihood of 98%. Both patterns increase the probability of hypotension by at least 80% and may be considered as interpretable explanations for hypotension, e.g., hypotension followed after three consecutive low blood pressure events, regardless of an initial high blood pressure. □

A key aspect for obtaining useful explanations, thus, consists of defining an appropriate α. Imposing a higher α leads to a stronger and more concise set of patterns that can explain a desired

outcome. Nonetheless, if α is too high, we may be left with little or no explaining patterns. This trade-off between values of α and the percentage of explained outcomes can be evaluated using regression hypothesis tests. Specifically, we determine if there exists a statistically significant linear relationship between the occurrence of outcome 1 in $\mathscr{S}$, denoted by Y, and the number of those outcomes that can be explained using a pattern $\mathcal{P}$ satisfying the α constraint, denoted by X. Note that we have N measurements for each X and Y, i.e., one for each sequence $\mathcal{S}_i$ in $\mathscr{S}$. Thus, our null hypothesis is if $\beta = 0$ in the best-fit line $Y = \beta X + \beta_0$.

The result of the regression hypothesis test is assessed using two measures. The first is the statistical confidence that the null hypothesis can be rejected. The second measure is the coefficient of determination, i.e., R^2, which determines the goodness of fit in linear regression. In the context of our analysis, R^2 is the percentage of the desired outcomes that can be explained by patterns satisfying Definition 4.13. Example 4.15 performs a regression hypothesis test for our case study.

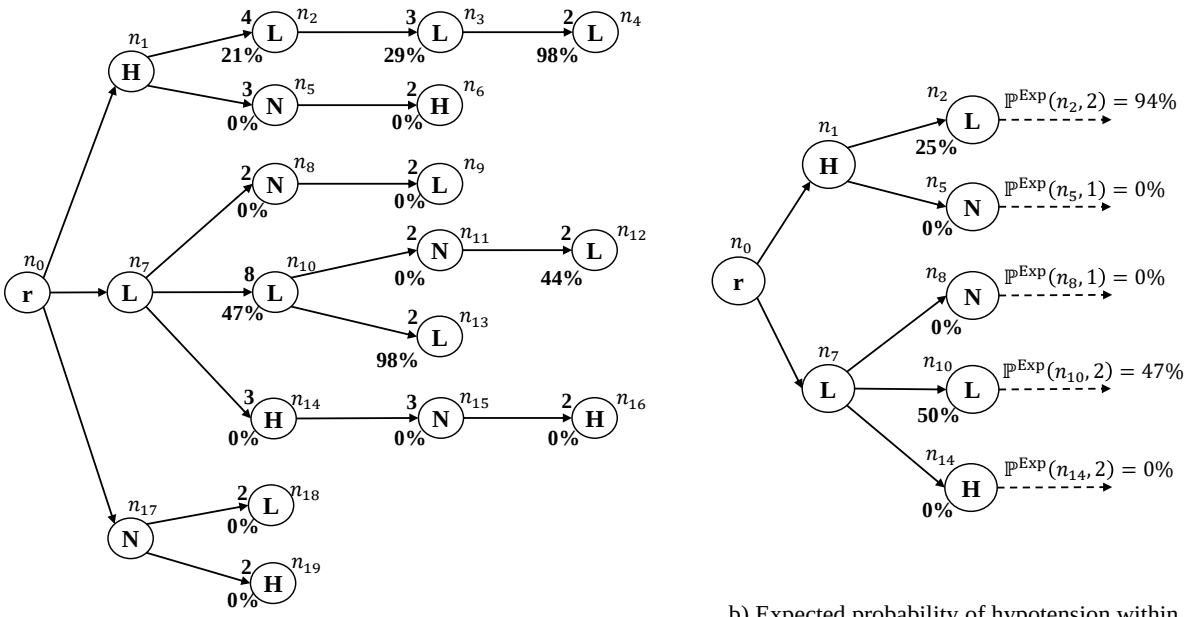
Example 4.15. From Example 4.14, $\alpha = 80\%$ leads to two explanatory patterns $\langle H, L, L, L\rangle$ and $\langle L, L, L\rangle$ (for patterns with adjusted frequency). Counting the number of hypothesis outcomes ($\mathcal{S}_i\{j\} = 1$) in each row of the original $\mathscr{S}$ gives the vector $Y = [1, 1, 1, 0, 0, 1, 1, 0, 1, 0]$. Counting the number of times a hypotension outcome is explained by either of the patterns gives the vector $X = [0, 0, 0, 0, 0, 1, 0, 0, 1, 0]$. A regression hypothesis test results in the linear relationship of $Y = 0.5X + 0.5$ with a confidence of 24%, and $R^2 = 17\%$. The two patterns are thus not statistically significant nor sufficient to explain hypotension outcomes in $\mathscr{S}$.

In order to explain hypotension outcomes, we decrease α to 25% leading the addition of $\langle L, L\rangle$ and $\langle N, L\rangle$ to the set of explanatory patterns. Using these new additions we have vector $X = [1, 1, 0, 0, 0, 1, 1, 0, 1, 0]$. The regression hypothesis test now gives the linear relationship of $Y = 0.8X + 0.2$ with a confidence of 99%, and $R^2 = 67\%$. The new four explanatory patterns are thus statistically significant, and sufficient to explain 67% of hypotension outcomes in $\mathscr{S}$. □

The coefficient of determination R^2, also indicates the degree to which variations in X, i.e., occurrences of explanatory patterns, influence Y, i.e., the occurrence of outcome. Therefore, a high value of R^2 indicates that by following (or avoiding) the sequence of events/decisions in mined patterns, we are statistically guaranteed to influence the occurrence of our outcome of interest $1 \in \mathbb{U}$. This provides the basis on using patterns as a decision support tool for sequential decision making, which we further exemplify in our case studies in §4.5 and §4.6.

4.4.3 Knowledge Trees for Decision Making

Our structural view of the decision making process is based on aggregating frequent patterns to construct a *knowledge tree*. Similar to data trees, knowledge trees provide a compact tree model of patterns and are more efficient in memory and analytical tasks performed over mined patterns. The main contribution of knowledge trees is, however, a visual representation of the overall decision making process for deriving insights.



a) Knowledge tree model of hypotension patterns.

b) Expected probability of hypotension within next two events.

Figure 4.3: Knowledge tree model of patterns mined from the database of Table 4.1, and the expected probability of hypotension within next two events.

A knowledge tree $\mathscr{K}$ is equivalent to a data tree $\mathscr{D}$ and is built using the same procedure depicted in Algorithm 4.2, except that the input database is substituted by the output patterns of the pattern mining algorithm. A path $(\mathbf{r}, n_1, \ldots, n_k)$ in a knowledge tree thus models k frequent patterns, i.e., $\mathcal{P} = \langle e_{n_1}, \ldots, e_{n_j} \rangle$, $\forall j = 1, \ldots, k$, each with a frequency of f_{n_j}. Furthermore, we let $\mathbb{P}(n_j) := \mathbb{P}^{H_0}(\mathcal{P})$ be the likelihood probability of the pattern $\mathcal{P}$ corresponding to $j = k$.

The knowledge tree brings to light how the sequence of events $e_1, \ldots, e_k$ affects the progression of outcome probabilities $\mathbb{P}(n_1), \ldots, \mathbb{P}(n_k)$. This information can aid decision making, e.g., by providing guidelines on how the next sequence of decisions should be made. For example, paths with increasing likelihood probabilities $\mathbb{P}(n_1) \leq \cdots \leq \mathbb{P}(n_k)$ serve as an example of the sequence of events that are increasing the chance of occurrence of the outcome. Example 4.16 shows the knowledge tree model of output patterns for our hypotension example, and how likelihood probabilities factor into a structural view of blood pressure events and their progression towards hypotension.

Example 4.16. Figure 4.3(a) shows the knowledge tree model $\mathscr{K}$ of the frequent patterns in Table 4.2. As with data trees, the label within each circle represents event e_n, and the label above the circle indicates frequency f_n. However, the label below the circle in knowledge trees indicates $\mathbb{P}(n)$ rather than outcome frequency $\hat{f}_n$. Path $(\mathbf{r}, n_1, n_2, n_3, n_4) \in \mathscr{K}$ models patterns $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$, with likelihood probabilities $0.21, 0.29, 0.98$ associated to nodes n_2, n_3, n_4, respectively. We can observe that the probability of hypotension increases as the number of low blood pressure events increase. Therefore, in order to prevent hypotension, the patient must be treated to avoid consecutive low

blood pressure events. □

Knowledge trees are primarily designed to provide a simple and interpretable view of the decision making process. However, as the number and length of outgoing paths from a node $n \in \mathscr{K}$ increases, the complexity of analyzing future events and outcome probabilities increases. Consequently, the interpretability of the knowledge tree may decrease. A possible solution to mitigate the complexity resulted from the size of a knowledge tree is to equip its nodes with statistics generated from future paths. For example, we can equip each node with the expected probability that an outcome occurs within a specific number of future events. This expected probability provides an estimate of future outcome probabilities after node n, without the need to analyze all its outgoing paths.

Formally, the expected probability $\mathbb{P}^{\mathbb{E}\text{xp}}(n, j)$ is defined as the percentage of the next j nodes n', following node n, that are associated with the desired outcome. That is,

$$\mathbb{P}^{\mathbb{E}\text{xp}}(n, j) = \frac{\sum\limits_{n':\exists(n,\ldots,n')\in\mathscr{K},|(n,\ldots,n')|\leq j} f_{n'} \times \mathbb{P}(n')}{\sum\limits_{n':(n,n')\in\mathscr{K}} f_{n'}}$$

where $|(n, \ldots, n')|$ is the length of the path from node n to node n'. The expected probability $\mathbb{P}^{\mathbb{E}\text{xp}}(n, j)$ thus averages the likelihood probabilities of the j future events. Moreover, $\mathbb{P}^{\mathbb{E}\text{xp}}(n, j)$ compliments the immediate probabilities $\mathbb{P}(n)$ by providing a possible way to anticipate outcome $1 \in \mathbb{U}$ earlier in the decision making process and hedge against myopic decision making. Example 4.17 demonstrates how expected probabilities are calculated and used for our hypotension case example.

Example 4.17. Consider the knowledge tree of Figure 4.3(a). A high blood pressure H followed by a drop to low blood pressure L leads to hypotension in all patients within the next 1-3 periods, and has an expected likelihood of $\mathbb{P}^{\mathbb{E}\text{xp}}(n_2, 2) = \frac{3\times 0.29+2\times 0.98}{3} = 94.3\%$. Similarly, two consecutive low blood pressure events L lead to hypotension in three patients out of a possible six, giving an expected hypotension likelihood of $\mathbb{P}^{\mathbb{E}\text{xp}}(n_{10}, 2) = 47\%$ within the next 2 periods. This information can be integrated into the knowledge tree as shown in Figure 4.3(b). □

4.5 Application: User Skips in Music Streaming Platforms

As our first application, we study online music streaming platforms where musical tracks are sequentially recommended to a user. The sequential decision making problem involves determining the next track to recommend based on the user's previously streamed songs. As with any recommender systems, the objective is to maximize user satisfaction through recommendations. Higher user satisfaction leads to higher opportunities to increase profits, e.g., with more user tolerance for advertisement. On the other hand, if a track recommendation does not satisfy the user's preference, he or she may skip the song or leave the platform.

The literature is rich with research on music recommender systems, which generally profile users by musical taste and recommend tracks similar to that profile (Van den Oord et al., 2013; Chen and Chen, 2001; Logan, 2004; Hijikata et al., 2006). Interestingly, despite the high accuracy of recommender systems and their capability to consider the user's contextual factors (e.g., Cheng and Shen (2016); Hariri et al. (2012); Wang et al. (2012)), the skip rate of users remains high in practice. A study published by Spotify,[1] one of the largest and most successful music streaming platforms with 190 million active users, shows that an astonishing 50% of recommended tracks are skipped by their users. Moreover, 25% of tracks are skipped within the first five seconds of streaming, which suggest recommendations may not be optimal.

Recently, Spotify released a "sequential skip prediction challenge"[2] where contestants were asked to predict whether users will skip their recommended track given their immediately preceding interactions with the system. The winning criteria was to maximize skip prediction accuracy, which entailed roughly 700 submissions of complex prediction models. In particular, the winning model was a recurrent neural network with 81.2% overall accuracy on test instances. The recurrent neural network model also achieved the highest weighted average accuracy of 65.1%, where skip predictions for tracks later in the user session were given higher weight.

Although the recurrent neural network model achieves high prediction accuracy, it faces two main challenges when used to reduce user skips rates. First, recurrent neural networks are highly non-linear supervised learners and do not provide an interpretable explanation for their prediction, i.e., why a skip outcome is predicted. Thus, it is difficult to use their prediction to determine how user skips may be prevented, e.g., by changing the recommendation sequence. Secondly, the prediction of recurrent neural networks, by design, only apply to the next immediate recommendation, and hence myopic for decision making beyond the next period. This is confirmed by the drop of accuracy from 81.2% to 65.1% in the winning model for Spotify when used to predict longer future sequences.

To better understand user skip behavior, we focus on generating interpretable explanations on why a skip outcome occurred. In particular, we find frequent patterns of track recommendation and analyze their association to user skip likelihoods. Moreover, we design a knowledge tree of frequent recommendations as a decision support tool for the track recommendation task. We begin by analyzing the sequential database provided by Spotify, its data tree model, and tailored pattern mining algorithm.

4.5.1 Data Tree Model of the Spotify Database and Pattern Mining

For our experiments, we use the database provided by Spotify for the sequential skip prediction challenge.[3] The database consists of approximately 130 million user sessions of 10-20 track recommendations, resulting in over 2 billion listening events. Each event is associated with user

[1] https://musicmachinery.com/2014/05/02/the-skip/
[2] https://www.crowdai.org/challenges/spotify-sequential-skip-prediction-challenge
[3] https://www.crowdai.org/challenges/spotify-sequential-skip-prediction-challenge/dataset_files

Table 4.3: Audio features of tracks, and discretization thresholds.

Feature	Data range	Discretization threshold	Feature	Data range	Discretization threshold threshold
Acousticness	0-100	0-5, 5-20, 20-50, 50-100.	Beat strength	0-100	0-30, 30-45, 45-55, 55-70, 70-100.
Bounciness	0-100	0-30, 30-45, 45-55, 55-70, 70-100.	Danceability	0-100	0-30, 30-45, 45-55, 55-70, 70-100.
Dynamic range	0-53	0-5, 5-9, 9-12, 12-15, 15-53.	Energy	0-100	0-30, 30-45, 45-55, 55-70, 70-100.
Flatness	0-117	0-95, 95-99, 99-102, 102-104, 104-108, 108-117.	Instrumentalness	0-100	0, 1-100.
Key	Pitch Class*	C, C♯ (D♭), D, D♯ (E♭), E, F, F♯ (G♭), G, G♯ (A♭), A, A♯ (B♭), B.	Liveness	0-100	0-15, 15-100.
			Loudness	0-66	0-40, 40-55, 55-66.
Mechanism	0-100	0-30, 30-45, 45-55, 55-70, 70-100.	Organism	0-100	0-30, 30-45, 45-55, 55-70, 70-100.
Popularity	90-100	90,91,92,93,94,95,96,97,98,99,100.	Speechiness	0-100	0-30, 30-45, 45-55, 55-70, 70-100.
Time signature	1-6	1,2,3,4,5,6.	Tempo	0-250	0-75, 75-90, 90-105, 105-120, 120-135, 135-150, 150-165, 165-250.
Release Year	1950-2019	1950-1969, 1970-1979, 1980-1989, 1990-1999, 2000-2009, 2010-2015, 2015-2017, 2018, 2019.	Valence	0-100	0-30, 30-45, 45-55, 55-70, 70-100.

*The interested reader is referred to Apel (2003) for pitch class definition. Only major key notes are shown, minor notes are defined analogously with lower case letters.

interactions, for example, whether the recommendation was skipped, or paused. In addition to user interactions, the dataset includes a set of track meta-data which are estimates of twenty one audio features for each track in the database, given in Table 4.3. The interested reader is referred to the Spotify website for the definition of these audio features.[4] We provide in Figure 4.4 the summary statistics of user interactions and distribution of example track audio features.

We consider finding frequent patterns of track recommendations based on track audio features. That is, we represent the recommended tracks by their audio feature in Table 4.3. For example, using the tempo audio feature, we represent recommended tracks by the value of their tempo as our sequential database. Next, for each derived database of audio feature values, we discretize the continuous values of audio features into a set of events $\mathbb{E}$. In particular, we represent similar audio feature values by the same event. For example, from a human perspective, a track with a tempo value of 45 is likely not distinguishable from track with a tempo value of 48. Such tracks should thus be considered as the same event.

To discretize an audio feature, we use any domain knowledge available from the Spotify website.[5] If no information is given, we use an equal-width discretization (Dougherty et al., 1995) with larger discretization intervals for the tails of the event distribution. The larger intervals are chosen to avoid events with little or no frequency, similar to the minimum frequency discretization approach (e.g., refer to Kotsiantis and Kanellopoulos (2006)). Figure 4.4 shows example thresholds used to discretize the continuous tempo and energy audio features. All discretization thresholds used in our experiments are given in Table 4.3.

An event e in our marketing application is thus the discretized value of audio features for a recommended track, and the associated outcome u is whether the track was skipped ($u = 1$) or not ($u = 0$). A regular tabular encoding of such a database is too large to fit in our system memory of 24 GB (ignoring the memory required for the pattern mining algorithm). On the other hand, a data

[4] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/
[5] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

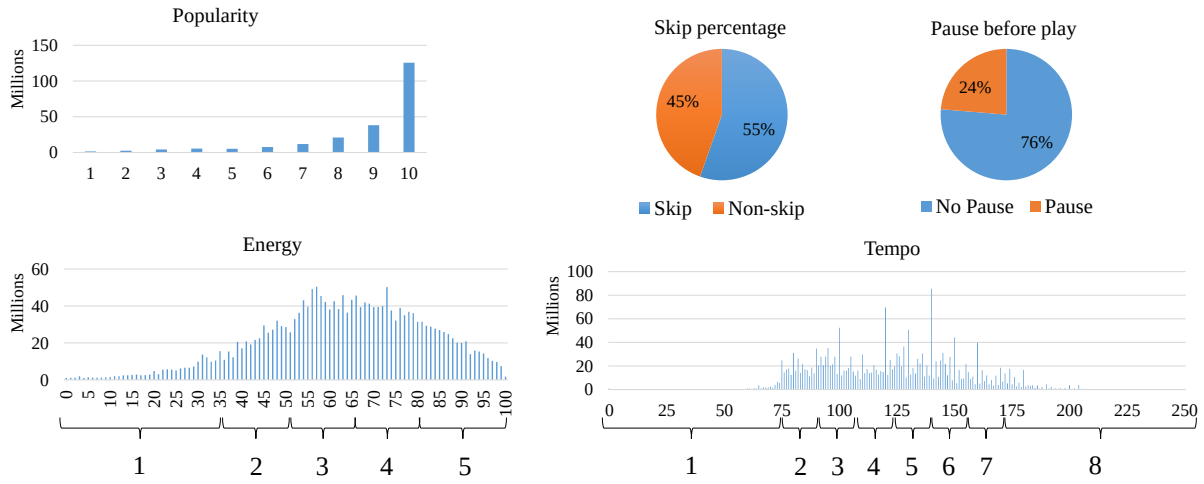


Figure 4.4: Example distribution of user interactions, audio feature distribution, and discretization thresholds.

tree model of the database fits the entire database in memory using as little as 4GB of memory. We note that, in general, memory requirements vary based on the number of discretized events $|\mathbb{E}|$ for different audio features.

For each constructed data tree, we perform pattern mining under a set of constraints as follows. First, we restrict the pattern mining algorithm to find patterns that consist of contiguous events in the data tree, with $|\mathcal{P}| \geq 2$. Furthermore, we impose a constraint on patterns $\mathcal{P} = \langle e_1, \ldots, e_{|\mathcal{P}|} \rangle$ to ensure events $e_1, \ldots, e_{|\mathcal{P}|-1}$ are all non-skipped. These constraints are imposed to capture the association of patterns $\mathcal{P}$ to user skip outcomes in the last events $e_{|\mathcal{P}|}$. The output of the mining algorithm is a set of patterns per audio feature, which we analyze in the following sections.

4.5.2 Sequential Association rules for User Skips

Given a data tree $\mathscr{D}$ (e.g., one constructed from a discretized audio feature representation of recommendations), we use our pattern mining algorithm to find all frequent patterns $\mathcal{P} : f(P) \geq \theta$. We set $\theta = 100$ as it is the minimum sample size required for a p-value hypothesis test with 95% confidence. For a frequent pattern $\mathcal{P}$, the skip likelihood probability of $\mathcal{P}$ is calculated using our p-value hypothesis tests from §4.4.1. Results show that $\mathbb{P}^{H_0}(\mathcal{P})$ is on average 2.7% lower than the ratio $\frac{\hat{f}(\mathcal{P})}{f(\mathcal{P})}$. Patterns that satisfy a minimum likelihood constraint $\mathbb{P}^{H_0}(\mathcal{P}) \geq \mathbb{P}(1) + \alpha$ are then used as interpretable explanations for user skips, as shown in Example 4.18.

Example 4.18. Table 4.4 gives a sample of patterns with high likelihood of user skips. Note that the average skip probability is $\mathbb{P}(1) = 55\%$ in the database, and all patterns in Table 4.4 satisfy $\mathbb{P}^{H_0}(\mathcal{P}) \geq \mathbb{P}(1) + 35\%$. In other words, the occurrence of these patterns increases user skip probability by at least 35%, and each individual probability is at least 90%. Patterns of Table 4.4 may be interpreted to provide insights into why users skip their recommended tracks. For example,

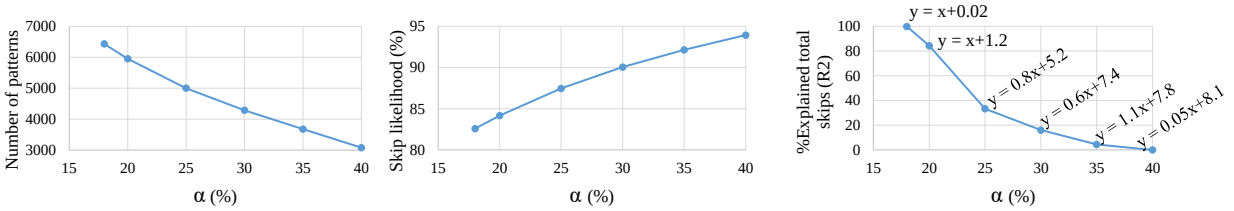
Table 4.4: Selection of significant patterns from audio features with high skip rates. The last event (or set of events) in patterns are frequently skipped by users.

Audio feature	First $\lvert\mathcal{P}\rvert - 1$ events of Pattern $\mathcal{P}$	Last event $e_{\lvert\mathcal{P}\rvert}$ (skipped)	$f(\mathcal{P})$	$\frac{\hat{f}(\mathcal{P})}{f(\mathcal{P})}$	$\mathbb{P}^{H_0}(\mathcal{P})$
Speechiness	$3 \to 2 \to 3 \to 1 \to 1 \to 2 \to 1$	$\to 1$	22,851	.98	.98
Dynamic range	$5 \to 5 \to 5 \to 3 \to 4 \to 3 \to 4$	$\to 3$	18,195	.98	.98
Accousticness	$2 \to 4 \to 3 \to 4 \to 1 \to 2 \to 2 \to 3$	$\to 3$	12,462	.98	.98
Energy	$2 \to 4 \to 2 \to 4 \to 5 \to 5 \to 3$	$\to 5$	10,129	.96	.96
Mechanism	$3 \to 3 \to 3 \to 3 \to 3 \to 3 \to 3$	$\to 5$	20,518	.96	.96
Key	C♯ (D♭) $\to$ a♯ (b♭) $\to$ B	$\to \{\text{F♯(G♭)}, \text{G}\}$	43,211	.95	.95
Time signature	$6 \to 6 \to 6 \to 6$	$\to 5$	21,705	.94	.94
Bounciness	$3 \to 3 \to 3 \to 3 \to 3 \to 3 \to 3$	$\to 5$	16,555	.94	.94
Danceability	$2 \to 2 \to 2 \to 2 \to 2 \to 2$	$\to 5$	16,784	.94	.94
Tempo	$8 \to 8 \to 8 \to 8 \to 8 \to 8 \to 8$	$\to \{1, \dots, 7\}$	33,089	.91	.91

if the user has listened to a series of high tempo tracks, recommending a track with even slightly lower tempo leads to a skip with 91% likelihood. Or, if the sequence of track energy follows a sinus patterns, it is 96% likely that she or he will skip the next recommendation. Overall, our observation is that the recommendation process should remain relatively stable with respect to audio features, and sudden changes generally lead to user skips. We emphasize that all patterns are interpretable, in that they provide an explanation for the user skip. Whether that explanation is valid and reliable beyond likelihood statistics depends on domain knowledge and judgment of the practitioner. □

We next analyze the trade-off between the imposed minimum likelihood threshold $\mathbb{P}(1) + \alpha$ and the percentage of user skips in the database that can be explained by feasible patterns. As previously discussed, this is done using regression hypothesis tests between vector Y and X. Here, Y denotes the total number of skips per user session, and X denotes the number of those skips that can be explained by a pattern. For our regression hypothesis test, we consider six different percentages of $\alpha = \{18, 20, 25, 30, 35, 40\}$ for the minimum likelihood threshold. For each value of α, we give the average number of feasible patterns over all audio features in Figure 4.5(a), and the average likelihood probability of the those patterns in Figure 4.5(b). The regression hypothesis results including the regression equation and goodness of fit R^2 are given in Figure 4.5(c).

All hypothesis tests give a near 100% confidence measure for a significant linear relationship between X and Y, which is expected given the high number of user sessions. Furthermore, the coefficient of determination R^2 shows that with a skip threshold of $\alpha = 18\%$, near 100% of variability in Y can be explained by variability in X. This means we are able to explain almost all one billion user skips in the database by approximately 6,400 patterns (number of patterns is averaged over all audio features), with an average 83% likelihood probability. Increasing the minimum likelihood threshold to $\mathbb{P}^{H_0}(\mathcal{P}) \geq \mathbb{P}(1) + 20\% = 75\%$ results in approximately 5,900 average number of patterns



a) Average number of patterns.

b) Average skip likelihood of patterns.

c) Regression equations and percentage of total skips explained (R^2) by patterns.

Figure 4.5: Music knowledge tree statistics and results of regression hypothesis tests.

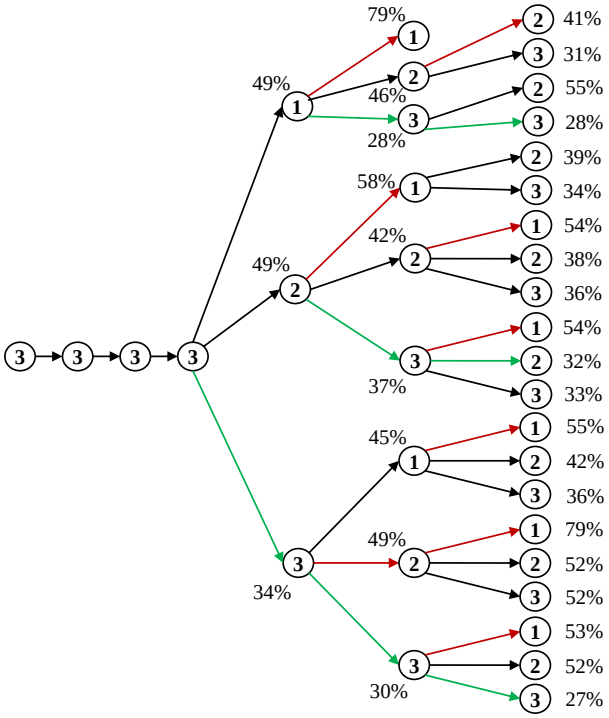
with an average of 84% likelihood probability, that are able to explain 84% of the variability in Y. Further increasing α gives 3,000 niche feasible patterns, with high 94% average likelihood, that are however not sufficient to explain the majority of user skips in the database.

4.5.3 Knowledge Tree for the Track Recommendation Process

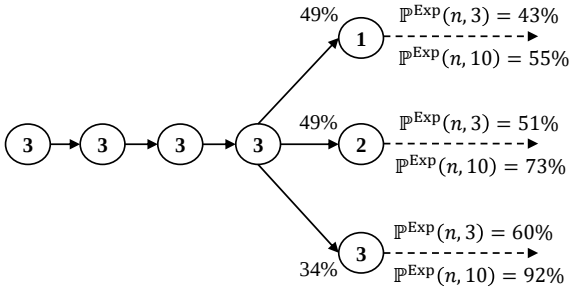
To develop a structural view of the track recommendation process, we aggregate mined patterns into a knowledge tree for each audio feature. Given a knowledge tree, we can now view the transition of skip likelihood probabilities $\mathbb{P}^{H_0}(\mathcal{P})$ based on the audio feature of recommendations. This can be used to support decision making, e.g., by displaying sequence of recommendation as guidelines to follow or avoid based on their associated skip likelihood. Example 4.19 gives an example of using knowledge trees as decision support tools in the recommendation process.

Example 4.19. Figure 4.6(a) displays a branch of the knowledge tree with three layers for the loudness audio feature. Here, we view the decision making task when the user has listened to four consecutive tracks with high loudness of 3. For illustration purposes, we color a number of recommendation paths with low skip likelihood in green, and with high skip likelihood in red. The path with the least skip likelihood is associated to recommendations (3,3,3) with skip likelihoods (34%,30%,27%), suggesting the recommender system should continue recommending tracks with high loudness. If a track with loudness of 2 is recommended, the next sequence of recommendations with minimum skip likelihood are (3,2) with (37%,33%) skip likelihood. Lastly, when lowering the loudness of the next immediate recommendation to 1, the knowledge tree still suggests to resume recommending tracks with high loudness of (3,3), giving a 28% likelihood of skip for both. On the other hand, in almost all cases, recommending a track with a low loudness of 1 has high skip likelihood, and should be avoided. □

As shown in Figure 4.6(a), the complexity of the knowledge tree grows with every layer, possibly displaying too much information to the decision maker and resulting in lower interpretability. Fortunately, paths of the knowledge tree can be used to generate more concise information for decision making, such as the expected skip probability of future recommendations. Example 4.20 gives an



a) Branch of the knowledge tree with three layers.



b) Expected skip probability within next 3 or 10 recommendations.

Figure 4.6: A branch of the recommendation knowledge tree for the loudness audio feature with three layers, and a concise representation of skip likelihoods using the expected skip probabilities.

example of such probabilities for the knowledge tree of Figure 4.6(a), and how it complements the information provided by the knowledge tree.

Example 4.20. Figure 4.6(b) gives the expected skip probability of recommendations for the knowledge tree of Figure 4.6(a). The expected skip probability is given for the next 3 or 10 recommendation. Results show that although the immediate skip likelihood is minimum for track recommendations with high loudness, the user eventually grows frustrated with loud recommendations. In particular, following a recommendation with loudness of 3, 60% of users skip any recommendation within the next three periods, with this growing to 92% within the next ten recommendations. In fact, in order to lower the expected skip likelihood for the next sequence of recommendations, the recommender system should recommend a track with lower loudness. Although such a recommendation has a higher likelihood of being skipped immediately, it avoids the long-term dissatisfaction of the user, and allows possible future recommendations with lower skip likelihoods. □

4.6 Application: Explaining Price Change Events in the Stock Market

For our next application, we consider sequential investment decision making in the stock market. Many factors influence a stock price and investment decisions, e.g., interest rates, dividends, economy, and political climate, to name a few. A compliment to using such factors in investments decisions are data-driven approaches, i.e., *technical analysis* of the market. Technical analysis is debated in the literature, with some believing that the market is perfect and thus impossible to predict future changes using historical data, and others believing that there is strong evidence for the contrary in practice. In this section, we take an unbiased stance, and investigate the effectiveness of technical analysis using historical patterns of stock price change events.

The most accurate price predictors for the stock market use deep neural networks (Kimoto et al., 1990; Pang et al., 2018; Selvin et al., 2017; Rather et al., 2015). However, the prediction process of such methods cannot be interpreted, and relying solely on a prediction value may be detrimental in the high risk environment of the market. In addition, the accuracy of such models decreases as the number of predicted future prices increase, lowering their effectiveness for long term investments.

On the other hand, a body of literature considers using interpretable and sequential price predictors and claims the existence of *stock chart patterns* (Bulkowski, 2011). Stock chart patterns are sequences of price changes that are used for stock price prediction, and are prevalent in technical analysis of the market. Although a number of papers study the detection of specific patterns in stock data and measure their confidence with positive results (e.g., refer to Leigh et al. (2007); Liu and Kwong (2007); Wu et al. (2014)), to the best of our knowledge, there is no explanation on how stock chart patterns were originally detected or how reliable they are in their prediction.

We analyze whether stock chart patterns exist, and if so, what is the likelihood that they

Table 4.5: Stock categories and number of their companies in the NASDAQ database.

Category	Number of stocks	Category	Number of stocks
Basic Industry	312	Capital goods	389
Consumer durables	134	Consumer non-durables	222
Consumer services	824	Energy	290
Finance	1,074	Healthcare	907
Technology	636	Transportation	117
Utilities	267	**Total**	**5,172**

correctly predict future prices. Unlike the literature, we do not assume the existence of any such patterns with a search and detect methodology, but rather rely on pattern mining and statistical significance to investigate their relevance to investment decision making. We find frequent patterns of price change events and analyze the likelihood they correctly predict future prices. Moreover, we design a knowledge tree of frequent patterns for a decision support tool in technical analysis. We begin by analyzing the sequential database, its data tree model, and tailored pattern mining algorithm.

4.6.1 Data Tree Model of Stock Price Changes and Pattern Mining Algorithm

For our database, we acquired the weekly closing price of a number of stocks, from January 1^{st} 2000 to January 1^{st} 2019. Stocks were derived from all available companies in the official NASDAQ website,[6] from 11 categories as shown in Table 4.5. The closing price of these companies were downloaded from Yahoo finance,[7] using an online bulk stock price download tool.[8]

In general, time series data are large, high dimensional, and noisy. Consequently, almost all pattern detection algorithms use a data transformation step to lower noise and memory requirements. Data transformation involves abstracting the database into a lower dimension by mitigating changes which are perceived to be noise. Many transformation methods are proposed in the literature, with the interested reader referred to Keogh et al. (2004) for a thorough review. Regardless of the method used, any type of transformation looses some aspects of the data due to abstraction, which may consequently remove valid patterns in data.

A common method of transformation for pattern mining in the stock market is linear interpolation between contiguous price changes (Linsay, 1991). In this approach, the difference between contiguous prices are calculated and discretized into symbolic events. A disadvantage of linear interpolation is its inability to distinguish noise in the data. For example, Table 4.6 shows two sequence

[6] https://www.nasdaq.com/screening/industries.aspx
[7] https://finance.yahoo.com
[8] http://finance.jasonstrimpel.com/bulk-stock-download/

Table 4.6: Sequence transformation by multilevel linear interpolation with maximum gap of one.

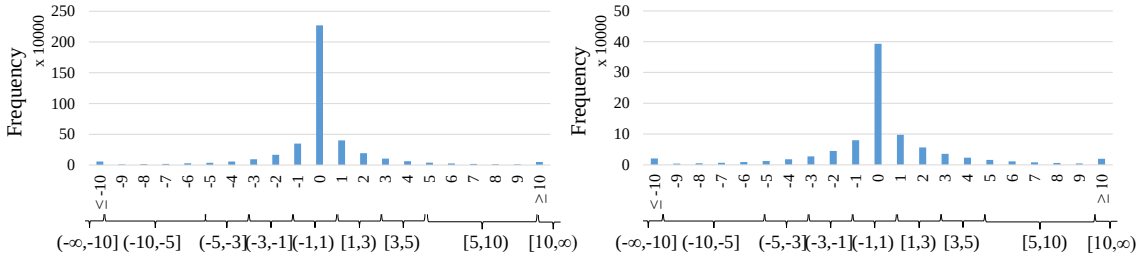
Original Sequence	Linear interpolation	Events used	Multilevel linear interpolation	Events used
$< 10, 10, 12, 13, 12 >$	$< 0, +2, +1, -1 >$	e_1, e_2, e_3, e_4, e_5	$< 0, +2, +1, -1 >$	e_1, e_2, e_3, e_4, e_5
			$< 0, +2, 0 >$	e_1, e_2, e_3, e_5
			$< 0, +3, -1 >$	e_1, e_2, e_4, e_5
			$< +2, +1, -1 >$	e_1, e_3, e_4, e_5
			$< +2, 0 >$	e_1, e_3, e_5
$< 10, 10, 12, 12 >$	$< 0, +2, 0 >$	e_1, e_2, e_3, e_4	$< 0, +2, 0 >$	e_1, e_2, e_3, e_4
			$< 0, +2 >$	e_1, e_2, e_3 or e_1, e_2, e_4
			$< +2, 0 >$	e_1, e_3, e_4

of stock prices, which are identical with the exception of an insertion of price 13 in the fourth position of the first sequence. This single insertion results in two different transformations by linear interpolation, and prevents the pattern mining algorithm from detecting their similar structure.

To detect common structure between noisy sequences of time series data and increase our chance of finding novel patterns, we propose a multilevel linear interpolation method. Multilevel linear interpolation involves allowing a gap between the interpolation of prices, and generates multiple transformations of a time series. For example, a maximum gap of one for the interpolation of sequences in the first column of Table 4.6, gives the transformations in the fourth column. Each transformation by multilevel interpolation is a possible abstraction by linear interpolation.

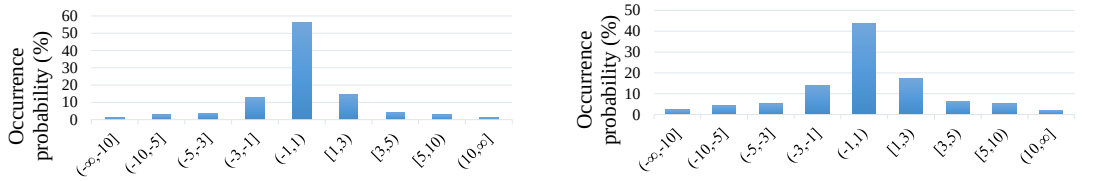
Using all such transformations, the pattern mining algorithm can detect similarities between sequences if any of their multilevel linear interpolation representations are identical. For example, two of the transformations $\langle 0, +2, 0\rangle, \langle +2, 0\rangle$ of the sequences in Table 4.6 are identical and used to detect their common pattern. Using multilevel linear interpolation, we can thus transfer the noise detection task to the pattern mining algorithm, where, prices that are not part of any frequent pattern are considered noise. For example, price 13 in the first sequence of Table 4.6 is considered noise, as it is not part of any common pattern with the second sequence. Although a multilevel linear interpolation considerably increases the already large database size of time series data, the larger memory requirement is mitigated by the compression provided by data trees.

As in §4.5, we next discretize price changes events using both domain knowledge and equal width distribution with larger tails. The distribution of price changes and the discretization thresholds are given in Figure 4.7. An event $e \in \mathbb{E}$ is thus a discretized price change in our finance application. The pattern mining task is to find frequent patterns of price change events that frequently lead to other price change events $e \in \mathbb{E}$. The outcome space in our finance application is thus the event space itself, i.e., $\mathbb{U} = \mathbb{E}$. Accordingly, sequential association rules are of the form $\langle e_1, \ldots, e_{|\mathcal{P}|-1}\rangle \rightarrow e_{|\mathcal{P}|}$, and $f_n = \hat{f}_n, \forall n \in \mathcal{N}$. Lastly, the ratio that a pattern $\langle e_1, \ldots, e_{|\mathcal{P}|}\rangle$ leads to its outcome $e_{|\mathcal{P}|}$ is defined by $f(\mathcal{P}) / \sum_{e' \in \mathbb{E}} f\left(\langle e_1, \ldots, e_{|\mathcal{P}|-1}, e'\rangle\right)$.



a) Weekly stock price changes.

b) Monthly stock price changes.



c) Distribution of weekly itemized stock price changes.

d) Distribution of monthly itemized stock price changes.

Figure 4.7: Distribution of stock price changes, and their discretization. Price changes are truncated to their integer part for clarity.

4.6.2 Sequential Association Rules for Stock Price Changes

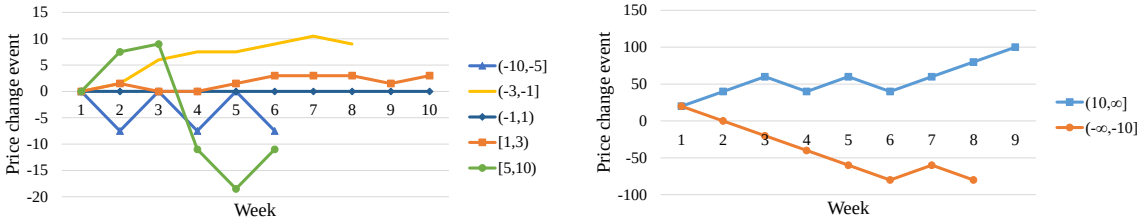
Given a data tree $\mathscr{D}$ of price change events, we use our pattern mining algorithm to find all weekly and monthly frequent patterns of price change event such that $\mathcal{P} : f(\mathcal{P}) \geq \theta$. As before, we set $\theta = 100$ as it is the minimum sample size required for a p-value hypothesis test with 95% confidence. In the multilevel linear interpolation procedure, we impose a maximum gap of 4 weeks. In other words, the maximum linear interpolation is between monthly stock prices. We restrict any pattern to a minimum span of 1 month and a maximum span of 12 months.

For a frequent pattern $\mathcal{P}$, the event likelihood probability of $\mathcal{P}$ is calculated using p-value hypothesis tests from §4.4.1. Results show that $\mathbb{P}^{H_0}(\mathcal{P})$ is on average 3% lower than ratios $f(\mathcal{P}) / \sum_{e' \in \mathbb{E}} f\left(\langle e_1, \ldots, e_{|\mathcal{P}|-1}, e' \rangle\right)$. Patterns that satisfy a minimum likelihood constraint $\mathbb{P}^{H_0}(\mathcal{P}) \geq \mathbb{P}(e) + \alpha$, are then used as interpretable explanations for event $e \in \mathbb{E}$, as shown in Example 4.21.

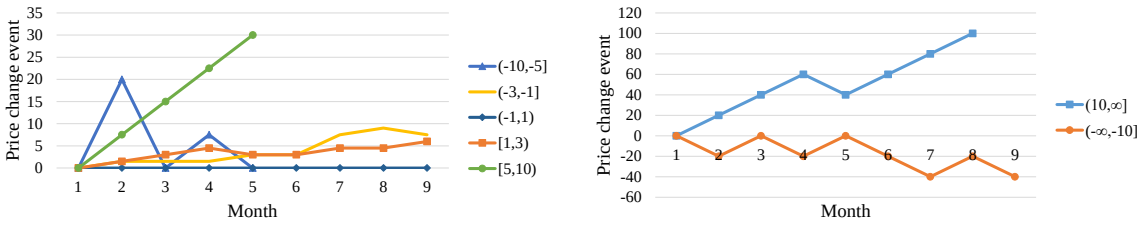
Example 4.21. Table 4.7 gives a sample of example patterns with high likelihood of leading to events $e \in \mathbb{E}$. Patterns are also depicted in Figure 4.8. The average probability of occurrence $\mathbb{P}(e)$ for all events is given in Figures 4.7(c) and 4.7(d), and all patterns in Table 4.7 satisfy at minimum $\mathbb{P}^{H_0}(\mathcal{P}) = \mathbb{P}(e) + 10\%$. No patterns are given for events $(-5, -3]$ and $[3, 5)$, as no frequent patterns exist (by our definition of patterns). Patterns of table 4.7 may be used as interpretable explanations on why an event $e \in \mathbb{E}$ occurred, namely, due the preceding sequence of price change events in each pattern. As shown in Table 4.7, the likelihood of these explanations are relatively low, except for event $(-1, 1)$. The overall trend of patterns shows that almost all price change events continue an increasing, decreasing, or a sinus trend. This trend is consistent for both monthly and weekly

Table 4.7: Selection of significant patterns for weekly and monthly stock price changes.

Type	First $\lvert\mathcal{P}\rvert - 1$ events of pattern $\mathcal{P}$	Last event $e_{\lvert\mathcal{P}\rvert}$	$f(\mathcal{P})$	$\frac{f(\mathcal{P})}{\sum_{e' \in \mathbb{E}} f(\langle e_1,\ldots,e_{\lvert\mathcal{P}\rvert-1},e'\rangle)}$	$\mathbb{P}^{H_0}(\mathcal{P} \to e_{\lvert\mathcal{P}\rvert})$
Weekly	$[10,\infty) \to (-\infty,-10] \to (-\infty,-10] \to (-\infty,-10] \to (-\infty,-10] \to (-\infty,-10] \to [10,\infty)$	$\to (-\infty,-10]$	2,699	0.54	0.52
	$(-10-5] \to [5,10) \to (-10-5] \to [5,10)$	$\to (-10,-5]$	577	0.15	0.13
	$[1,3) \to [3,5) \to [1,3) \to (-1,1) \to [1,3) \to [1,3)$	$\to (-3,-1]$	888	0.22	0.21
	$(-1,1) \to (-1,1) \to (-1,1) \to (-1,1) \to (-1,1) \to (-1,1) \to (-1,1) \to (-1,1)$	$\to (-1,1)$	2,411,382	0.91	0.9
	$[1,3) \to (-3,-1] \to (-1,1) \to [1,3) \to [1,3) \to (-1,1) \to (-1,1) \to (-3,-1]$	$\to [1,3)$	518	0.25	0.23
	$[5,10) \to [1,3) \to (-\infty,-10] \to (-10,-5]$	$\to [5,10)$	326	0.13	0.12
	$[10,\infty) \to [10,\infty) \to [10,\infty) \to (-\infty,-10] \to [10,\infty) \to (-\infty,-10] \to [10,\infty) \to 8$	$\to [10,\infty)$	642	0.46	0.43
Monthly	$(-\infty,-10] \to [10,\infty) \to (-\infty,-10] \to [10,\infty) \to (-\infty,-10] \to (-\infty,-10] \to [10,\infty)$	$\to (-\infty,-10]$	696	0.55	0.52
	$[10,\infty) \to (-\infty,-10] \to [5,10)$	$\to (-10,-5]$	308	0.15	0.14
	$[1,3) \to (-1,1) \to (-1,1) \to [1,3) \to (-1,1) \to [3,5) \to [1,3)$	$\to (-3,-1]$	245	0.24	0.21
	$(-1,1) \to (-1,1) \to (-1,1) \to (-1,1) \to (-1,1) \to (-1,1) \to (-1,1)$	$\to (-1,1)$	153,495	0.84	0.84
	$[1,3) \to [1,3) \to [1,3) \to (-3,-1] \to (-1,1) \to [1,3) \to (-1,1)$	$\to [1,3)$	378	0.26	0.24
	$[5,10) \to [5,10) \to [5,10)$	$\to [5,10)$	447	0.14	0.13
	$[10,\infty) \to [10,\infty) \to [10,\infty) \to (-\infty,-10] \to [10,\infty) \to [10,\infty)$	$\to [10,\infty)$	450	0.41	0.38

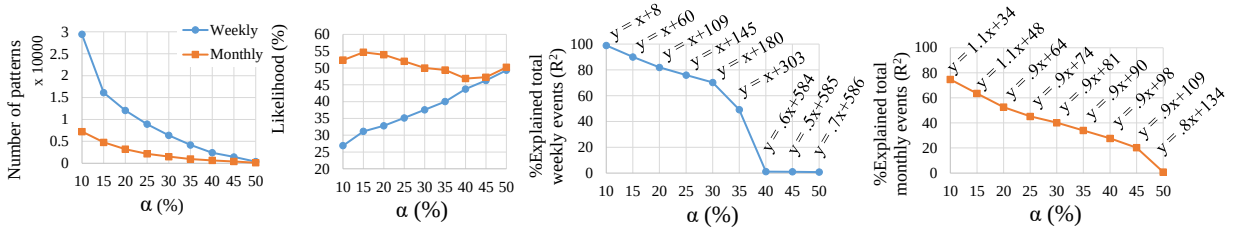


a) Weekly stock price change patterns.



b) Monthly stock price change patterns.

Figure 4.8: Graphical representation of significant patterns given in Table 4.7.



a) Average number of patterns.

b) Average likelihood of patterns.

c) Regression equations and percentage of weekly events (R^2) explained.

d) Regression equations and percentage of monthly events (R^2) explained.

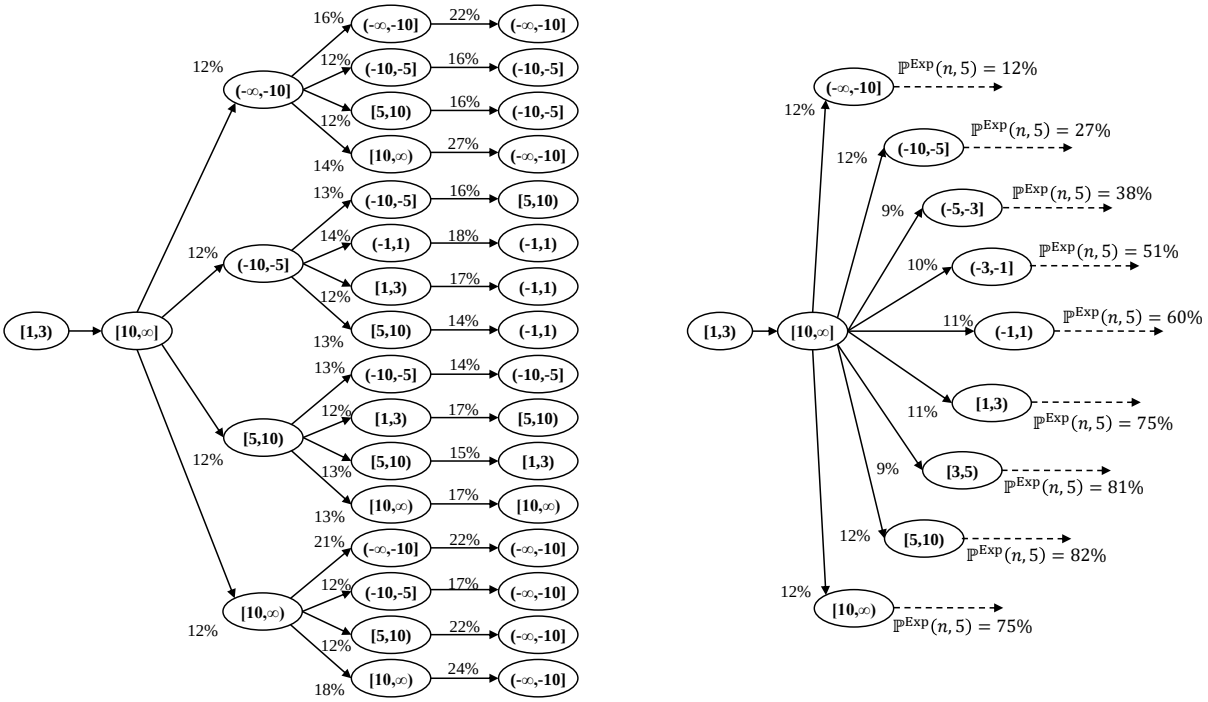
Figure 4.9: Stock knowledge tree statistics and results of regression hypothesis tests.

patterns. □

In the next step, we analyze the trade-off between imposed minimum likelihood thresholds $\mathbb{P}^{H_0}(\mathcal{P}) \geq \mathbb{P}(e) + \alpha$ and the percentage of stock price change events in the database that can be explained by feasible patterns using regression hypothesis tests. Here, Y denotes the total number of events in a sequence that may be explained using a pattern $\mathcal{P}$, i.e., Y is a vector with $|\mathcal{S}_i| - 1$ for all $\mathcal{S}_i \in \mathscr{S}$. This follows from constraint $|\mathcal{P}| \geq 2$, where no single event $e_1 \in \mathcal{S}$ can be explained by a pattern. As before, vector X contains the number of events that can be explained using any of our mined patterns satisfying the minimum likelihood threshold. We consider nine different values for α, ranging from $\alpha = 10\%$ to $\alpha = 50\%$ and increasing in increments of 5%. Figure 4.9(a) gives the number of feasible patterns, with the average likelihood of such patterns given in Figure 4.9(b). The regression hypothesis test results including the regression equation and goodness of fit R^2 are given in Figures 4.9(c) and 4.9.(d).

All hypothesis tests give a near 100% confidence rate for a significant linear relationship between X and Y, which is expected given the high number of considered stocks. Figures 4.9(a) to 4.9(c), show that close to 100% of the four million weekly price changes can be explained using approximately 30,000 patterns. However, the average occurrence probability of such patterns is considerably low at 27%. Increasing α to 35% gradually decreases the percentage of explained events to 50% using approximately 5,000 patterns with an average 40% likelihood. Further increasing α drops the percentage of explained events close to zero, as only hundreds of patterns satisfy the higher threshold. This shows that explaining all weekly stock prices using patterns remains difficult, although a number of patterns have high likelihood probabilities. Figures 4.9(a), 4.9(b), 4.9(d) show that long term monthly patterns are even less explained by mined patterns. For example, with $\alpha = 10\%$, approximately 7,000 patterns explain at most 80% of 900,000 monthly price changes, with a low average likelihood of 53%.

The results of our regression hypothesis tests show that only a few niche stock chart patterns have a high likelihood probability, and the average likelihood of anticipating price change events is generally low and at most 55%. Moreover, stock chart patterns are overall not sufficient to explain all price change events in the database. In other words, using a distinct number of patterns with



a) Branch of the knowledge tree with three layers.

b) Expected probability of price increase.

Figure 4.10: A branch of the stock price change knowledge tree with three layers, and a concise representation of price increase using the expected event probabilities.

high likelihood may be accurate for rare events, but not for all events.

4.6.3 Knowledge Tree for Stock Price Change Events

We aggregate patterns of price changes to construct a knowledge tree and develop a structural view of the stock price movements. Given a knowledge tree, we can now view the transition of price change likelihoods $\mathbb{P}^{H_0}(\mathcal{P})$. This can be used to support investment decision making, e.g., by displaying the most probable future sequence of price changes. Example 4.22 shows how a knowledge tree can be used to provide insights for investment decision making.

Example 4.22. Figure 4.10(a) displays a partial branch of the knowledge tree for monthly price movements. Here, we analyze whether the decision maker should make an investment given a sequence of observed events $\langle [1,3), [10,\infty) \rangle$. For succinctness, we only display the four most probable price changes in the first two future periods after $\langle [1,3), [10,\infty) \rangle$, and only the price change with maximum likelihood in the last layer. Following an observation of $\langle [1,3), [10,\infty) \rangle$, all future price change events are equally likely with a 12% likelihood. If a sharp decrease of $(-\infty, -10]$ occurs, the future outgoing paths from $(-\infty, -10]$ show that the price decrease is likely to continue, regardless of any intermediate price increase. If a price decrease of $(-10, -5]$ occurs, future paths show that price change most likely stabilizes to $(-1, 1)$ within the next two periods. The highest price increase

is most likely for a future sequence of $[10, \infty), [10, \infty), [10, \infty)$, while an initial increase of $[10, \infty)$ is most likely followed by a sharp decrease of $(-\infty, -10]$. □

The stock price knowledge tree can be used to provide more concise and interpretable information to the decision maker, e.g., by generating expected probabilities $\mathbb{P}^{\mathbb{E}xp}(\mathcal{P})$. Here, we analyze the expected probability that a stock price increases above its price at time of decision making. Note that this is different to observing a future price increase event e, as, e.g., the overall price change may still be negative despite a price increase event. Example 4.23 shows how expected likelihood probabilities are used to complement the insights of the knowledge tree in Figure 4.10(a).

Example 4.23. Figure 4.10(b) gives the expected likelihood probability of price increase within the next five periods, after the events following $\langle [1, 3), [10, \infty) \rangle$ in Figure 4.10(a). The expected probabilities show that unless a decrease between $(-\infty, -3]$ is observed, it is likely that the overall trend of price change is positive in the next five periods. In particular, the expected probabilities show that an investment may eventually payoff in the next five months, despite early intermediate price decrease events. □

4.7 Conclusion

In this paper, we developed a data-driven framework for sequential decision making with an emphasize on interpretability and sequential structure. We designed a novel data tree model of the database for pattern mining algorithms, that is able to fit in memory orders of magnitude larger datasets compared to traditional tabular encodings. We tailored the prefix-projection algorithm to mine patterns from data trees, and showed its efficiency compared to traditional algorithms. Using the capability of our pattern mining algorithm, we mined novel sequential patterns from data that are otherwise not possible using a tabular encoding of the database. We showed how sequential patterns can be used to develop interpretable explanations for events and outcomes in the database. We generated statistically guaranteed likelihood probabilities for such explanations, and analyzed the trade-off between minimum likelihood constraints and overall explainability of patterns. Lastly, we showed how patterns can be aggregated into a knowledge tree to provide an interpretable data-driven decision support tool for sequential decision making.

Using our methodology, we studied two applications in marketing and finance. In marketing, we considered explaining user skips in online music streaming platforms. We generated interpretable explanations on why users may skip their recommended tracks, and showed how the knowledge tree can be used to provide guidelines to avoid recommendations with high-skip likelihood. By using an average 6,400 sequential patterns, we can explain all one billion user skips in the database with a high average likelihood of 83%. In our finance application, we assessed whether sequential patterns of price change can be used to aid technical analysis for investment decision making in the stock market. Our results show that although we can find a number of explanations with high likelihood of

leading to specific price changes, most price change events cannot be reliably explained by patterns. In particular, at best 80% of nine hundred thousand monthly price changes can be explained using 7,000 patterns with a lower average likelihood of 53%.

Chapter 5

Provider Network Selection and Transition under Competition

5.1 Introduction

The United States healthcare market is the largest in the world with $3.6 trillion expenditure in 2018. The United States healthcare costs continue to grow with an estimated annual rate of 5.4% for 2019-28, to reach $6.2 trillion by 2028.[1] Adhering to this changing environment, health insurance firms in the united states annually revisits the design of their healthcare plans. This includes the evaluation of contracts made with healthcare providers such as hospitals, physicians, retail clinics, and pharmacies; and the premiums charged to patients for insurance plans.

An insurance plan is generally characterized by its set of *in-network* providers and the restrictions on receiving care from outside providers. For example, a Health Maintenance Organization (HMO) plan restricts patients to only visit in-network providers, with the exception of an emergency. Patients enrolled in an HMO plan who visit an outside provider must pay the entire healthcare cost out of pocket. Similarly, a Point of Service (POS) plan requires patients to visit in-network providers, but also provides the option to visit an outside provider if referred to by the patient's Primary Care Physician (PCP). The least restrictive plans are perhaps Preferred Provider Organizations (PPO), where patients benefit from lower rates if they receive care from in-network providers, and are charged higher out of pocket costs for visiting outside providers.

Insurance firms are thus tasked with strategically composing different insurance plans for targeted patients, including their set of in-network providers and their type of insurance. Insurance plans are then offered to patients, who evaluate and choose them by an unobserved internal mechanism. This involves patients negotiating the insurance premium paid to the insurer for their desired plan, generally through their employer. After enrolling in a plan, patients who require medical care may visit one of their in-network providers at zero or little cost.

[1] https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/

Under competition, insurance firms compete over profitable patients, while trying to guide costly patients to their rival. This strategy is known as cream skimming (Barros, 2003) and controlled by the design and plan premium for different individuals. For example, by excluding the desirable providers for costly patients, or increasing their premium, insurers hope to evade such patients. On the other hand, designing poor provider networks leads to the loss of profitable patients to rival firms and decrease profits.

Furthermore, healthcare providers are generally affiliated with different insurance firms, and accordingly, charge a higher contract fee to rival firms. This brings a different layer of decision making into the network provider selection problem, where the insurer seeks to mainly contract its affiliated providers while avoiding those of its competitor. The challenge is that a provider affiliated with a rival firm may be popular among patients, and excluding it from the provider network may prove to be detrimental.

In this chapter, we focus on the above challenges faced by health insurance companies in designing optimal HMO plans. Unlike the literature which evaluates and selects providers on an individual basis, we take an optimization approach and optimize the provider network as a whole. This allows us to capture the relationships between providers, together with the joint preference of patients in provider selection. Moreover, instead of generating one insurance plan for the entire set of patients, our optimization framework designs and generates several insurance plans. Each plan targets a specific set of customers with the objective of maximizing the insurer's profit while maintaining patient satisfaction.

An instrumental factor in provider network selection is predicting patient choice. Our emphasis is on developing an accurate yet interpretable methodology for patient behavior, that can be exploited in the network provider selection model. Interpretability is important in our approach as prediction is performed as an intermediate step to better design the provider network. It is thus required that the results of prediction can be analyzed to accommodate the provider selection model. On the other hand, the prediction methodology should have high accuracy, as inaccurate predictions may result in a sub-optimal provider network where patients differ from predictions and enroll in unintended plans. Indeed, we analyze and show the effects of such inaccurate predictions in our numerical results.

Our optimization model for provider network selection is of large size, and hard to solve in practice. We thus develop a simultaneous multi-column-and-row generation approach to effectively solve our model. More importantly, a simultaneous multi-column-and-row scheme allows us to model the transition of the provider network and insurance plans from the current settings of the insurer. We provide interpretable insights into this transition, and discuss how they can be user for data-driven decision making.

Finally, we test our solution procedure and compare the results in two aspects. First, we compare the possible gains in an optimization approach towards provider network selection compared to an individual and heuristic evaluation of providers. Second, we analyze the consequences of inaccurate

prediction in patient behavior and discuss its the disadvantages.

The rest of this chapter is organized as follows. In §5.2 we review the relevant literature on provider network selection and patient choice prediction. In 5.3 we layout our approach towards provider network selection, and develop its multi-column-and-row generation solution algorithm in §5.4. Numerical results are given in §5.5 and the chapter is concluded in §5.6.

5.2 Literature Review

The literature prominently addresses the provider selection problem using discrete choice models (Robinson and Gardner, 1995; Shepard, 2016; Raval et al., 2016). Discrete choice models assess providers based on factors such as service type, cost, quality, overall patient demand, or bargaining and contract price. Providers are evaluated individually, and either pass or fail the evaluation. The provider network of a discrete choice model approach is thus the union of all providers that pass the evaluation.

Provider network selection may also be implicitly considered in insurance plan design. For example, Ho and Lee (2017) consider insurance design as a game between the insurer and patients, where providers are strategically chosen to attract or exclude certain patient segments (McGuire et al., 2014). Bargaining games are studied between the insurer and provider by exploiting the possibility of substituting providers and the strategies it entails in the bargain (Ho and Lee, 2019). Provider network selection is also studied using demand and price equilibrium, and the shift of demand/price given a change in the provider network (Geruso and Layton, 2017; Ho and Lee, 2019). In this perspective, a provider is analyzed by the change in demand and price of the insurance plan when it is selected to be in-network.

An important aspect of provider network selection is predicting patient behavior given a change in the network. The literature analyzes patients based on factors such as their distance to different providers, their healthcare expectations and loyalty to their current provider (McGuire et al., 2014; Raval et al., 2016; Raval and Rosenbaum, 2018). These factors contribute to an overall “switching cost" for patients that is used to determine the likelihood a patient would switch their current provider given a change in the insurance plan. In that regard, Shepard (2016); Crespin (2016) show that patients are generally more influenced by price of insurance and not quality of service or provider reputation. In other words, patients generally have low switching costs between providers if the price is right. Similarly, Irace (2018) shows that patients generally choose to receive service from their most recently visited provider, and not their most long-standing provider. Such results show that it may beneficial to design optimal provider networks with higher profit and guide patients to switch their providers to the newly designed plans while maintaining their satisfaction. For example, Shepard (2016) show that it is profitable for insurers to design limited networks that exclude costly “star providers", and guide patients to choose lower cost healthcare provider in-network of the newly designed plans.

5.3 Provider Network Selection under Competition

Let $\mathbb{I}$ be a finite set of patients seeking to purchase an HMO health insurance plan, and $\mathbb{J}$ be a finite set of healthcare providers. The provider network selection problem involves selecting a subset of providers $j \in \mathbb{J}$ to contract in order to compose insurance plans. An insurance plan $\mathbf{P}$, is characterized by its in-network providers that are a subset of contracted providers, and the premiums r_{ip} charged to patients $i \in \mathbb{I}$. The insurer may compose many such insurance plans in order to target different patient segments and maximize its profit.

Let $\mathbb{P} = \{\mathbf{P}_1, \ldots, \mathbf{P}_{|\mathbb{P}|}\}$ be the set of all potential plans composed by the insurer, indexed by p. The insurer is next tasked by choosing which of the insurance plans $p \in \mathbb{P}$ to offer to patients $i \in \mathbb{I}$. If the insurer decides to offer a plan $p \in \mathbb{P}$ to patients, it incurs the fixed cost $\sum_{j \in \mathbf{P}_p} f_j$ of contracting its in-network providers $j \in \mathbf{P}_p$. Thereafter, if a patient $i \in \mathbb{I}$ chooses to enroll in an offered plan $p \in \mathbb{P}$, the insurer receives an insurance premium r_{ip}, and additionally incurs an expected cost of service c_{ip} for the enrolled patient i.

Under competition, patients $i \in \mathbb{I}$ have available an additional set of insurance plans $\mathbb{P}^c$ offered by the insurer's competitors. Patients thus evaluate and select the best plan on offer from the set $\mathbb{P} \cup \mathbb{P}^c$, according to their unobserved internal decision making mechanism. We follow the literature in using a utility-based approach to predict patient behavior (see, e.g., Capps et al. (2003); Ho and Lee (2017, 2019); Geruso and Layton (2017); Raval and Rosenbaum (2018)). In a utility-based approach, we measure the value of a provider $j \in \mathbb{J}$ for a patient $i \in \mathbb{I}$ by a value u_{ij}. Similarly, the value of a plan $p \in \mathbb{P} \cup \mathbb{P}^c$ for a patient $i \in \mathbb{I}$ is measured by the utility $\mathcal{U}_{ip}$. Higher utility values of utility $\mathcal{U}_{ip}$ indicate a higher worth of plan p for patient i. In particular, patient i selects the highest value plan $p^* : \mathcal{U}_{ip^*} = \max_{p \in \mathbb{P} \cup \mathbb{P}^c} \{\mathcal{U}_{ip}\}$. From the selected plan p^*, patient i selects the highest utility provider $j^* : u_{ij^*} = \max_{j \in \mathbf{P}_{p^*}} \{u_{ij}\}$. We detail the procedure to estimate utility values in Appendix 5.A, and use their estimated values in our provider selection optimization model.

5.3.1 Provider Network Selection Model

In this section, We develop an optimization model that takes as input the current provider network of the insurer, and transitions it into an optimized network. The provider network selection problem may be naturally considered as a bilevel optimization program. In a bi-level setting, the insurer acts as the leader, and selects the providers $j \in \mathbb{J}$ to contract, and plans $p \in \mathbb{P}$ to offer to patients. Patients act as the follower, and select among the plans offered by the insurer (and its competitor) to enroll in.

We recall that if a patient $i \in \mathbb{I}$ selects a plan $p \in \mathbb{P}$, the insurer receives a premium r_{ip}, and incurs an expected cost of healthcare c_{ip}. This cost is calculated based on the provider $j^* \in p$ selected by patient i, if patient i selects to enroll in plan p. Specifically, by the utility theory, patient i chooses to receive care from the provider $j^* \in \mathbf{P}_p$ that has the maximum utility $u_{ij^*} = \max_{j \in p} \{u_{ij}\}$.

The expected cost of service for patient $i \in \mathbb{I}$ that selects plan $p \in \mathbb{P}$ is thus $c_{ip} = c_{ij^*}$. Cost c_{ij^*} may be estimated using any method, such as the one detailed in Appendix 5.C. $= \mathbf{C}_i \times \mathbf{I}_i \times \mathbf{J}_{j^*}^T$, where $j^* : u_{ij^*} = \max_{j \in \mathbf{P}_p} \{u_j\}$.

We can now define our bilevel optimization model. Let x_j be a binary variable that denotes whether providers $j \in \mathbb{J}$ is contracted by the insurer. Further let z_p be a binary variable that denotes whether the insurer offers plan p to patients, and binary variable y_{ip} denote whether patient i selects to enroll in plan $p \in \mathbb{P} \cup \mathbb{P}^c$. The bilevel model **BL** is formulated as:

$$\textbf{BL}: \quad \max \quad \sum_{i \in \mathbb{I}} \sum_{p \in \mathbb{P}} (r_{ip} - \mathcal{C}_{ip})\, y_{ip} - \sum_{j \in \mathbb{J}} f_j x_j \tag{5.1}$$

$$\text{s.t.} \quad z_p \leq x_j \qquad \forall j \in \mathbb{J}, \forall p \in \mathbb{P} : j \in \mathbf{P}_p, \tag{5.2}$$

$$\sum_{p \in \mathbb{P}} z_p \geq 1 \tag{5.3}$$

$$x_j \in \{0,1\} \qquad \forall j \in \mathbb{J}, \tag{5.4}$$

$$z_p \in \{0,1\} \qquad \forall p \in \mathbb{P}, \tag{5.5}$$

$$\text{where} \quad y_{ip}, \forall i \in \mathbb{I}, \forall p \in \mathbb{P} \cup \mathbb{P}^c \quad \text{solve:}$$

$$\max \quad \sum_{i \in \mathbb{I}} \sum_{p \in \mathbb{P} \cup \mathbb{P}^c} \mathcal{U}_{ip} y_{ip} \tag{5.6}$$

$$\text{s.t.} \quad y_{ip} \leq z_p \qquad \forall i \in \mathbb{I}, \forall p \in \mathbb{P}, \tag{5.7}$$

$$\sum_{p \in \mathbb{P} \cup \mathbb{P}^c} y_{ip} = 1 \qquad \forall i \in \mathbb{I}, \tag{5.8}$$

$$y_{ip} \in \{0,1\} \qquad \forall i \in \mathbb{I}, \forall p \in \mathbb{P} \cup \mathbb{P}^c. \tag{5.9}$$

The objective function (5.1) maximizes the profit of the insurer, composed of the revenue generated by insurance premiums rip minus the expected cost of patient healthcare c_{ip} (calculated in Appendix 5.A), and a fixed annual cost of contracting providers f_j. Constraints (5.2) restrict the insurer from offering a plan $\mathbf{P}$ that includes a provider $j \in \mathbb{J}$ not contracted by the insurer. Constraint (5.3) enforces the insurer to offer at least one plan $p \in \mathbb{P}$ to patients. This constraint is imposed to analyze the quality of plans in our numerical tests, when no profitable plan is available for the insurer. The second level objective function (5.6) models patient choice, and maximizes the utility of patients $i \in \mathbb{I}$ in selecting a plan $p \in \mathbb{P} \cup \mathbb{P}^c$. Constraints (5.7) permit patients to choose a plan $p \in \mathbb{P}$, only if plan p is offered by the insurer. Note that constraints (5.7) do not impose any restriction on patient choice for plans $p \in \mathbb{P}^c$. Constraints (5.8) ensure that all patients choose exactly one insurance plan, and constraints (5.4), (5.5) and (5.9) are binary restrictions on the decision variables.

The bilevel program may be transformed into a single level program by exploiting the fact that patients choose the offered insurance plan that maximizes their utility. Specifically, we may

reformulate the bilevel program **BL** into a single level model **SL** using constraints (5.10)-(5.12):

$$
\begin{aligned}
\textbf{SL}: \quad \max \quad & (5.1) \\
\text{s.t.} \quad & (5.2)-(5.5),(5.7)-(5.9), \\
& \sum_{\substack{p\in\mathbb{P}\cup\mathbb{P}^c: \\ \mathcal{U}_{ip}<0}} y_{ip'} \leq 0 && \forall i \in \mathbb{I}, && (5.10) \\
& \sum_{\substack{p'\in\mathbb{P}\cup\mathbb{P}^c: \\ \mathcal{U}_{ip'}<\mathcal{U}_{ip}}} y_{ip'} \leq 1 - z_p && \forall i \in \mathbb{I}, \forall p \in \mathbb{P}, && (5.11) \\
& \sum_{\substack{p'\in\mathbb{P}\cup\mathbb{P}^c: \\ \mathcal{U}_{ip'}<\mathcal{U}_{ip}}} y_{ip'} \leq 0 && \forall i \in \mathbb{I}, \forall p \in \mathbb{P}^c. && (5.12)
\end{aligned}
$$

Constraint (5.10) restricts the selection of any plan with negative utility, and constraints (5.11) ensure that patients do not choose a plan with lower utility than any plan offered by the insurer. Similarly, constraints (5.12) ensure that patients do not select a plan with lower utility than any competitor plan $p \in \mathbb{P}^c$.

If set $\mathbb{P}$ contains every possible plan $\mathbf{P} \subseteq \mathbb{J}$, i.e., every $2^{|\mathbb{J}|}$ subset of providers, the solution of model **SL** determines the best plan the insurer should offer to patients in order to maximize profits. Such a solution approach is however inefficient, and impractical in practice. We thus use a simultaneous multi-column-and-row generation algorithm to solve **SL**.

5.4 A Simultaneous Multi-Column-and-Row Generation Algorithm for Network Provider Selection

Our algorithm to converge to the optimal solution of **SL** involves solving the linear relaxation of **SL**, under a restricted number of plans $p \in \mathbb{P}$. The solution of this *restricted mater problem* **RMP** is then used to determine whether the addition of new plan **P** improves the objective of **RMP**, or prove that no other improving plan exists. A possible plan that improves the solution of **RMP** is found by solving a subproblem **SP** defined based on the dual information of a **RMP** solution, and the structure of a feasible plan **P**. The algorithm recursively solves **RMP** and **SP** until the plan generated by **SP** does not improve the objective of **RMP**. This indicates that no other improving plan exists, providing the optimality condition of the **RMP** solution.

Let $\bar{\mathbb{P}} \subset \mathbb{P}$ be a restricted plan set populated by an initial number of plans, e.g., the ones currently offered by the insurer. We define the restricted master problem **RMP**, as the linear-relaxation of model **SL**:

$$
\textbf{RMP}: \quad \max \quad (5.1)
$$

$$\text{s.t.} \quad (5.2), (5.3), (5.7), (5.8), (5.10) - (5.12),$$

$$0 \leq x_j \leq 1 \qquad \forall j \in \mathbb{J}, \tag{5.13}$$

$$0 \leq z_p \leq 1 \qquad \forall p \in \bar{\mathbb{P}}, \tag{5.14}$$

$$0 \leq y_{ip} \leq 1 \qquad \forall i \in \mathbb{I}, \forall p \in \bar{\mathbb{P}}. \tag{5.15}$$

RMP is a linear program, and its optimal solution may turn out to be fractional. Although this is common in column generation algorithms, we prove that under a *price-out* strategy, the solution of **RMP** is integral or can be made integral by inspection. As our first step, we define the notion of patient *targeting* by a plan $\mathbf{P}$.

Definition 5.1. A plan $\mathbf{P}$ targets a patient $i \in \mathbb{I}$ if

1. $\mathcal{U}_{i\mathbf{P}} > 0$, and
2. $\mathcal{U}_{i\mathbf{P}} > \mathcal{U}_{ip}, \forall p \in \mathbb{P}^c$.

The set of all such patients define the set of targeted patients $\bar{\mathbb{I}}_{\mathbf{P}}$ of plan $\mathbf{P}$. □

We next define the price-out strategy as

Definition 5.2. A price-out strategy of plan $\mathbf{P}$ involves:

1. increasing the premiums $r_{i\mathbf{P}}$ for all $i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}$, until $\mathcal{U}_{i\mathbf{P}}$ becomes negative.
2. increasing the premiums r_{ip} for all $i \in \bar{\mathbb{I}}_{\mathbf{P}}, \forall p \in \mathbb{P} : \mathbf{P}_p \neq \mathbf{P}$, until $\mathcal{U}_{ip}$ becomes negative. □

A price-out strategy ensures that at most one plan $p \in \mathbb{P}$ targets a patient $i \in \mathbb{I}$. This is not a restricting assumption, as proved in Proposition 5.3.

Proposition 5.3. *The optimal plan premiums and access to in-network providers determined by model **SL** remains unchanged for all patients $i \in \mathbb{I}$ under a price-out strategy.*

Proof. Let z_{p^*} be the optimal plan offered to a patient $i \in \mathbb{I}$. An optimal solution of **SL** thus satisfies $z_p = 1, y_{ip^*} = 1$, and by constraint (5.8) we have $y_{ip} = 0, \forall p \in \mathbb{P} : p \neq p^*$. Under a price-out strategy, we increase premiums $r_{ip} : \forall p \in \mathbb{P} : p \neq p^*$ to enforce $\mathcal{U}_{ip} < 0$ and consequently $y_{ip} = 0, \forall p \in \mathbb{P} : p \neq p^*$ by constraints (5.10). A price-out strategy thus has no effect on the optimal plan offered to patients, and the optimal premium and in-network access for patients remains unchanged. □

We can now show that under a price-out strategy the optimal solution of **RMP** is integral if it has a positive objective value, or otherwise, can be made integral by inspection. This is proved in Theorem 5.4.

Theorem 5.4. *Under a price-out strategy, either an optimal solution* $(\bar{y}_{ip}, \bar{z}_p, \bar{x}_j)$ *of* ***RMP*** *is integral, or can be made integral by inspection.*

Proof. We first prove that if an optimal solution $(\bar{y}_{ip}, \bar{z}_p, \bar{x}_j)$ of **RMP** has positive objective value, it is integral. Due to a price-out strategy, we can only have $\bar{y}_{ip} > 0$ for a single plan $p \in \bar{\mathbb{P}}$. Moreover, as $\bar{y}_{ip} \leq \bar{z}_p \leq \bar{x}_j, \forall i \in \mathbb{I}, \forall p \in \bar{\mathbb{P}}, \forall j \in \mathbb{J} : j \in \mathbf{P}_p$ imposed by constraints (5.7) and (5.2), and due to optimality, we must have $\bar{y}_{ip} = \bar{z}_p = \bar{x}_j, \forall i \in \mathbb{I}, \forall p \in \bar{\mathbb{P}}, \forall j \in \mathbb{J} : j \in \mathbf{P}_p$. Otherwise, we can increase the values of $\bar{y}_{ip}$ and $\bar{z}_p$ up to $\bar{x}_j$ and obtain a higher objective value in (5.1). Now, as solution $(\bar{y}_{ip}, \bar{z}_p, \bar{x}_j)$ has positive objective value, we can conclude that $\sum_{i \in \mathbb{I}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - c_{ip}) > \sum_{j \in \mathbb{J}} f_j$. Therefore, a marginal increase in variables y_{ip}, z_p such that $\bar{y}_{ip} > 0, \bar{z}_p > 0, \forall i \in \mathbb{I}, \forall p \in \bar{\mathbb{P}}$ increases the objective value in (5.1). An optimal solution must thus satisfy $\bar{y}_{ip} = 1$ for all $\bar{y}_{ip} > 0$. Consequently, $\bar{z}_p = 1$ and $\bar{x}_j = 1$ for all $\bar{z}_p > 0$ and $\bar{x}_j > 0$.

We now prove that if an optimal solution $(\bar{y}_{ip}, \bar{z}_p, \bar{x}_j)$ of **RMP** has negative objective value, it can be made integral by inspection. As shown in the proof of Theorem 5.5, we have $\sum_{p \in \bar{\mathbb{P}}} z_p = 1$ in the optimal solution of **RMP**. To obtain an integral solution, we can thus inspect each $p \in \bar{\mathbb{P}}$ by its profit $\sum_{i \in \bar{\mathbb{I}}_p} (r_{ip} - c_{ip}) - \sum_{j \in p}$, and select the plan $\mathbf{P}^*$ with the maximum (negative) profit. By setting $y_{ip} = 1 \forall i \in \bar{\mathbb{I}}_{p^*}, z_{p^*} = 1$ and $x_j = 1, \forall j \in \mathbf{P}_{p^*}$, we obtain an integral optimal solution of **RMP**. □

Our next task is to use the dual information from **RMP** to possibly generate a plan that improves its solution. This is done using pricing subproblems **SP**.

5.4.1 Pricing Subproblems

A plan $\mathbf{P}$ in our setting corresponds to a tuple of variables $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ in **RMP**. This is unlike column generation algorithms, where the addition of a new column involves adding only a single variable. In particular, in the formulation of **RMP**, variables y_{ip} and z_p are dependent on one another and the addition of one variable to **RMP** is meaningless without the other. Moreover, the addition of a variable y_{ip} to **RMP** also entails adding new constraints (5.7) to **RMP**. Similarly, the addition of a new variable z_p to **RMP** involves adding new constraints (5.2), (5.7), and (5.11) to **RMP**. This means the reduced cost of a new variable y_{ip} or z_p not only depends of the value of dual variables generated by the current formulation of **RMP**, but also depends on the value of dual variables associated to the newly added constraints. Converging to an optimal solution of **SL** using the relaxation **RMP** thus require a simultaneous multi-column-and-row algorithm, where multiple columns $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$, and multiple corresponding rows (5.2), (5.7), and (5.11) are simultaneously added to **RMP**. The challenge, however, is to determine the reduced cost of adding a new plan to **RMP** using only the information of the dual variables from its current solution.

Our price-out strategy can be further used to simplify the procedure of calculating the reduced cost of a new plan $\mathbf{P}$. This procedure involves determining its objective value (5.1) of **RMP** after

the addition of plan $\mathbf{P}$, and comparing it to its current value. If the objective value increases with the addition of plan $\mathbf{P}$, then plan $\mathbf{P}$ has a positive reduced cost; otherwise, it has a reduced cost of zero. Theorem 5.5 employs this procedure and shows how to determine whether a plan $\mathbf{P}$ has positive reduced cost.

Theorem 5.5. *Under a price-out strategy and given a solution* $(\bar{x}_j, \bar{z}_p, \bar{y}_{ip})$ *of* ***RMP****, a tuple* $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ *has positive reduced cost if and only if*

$$\sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} (r_{i\mathbf{P}} - \mathcal{C}_{i\mathbf{P}}) - \sum_{j \in \mathbf{P}} f_j(1 - \bar{x}_j) - \sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} + \sum_{j \in \mathbb{J}} f_j \bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} > 0.$$

Proof. See Appendix 5.B. □

Here, $\mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}}$ denotes whether the addition of plan $\mathbf{P}$ to **RMP** leads to the value of variable x_j that has a positive value $\bar{x}_j > 0$ in the current solution of **RMP**, to become zero. Similarly, $\mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}}$ denotes the same for variable z_p. The outcome of both scenarios can be determined under a price-out strategy, as shown in the proof of Theorem 5.5. However, determining the outcome of these scenarios is computationally costly. For a more effective solution algorithm, we use corollary 5.6 which gives a sufficient condition for a positive reduced cost plan $\mathbf{P}$.

Corollary 5.6. *Under a price-out strategy and given a solution* $(\bar{x}_j, \bar{z}_p, \bar{y}_{ip})$ *of* ***RMP****, a tuple* $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ *has positive reduced cost if*

$$\sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} (r_{i\mathbf{P}} - \mathcal{C}_{i\mathbf{P}}) - \sum_{j \in \mathbf{P}} f_j(1 - \bar{x}_j) - \sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} > 0.$$

Proof. As discussed in the proof of Theorem 5.5, we have $\mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} = 1$ or $\mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} = 1$ only when $\sum_{j \in \mathbb{J}} f_j \bar{x}_j - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} > 0$. □

Using Corollary 5.6 we define the first subproblem of our simultaneous multi-column-and-row procedure, to possibly generate a new positive plan $\mathbf{P}$ for **RMP**. Subproblem **SP1** generates a new plan $\mathbf{P}$ with positive reduced cost, with the assumption that the addition of plan $\mathbf{P}$ to **RMP** satisfies $\mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} = \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} = 0$. We later relax this assumption to ensure the optimality of the solution by **RMP** for model **SL**.

Let r_i be a non-negative continuous variable that denotes the premium charged to patient $i \in \mathbb{I}$ for plan $\mathbf{P}$ under construction, and w_{ij} be a binary variable that denotes whether patient i chooses to receive care from provider $j \in \mathbf{P}$. Further, let v_j be a binary variable that denotes whether provider $j \in \mathbb{J}$ is selected in-network of plan $\mathbf{P}$, and b_i be a binary variable that denotes whether plan $\mathbf{P}$ targets patient $i \in \mathbb{I}$. Subproblem **SP1** is defined as:

$$\textbf{SP1}: \quad \max \quad \sum_{i \in \mathbb{I}} r_i - \sum_{i \in \mathbb{I}} \sum_{j \in \mathbb{J}} c_{ij} w_{ij} - \sum_{j \in \mathbb{J}} f_j(1 - \bar{x}_j) v_j - \sum_{i \in \mathbb{I}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} + \mathcal{C}_{ip})\, \bar{y}_{ip} b_i \tag{5.16}$$

$$
\begin{aligned}
\text{s.t.}\quad & w_{ij} \leq v_j && \forall i \in \mathbb{I}, \forall j \in \mathbb{J}, && (5.17)\\
& \sum_{j \in \mathbb{J}} w_{ij} = b_i && \forall i \in \mathbb{I}, && (5.18)\\
& \sum_{j' : u_{ij'} < u_{ij}} w_{ij'} \leq 1 - v_j && \forall i \in \mathbb{I}, \forall j \in \mathbb{J}, && (5.19)\\
& r_i \leq M b_i && \forall i \in \mathbb{I}, && (5.20)\\
& \beta^n \left(\sum_{j \in \mathbb{J} : u_{ij} > 0} u_{ij} v_j \right) - \beta^r r_i \geq b_i - 1 + \epsilon && \forall i \in \mathbb{I}, && (5.21)\\
& \beta^n \left(\sum_{j \in \mathbb{J} : u_{ij} > 0} u_{ij} v_j \right) - \beta^r r_i \geq (\mathcal{U}_{ip} + \epsilon) b_i && \forall i \in \mathbb{I}, \forall p \in \mathbb{P}^c, && (5.22)\\
& r_i \geq 0, \quad b_i \in \{0,1\} && \forall i \in \mathbb{I}, && (5.23)\\
& v_j \in \{0,1\} && \forall j \in \mathbb{J}, && (5.24)\\
& w_{ij} \in \{0,1\} && \forall i \in \mathbb{I}, \forall j \in \mathbb{J}. && (5.25)
\end{aligned}
$$

Objective function (5.16) maximizes the reduced cost of a tuple $(y_{0p}, \ldots, y_{|\mathbb{I}|p}, z_p)$ for plan $\mathbf{P}$, according to corollary 5.6. Constraints (5.17)-(5.19) model patient choice of in-networks providers. In particular, constraint (5.17) ensures that patients $i \in \mathbb{I}$ only visit in-network providers, constraint (5.18) ensure that if a patient $i \in \mathbb{I}$ is targeted by plan $\mathbf{P}$ it visits exactly one in-network provider, and otherwise restricts the patient from visiting any provider. Constraints (5.19) model patient preference and ensure that a patient i selects to receive care from the provider that gives him/her the maximum utility u_{ij}. Constraint (5.20) restricts the insurer from charging a premium to patients $i \in \mathbb{I}$ who are not targeted by plan $\mathbf{P}$. Such patients are later priced-out of plan $\mathbf{P}$ to ensure they do not enroll in it. Constraints (5.21) impose feasibility of a plan for a targeted patient i, such that his/her utility is positive for plan $\mathbf{P}$. Here, ϵ is a small number to enforce strict positivity. Similarly, constraints (5.22) ensure that if a patient i is targeted by plan $\mathbf{P}$, then their utility for plan $\mathbf{P}$ is higher than any competitor's plan. Constraints (5.23)-(5.25) are non-negative and binary restrictions on the decision variables.

If a tuple $(y_{0p}, \ldots, y_{|\mathbb{I}|p}, z_p)$ generated by **SP** has positive reduced cost, we add it to **RMP**. Adding a plan $\mathbf{P}$ to **RMP** involves a price-out strategy for the plan itself, and for all plans $p \in \mathbb{P}$. For tuple $(y_{0p}, \ldots, y_{|\mathbb{I}|p}, z_p)$ corresponding to plan $\mathbf{P}$, we increase premiums $r_{i\mathbf{P}}$ for non-targeted patients $i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}$ such that $u_{i\mathbf{P}} < 0$. That is, we set

$$
r_{i\mathbf{P}} = \beta^n \left(\sum_{j \in \mathbb{J} : u_{ij} > 0} u_{ij} v_j \right) / \beta^r + \epsilon, \quad \forall i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}. \tag{5.26}
$$

Next, we price-out all patients $i \in \bar{\mathbb{I}}_{\mathbf{P}}$ from all plans $p \in \bar{\mathbb{P}}$, by increasing their premiums r_{ip}

according to (5.26). Finally, we add plan $\mathbf{P}$ to set $\bar{\mathbb{P}}$ by adding its corresponding columns and rows to **RMP** and repeat the multi-column-and-row generation procedure. The algorithm iterates until no positive reduced column is found by **SP1**.

To ensure no other positive reduced cost plan exists, we next solve subproblem **SP2**, which is a generalization of **SP1** and takes into account the possibility that $\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}} = \mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}} = 1$. To consider these scenarios, we define additional variables as follows. Let a_j be a binary variable that denotes $\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}}$, and h_p be a binary variable that denotes $\mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}} = 1$. Subproblem **SP2** is formulates as

$$\textbf{SP2}: \quad \max \quad (5.16) + \sum_{j\in\mathbb{J}} f_j \bar{x}_j a_j - \sum_{i\in\mathbb{I}}\sum_{p\in\bar{\mathbb{P}}} (r_{ip}-c_{ip})\bar{y}_{ip}\bar{z}_p(1-b_i)h_p \tag{5.27}$$

$$\text{s.t.} \quad (5.17)-(5.25),$$

$$a_j \le 1 - v_j \qquad \forall j\in\mathbb{J}:\bar{x}_j>0, \tag{5.28}$$

$$a_j \le h_p \qquad \forall p\in\bar{\mathbb{P}}:\bar{z}_p>0, \forall j\in\mathbf{P}_p, \tag{5.29}$$

$$\sum_{i\in\mathbb{I}}(r_{ip}-c_{ip})\bar{y}_{ip}(1-b_i) - \sum_{j\in\mathbf{P}_p} f_j \le M(1-h_p) \qquad \forall p\in\bar{\mathbb{P}}:\bar{z}_p>0, \tag{5.30}$$

$$a_j \in \{0,1\} \qquad \forall j\in\mathbb{J}:\bar{x}_j>0, \tag{5.31}$$

$$h_p \in \{0,1\} \qquad \forall p\in\bar{\mathbb{P}}:\bar{z}_p>0. \tag{5.32}$$

The objective function (5.27) maximizes the reduced cost of a new plan $\mathbf{P}$, and unlike objective (5.16), takes into account the possibility that the addition of plan $\mathbf{P}$ to **RMP** may satisfy $\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}} = \mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}} = 1$. Constraints (5.28) and (5.29) model the scenario where $\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}} = 1$. For a new plan $\mathbf{P}$ and a provider $j\in\mathbb{J}:\bar{x}_j>0$, we have $\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}}=1$ if $j\notin\mathbf{P}$, and all plans $p\in\bar{\mathbb{P}}: j\in\mathbf{P}_p, \bar{z}_p>0$ satisfy $z_p=0$. The former condition is imposed by constraints (5.28), and the latter by constraints (5.29). Constraints (5.30) model the scenario where $\mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}}=1$. This scenario may occur when the cost of offering a plan $p\in\bar{\mathbb{P}}$, i.e., $\sum_{j\in\mathbf{P}_p} f_j$, is higher than its revenue $\sum_{i\in\mathbb{I}}(r_{ip}-c_{ip})\bar{y}_{ip}(1-b_i)$. As discussed in the proof of Theorem 5.5, if the revenue of all plans $p\in\bar{\mathbb{P}}$ that include a provider $j\in\mathbb{J}$ become negative, then it is optimal not to contract provider j. This scenario satisfies $\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}} = \mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}} = 1$, and is modelled by the combination of constraints (5.29) and (5.30). Lastly, constraints (5.31) and (5.32) are binary restrictions on the subproblem variables.

We note that the objective function (5.27) is nonlinear due to the multiplication of binary variables $(1-b_i)h_p$. However, this can be made linear by introducing a new variable $e_{ip}=(1-b_i)h_p$, and constraints

$$e_{ip} \le h_p, \quad e_{ip} \le 1-b_i, \quad h_p - b_i \le e_{ip},$$

for all $i\in\mathbb{I}, \forall p\in\bar{\mathbb{P}}$.

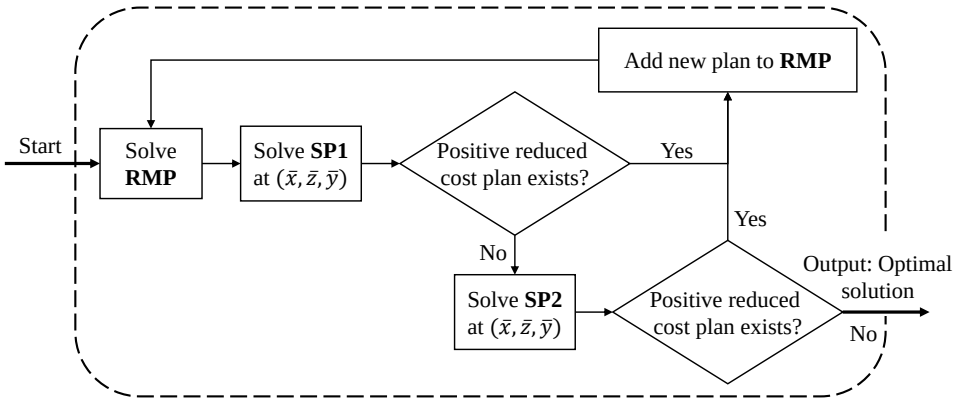


Figure 5.1: Overall Multi-column-and-row generation algorithm.

Solving subproblem **SP2** is computationally more expensive than solving subproblem **SP1**. We therefore, only solve **SP2** when **SP1** does not generate any positive reduced cost plan. This ensures that we find all positive reduced plans by subproblem **SP2**, while avoiding its computational burden at every iteration of the multi-column-and-row generation algorithm. The overall solution algorithm is depicted in Figure 5.1.

5.4.2 Interpretable Insights for Decision Making

In addition to optimal provider network selection and insurance plan design, our framework provides valuable insights for data-driven decision making. In particular, we can use the sensitivity report of the relaxed master problem **RMP** to identify and analyze key aspects of its decision making process. This section discusses a number of such benefits and how they can be used for better decision making by the insurer. We assume the reader has sufficient knowledge on sensitivity analysis, and otherwise refer to Saltelli et al. (2004) for a thorough explanation of sensitivity analysis concepts and practices.

We first consider the information generated for providers currently contracted by the insurer, i.e., $j \in \mathbb{J} : \bar{x}_j = 1$ in **RMP**. The allowable increase of the objective coefficients f_j for variable x_j, gives the maximum contract fee that provider j can ask for while remaining profitable for the insurer. That is, the insurer can guarantee optimality of the current solution despite an increase of fee f_j is within the range of the allowable increase. The insurer can accommodate the increase of f_j without needing to re-optimize its provider network, and possibly remove provider j from its network.

Similarly, the insurer has information on the allowable decrease of contract fees f_j for providers currently not contracted, i.e., $\bar{x}_j = 0$. This information gives the minimum decrease in the contract fee f_j, such that including provider j becomes profitable for the insurer. On the other hand, any change of f_j within the allowable decrease has no effect on the optimal solution, and is not worth anything to the insurer. Such interpretable information is particularly useful in the negotiations

between the insurer and providers on contract fees, and may provide an edge to the insurer in its bargaining power.

We next analyze the the information generated by **RMP** for variables y_{ip}. The allowable increase of objective coefficient for variables $\bar{y}_{ip} = 0$, gives the required increase in premium r_{ip} such that patient i becomes profitable in plan p. That is, if patient i is willing to pay the extra cost of the allowable increase, it can be offered to enroll in plan p without changing the optimal objective value. In the case that patient i is enrolled in a competitor plan, the reduced cost of variable y_{ip} implies how costly it is to attract patient i to plan p using its current premium r_{ip}. This information can thus be used, e.g., in the premium negotiations between the insurer and patients $i \in \mathbb{I}$.

All such information may additionally be used to hedge against uncertainty in patient utility predictions. For targeted patient by plan $p \in \mathbb{P}$, i.e., $y_{ip} = 1$, we can determine using the allowable decrease of its objective coefficient, the degree to which an inaccuracy in utility u_{ij} affects the optimal solution. For example, if the decrease in premium r_{ip}, due to a decrease in utility $\mathcal{U}_{ip}$, is within the allowable decrease, the optimal provider network and plans remain unchanged. This information can be used, e.g., in price-out strategies to determine premiums r_{ip} that can be charged to patient i to ensure he/she does not enroll in plan p.

Lastly, we can analyze the impact of offering plans that are currently not offered by the insurer, i.e., $\bar{z}_p = 0$. The reduced cost of a variables $z_p : \bar{z}_p = 0$ gives the marginal change in the objective function if plan p were to offered by the insurer. This information can be used, e.g., to analyze the quality of different plans and provider networks, such as the ones currently used by the insurer.

5.5 Numerical Results

We test our multi-column-and-row generation approach for the provider network selection problem by two sets of experiments. In the first set, we compare the plans generated by our solution approach to plans generated by heuristic and individual provider selection methods. In our second set of experiments, we evaluate the effect of inaccurate patient utility predictions on the outcome of the multi-column-and-row generation approach. We evaluate the results in terms of the insurer's profit and market share, and the social welfare of patients in paid premiums and healthcare costs.

We generate test data based on the number of patients $|\mathbb{I}| \in \{100, 200\}$ and number of providers $|\mathbb{J}| \in \{10, 20, 30\}$, using the procedure detailed in Appendix 5.C. We generate three datasets for each possible size of the database, giving a total of 18 test instances.

5.5.1 Comparison to Heuristic and Individual Provider Selection

This sections evaluates the performance of the multi-column-and-row generation solution approach for the provider network selection problem, compared to heuristic solutions used in practice and the literature. In particular, we first generate a number of potential insurance plans for the insurer and

its competitor using heuristic algorithms and add them to sets $\bar{\mathbb{P}}, \mathbb{P}^c$. Furthermore, we generate an additional plan for the insurer and its competitor based on individual provider selection motivated by the literature (Geruso and Layton, 2017), and add it to sets $\bar{\mathbb{P}}, \mathbb{P}^c$. Sets $\bar{\mathbb{P}}, \mathbb{P}^c$ are then fed into the relaxed master problem **RMP**, where it is assumed that all competitor plans $\mathbf{P} \in \mathbb{P}^c$ are available to patients, while the insurer must decide which plans $\mathbf{P} \in \bar{\mathbb{P}}$ to offer to patients. After determining the best plans to offer among the ones in set $\bar{\mathbb{P}}$, we use our multi-column-and-row generation framework to possibly construct new insurance plans and optimize the provider network to improve over the initial set of plans in $\bar{\mathbb{P}}$.

To evaluate the impact of the multi-column-and-row generation approach, we assume exact information is available for patient utilities u_{ij} and coefficients β^n, β^r. This allows us to evaluate the results without the influence of prediction accuracy and pricing decisions. Exact information is thus used in both the multi-column-and-row generation framework and all heuristic and individual provider selection methods.

We use four heuristic approaches for provider selection motivated by strategies currently employed by a major health insurance provider in the USA. Each heuristic approach evaluates a provider $j \in \mathbb{J}$ by a specific set of criteria such as quality and affiliation, cost efficiency, patient demand, and the combination of quality and efficiency. The same strategies are used by the insurer and its competitor, but result in distinct plans due the different affiliations of providers $j \in \mathbb{J}$ to the insurer or its competitor. For succinctness we detail each plan with the perspective of only the insurer. The perspective of the competitor is defined analogously. The heuristic provider selection frameworks are as follows.

(i) The quality and affiliation heuristic selects to contract a provider $j \in \mathbb{J}$ if it meets a high quality threshold on its healthcare services, i.e., $q_j \geq 2$, or satisfies a minimum quality threshold $q_j \geq 1$ but is affiliated with the insurer.

(ii) The efficiency heuristic selects to contract a provider $j \in \mathbb{J}$ if it meets a maximum threshold on the average healthcare costs and contract cost charged to the insurer. Recall that a provider j may charge different prices to the insurer based on whether it is affiliated with the insurer or its competitor. In this heuristic algorithm, the insurer determines an average price for the provider, and selects to contract the provider if their contract fee does not exceed its expected cost.

(iii) The demand heuristic selects to contract a provider $j \in \mathbb{J}$ if a minimum of $\alpha\%$ of the patients have a positive utility $u_{ij} > 0$ for that provider. The value of α is 60% for providers affiliated to the insurer and is increased to 80% otherwise.

(iv) The last heuristic strategy combines the first two heuristics and selects to contract a provider if it satisfies both their conditions.

Following the pricing setting of the subproblems **SP1** and **SP2**, we set premiums $r_{i\mathbf{P}}$ as

$$r_{i\mathbf{P}} = \beta^n \left(\sum_{j \in \mathbf{P}} u_{ij} - \epsilon \right) / \beta^r.$$

for all heuristically created plans. That is, we charge patients $i \in \mathbb{I}$ by the highest possible premium that gives them a minimum plan utility of $\mathcal{U}_{i\mathbf{P}} = \epsilon$. Moreover, if $r_{i\mathbf{P}} - c_{ij} < 0$, we price-out patient i from plan $\mathbf{P}$ by charging a higher premium according to (5.26). This ensures that the insurer receives the maximum possible profit from any of the plans generated by the heuristic or individual selection methodologies, and any improvement in the multi-column-and-row generation algorithm is due to higher quality plans, rather then higher charged premiums.

All above heuristic strategies evaluate providers $j \in \mathbb{J}$ individually, with no analysis on how the inclusion or exclusion of a provider may affect the overall insurance plan. Motivated by the literature (see e.g., Geruso and Layton (2017) and references therein) we next evaluate a provider $j \in \mathbb{J}$ by the effect that its exclusion entails for the insurance plan under construction. We start with a plan $\mathbf{P} = \mathbb{J}$, i.e., one that included all providers $j \in \mathbb{J}$. We next compare for a provider $j \in \mathbf{P}$, whether excluding it from plan $\mathbf{P}$ results in positive savings. The savings of excluding provider j is defined as:

$$f_j - \sum_{i \in \mathbb{I}} \left(\max\{0, r_{i\mathbf{P}} - c_{ij}\} \mathbb{1}_{\{u_{ij} = \max\limits_{j \in \mathbf{P}} \{u_{ij}\}, u_{ij} > 0\}} + \max\{0, r_{i\mathbf{P}\setminus j} - c_{ij'}\} \mathbb{1}_{\{u_{ij'} = \max\limits_{j \in \mathbf{P}\setminus j} \{u_{ij}\}, u_{ij'} > 0\}} \right).$$

Here, excluding a provider j from plan $\mathbf{P}$, initially saves on the contract fee f_j. The rest of the savings depend on whether provider j is the highest utility provider for patients $i \in \mathbb{I}$. We recall that by the utility theory, patients $i \in \mathbb{I}$ select to receive care from the provider $j \in \mathbf{P}$ that gives them the highest utility $u_{ij} > 0$. Therefore, if $\mathbb{1}_{\{u_{ij} = \max\limits_{j \in \mathbf{P}} \{u_{ij}\}, u_{ij} > 0\}} = 1$ for a patient $i \in \mathbb{I}$, excluding provider j from plan $\mathbf{P}$ loses the premium $r_{i\mathbf{P}}$ patient i is willing to pay for plan $\mathbf{P}$ and saves on its expected healthcare costs c_{ij}. This is conditioned on $r_{i\mathbf{P}} - c_{ij} > 0$, as otherwise patient i is priced out from plan $\mathbf{P}$. Given the exclusion of provider j, we next calculate the expected profit the insurer may generate from plan $\mathbf{P} \setminus j$. Let j' be the possible substitute provider patient i chooses to receive care from given the exclusion of their highest utility provider j. If such a provider exists, i.e., $\mathbb{1}_{\{u_{ij'} = \max\limits_{j \in \mathbf{P}\setminus j} \{u_{ij}\}, u_{ij'} > 0\}} = 1$, then we generate a profit of $\max\{0, r_{i\mathbf{P}\setminus j} - c_{ij'}\}$ from patient i. Analyzing this effect for all patients $i \in \mathbb{I}$ gives us the above total savings for excluding provider j.

If the total savings from excluding a provider $j \in \mathbb{J}$ is positive, then we exclude provider j from plan $\mathbf{P}$. We perform this analysis for all providers $j \in \mathbf{P}$, in the order of providers with the highest contract fee f_j. This procedure results in a set of providers that are all beneficial to keep in plan $\mathbf{P}$, not just based on individual evaluation but also their individual effect on the entire plan $\mathbf{P}$.

Using the above heuristics, we generate five insurance plans for the insurer and its competitor. We assume the competitor offers all such plans to patients to maximize its competition on market

Table 5.1: Improvements for the insurer by the multi-column-and-row generation framework. Results include the number of plans offered (#Pln) and the percentage of providers contracted (%Cont Prv) for the heuristic and multi-column-and-row generation approaches, the percentage of increase in the insurer's profit made by multi-column-and-row generation (%Profit), the percentage of change in contract fees incurred by the insurer (%Cont Cost), the percentage of change in patient healthcare costs incurred by the insurer (%Pat Cost), and the percentage of change in the insurer's revenue from patient premiums (%Pat Prem).

Instance	Heuristic Selection		Multi-Column-and-Row Generation					
	#Pln	%Cont Prv	#Pln	%Cont Prv	%Profit	%Cont Cost	%Pat Cost	%Pat Prem
$\mathbb{I}100\mathbb{J}10$	1	33	4.3	53	526	200	550	1,677
$\mathbb{I}100\mathbb{J}20$	1	40	7.6	57	296	97	160	1,020
$\mathbb{I}100\mathbb{J}30$	1	26	8.6	52	722	319	95	13,934
$\mathbb{I}200\mathbb{J}10$	1	23	3.3	33	265	25	699	1,524
$\mathbb{I}200\mathbb{J}20$	1	22	13.3	55	1057	213	319	2,796
$\mathbb{I}200\mathbb{J}30$	1	43	25.3	67	423	467	108	1,125

share with the insurer, without considering its profit or loss. The insurer, on the other hand, decides which of the potential plans to offer patients in order to maximize its profit. We then use our multi-column-and-row generation approach to determine whether better plans may be added to $\bar{\mathbb{P}}$. We analyze results by the effects on the insurer given in Table 5.1, its competition with the competitor and patient market share in Table 5.2, and the social welfare of patients given in Table 5.3.

Results of Table 5.1 compare the decisions of the insurer when set $\bar{\mathbb{P}}$ includes only the plans generated by the heuristic algorithms, to the final optimal set of plans generated by the multi-column-and-row generation algorithm. Comparison is on the number of plans offered by the insurer given in columns (#Pln), and the percentage of providers contracted by the insurer given in columns (%Cont Prv). Furthermore, we give the percentage of increase in the insurer's profit using the plans generated by multi-column-and-row generation in column (%Profit), and the percentage of change in contract fees incurred by the insurer in column (%Cont Cost). Lastly, we give the percentage of change in average patient healthcare costs incurred by the insurer (%Pat Cost), and the percentage of change in the insurer's average charged premiums to patients(%Pat Prem).

Results of Table 5.1 show that the multi-column-and-row generation algorithm is able to significantly improve the provider network selection of the insurer. When the insurer is limited to the five plans generated by the heuristic procedures, its best decision is to only offer one plan, often incurring a loss in profits. Using the multi-column-and-row generation algorithm, the insurer is able to offer on average 10.4 insurance plans that effectively target different subsets of patients, and increase profits by an average of 548%. The higher profit is due to higher market share and better patient targeting, that results in an average increase revenue by 3,679%. The increase in revenue is also accompanied by an increase in contract costs, on average by 22%, and patient healthcare

Table 5.2: Effects of multi-column-and-row generation on market share between the insurer and competitor. Results include market share of patients for the insurer (Insr) and competitor (Comp) using heuristic and individual provider selection, and the multi-column-and-row generation framework.

Instance	Heuristic Selection		Multi-Column-and-Row Generation	
	%Share Insr	%Share Comp	%Share Insr	%Share Comp
$\mathbb{I}100\mathbb{J}10$	4	89	65	30
$\mathbb{I}100\mathbb{J}20$	2	93	71	27
$\mathbb{I}100\mathbb{J}30$	2	91	73	26
$\mathbb{I}200\mathbb{J}10$	4	87	47	47
$\mathbb{I}200\mathbb{J}20$	1	93	70	28
$\mathbb{I}200\mathbb{J}30$	8	87	77	22

costs, on average by 322%. Note that premiums and patient healthcare costs of Table 5.1 only give the change in cost for the insurer, and do not represent the welfare costs and premiums incurred by patients. The overall results of the multi-column-and-row generation algorithm show that the insurer should broaden the set of contracted providers, but tailor its insurance plans to different customer segments to maximize its profits.

Results of Table 5.2 compare the market share between the insurer and its competitor when the insurer is limited to offering plans generated heuristically, and when it has the optimal set of plans to offer generated by the multi-column-and-row generation algorithm. Results show that due to negative profit, the insurer should only offer a single plan when limited to heuristically generated plans. Recall that we do not consider the profits of the competitor in our numerical tests, and enforce the competitor to offer all its heuristically generated plans to patients. In such a case, the insurer cannot capture a large portion of patients, as most patient find the plans offered by competitor more attractive. The competitor is able to capture on average 90% of the health insurance market, albeit by incurring a negative profit. On the other hand, when the insurer is able to offer plans generated by the multi-column-and-row generation algorithm it can effectively capture on average 67% of the market and make a positive profit. Despite the broader network offered by the competitor, the insurer is able to attract specific patients segments, and conclude that the remaining patients are not profitable and should be priced out. Priced out patients remain with the competitor, which captures on average 30% of the market.

Lastly, we compare the social welfare of patients when the insurer can select among the heuristically generated plans, or the plans generated by the multi-column-and-row generation algorithm. We observe in Table 5.3 that despite an increase of revenue by the insurer from patient premiums, patients on average pay 21% less for their health insurance. This indicates that the higher revenue by the insurer is not due to an overall increase in charged premiums, but rather due to better in-

Table 5.3: Effects of multi-column-and-row generation on patient welfare. Results include the percentage of patients insured by the insurer or competitor (%Insured) using heuristic and individual provider selection, the percentage of change in total healthcare costs (%Pat Cost) by multi-column-and-row generation, and the percentage of change in total premium (%Prem) payed by patients to the insurer or the competitor by multi-column-and-row generation.

Instance	Heuristic Selection	Multi-Column-and-Row Generation		
	%Insured	%Insured	%Pat Cost	%Prem
$\mathbb{I}100\mathbb{J}10$	93	95	-29	-8
$\mathbb{I}100\mathbb{J}20$	95	98	-41	-27
$\mathbb{I}100\mathbb{J}30$	94	99	-53	-30
$\mathbb{I}200\mathbb{J}10$	91	93	-31	-43
$\mathbb{I}200\mathbb{J}20$	94	98	-47	-29
$\mathbb{I}200\mathbb{J}30$	95	99	-17	12

surance plan design. In particular, the insurer is able to target patients by the optimal plan that is both attractive to the patient and provides the highest profit for the insurer. Moreover, healthcare costs of the entire set of patients decreases under an optimal plan design by the insurer. This is due to better plan design, where patients are guided to receive care from more cost efficient providers. The total number of patients that choose to enroll in any healthcare plans also increase from 93% to 97%, meaning that the new insurance plans are able to attract on average 4% of patients that previously has negative utility for the plans offered by the insurer and competitor.

In summary, using multi-column-and-row generation significantly impacts the profit made by the insurer, increases its market share, and also increases social welfare by decreasing healthcare costs and patient premiums.

5.5.2 Effects of Utility Prediction Accuracy

The multi-column-and-row generation algorithm outputs the optimal set of plans to offer to patients. The optimality of the plans, however, is conditional on the accuracy of predicting patient behavior in the utility function approach. An inaccurate prediction of patient utilities leads to sub-optimal plans that cannot target the intended patients. Inaccuracies in coefficients β^n, β^r further lower solution quality as they influence pricing and price-out strategies. In particular, patients predicted to have positive utility $\mathcal{U}_{i\mathbf{P}} > 0$ for plan $\mathbf{P}$ may turn out to have negative utility $\mathcal{U}_{i\mathbf{P}} < 0$ due to higher prices. Consequently, such patients turn to other unintended insurance plans $\mathbf{P} \in \mathbb{P}$ by the insurer or even a plan $\mathbf{P} \in \mathbb{P}^c$ from the competitor. Similarly, patients predicted to be priced-out of a plan $\mathbf{P}$ may turn out to have positive utility $\mathcal{U}_{i\mathbf{P}} > 0$ due to an underestimation of utility values, and lead to loss for the insurer. Such consequences motivate us to evaluate the influence of utility

Table 5.4: Effect of inaccurate utility predictions on provider network selection. Results include the percentage of change in profit (%Profit), percentage of change in market chare (%Share), percentage of change in patient healthcare costs (%PAt Cost), and percentage of change revenue from patient premiums (%Prem), for the actual choice of patients on the plans generated by the insurer using estimated data.

Uncertainty	Instance	%Profit	%Share	%Pat Cost	%Pat Prem
u_{ij}	$\mathbb{I}100\mathbb{J}10$	-215	-54	-73	-72
	$\mathbb{I}100\mathbb{J}20$	-185	-76	-99	-100
	$\mathbb{I}100\mathbb{J}30$	-166	-75	-98	-99
	$\mathbb{I}200\mathbb{J}10$	-176	-62	-67	-99
	$\mathbb{I}200\mathbb{J}20$	-140	-70	-93	-94
	$\mathbb{I}200\mathbb{J}30$	-142	-75	-98	-99
u_{ij}, β^n, β^r	$\mathbb{I}100\mathbb{J}10$	-243	-100	-100	-100
	$\mathbb{I}100\mathbb{J}20$	-267	-100	-100	-100
	$\mathbb{I}100\mathbb{J}30$	-167	-100	-100	-100
	$\mathbb{I}200\mathbb{J}10$	-615	-100	-100	-100
	$\mathbb{I}200\mathbb{J}20$	-411	-100	-100	-100
	$\mathbb{I}200\mathbb{J}30$	-237	-100	-100	-100

prediction accuracy on the outcome of the multi-column-and-row generation algorithm.

We consider two uncertainty settings in evaluating the effects of utility inaccuracies. In our first setting, we assume that utility values u_{ij} are unknown and estimated, but coefficients β^n, β^r are known with certainty for all patients $i \in \mathbb{I}$. This allows us to evaluate the influence of inaccurate utility predictions with less emphasis on the pricing decisions made based on coefficients β^n, β^r. We next consider uncertainty in both utility values u_{ij} and coefficients β^n, β^r. Using the estimated values of these parameters, we generate the initial set of plans using the heuristic approaches. Estimated parameters are then passed to the multi-column-and-row generation framework to construct and optimize the insurance plan and provider network. After the multi-column-and-row generation algorithm terminates with the optimal set $\bar{\mathbb{P}}$ of plans based on the estimated parameters, we use the exact knowledge of utility values u_{ij} and coefficients β^n, β^r to determine the actual selection of patients $i \in \mathbb{I}$ for plans $\bar{\mathbb{P}} \cup \mathbb{P}^c$. We compare the actual selection of patients to their predicted behavior, and evaluate its influence on the profit and market share of the insurer and its competitor.

Table 5.4 gives the percentage of change in profits, market share, patient healthcare costs, and patient premiums, when actual patient choice is taken into account over the plans optimized using estimated utilities and possibly coefficient β^n, β^r. Results show that despite patient targeting and better plan design by the multi-column-and-row generation algorithm, patients do not choose their intended plans due to inaccurate predictions. In fact, when all parameters are uncertain, the insurer

Table 5.5: Effects of inaccurate utility predictions on market share between the insurer and competitor. Results include the estimated and actual market share of patients for the insurer (Insr) and competitor (Comp).

Uncertainty	Instance	Estimated		Actual	
		%Share Insr	%Share Comp	%Share Insr	%Share Comp
u_{ij}	$\mathbb{I}100\mathbb{J}10$	70	30	15	84
	$\mathbb{I}100\mathbb{J}20$	77	23	1	98
	$\mathbb{I}100\mathbb{J}30$	78	22	3	97
	$\mathbb{I}200\mathbb{J}10$	62	38	1	99
	$\mathbb{I}200\mathbb{J}20$	76	24	6	94
	$\mathbb{I}200\mathbb{J}30$	78	23	3	97
u_{ij}, β^n, β^r	$\mathbb{I}100\mathbb{J}10$	24	76	0	84
	$\mathbb{I}100\mathbb{J}20$	42	58	0	86
	$\mathbb{I}100\mathbb{J}30$	50	50	0	85
	$\mathbb{I}200\mathbb{J}10$	9	91	0	90
	$\mathbb{I}200\mathbb{J}20$	46	54	0	91
	$\mathbb{I}200\mathbb{J}30$	50	50	0	87

is not able to capture any of the market share and its revenue from patient premiums decreases by 100%. Results generally remain the same when coefficients β^n, β^r are known with certainty, where on average, the insurer looses 74% of the estimated market with an average -171% decrease in profits. In both cases, the insurer generally incurs negative profits as it wrongly chose to contract a broad network of providers. Patients in this case choose to receive care from the competitor or remain uninsured, as they have negative utility for the plans offered by the insurer, or have higher utility for the plans offered by the competitor.

As shown in Table 5.5, when all parameters are uncertain the estimated market share of the insurer is on average 37% lower than when exact information is available. This is due to restrictions on optimal design of insurance premiums and utility predictions of patients. This market share, however, is in fact 0%, as the insurer incorrectly predicted patient choice and optimized on estimated values. When parameters β^n, β^r are know with certainty, the insurer is able to retain 5% of its original market share but still looses the majority to the competitor.

In summary, results show that despite the high attractiveness of the utility function approach to estimate and explain patient choice, its low accuracy is problematic in optimizing the provider network selection problem. Under the inaccuracy of the utility function approach, neither the heuristic selection methods nor the multi-column-and-row generation algorithms are able to exploit patient behavior for better provider selection and insurance design. This motivates future work on

developing a more accurate prediction method for patient choice, which is also interpretable and suitable to be used in the multi-column-and-row generation framework.

5.6 Conclusion

We studied the provider network selection problem through an optimization lens, where providers are selected based on an overall optimized network model. We developed a multi-column-and-row generation algorithm to solve our model, and proved properties of positive reduced plans and integrality of the relaxed master problem. We showed that our optimal solution significantly improves over heuristic provider selection methods, and individual selection algorithms used in the literature. Our results show that under accurate prediction of patient choice, the multi-column-and-row generation algorithm increases the insurer's profit by an average 548%, while decreasing the overall healthcare costs by 36% and the overall insurance premium payed by patient by 21%. In addition, the multi-column-and-row generation algorithm is able to attract and increase overall insurance enrollment from an average of 93% to 97%.

We used the utility function approach of the literature as an interpretable method to predict patient choice, and showed how it can be incorporated into our multi-column-and-row generation framework for optimal provider selection. We discussed how interpretable knowledge can be extracted from the multi-column-and-row generation model to aid decision making and increase the bargaining power of the insurer over providers and patients. Lastly, we showed that the accuracy of utility prediction significantly impacts the results of the multi-column-and-row generation algorithm, motivating the need to research more accurate models of patient behavior that are also interpretable.

5.A Predicting Patient Choice

We seek to learn patient behavior in selecting the plans offered by the insurer. This is done by taking as input characteristics of the patient and the plans offered by the insurer, and training a learning algorithm to predict patient choice. For example, we seek to learn patient choice by considering patient demographics, income, and healthcare history together with the breadth, quality, and distance of in-network providers for each plan $p \in \mathbb{P} \cup \mathbb{P}^c$.

One approach to learning patient choice is to use the accurate prediction capabilities of supervised learners, such as deep learning methodologies. By using enough data entries and the right learning architecture, it is possible to use their complex learning procedure to accurately predict patient choice. The challenge, however, is that such methods provide little interpretability. This means limited analysis can be done on the underlying reasons behind patient decisions, which we seek to learn and use in the provider selection model.

To learn patient choice in their preferred in-network provider, we consider using the utility theory (Fishburn, 1968). In a utility-based approach, patients $i \in \mathbb{I}$ evaluate the offered plans by the insurer or its competitor based on a number of factors, e.g., based on in-network providers and charged premium, and select to enroll in a plan $\mathbf{P}^*$ with the highest perceived utility $\mathcal{U}_{i\mathbf{P}^*}$. Patients then select to receive healthcare from the in-network provider $j^* \in \mathbf{P}^*$ of their selected plan $\mathbf{P}^*$, that they perceive to have the highest utility u_{ij^*}. Our observed data is thus the current choice of provider j^* per patient $i \in \mathbb{I}$, among the in-network providers of their current plan $\mathbf{P}^*$.

Accordingly, we estimate patient choice in two steps. First, we consider learning how patients select their preferred in-network provider j^* from their chosen plan $\mathbf{P}^*$. This gives a generic utility function u_{ij} that can be used to evaluate the fit of any provider $j \in \mathbb{J}$. Next, by evaluating the total utility of in-networks of providers for a plan $p \in \mathbb{P} \sup \mathbb{P}^c$, and the premium r_{ip}, we learn how patients select their preferred insurance plan.

The first step to learn patient behavior by the utility theory is to identify the most influential factors in patient choice. Following the literature, (e.g., see works of Ho (2006); Shepard (2016); Ericson and Starc (2015); Raval and Rosenbaum (2018) and references therein), we consider factors such as quality of service q_j, patient loyalty l_i, distance of patients to providers d_{ij}, and fit of provider j to patient i. The fit of a provider $j \in \mathbb{J}$ to patient $i \in \mathbb{I}$ is determined by the types of services that are of value for patients $i \in \mathbb{I}$, and the the types of services offered by provider j. In particular, let $\mathbb{S}$ be the set of all possible healthcare services that may be of interest to patients $i \in \mathbb{I}$ or provided by providers $j \in \mathbb{J}$; e.g., primary care, emergency room, ancillary facilities, surgical, women's hospital, children's hospital, and/or specialty physicians. A patient $i \in \mathbb{I}$ may be interested in any number of these healthcare services, and a provider $j \in \mathbb{J}$ may offer any number of such service to patients.

To identify which services are offered by a provider $j \in \mathbb{J}$, we define $\mathbf{J}_j$ as a $1 \times |\mathbb{S}|$ vector of binaries where the s-th cell has value 1 if provider j offers service $s \in \mathbb{S}$, and zero otherwise. To estimate the interest of patients $i \in \mathbb{I}$ for different services $s \in \mathbb{S}$, we use their healthcare history and

let $\mathbf{I}_i$ be a $1 \times |\mathbb{S}|$ vector of integers where the s-th cell gives the number of times patient i received care of type $s \in \mathbb{S}$ in the past year. Vector $\mathbf{I}_i$ may be used as an estimation of the value a patient $i \in \mathbb{I}$ has for a service $s \in \mathbb{S}$. That is, the value of service type $s \in \mathbb{S}$ for a patient $i \in \mathbb{I}$ is indicated by the number of times patient i has received healthcare of type s. Given vectors $\mathbf{I}_i$ and $\mathbf{J}_j$, we can thus estimate the fit of a provider $j \in \mathbb{J}$ for a patient $i \in \mathbb{I}$ in terms of healthcare services $s \in \mathbb{S}$. Specifically, the sum $\mathbf{I}_i \times \mathbf{J}_j^T$ gives a comparable value of providers $j \in \mathbb{J}$ for patients $i \in \mathbb{I}$.

Using all considered factors on patient utilities, we can estimate values u_{ij} by any concave non-decreasing function, such as

$$u_{ij} = \beta^s \mathbf{I}_i \times \mathbf{J}_j^T + \beta^q q_j + \beta^l l_{ij} - \beta^d d_{ij} + \epsilon_{ij}.$$

Here, ϵ_{ij} is a patient-provider error term, and $\beta^s, \beta^q, \beta^l, \beta^d$ are coefficients of factors that are considered to be influential in decision making, and must be estimated. Assuming that error terms ϵ_{ij} have a standard type I extreme value distribution with density

$$f(\epsilon_{ij}) = exp(-exp(-\epsilon_{ij})),$$

it can be shown (see e.g., Maddala (1986)) that the probability that patient i selects to receive care from provider j is given by

$$\mathbf{Pr}(i, j^*) = \frac{exp(u_{ij^*} - \epsilon_{ij^*})}{\sum\limits_{j \in \mathbb{J}} exp(u_{ij} - \epsilon_{ij})}.$$

This is commonly known as the conditional logit model (McFadden et al., 1973). Given probabilities $\mathbf{Pr}(i, j)$, we can define the overall likelihood function of patient provider selection as

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i \in \mathbb{I}} \prod_{j \in \mathbb{J}} \mathbf{Pr}(i, j)^{\bar{w}_{ij}}.$$

where $\bar{w}_{ij}$ is a binary variable indicating whether patient i selected provider j in the observed dataset. Using maximum likelihood estimation, we can estimate coefficients $\boldsymbol{\beta} = \beta^s, \beta^q, \beta^l, \beta^d$. Note that unlike linear regression, an exact analytical solution does not exist. As the utility function is concave and non-decreasing, coefficients $\boldsymbol{\beta}$ can be estimated using a step-wise gradient ascent algorithm, e.g., such as the Newton-Raphson algorithm.

Having estimated utility values for each provider $j \in \mathbb{J}$, we can now estimate patient utilities $\mathcal{U}_{ip}$ for plans $p \in \mathbb{P} \sup \mathbb{P}^c$. We consider two influential factors in patient plan selection, namely, network breadth and the charged premium. Similar to Ericson and Starc (2015), network breadth is determined by the sum of positive utility values $u_{ij} > 0 : j \in p$. The utility of a plan $p \in \mathbb{P} \cup \mathbb{P}^c$

for patient $i \in \mathbb{I}$ can be calculated using any concave non-decreasing function such as:

$$\mathcal{U}_{ip} = \beta^n \left(\sum_{j \in \mathbf{P}: u_{ij} > 0} u_{ij} \right) - \beta^r r_{ip} + \epsilon_{ip}.$$

Coefficients β^n, β^r are estimated using the same procedure discussed above.

A utility function approach has several benefits in our framework. It generates a utility value for all providers $j \in \mathbb{J}$ and not just the current chosen provider. A utility value is interpretable, in that patients select the highest utility provider available to them, and are also reluctant to pick a provider with negative utility. Furthermore, as the process of generating utility values is known, we can determine the precise reasons why a patient $i \in \mathbb{I}$ has high or low utility for a provider $j \in \mathbb{J}$ or a plan $p \in \mathbb{P} \cup \mathbb{P}^c$. In particular, coefficients β^n, β^r allow us to analyze how the composition of a plan and its charged premium affects patients. This information can be used, e.g., in insurance plan design to target specific patient segments, as we do in our optimization framework.

On the other hand, a utility function may turn out to be an under-estimation of the complex decision making mechanism of patients $i \in \mathbb{I}$. The relationship between the different factors may not be linear as imposed in the utility function, and there may be more important factors not considered in its calculation. Furthermore, estimates of coefficients $\boldsymbol{\beta}$ are generally not optimal, and may lead to inaccuracies in estimating the utility values. We evaluate the accuracy of the utility function approach and its effects on the provider network selection problem in §5.5.

5.B Proof of Theorem 5.5

Proof. The addition of a tuple $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ to **RMP**, involves a price-out strategy for all $y_{ip} : i \in \bar{\mathbb{I}}_{\mathbf{P}}$, the additions of $|\mathbb{I}|$ variables $y_{i\mathbf{P}}$, and variable $z_{\mathbf{P}}$ to **RMP**. As sensitivity analyzes is valid only for the change or addition of a single variable, we proceed by pricing-out variables $y_{ip} : i \in \bar{\mathbb{I}}_{\mathbf{P}}$, and adding variables $z_{\mathbf{P}}$ and $y_{i\mathbf{P}}, \forall i \in \mathbb{I}$ to **RMP** one by one. The overall change in the objective of **RMP** by performing these steps gives the total reduced cost of adding tuple $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ to **RMP**. Without loss of generality, we first perform the price-out strategy, add variable $z_{\mathbf{P}}$, and thereafter add variables $y_{i\mathbf{P}}$ to **RMP**.

We recall that a price-out strategy performed for the addition of plan $\mathbf{P}$ to **RMP**, increases the premiums $r_{ip}, \forall p \in \bar{\mathbb{P}}$ to ensure $y_{ip} = 0, \forall p \in \bar{\mathbb{P}}$ by constraints (5.10). Pricing-out all patient $i \in \bar{\mathbb{I}}_{\mathbf{P}}$ thus decreases the objective (5.1) of **RMP** by

$$\sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - c_{ip})\, \bar{y}_{ip}.$$

A possible consequence of decreasing the revenue generated from offering plan $p \in \bar{\mathbb{P}}$ by $\sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} (r_{ip} - c_{ip})\, \bar{y}_{ip}$, is that the total profit from plan p may become negative. That is, $\sum_{i \in \mathbb{I}} (r_{ip} - c_{ip})\, \bar{y}_{ip} - \sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} (r_{ip} - c_{ip})\, \bar{y}_{ip} - \sum_{j \in \mathbb{J}} f_j \bar{x}_j = \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} (r_{ip} - c_{ip})\, \bar{y}_{ip} - \sum_{j \in \mathbb{J}} f_j \bar{x}_j < 0$. Now, if for a provider $j \in \mathbb{J} : \bar{x}_j = 1$, the profit of all plans $p \in \bar{\mathbb{P}} : j \in \mathbf{P}_p$ become negative, and we have $\sum_{p \in \bar{\mathbb{P}}} z_p > 1$, then the optimal solution of **RMP** updated by the addition of plan $\mathbf{P}$ satisfies $x_j = 0, z_p = 0, \forall p : j \in \mathbf{P}_p$. Consequently, the objective (5.1) is increased by the contract cost f_j of provider j, and decreased by the revenue of all remaining patients serviced by plan p. This changes the objective value (5.1) by

$$\left(\sum_{j \in \mathbb{J}} f_j \bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} \right) \mathbb{1}_{\{\sum_{p \in \bar{\mathbb{P}}} z_p > 1\}}.$$

The overall change in objective (5.1) by performing a price-out strategy is thus

$$- \sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - c_{ip})\, \bar{y}_{ip} + \left(\sum_{j \in \mathbb{J}} f_j \bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} \right) \mathbb{1}_{\{\sum_{p \in \bar{\mathbb{P}}} z_p > 1\}}. \tag{5.33}$$

We next add variable $z_{\mathbf{P}}$ to **RMP**, which involves adding new constraints (5.2) $\forall j \in \mathbf{P}$, updating constraint (5.3), and adding new constraints $z_{\mathbf{P}} \geq 0, \forall i \in \mathbb{I}$ (corresponding to constraints (5.7) without variables y_{ip}) to **RMP**. Note that unlike traditional column generation where it is assumed that an added variable takes a positive value, we add $z_{\mathbf{P}}$ to **RMP** despite it possibly taking a zero

value $z_{\mathbf{P}} = 0$ in **RMP**. This is because the addition of z_p is only part of adding plan $\mathbf{P}$ to **RMP**, and its reduced cost does not reflect its overall effect on **RMP**. The reduced cost of adding $z_{\mathbf{P}}$ to **RMP** is thus its final value in **RMP** multiplied by its objective coefficient 0 minus the sum of dual values for all added and updated constraints, i.e.,

$$\left(0 - \sum_{j \in \mathbf{P}} \delta_{j\mathbf{P}} - \alpha - \sum_{i \in \mathbb{I}} \lambda_{i\mathbf{P}}\right) \bar{z}_{\mathbf{P}}.$$

Here, $\delta_{j\mathbf{P}}, \alpha$, and $\lambda_{i\mathbf{P}}$ are the dual values of constraints (5.2), (5.3), and (5.7), respectively. We propose and prove the value for each dual value as follows.

(i) $\delta_{jp}\bar{z}_{\mathbf{P}} = f_j(1 - \bar{x}_j)\bar{z}_{\mathbf{P}}$.

Assume $\bar{x}_j = 1$ in the solution of **RMP**. The addition of variable $z_{\mathbf{P}}$ to **RMP** adds a new constraint (5.2) of the form $z_{\mathbf{P}} \leq x_j$ to **RMP** that enforces $x_j = 1$ if $z_{\mathbf{P}} = 1$. If $\bar{x}_j = 1$ in the current solution of **RMP** the added constraint (5.2) is redundant and has dual value $\delta_{j\mathbf{P}} = f_j(1 - \bar{x}_j) = 0$. On the other hand if $\bar{x}_j = 0$, the addition of the new constraint (5.2) to **RMP** with variable $z_{\mathbf{P}} = 1$ incurs the contract cost of provider j, with dual value $\delta_{j\mathbf{P}} = f_j\bar{z}_{\mathbf{P}}$. Combining both cases gives the statement.

(ii) $\alpha\bar{z}_{\mathbf{P}} = \left(\sum_{j \in \mathbb{J}} f_j\bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}}\right) \mathbb{1}_{\{\sum_{p \in \bar{\mathbb{P}}} z_p = 1\}}$.

To prove this statement, we first prove that $\alpha = \max\{0, \sum_{j \in \mathbb{J}} f_j\bar{x}_j - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip}\}$. Assume first that $\sum_{j \in \mathbb{J}} f_j\bar{x}_j - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip} < 0$. It is thus profitable and optimal to have $z_p = 1$ for at least one $p \in \bar{\mathbb{P}}$ which also satisfies $\sum_{p \in \bar{\mathbb{P}}} z_p \geq 1$ regardless of the restriction imposed by constraint (5.3). Constraint (5.3) is thus redundant and has shadow price $\alpha = 0$. Now assume $\sum_{j \in \mathbb{J}} f_j\bar{x}_j - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip} > 0$. This implies that offering any plan $p \in \bar{\mathbb{P}}$ has negative profit, and it is optimal to offer exactly one plan $p \in \bar{\mathbb{P}}$ with minimum loss, due to the restriction imposed by constraint (5.3). Constraint (5.3) is thus binding, and its removal from **RMP** gives a solution $x_j = 0, \forall j \in \mathbb{J}$ and $y_{ip} = 0, \forall i \in \mathbb{I}, \forall p \in \bar{\mathbb{P}}$ with objective 0. Therefore, we have $\alpha = \sum_{j \in \mathbb{J}} f_j\bar{x}_j - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip}$. Combining both cases gives the statement $\alpha = \max\{0, \sum_{j \in \mathbb{J}} f_j\bar{x}_j - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip}\}$.

It is now sufficient to prove that $\max\{0, \sum_{j \in \mathbb{J}} f_j\bar{x}_j - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip}\}\bar{z}_{\mathbf{P}} = \left(\sum_{j \in \mathbb{J}} f_j\bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\,\bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}}\right) \mathbb{1}_{\{\sum_{p \in \bar{\mathbb{P}}} z_p = 1\}}$. A value of $\bar{z}_{\mathbf{P}} = 0$ implies that the addition of variable $z_{\mathbf{P}}$ has no effect on the current solution of **RMP**, i.e.,

$\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}} = \mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}} = 0$. Therefore, given $\bar{z}_{\mathbf{P}} = 0$, the equation holds with both sides taking value 0. A value of $\bar{z}_{\mathbf{P}} = 1$ implies that offering plan $\mathbf{P}$ which incurs a contract cost of $\sum_{j\in\mathbf{P}} f_j$ with no revenue, is more beneficial than the current solution of **RMP**. This occurs when the current objective of **RMP** is more costly than the cost of setting $z_{\mathbf{P}} = 1$, i.e., $\sum_{j\in\mathbb{J}} f_j\bar{x}_j - \sum_{i\in\mathbb{I}\setminus\bar{\mathbb{I}}_{\mathbf{P}}}\sum_{p\in\bar{\mathbb{P}}}(r_{ip}-\mathcal{C}_{ip})\,\bar{y}_{ip} > \sum_{j\in\mathbf{P}} f_j > 0$. Moreover, when the objective of **RMP** is negative we have $\sum_{p\in\bar{\mathbb{P}}} z_p = 1$, as discussed above. Therefore, the addition of variable $z_{\mathbf{P}}$ satisfies $\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}} = \mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}} = 1$. Both sides of the equation thus take value $\sum_{j\in\mathbb{J}} f_j\bar{x}_j - \sum_{i\in\mathbb{I}\setminus\bar{\mathbb{I}}_{\mathbf{P}}}\sum_{p\in\bar{\mathbb{P}}}(r_{ip}-\mathcal{C}_{ip})\,\bar{y}_{ip}$ for the case where $\bar{z}_{\mathbf{P}} = 1$, which concludes this part of the proof.

(iii) $\lambda_{i\mathbf{P}} = 0$.

The addition of variable $z_{\mathbf{P}}$ to **RMP** adds new constraints (5.7) to **RMP**. However, as no variable $y_{i\mathbf{P}}$ is yet added to **RMP**, all constraints (5.7) are of the form $z_{\mathbf{P}} \geq 0$, which are redundant and have shadow price of zero.

Due to proofs (i)-(iii), we have the reduced cost of adding variable $z_{\mathbf{P}}$ to **RMP** as

$$\left(\sum_{j\in\mathbb{J}} f_j\bar{x}_j\mathbb{1}_{\{\mathbf{P}\Rightarrow x_j=0\}} - \sum_{i\in\mathbb{I}\setminus\bar{\mathbb{I}}_{\mathbf{P}}}\sum_{p\in\bar{\mathbb{P}}}(r_{ip}-\mathcal{C}_{ip})\,\bar{y}_{ip}\mathbb{1}_{\{\mathbf{P}\Rightarrow z_p=0\}}\right)\mathbb{1}_{\{\sum_{p\in\bar{\mathbb{P}}} z_p=1\}} + f_j(1-\bar{x}_j)\bar{z}_{\mathbf{P}}. \tag{5.34}$$

We next add variables $y_{i\mathbf{P}}, \forall i \in \mathbb{I}$ to **RMP** one by one. Note variables $y_{i\mathbf{P}}$ are independent on patients, in that the addition of one $y_{i\mathbf{P}}$ does not affect the value of any other $y_{i'\mathbf{P}} : i' \neq i$. We thus propose and prove the reduced cost of variable $y_{i\mathbf{P}}$ for a single patient i, and the remaining patients follow analogously.

The addition of a variable $y_{i\mathbf{P}} : i \notin \bar{\mathbb{I}}_{\mathbf{P}}$ to **RMP** has zero reduced cost, as it satisfies $y_{i\mathbf{P}} = 0$ due to constraints (5.10). The reduced cost of adding variable $y_{i\mathbf{P}} : i \in \bar{\mathbb{I}}_{\mathbf{P}}$ to **RMP** is its objective coefficient $r_{i\mathbf{P}} - c_{i\mathbf{P}}$ minus the sum of dual values for all updated constraints, i.e.,

$$r_{i\mathbf{P}} - c_{i\mathbf{P}} - \lambda_{i\mathbf{P}} - \mu_i.$$

where μ_i are the dual variables associated to constraints (5.8). Note that due to the price-out strategy, we have $u_{i\mathbf{P}} > 0, u_{i\mathbf{P}} > u_{ip}, \forall i \in \bar{\mathbb{I}}_{\mathbf{P}}, \forall p \in \mathbb{P}\cup\mathbb{P}^c$ and thus variables $y_{i\mathbf{P}} : i \in \bar{\mathbb{I}}_{\mathbf{P}}$ do not update any of the constraints (5.10)-(5.12). We propose and prove the value for each shadow price as follows.

(iv) $\lambda_{i\mathbf{P}} = \sum_{j\in\mathbf{P}} f_j(1-\bar{x}_j)(1-\bar{z}_{\mathbf{P}})$.

Assume $\bar{z}_{\mathbf{P}} = 1$, then adding variable $y_{i\mathbf{P}}$ to its corresponding constraint (5.7) is equivalent to adding redundant constraint $y_{i\mathbf{P}} \leq 1$ to **RMP**, and has dual value of $\lambda_{i\mathbf{P}} = 0$. On the other

hand, if $\bar{z}_{\mathbf{P}} = 0$, adding variable $y_{i\mathbf{P}} = 1$ to its corresponding constraint (5.7) sets $z_{\mathbf{P}} = 1$ and changes the objective of **RMP** by the reduced cost of $z_{\mathbf{P}}$, i.e., $\sum_{j \in \mathbf{P}} f_j(1 - \bar{x}_j)$. Combining both cases gives the statement.

(v) $\mu_i = 0$.

Due to the price-out strategy, patients $i \in \bar{\mathbb{I}}_{\mathbf{P}}$ are priced-out of all plans $p \in \bar{\mathbb{P}}$ and are currently served by a plan $p \in \mathbb{P}^c$. Therefore, the addition of variable $y_{i\mathbf{P}}$ to constraint (5.8), which sets some $y_{ip} : p \in \mathbb{P}^c$ to zero and has no effect on the objective value (5.1). This implies $\mu_i = 0$.

Due to proofs (iv),(v), we have the reduced cost of variable $y_{i\mathbf{P}}$ as

$$r_{i\mathbf{P}} - c_{i\mathbf{P}} - \sum_{j \in \mathbf{P}} f_j(1 - \bar{x}_j)(1 - \bar{z}_{\mathbf{P}}). \tag{5.35}$$

We can now determine the overall reduced cost of adding a tuple $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ to **RMP**. In particular, by (5.33)-(5.35), the sum of reduced costs of pricing out patients $i \in \bar{\mathbb{I}}_{\mathbf{P}}$, and adding variables $z_{\mathbf{P}}, y_{i\mathbf{P}} \forall i \in \mathbb{I}$ to **RMP**, gives the reduced cost of tuple $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ as

$$\begin{aligned}
&\sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} (r_{i\mathbf{P}} - c_{i\mathbf{P}}) - \sum_{j \in \mathbf{P}} f_j(1 - \bar{x}_j)(1 - \bar{z}_{\mathbf{P}}) - \sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - c_{ip})\, \bar{y}_{ip} + \\
&\left(\sum_{j \in \mathbb{J}} f_j \bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} \right) \mathbb{1}_{\{\sum_{p \in \bar{\mathbb{P}}} z_p > 1\}} + \\
&\left(\sum_{j \in \mathbb{J}} f_j \bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} \right) \mathbb{1}_{\{\sum_{p \in \bar{\mathbb{P}}} z_p = 1\}} + f_j(1 - \bar{x}_j) \bar{z}_{\mathbf{P}} = \\
&\sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} (r_{i\mathbf{P}} - \mathcal{C}_{i\mathbf{P}}) - \sum_{j \in \mathbf{P}} f_j(1 - \bar{x}_j) - \sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} + \sum_{j \in \mathbb{J}} f_j \bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \\
&\sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}}.
\end{aligned}$$

We lastly prove the converse, that if tuple $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ has positive reduced cost, the statement of the theorem holds. The statement gives the change in the objective function (5.1) if plan $\mathbf{P}$ were to be offered to patients. As the tuple $(y_{0\mathbf{P}}, \ldots, y_{|\mathbb{I}|\mathbf{P}}, z_{\mathbf{P}})$ has positive reduced cost, its addition to **RMP** must increase its objective and thus

$$\begin{aligned}
&\sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} (r_{i\mathbf{P}} - \mathcal{C}_{i\mathbf{P}}) - \sum_{j \in \mathbf{P}} f_j(1 - \bar{x}_j) - \sum_{i \in \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} + \sum_{j \in \mathbb{J}} f_j \bar{x}_j \mathbb{1}_{\{\mathbf{P} \Rightarrow x_j = 0\}} - \\
&\sum_{i \in \mathbb{I} \setminus \bar{\mathbb{I}}_{\mathbf{P}}} \sum_{p \in \bar{\mathbb{P}}} (r_{ip} - \mathcal{C}_{ip})\, \bar{y}_{ip} \mathbb{1}_{\{\mathbf{P} \Rightarrow z_p = 0\}} > 0
\end{aligned}$$

. □

5.C Generating Patients and Providers

A patient $i \in \mathbb{I}$ is identified by its demographics, healthcare history, and coefficients $\boldsymbol{\beta}$ of the utility function denoting his/her internal preference. Patients demographics include age, gender, location of residence, expected interest in healthcare quality, overall frequency of healthcare visits, and loyalty to the insurer or its competitor. Patients healthcare history includes the types of healthcare services the patient received and their frequency.

Table 5.6: Patient demographic classes, labels, and classification probability thresholds.

Demographic	Class	Class label	Probability threshold
Age	$[[0-20), [20-30), [30-40), [40-50), [50-60), [60-70), [70, \infty)]$	$[1, \ldots, 7]$	(25,39,52,65,78,89,100)
Gender	[Male, Female]	[1,2]	(50,100)
Location	[Central,North,East,West,South]	[1,2,3,4,5]	(20,40,60,80,100)
Quality	[Low, High]	[1,2]	(30,100)
Frequency	[Low, Medium, High]	[1,2,3]	(20,80,100)
Loyalty	[Insurer, Competitor]	[1,2]	(50,100)

Patient demographics are generated according Table 5.6, where the second column gives the classifications, the third column gives a numerical label for each class, and the last column gives the probability thresholds $(\mathbf{Pr}_1, \ldots, \mathbf{Pr}_k)$ used to classify a random probability into each class. For example, patient age is categorized into six categories of $[[0-20), [20-30), [30-40), [40-50), [50-60), [60-70), [70, \infty)]$, labeled $[1, \ldots, 7]$, and classified according to probability thresholds $(25, 39, 52, 65, 78, 89, 100)$. To classify a patient into an age class $[1, \ldots, 7]$, we generate a random probability $\mathbf{Pr}$ according to a uniform distribution $\mathbf{Pr} = \text{unif}\{0, 100\}$ and classify it into class k^* such that $\mathbf{Pr}_{k^*}$ is the minimum probability threshold that satisfies $\mathbf{Pr} \leq \mathbf{Pr}_{k^*}$. Wherever possible, probability thresholds are generate using real data from the demographics of the United State's population, e.g., in the case of age distribution.

The frequency of healthcare services received by a patient $i \in \mathbb{I}$ primarily depends on patient i's age, and a random frequency class of healthcare visits that may be low, medium, or high according to Table 5.6. The total frequency of each patient is calculated based on the functions given in Table

Table 5.7: Frequency of healthcare visits by patient age and frequency.

Frequency class	Age Class						
	1	2	3	4	5	6	7
Low	[0,...,1]	[0,...,2]	[0,...,3]	[0,...,4]	[0,...,5]	[0,...,6]	[0,...,7]
Medium	[3,...,7]	[3,...,8]	[3,...,9]	[3,...,10]	[3,...,11]	[3,...,12]	[3,...,13]
High	[6,...,14]	[6,...,15]	[6,...,16]	[6,...,17]	[6,...,18]	[6,...,19]	[6,...,20]

Table 5.8: Class of healthcare services for categories of patients.

Patient Category	Types of healthcare services	Probability threshold
Younger Male	[Emergency services, Primary care, Ancillary, Surgical, Children's, Specialty]	(10,60,70,80,90,100)
Younger Female	[Emergency services, Primary care, Ancillary, Surgical, Women's, Children's, Specialty]	(10,40,50,60,80,90,100)
Older Male	[Emergency services, Primary care, Ancillary, Surgical, Specialty]	(20,50,70,80,100)
Older Female	[Emergency services, Primary care, Ancillary, Surgical, Women's, Children's, Specialty]	(20,50,70,80,100)

Table 5.9: Patient coefficients for utility generation.

Coefficient	Probability distribution
β^d	unif$\{2, 10\}$
β^q	unif$\{1.2, 2.8\}$ + $\mathbb{1}_{q_i \geq 1}$
β^l	unif$\{0.2, 2\}$
β^s	unif$\{1, 10\}$
β^n	unif$\{1, 10\}$
β^r	unif$\{0.25, 2.5 - \beta^n/4\}$

5.7. As shown in Table 5.7, older patients are modelled to have higher frequency of healthcare visits.

The types of healthcare services received by each patient depends on the patient's age and gender, as shown in Table 5.8. The first column of Table 5.8 gives the possible class of services for each category of patients, and the second column gives the probability threshold used in classifying a healthcare visits. Lastly, the coefficients of the utility function theory are generated randomly according to the probability thresholds given in Table 5.9.

We now discuss the generation of providers $j \in \mathbb{J}$. A provider $j \in \mathbb{J}$ is identified by its characteristics including location, affiliation to the insurer or its competitor, overall quality of its healthcare services, the types of healthcare services it offers to patients, and its minimum annual cost of contract for the insurer and its competitor. The provider characteristics are generated according to Table 5.10. We assume that a provider $j \in \mathbb{J}$ affiliated with an insurer charges the competitor a 20% higher contract fee f_j.

As the final step, the types of services offered by a provider $j \in j$ are generated independently

Table 5.10: Provider characteristics and probability thresholds.

Characteristic	Class	Class label	Probability threshold
Location	[Central,North,East,West,South]	[1,2,3,4,5]	(20,40,60,80,100)
Affiliation	[Insurer, Competitor]	[1,2]	(50,100)
Quality	[Low, Medium, High]	[1,2,3]	(33,66,100)
Frequency	[Low, Medium, High]	[1,2,3]	(20,80,100)

Table 5.11: Provider services and probability thresholds.

Type of service	Probability that provider provides service
Emergency services	20%
Primary care physician	70%
Ancillary services	30%
Surgical services	30%
Women's hospital	10%
Children's hospital	10%
Specialty services	30%

as binary values according to the probability thresholds given in Table 5.11.

Chapter 6

Conclusion

Interpretable machine learning is a powerful tool for data-driven decision making. This dissertation studied algorithms and applications of interpretable learning, with an emphasis on pattern mining algorithms and data-driven applications. Our focus was on developing interpretable, accurate, and scalable algorithms, with the final goal of accommodating large-size real-world problems in management science.

We contributed to the multiple sequence alignment problem by developing a novel exact solution algorithm that improves both the scalability and accuracy of results generated in the literature. Our algorithm is the first to model and solve multiple sequence alignment in polynomial space requirements in the size of the problem. We further close 37 out of 51 benchmark instances to optimality for the first time, and significantly increase the accuracy of alignments in the remaining instances.

We contributed to Constraint-based Sequential Pattern mining, by developing a novel algorithm which is able to handle complex constraints over the mined patterns. We showed that our algorithm is scalable both in time and space, and provides a concise set of patterns that are better fit for decision making tasks. While our algorithm is primarily designed to focus on complex constraint satisfaction, we showed that it remains competitive with algorithms tailored to handle more simple constraint. We solved instances with up to 10 million data entries, increasing the size of instances solved in the literature by an order of magnitude.

We next developed an interpretable data-driven decision making framework using pattern mining. As our first contribution we developed novel methodologies and models to accommodate large size datasets in pattern mining, which common in the practice of management. We then showed how pattern mining can be used to mine knowledge for sequential decision making tasks. Our approach focused on generating interpretable explanations with statistical guarantees for the occurrence of outcomes that are of interest to the practitioner. We showed how such explanations are generated, statistically guaranteed by hypothesis tests, and aggregated into a knowledge tree for an interpretable and visual decision support tool for decision making.

We applied our methodology to two large-size problems in marketing and finance. While the pattern mining algorithms in the literature are limited to datasets with at most millions of data entries, we showed that our methodology scales up to three billion data entries. We used this capability to generate interpretable explanations for user skips in online music streaming platforms, and evaluated the role of technical analysis in the stock market. We showed that by using an average number of 6,400 patterns, we can explain 100% of approximately 1.1 billion user skips in our marketing application. On the other hand, despite novel pattern mining techniques to find hidden patterns in data, we showed that technical analysis in the stock market is generally not sufficient for investment decision making.

As our final contribution, we focused on the provider network selection model in insurance plan design. We developed a novel solution methodology based on a simultaneous multi-column-and-row generation algorithm and interpretable learning of patient behavior. We showed that our algorithm significantly increases solution quality for the insurer compared to individual provider selection strategies used in the literature. Moreover, we showed that inaccuracies in predicting patient behavior using the utility function approach of the literature considerably decreases solution quality in the provider network selection model.

The contributions of this dissertation show how techniques from optimization and mathematical programming can benefit interpretable learning methodologies in accuracy and scalability. Interpretable learning algorithms such as pattern mining are, however, still faced by challenges in prediction accuracy.

In the multiple sequence alignment problem, the scalability of exact algorithms is still far below the requirements of their application, and the accuracy of heuristic learners is far below the optimal solution. Therefore, exact algorithms for sequence alignment remain highly desirable. In pattern mining, we showed the power that interpretability brings in solving real-life applications in management science. Numerous unexplored applications exist, e.g., in healthcare and clinical trials, where interpretability is critical for decision making while no such algorithm exists for knowledge discovery. Another interesting line of future work for pattern mining is to integrate the interpretability provided by sequential patterns into accurate learning models. This may benefit both the accuracy of prediction in supervised learners, while maintaining the interpretability provided by sequential patterns. Similarly, an integration of interpretable and supervised algorithms may increase the prediction accuracy for the provider network selection problem. Our results show that despite the attractiveness of the utility function approach, its results are often not accurate enough for the provider network selection problem. This motivates the need to design and integrate a more accurate predictor of patient choice in order to achieve optimal provider network design.

Bibliography

Agrawal, R., Imieliński, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

Agrawal, R., Srikant, R., et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

Akter, S. and Wamba, S. F. Big data analytics in e-commerce: a systematic review and agenda for future research. *Electronic Markets*, 26(2):173–194, May 2016.

Althaus, E., Caprara, A., Lenhof, H.-P., and Reinert, K. A branch-and-cut algorithm for multiple sequence alignment. *Mathematical Programming*, 105(2-3):387–425, 2006.

Andersen, H. R., Hadzic, T., Hooker, J. N., and Tiedemann, P. A constraint store based on multi-valued decision diagrams. In *International Conference on Principles and Practice of Constraint Programming*, pages 118–132. Springer, 2007.

Aoga, J. O. R., Guns, T., and Schaus, P. Mining time-constrained sequential patterns with constraint programming. *Constraints*, pages 1–23, 2017.

Apel, W. *The Harvard dictionary of music*. Harvard University Press, 2003.

Appelbaum, D., Kogan, A., and Vasarhelyi, M. A. Big data and analytics in the modern audit engagement: Research needs. *AUDITING: A Journal of Practice & Theory*, 36(4):1–27, 2017.

Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435. ACM, 2002.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

BaliBASE4. Balibase 4, 2017. URL http://www.lbgi.fr/balibase/. Online; accessed 5-November-2017.

Barros, P. P. Cream-skimming, incentives for efficiency and payment system. *Journal of health economics*, 22(3):419–443, 2003.

Becker, B., Behle, M., Eisenbrand, F., and Wimmer, R. Bdds in a branch and cut framework. In *International Workshop on Experimental and Efficient Algorithms*, pages 452–463. Springer, 2005.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

Bergman, D., Cire, A. A., van Hoeve, W.-J., and Hooker, J. N. *Decision Diagrams for Optimization*. Springer, 2016a.

Bergman, D., Cire, A. A., and van Hoeve, W. Mdd propagation for sequence constraints. *Journal of Artificial Intelligence Research*, 50:697–722, 2014.

Bergman, D., Cire, A. A., van Hoeve, W.-J., and Hooker, J. N. Discrete optimization with decision diagrams. *INFORMS Journal on Computing*, 28(1):47–66, 2016b.

Bistarelli, S. and Bonchi, F. Soft constraint based pattern mining. *Data & Knowledge Engineering*, 62(1):118–137, 2007.

Bonchi, F. and Lucchese, C. Pushing tougher constraints in frequent pattern mining. In *PAKDD*, volume 5, pages 114–124. Springer, 2005.

Bonchi, F. and Lucchese, C. Extending the state-of-the-art of constraint-based pattern discovery. *Data & Knowledge Engineering*, 60(2):377–399, 2007.

Borah, A. and Nath, B. Fp-tree and its variants: Towards solving the pattern mining challenges. In *Proceedings of First International Conference on Smart System, Innovations and Computing*, pages 535–543. Springer, 2018.

Bouquet, F. and Jégou, P. Solving over-constrained csp using weighted obdds. In *International Workshop on Over-Constrained Systems*, pages 293–308. Springer, 1995.

Bryant, R. E. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

Bulkowski, T. N. *Encyclopedia of chart patterns*, volume 225. John Wiley & Sons, 2011.

Cambazard, H., Hadzic, T., and O'Sullivan, B. Knowledge compilation for itemset mining. In *ECAI*, volume 10, pages 1109–1110, 2010.

Capps, C., Dranove, D., and Satterthwaite, M. Competition and market power in option demand markets. *RAND Journal of Economics*, pages 737–763, 2003.

Carrillo, H. and Lipman, D. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988.

Cayrol, C., Lagasquie-Schiex, M.-C., and Schiex, T. Nonmonotonic reasoning: from complexity to algorithms. *Annals of Mathematics and Artificial Intelligence*, 22(3-4):207–236, 1998.

Chakraborty, A. and Bandyopadhyay, S. Fogsaa: Fast optimal global sequence alignment algorithm. *Scientific reports*, 3, 2013.

Chen, H.-C. and Chen, A. L. A music recommendation system based on music data grouping and user interests. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 231–238. ACM, 2001.

Chen, Y.-L. and Hu, Y.-H. Constraint-based sequential pattern mining: The consideration of recency and compactness. *Decision Support Systems*, 42(2):1203–1215, 2006.

Cheng, Z. and Shen, J. On effective location-aware music recommendation. *ACM Transactions on Information Systems (TOIS)*, 34(2):13, 2016.

Cire, A. A. and van Hoeve, W.-J. Multivalued decision diagrams for sequencing problems. *Operations Research*, 61(6):1411–1428, 2013.

Coleman, B. J. Identifying the "players" in sports analytics research. *INFORMS Journal on Applied Analytics*, 42(2):109–118, 2012.

Corea, F. Big data and risk management in financial markets: a survey. Technical report, Montreal Institute of Structured Finance and Derivatives, 04 2016.

Crespin, D. Primary care clinic switching due to provider network restrictions: Effects on costs, utilization, and quality, 2016.

Cui, G., Wong, M. L., and Lui, H.-K. Machine learning for direct marketing response models: Bayesian networks with evolutionary programming. *Management Science*, 52(4):597–612, 2006.

Darwiche, A. and Marquis, P. Compiling propositional weighted bases. *Artificial Intelligence*, 157 (1-2):81–113, 2004.

Detienne, B., Sadykov, R., and Tanaka, S. The two-machine flowshop total completion time problem: A branch-and-bound based on network-flow formulation. In *7th Multidisciplinary International Conference on Scheduling: Theory and Applications*, pages 635–637, 2015.

Detienne, B., Sadykov, R., and Tanaka, S. The two-machine flowshop total completion time problem: branch-and-bound algorithms based on network-flow formulation. *European Journal of Operational Research*, 252(3):750–760, 2016.

Dougherty, J., Kohavi, R., and Sahami, M. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*, pages 194–202. Elsevier, 1995.

Došilović, F. K., Brčić, M., and Hlupić, N. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0210–0215, 2018.

Drew, J., Tysiac, K., and White, S. Ethical implications of artificial intelligence. https://www.fm-magazine.com/issues/2018/dec/ethical-implications-of-artificial-intelligence.html, 2019. Accessed: 2019-08-14.

Edgar, R. C. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

Ericson, K. M. and Starc, A. Measuring consumer valuation of limited provider networks. *American Economic Review*, 105(5):115–19, 2015.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., et al. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996.

Fischetti, M. and Toth, P. An additive bounding procedure for combinatorial optimization problems. *Operations Research*, 37(2):319–328, 1989.

Fishburn, P. C. Utility theory. *Management science*, 14(5):335–378, 1968.

Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., and Thomas, R. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

Garofalakis, M. N., Rastogi, R., and Shim, K. Spirit: Sequential pattern mining with regular expression constraints. In *VLDB*, volume 99, pages 7–10, 1999.

Geruso, M. and Layton, T. J. Selection in health insurance markets and its policy remedies. *Journal of Economic Perspectives*, 31(4):23–50, 2017.

Ghosh, S., Feng, M., Nguyen, H., and Li, J. Hypotension risk prediction via sequential contrast patterns of icu blood pressure. *IEEE journal of biomedical and health informatics*, 20(5):1416–1426, 2015.

Guns, T., Dries, A., Nijssen, S., Tack, G., and De Raedt, L. Miningzinc: A declarative framework for constraint-based mining. *Artificial Intelligence*, 244:6–29, 2017.

Hajek, P. and Henriques, R. Mining corporate annual reports for intelligent detection of financial statement fraud–a comparative study of machine learning methods. *Knowledge-Based Systems*, 128:139–152, 2017.

Hämäläinen, W. and Nykänen, M. Efficient discovery of statistically significant association rules. In *2008 Eighth IEEE International Conference on Data Mining*, pages 203–212. IEEE, 2008.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M. C. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, pages 355–359, 2000.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224, 2001.

Han, J., Pei, J., Yin, Y., and Mao, R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.

Hariri, N., Mobasher, B., and Burke, R. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 131–138. ACM, 2012.

Hijikata, Y., Iwahama, K., and Nishida, S. Content-based music filtering system with editable user profile. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1050–1057. ACM, 2006.

Hirschberg, D. S. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

Ho, K. and Lee, R. S. Insurer competition in health care markets. *Econometrica*, 85(2):379–417, 2017.

Ho, K. and Lee, R. S. Equilibrium provider networks: Bargaining and exclusion in health care markets. *American Economic Review*, 109(2):473–522, 2019.

Ho, K. The welfare effects of restricted hospital choice in the us medical care market. *Journal of Applied Econometrics*, 21(7):1039–1079, 2006.

Hoda, S., Van Hoeve, W.-J., and Hooker, J. N. A systematic approach to mdd-based constraint programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 266–280. Springer, 2010.

Hooker, J. N. Logic, optimization, and constraint programming. *INFORMS Journal on Computing*, 14(4):295–321, 2002.

Hooker, J. N. and Ottosson, G. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

Hosseininasab, A. and van Hoeve, W.-J. Exact multiple sequence alignment by synchronized decision diagrams. *INFORMS Journal on Computing*, 2019. (to appear).

Hosseininasab, A., van Hoeve, W.-J., and Cire, A. A. Constraint-based sequential pattern mining with decision diagrams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1495–1502, 2019.

Hung, C.-L., Lin, Y.-S., Lin, C.-Y., Chung, Y.-C., and Chung, Y.-F. Cuda clustalw: An efficient parallel algorithm for progressive multiple sequence alignment on multi-gpus. *Computational biology and chemistry*, 58:62–68, 2015.

Irace, M. *Patient Loyalty in Hospital Choice: Evidence from New York*. PhD thesis, University of Chicago, Becker Friedman Institute for Economics Working Paper, 2018.

Katoh, K., Rozewicki, J., and Yamada, K. D. Mafft online service: multiple sequence alignment, interactive sequence choice and visualization. *Briefings in Bioinformatics*, 2017.

Kececioglu, J. The maximum weight trace problem in multiple sequence alignment. In *Annual Symposium on Combinatorial Pattern Matching*, pages 106–119. Springer, 1993.

Kececioglu, J. D., Lenhof, H.-P., Mehlhorn, K., Mutzel, P., Reinert, K., and Vingron, M. A polyhedral approach to sequence alignment problems. *Discrete applied mathematics*, 104(1):143–186, 2000.

Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P., and Charnois, T. A global constraint for mining sequential patterns with gap constraint. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 198–215. Springer, 2016.

Kemmar, A., Lebbah, Y., Loudni, S., Boizumault, P., and Charnois, T. Prefix-projection global constraint and top-k approach for sequential pattern mining. *Constraints*, 22(2):265–306, 2017.

Keogh, E., Chu, S., Hart, D., and Pazzani, M. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*, pages 1–21. World Scientific, 2004.

Kimoto, T., Asakawa, K., Yoda, M., and Takeoka, M. Stock market prediction system with modular neural networks. In *1990 IJCNN international joint conference on neural networks*, pages 1–6. IEEE, 1990.

Kinable, J., Cire, A. A., and van Hoeve, W.-J. Hybrid optimization methods for time-dependent sequencing problems. *European Journal of Operational Research*, 259(3):887–897, 2017.

Klayman, J. and Ha, Y.-w. Hypothesis testing in rule discovery: Strategy, structure, and content. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(4):596, 1989.

Kotsiantis, S. and Kanellopoulos, D. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

Lambiotte, R., Rosvall, M., Schaub, M., Scholtes, I., and Xu, J. Beyond graph mining: Higher-order data analytics for temporal network data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'19, 05 2018.

Le Bras, Y., Lenca, P., and Lallich, S. On optimal rule mining: A framework and a necessary and sufficient condition of antimonotonicity. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 705–712. Springer, 2009.

Lee, C.-Y. Representation of switching circuits by binary-decision programs. *Bell System Technical Journal*, 38(4):985–999, 1959.

Leigh, W., Frohlich, C. J., Hornik, S., Purvis, R. L., and Roberts, T. L. Trading with a stock chart heuristic. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(1):93–104, 2007.

Leung, C. K.-S., Jiang, F., Sun, L., and Wang, Y. A constrained frequent pattern mining system for handling aggregate constraints. In *Proceedings of the 16th International Database Engineering & Applications Sysmposium*, pages 14–23. ACM, 2012.

Lin, M.-Y. and Lee, S.-Y. Efficient mining of sequential patterns with time constraints by delimited pattern growth. *Knowledge and Information Systems*, 7(4):499–514, 2005.

Lin, W., Alvarez, S. A., and Ruiz, C. Collaborative recommendation via adaptive association rule mining. *Data Mining and Knowledge Discovery*, 6:83–105, 2000.

Lin, W., Alvarez, S. A., and Ruiz, C. Efficient adaptive-support association rule mining for recommender systems. *Data mining and knowledge discovery*, 6(1):83–105, 2002.

Linsay, P. S. An efficient method of forecasting chaotic time series using linear interpolation. *Physics Letters A*, 153(6-7):353–356, 1991.

Lipton, Z. C., Berkowitz, J., and Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Liu, J. N. and Kwong, R. W. Automatic extraction and identification of chart patterns towards financial forecast. *Applied Soft Computing*, 7(4):1197–1208, 2007.

Loekito, E. and Bailey, J. Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 307–316. ACM, 2006.

Loekito, E. and Bailey, J. Are zero-suppressed binary decision diagrams good for mining frequent patterns in high dimensional datasets? In *Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70*, pages 139–150. Australian Computer Society, Inc., 2007.

Loekito, E., Bailey, J., and Pei, J. A binary decision diagram based approach for mining frequent subsequences. *Knowledge and Information Systems*, 24(2):235–268, 2010.

Logan, B. Music recommendation from song sets. In *ISMIR*, pages 425–428, 2004.

Maddala, G. S. *Limited-dependent and qualitative variables in econometrics*. Cambridge university press, 1986.

Magis, C., Taly, J.-F., Bussotti, G., Chang, J.-M., Di Tommaso, P., Erb, I., Espinosa-Carrasco, J., and Notredame, C. T-coffee: tree-based consistency objective function for alignment evaluation. *Multiple Sequence Alignment Methods*, pages 117–129, 2014.

Mallick, B., Garg, D., and Grover, P. S. Constraint-based sequential pattern mining: a pattern growth algorithm incorporating compactness, length and monetary. *Int. Arab J. Inf. Technol.*, 11(1):33–42, 2014.

Martin, R. K. Generating alternative mixed-integer programming models using variable redefinition. *Operations Research*, 35(6):820–831, 1987.

Martin, R. K., Rardin, R. L., and Campbell, B. A. Polyhedral characterization of discrete dynamic programming. *Operations research*, 38(1):127–138, 1990.

Martinez, J., Black, M. J., and Romero, J. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017.

Masseglia, F., Poncelet, P., and Teisseire, M. Efficient mining of sequential patterns with time constraints: Reducing the combinations. *Expert Systems with Applications*, 36(2):2677–2690, 2009.

McDaniel, D. and Devine, M. A modified benders' partitioning algorithm for mixed integer programming. *Management Science*, 24(3):312–319, 1977.

McFadden, D. et al. Conditional logit analysis of qualitative choice behavior, 1973.

McGuire, T. G., Newhouse, J. P., Normand, S.-L., Shi, J., and Zuvekas, S. Assessing incentives for service-level selection in private health insurance exchanges. *Journal of Health Economics*, 35: 47–63, 2014.

Molnar, C. et al. Interpretable machine learning: A guide for making black box models explainable. *E-book at< https://christophm. github. io/interpretable-ml-book/>, version dated*, 10, 2018.

Mott, R. Local sequence alignments with monotonic gap penalties. *Bioinformatics (Oxford, England)*, 15(6):455–462, 1999.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

Najafabadi, M. K., Mahrin, M. N., Chuprat, S., and Sarkan, H. M. Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data. *Computers in Human Behavior*, 67:113–128, 2017.

Needleman, S. B. and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

Negrevergne, B. and Guns, T. Constraint-based sequence mining using constraint programming. In *Proceedings of CPAIOR*, volume 9075 of *LNCS*, pages 288–305. Springer, 2015.

Nijssen, S. and Zimmermann, A. Constraint-based pattern mining. In *Frequent pattern mining*, pages 147–163. Springer, 2014.

Nuin, P. A., Wang, Z., and Tillier, E. R. The accuracy of several multiple sequence alignment programs for proteins. *BMC bioinformatics*, 7(1):471, 2006.

Pang, X., Zhou, Y., Wang, P., Lin, W., and Chang, V. An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, pages 1–21, 2018.

Pei, J., Han, J., and Wang, W. Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems*, 28(2):133–160, 2007.

Pyun, G., Yun, U., and Ryu, K. H. Efficient frequent pattern mining based on linear prefix tree. *Knowledge-Based Systems*, 55:125–139, 2014.

Rather, A. M., Agarwal, A., and Sastry, V. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6):3234–3241, 2015.

Raval, D. and Rosenbaum, T. Why do previous choices matter for hospital demand? decomposing switching costs from unobserved preferences. *Review of Economics and Statistics*, 100(5):906–915, 2018.

Raval, D., Rosenbaum, T., and Wilson, N. Industrial reorganization: Learning about patient substitution patterns from natural experiments, 2016.

Reinert, K., Lenhof, H.-P., Mutzel, P., Mehlhorn, K., and Kececioglu, J. D. A branch-and-cut algorithm for multiple sequence alignment. In *Proceedings of the first annual international conference on Computational molecular biology*, pages 241–250. ACM, 1997.

Riondato, M. and Upfal, E. Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 25–41. Springer, 2012.

Robinson, J. C. and Gardner, L. B. Adverse selection among multiple competing health maintenance organizations. *Medical Care*, pages 1161–1175, 1995.

Rozenshtein, P. and Gionis, A. Mining temporal networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'19, pages 3225–3226, 2019. ISBN 978-1-4503-6201-6.

Sagi, O. and Rokach, L. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. *Sensitivity analysis in practice: a guide to assessing scientific models*, volume 1. Wiley Online Library, 2004.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., and Soman, K. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647. IEEE, 2017.

Shepard, M. Hospital network competition and adverse selection: Evidence from the massachusetts health insurance exchange. Technical report, National Bureau of Economic Research, 2016.

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, 7(1):539, 2011.

Smith, T. F. and Waterman, M. S. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

Soulet, A. and Crémilleux, B. Exploiting virtual patterns for automatically pruning the search space. In *International Workshop on Knowledge Discovery in Inductive Databases*, pages 202–221. Springer, 2005.

Suchacka, G. and Chodak, G. Using association rules to assess purchase probability in online stores. *Information Systems and e-Business Management*, 15(3):751–780, 2017.

Thompson, J. D., Linard, B., Lecompte, O., and Poch, O. A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PloS one*, 6 (3):e18093, 2011.

Van den Oord, A., Dieleman, S., and Schrauwen, B. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

Vayena, E., Blasimme, A., and Cohen, I. G. Machine learning in medicine: Addressing ethical challenges. *PLoS medicine*, 15(11):e1002689, 2018.

Vingron, M. and Waterman, M. S. Sequence alignment and penalty choice: Review of concepts, case studies and implications. *Journal of molecular biology*, 235(1):1–12, 1994.

Wang, L. and Jiang, T. On the complexity of multiple sequence alignment. *Journal of computational biology*, 1(4):337–348, 1994.

Wang, X., Rosenblum, D., and Wang, Y. Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 99–108. ACM, 2012.

Wegener, I. *Branching programs and binary decision diagrams: theory and applications*, volume 4. SIAM, 2000.

Wu, K.-P., Wu, Y.-P., and Lee, H.-M. Stock trend prediction by using k-means and aprioriall algorithm for sequential chart pattern mining. *J. Inf. Sci. Eng.*, 30(3):669–686, 2014.

Wu, Y., Liu, L., Pu, C., Cao, W., Sahin, S., Wei, W., and Zhang, Q. A comparative measurement study of deep learning as a service framework. *IEEE Transactions on Services Computing*, 2019.

Yen, I. E.-H., Lin, X., Zhang, J., Ravikumar, P., and Dhillon, I. A convex atomic-norm approach to multiple sequence alignment and motif discovery. In *International Conference on Machine Learning*, pages 2272–2280, 2016.

Zaki, M. J. Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 422–429. ACM, 2000.