2016

# PREFERENCES: OPTIMIZATION, IMPORTANCE LEARNING AND STRATEGIC BEHAVIORS

Ying Zhu
*University of Kentucky*, isaswing@gmail.com
Digital Object Identifier: http://dx.doi.org/10.13023/ETD.2016.197

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Ying Zhu, Student

Dr. Miroslaw Truszczynski, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

# PREFERENCES: OPTIMIZATION, IMPORTANCE LEARNING AND STRATEGIC BEHAVIORS

---

DISSERTATION

---

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Ying Zhu
Lexington, Kentucky

Director: Dr. Mirosław Truszczyński, Professor of Computer Science
Lexington, Kentucky 2016

ABSTRACT OF DISSERTATION

# PREFERENCES: OPTIMIZATION, IMPORTANCE LEARNING AND STRATEGIC BEHAVIORS

Preferences are fundamental to decision making and play an important role in artificial intelligence. Our research focuses on three group of problems based on the preference formalism *Answer Set Optimization* (ASO) [27]: preference aggregation problems such as computing optimal (near optimal) solutions, strategic behaviors in preference representation, and learning ranks (weights) for preferences.

In the first group of problems, of interest are optimal outcomes, that is, outcomes that are optimal with respect to the preorder defined by the preference rules. In this work, we consider computational problems concerning optimal outcomes. We propose, implement and study methods to compute an optimal outcome; to compute another optimal outcome once the first one is found; to compute an optimal outcome that is similar to (or, dissimilar from) a given candidate outcome; and to compute a set of optimal answer sets each significantly different from the others. For the decision version of several of these problems we establish their computational complexity.

For the second topic, the strategic behaviors such as manipulation and bribery have received much attention from the social choice community. We study these concepts for preference formalisms that identify a set of optimal outcomes rather than a single winning outcome, the case common to social choice. Such preference formalisms are of interest in the context of combinatorial domains, where preference representations are only approximations to true preferences, and seeking a single optimal outcome runs a risk of missing the one which is optimal with respect to the actual preferences. In this work, we assume that preferences may be ranked (differ in importance), and we use the Pareto principle adjusted to the case of ranked preferences as the preference aggregation rule. For two important classes of preferences, representing the extreme ends of the spectrum, we provide characterizations of situations when manipulation and bribery is possible, and establish the complexity of the problem to decide that.

Finally, we study the problem of learning the importance of individual preferences in preference profiles aggregated by the ranked Pareto rule or positional scoring rules. We provide a polynomial-time algorithm that finds a ranking of preferences such that the ranked profile correctly decided all the examples, whenever such a ranking exists. We

also show that the problem to learn a ranking maximizing the number of correctly decided examples is NP-hard. We obtain similar results for the case of weighted profiles.

KEYWORDS: Artificial Intelligence, Preference Reasoning and Learning, Answer Set Optimization, Manipulation and Bribery

Author's signature: Ying Zhu

Date: May 2, 2016

# PREFERENCES: OPTIMIZATION, IMPORTANCE LEARNING AND STRATEGIC BEHAVIORS

By
Ying Zhu

Director of Dissertation: Mirosław Truszczyński

Director of Graduate Studies: Mirosław Truszczyński

Date: May 2, 2016

ACKNOWLEDGMENTS

Though only my name appears on the cover of this dissertation, I would never have been able to finish it without the support from many people.

I would like to express my deepest gratitude to my advisor, Dr. Mirosław Truszczyński. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time offered guidance and pushed me in the right direction. His patience and encouragement helped me overcome many difficulties and finish this dissertation. I would also like to thank my committee members, Dr. Samson Cheung, Dr. Judy Goldsmith and Dr. Jerzy Jaromczyk, for their invaluable feedback and comments over the years. I am also grateful to my colleague Xudong Liu who was always willing to help and give his best suggestions, and to my TA supervisor Debby Keen who showed me how to be a good instructor.

I am grateful to all former and current faculty and staff at the University of Kentucky for their various forms of support during my graduate study. Many thanks to colleagues in the lab and my friends who made my graduate experience has been one that I will cherish forever.

Last and most to my family and boyfriend, for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

## Chapter 1 Introduction

Preferences appear in every scenario requiring a decision to be made among multiple choices. When choosing a restaurant to have dinner at, people consider the menu, price, location, opening hours and table size. When selecting a college to attend, students take into account the rank of the college, the reputation of the department, its location, the faculty and the cost. When buying a house people may inspect the location, the price, the size, whether it has a garage, and how large the backyard is.

Preferences are fundamental to decision making and play an important role in artificial intelligence (AI), economics, operations research and databases. In AI, preferences play a role whenever the decision support is needed in, recommendation systems, multi-agent systems, product configuration, and planning. For example, if the online retailers understand a customer's preferences, they can recommend the most desirable products to him or her, and robot vacuums, like the Roomba, can plan an optimal cleaning path if they have the user's preferences.

There are many important questions about preferences. For instance, how to collect preferences from people? How to represent preferences accurately and concisely with a model which is easy for computation? How to compute optimal choices according to the preferences? How to construct people's preferences based on the observation of their behaviors? Given that preferences may come with conditions, from multiple agents, may concern combinatorial domains, and have different importance levels, preference representation, reasoning and learning are nontrivial.

There are two typical models for preferences. In one model, preferences are represented by utility functions which map the feasible choices of the decision making problem to numeric values. In the other, preferences are partial orders over feasible choices determining whether one choice is better than another. People rarely express their preferences

by utility functions especially when there are a large number of alternatives over multiple attributes. Moreover, in most cases it is not necessary to gather the quantitative information by how much one choice is better than another. Instead of rating all choices directly, it is normally more natural for people to provide their preferences in a qualitative way. For instance, given two alternatives, it is usually not difficult for people to answer which one is better (but not necessarily, by how much one is better than the other). However, with a large number of alternatives, it requires too many pairwise comparisons to express the entire preference order explicitly. In many situations, the set of alternatives has a *combinatorial structure*. That is, outcomes are described in terms of values they assume on several attributes (or variables). Thus, for decision making, each alternative is uniquely characterized by a vector of the values these attributes take. The size of combinatorial domains grows exponentially with the number of variables. Therefore, representing preferences by listing all alternatives with their utility or rank in the preference order is infeasible. To circumvent the problem of size, one resorts to implicit representation languages that aim to provide concise and intuitive approximations to agents' true preferences. The survey by Domshlak et al. [40] and the monograph by Kaci [69] discuss several of them.

Many of these approaches express preferences by describing desirable properties of outcomes in terms of the values of attributes. For example, when ordering a dinner, people may express their preference stating that dinners with beef are better than dinners with fish when having beer. When planning a vacation, they may like vacations in London more than vacations in Paris. If it is raining, they may prefer reading at home to any outdoor activities. Such preference statements can be represented with graphical or logical representations.

One well known graphical representation is offered by conditional preference networks (CP-nets) [13, 12] with its extensions and variants [14, 11, 15]. CP-nets represent preference relations in a compact, intuitive and structured manner using conditional *ceteris paribus* (all else being equal) preference statements. Each preference statement expresses the user's preference over a single variable. A statement "I prefer $X = x_1$ to $X = x_2$" means

that given two alternatives with the same value for every attribute except for $X$, the user prefers the one with $x_1$ for $X$ to the one with $x_2$. If these two alternatives have different values on some other variables, they cannot be compared based on this single preference statement. A conditional statement "I prefer $X = x_1$ to $X = x_2$ if $Y = y_1$" is interpreted exactly as above but applies only to the alternatives that have the value $y_1$ for $Y$. CP-nets consist of a directed graph with attributes (variables) as nodes that describes the parent dependence between variables. Each node (variable) in the graph is associated with a conditional preference table expressing the local preferences conditioned on values taken by parent variables.

The basic idea of logic-based representations is to distinguish the outcomes satisfying a formula expressing people's desirable properties from the outcomes violating it. This idea can be found in formalisms such as *penalty logic* [85], *possibilistic logic* [41], and *discrimin* and *lexmin logics* [8]. The goals with different priorities can be modeled by assigning ranks or weights to the formulas. When comparing two alternatives, the penalty logic compares the sum of weights for the violated goals, while the possibilistic logic compares the maximum weight of the violated goals. The leximin ordering compares the cardinalities of satisfied goals at each level of priority, and the discrimin ordering considers the sets of falsified goals.

Our work is based on the logic-based preference formalism called *Answer Set Optimization* (ASO) [27] introduced by Brewka, Niemelä and Truszczyński in 2003. The formalism is a combination of answer set programming [80] and qualitative preferences similar to those used in possibilistic logic. The ASO programs contain two parts, a generator which is an answer set program defining the feasible solutions and a selector which is a set of preference rules. The preference program compares the solutions based on their satisfaction degree on the preference rules, using to this end the Pareto principle. The ASO programs also allow preference rules with ranks that model importance of preferences in an intuitive and accurate way. In our work, we also propose and study a weighted version of ASO that

uses numerical values to represent the importance of preferences, and applies voting rules to aggregate preferences. Our research focuses on three group of problems in the ASO formalism: preference aggregation problems such as computing optimal (near optimal) solutions, strategic behaviors in preference representation, and learning ranks (weights) for preferences. Some basic optimization problems for ASO formalism were studied in the original paper that introduced it [27]. Our results concern problems not studied before and alternative computational methods.

### 1.1 Motivation

In our work, we focus on the following three areas of research concerning preference representation and reasoning: optimization of preferences, including computing *sets* of optimal outcomes with properties useful in decision making; strategic behaviors when preferences are aggregated by the Pareto rule, possibly in its ranked version; and learning the importance of preference rules when a user or a group are making decisions.

Preference aggregation is a fundamental aspect of preference reasoning. In many practical problems, hard constraints still leave many feasible solutions and a mechanism is needed for selecting those which have desirable properties. A typical approach consists of eliciting from the user her preferences on the space of solutions, and returning to the user only those solutions that “score" high on the user’s preference criteria. In most cases, only the solutions that are optimal, with respect to these criteria, are of interest.

To help the user make that selection, one needs computational support for the key preference reasoning tasks such as computing an optimal solution, and computing an alternative optimal solution once the first optimal solution was found. In some cases, the user can give examples of what might be desirable or what has to be avoided. Given such examples, the reasoning support system should return optimal solutions that come close to examples that are desirable (or, are dissimilar from those that are undesirable). The problem of computing similar/dissimilar solutions was identified as important in the setting of problems defined

by hard constraints, and was studied for answer sets of programs by Eiter, Erdem, Erdogan and Fink [42]. Another computational reasoning task arises when preference optimization still leaves a significant number of choices. This is often the case when outcomes come from combinatorial domains, which are almost always large, and when outcomes subject to conflicting preferences are aggregated by weak aggregation mechanisms such as the Pareto principle. In such cases, to help the user make the final selection, it is useful to present her with a small and diverse sample of optimal outcomes.

The second part of our work studies the strategic behaviors in preference aggregation. In a common preference reasoning scenario, a group of agents is presented with a set of configurations or outcomes. These outcomes come from a combinatorial domain, that is, they are characterized by several multivalued attributes and are represented as tuples of attribute values. Each agent has her individual preferences on the outcomes. The problem is to aggregate these preferences, that is, to define a “group" preference relation or, at the very least, to identify outcomes that could be viewed by the entire group as good consensus choices. This scenario has received much attention in the AI and decision theory communities [40, 64, 69].

In this scenario, the goal is to aggregate diverse preferences of a group of agents into a single consensus preference ordering on outcomes or, for some applications, into a set of consensus optimal outcomes. This process may cause agents behave strategically. They may misrepresent their true preferences, or coerce others to do so, in order to secure consensus preference aggregation outcomes that are more favorable to them. Since such strategic behaviors may have a negative effect on the group overall welfare, it is important to characterize situations when they can occur and to understand how to defend against them. This is the second topic in our work.

The third problem we study concerns preference learning. The problems discussed above are based on the preferences which are total orders over the candidates. However, in many cases, it is impractical to collect complete preferences from people. Instead, one

might try to *learn* the preferences based on observations. That problem has received much attention [60, 59, 20, 33, 39]. In some cases, preferences have different importance. For example, people may think that the safety is more important than the appearance when buying a car. Therefore, if their preferences on safety and appearance conflict, they may want to consider the preferences on safety first. For group decision making, it is common that agents are not equally important. In a job interview, the interviewers may include recruiters and various-level managers. The opinion from some of them may play a decisive role on whether a candidate can get the job. In our work, we assume that individual preferences are known but not their importance. Our goal is to develop methods of learning the importance. Similar problems are considered in the social choice community. An example is the possible winner problem in an election with uncertain weights [7]. However, the problem of learning the importance of preferences has not been studied so far. We propose a formal statement of this problem and present several results.

### 1.2 Contributions

In this section, we outline the results we obtained in the three areas discussed above.

Basic optimization problems were considered in the original paper on answer set optimization (ASO) [27], such as deciding whether a given outcome is optimal and how to compute an optimal outcome. The first group of our contributions concerns optimization problems subject to additional conditions, their computational complexity, and effective computing methods to solve them. These computational methods are built on the basis of answer set programming [81, 83] and answer-set programming tools.

In particular, we show that for ASO programs, the problems of the existence of optimal answer sets that are similar to (respectively, dissimilar from) a given interpretation are $\Sigma_2^p$-complete. Further, we show that the problem of deciding the existence of $k$ optimal solutions at distance at least $d$ from each other is $\Sigma_2^p$-complete, assuming that $d$ is part of the input and $k$ is fixed. If $d$ is fixed, the problem is in $\Delta_2^p$. Second, we design techniques

to solve optimization problems for ASO programs. To this end, we propose two main approaches to support optimization tasks in ASO. Both take advantage of answer set solvers. One approach is fully declarative. It relies on representations of optimization problems as disjunctive logic programs. The other approach reduces optimization tasks to be solved to an iterative process of answer set computations on specially designed answer set programs. Finally, we propose methods to generate random instances of ASO programs for testing and present experimental results. The results provide insights into the feasibility of the two approaches and suggest a class of challenging bench-marks for disjunctive ASP solvers. We presented this work on the international conference of Logic Programming and Nonmonotonic Reasoning 2013 and it was published in the proceedings of that conference [96].

In the second part, we study manipulation and bribery problems in preference formalisms which have attracted a substantial amount of attention from the social choice community. Preference formalisms identify a set of optimal outcomes rather than a single winning outcome, the case common to social choice. Such formalisms are of interest in the context of combinatorial domains, where preference representations are only approximations to true preferences, and seeking a single optimal outcome runs a risk of missing the one which is optimal with respect to the actual preferences. In this work, we assume that preferences may be ranked (differ in importance), and we use the Pareto principle adjusted to the case of ranked preferences as the preference aggregation rule. For two important classes of preferences, representing the extreme ends of the spectrum, we provide characterizations of situations when manipulation and bribery is possible, and establish the complexity of the problem to decide that. Our results shows that deciding the existence of manipulation or bribery is not tractable. We presented this work on the international conference of Algorithmic Decision Theory 2015 and it was published in the proceedings of that conference [97].

In preference learning, the third area of our study, we proposed a polynomial time algo-

rithm to decide, given a preference profile and a set of pairwise ordered candidates (examples), whether the preferences can be ranked such that the ranked profile orders candidates consistently with the examples. The algorithm computes a rank assignment with minimal number of ranks for the profile if it is consistent with the examples. We also consider the same problem in the voting theory setting with positional scoring rules as the aggregation mechanism. We proved that it can be decided in polynomial time whether, given an election and a set of examples, there exists a weight assignment such that the weighted election is consistent with all ordered candidates. For both preference profiles and voting profiles, we proved that the problem to decide whether a rank/weight assignment exists such that the ranked/weighted profile is consistent with at least $k$ examples is NP-complete.

### 1.3 Outline

Here is a brief outline of the thesis. Before presenting our work, we start with a brief discussion of the related work in Chapter 2 and provide the technical preliminaries in Chapter 3. The preliminaries mainly discuss the preference formalism of Answer Set Optimization, and also provide a general introduction of the computational complexity. Our three research topics are then presented in Chapter 4, 5 and 6. Chapter 4 introduces our work on preference aggregation and presents results on the computational complexity and methods of finding optimal and near optimal solutions based on answer set optimization. Chapter 5 presents our work on preference misrepresentation, specifically, the manipulation and bribery problems in the setting of preference with combinatorial domain in a multi-agent scenario. In Chapter 6, we introduce our work on learning ranks in a preference setting and learning weights in the social choice voting setting when the preference profile and some examples are given. Finally, the summary of our contributions and potential future work are discussed in Chapter 7.

# Chapter 2 Related Work

Our work is based on the logic-based preference formalism *Answer Set Optimization* (ASO) [27] introduced by Brewka, Niemelä and Truszczyński in 2003 which is a combination of answer set programming [62] and qualitative preferences similar to those used in possibilistic logic. The answer set program defines the feasible solutions and the qualitative preferences define the preference ordering on solutions. The optimal outcome of an ASO problem is a non-dominated answer set, which means there is no answer set strictly better than it exists. Our research addresses three aspects of the ASO formalism: preference aggregation problems, strategic behaviors in preference representation, and learning ranks (weights) for preferences.

In this section, we introduce the related work for logic-based preference representations, preferences in answer set programming, preference learning for other preference representations, like lexicographic order and CP-nets, and strategic behaviors in voting.

## 2.1 Logic-based Preference Representation

The basic idea of logical preference statements is to discriminate between models satisfying formulas expressing the desirable properties and models violating them. This idea can be found in formalisms such as *penalty logic* [85], *possibilistic logic* [41], and *discrimin and lexmin logics* [8]. These languages represent preferences by a set of weighted formulas of the form $T = \{(\phi_i, \alpha_i) \mid i = 1, \ldots, n\}$, where $\phi_i$ is a propositional logic formula and $\alpha_i$ is a numerical value representing the importance of $\phi_i$. For penalty logic, $\alpha_i$ is an integer representing the penalty of falsifying the preference formula $\phi_i$. The outcomes are compared based on the sum of penalties they have. For the other three languages, $\alpha_i$, now restricted to the interval $(0, 1]$, represents the importance of $\phi_i$. The larger $\alpha_i$ is, the more important $\phi_i$ is. Possibilistic logic compares outcomes based on the most important formula

they violate. The leximin logic compares the outcomes according to the cardinalities of falsified formulas at each level of importance. The discrimin logic compares the outcomes based on the sets of falsified formulas at each level of importance.

The specific importance values are immaterial in these three languages and the importance information can be represented by ranks. Preferences can be grouped into levels based on their importance, each level gathering formulas of the same importance, and different levels consisting of formulas with different importance. Thus, theories with $k$ importance values can be written as:

$$\varphi_1 > \varphi_2 > \cdots > \varphi_k,$$

where $\varphi_1$ is a set of formulas with the highest importance value, $\varphi_2$ is a set of formulas with the next highest importance value, etc.

In the possibilistic logic, the conjunction of formulas in $\varphi_i$ is denoted by $\Phi_i$. The complete relation over all outcomes is represented as follows:

$$\Phi_1 \wedge \cdots \wedge \Phi_k > \Phi_1 \wedge \cdots \wedge \Phi_{k-1} > \cdots > \Phi_1 > \neg\Phi_1.$$

This means that the outcomes satisfying all $\Phi_i$s are preferred to the outcomes satisfying only $\Phi_1$ to $\Phi_{k-1}$. And the outcomes satisfying $\Phi_1$ to $\Phi_{k-1}$ are better than the outcomes satisfying only $\Phi_1$ to $\Phi_{k-2}$. The position of an outcome in this order is determined by the most important formula it violates.

For lexmin logic, given two outcomes $\omega$ and $\omega'$, we say $\omega$ is preferred to $\omega'$ if and only if there is $\varphi_l$ such that

$$\mid \{\phi_i \mid \phi_i \in \varphi_j, \omega \not\models \phi_i\} \mid = \mid \{\phi_i \mid \phi_i \in \varphi_j, \omega' \not\models \phi_i\} \mid$$

for every $j < l$ and

$$\mid \{\phi_i \mid \phi_i \in \varphi_l, \omega \not\models \phi_i\} \mid < \mid \{\phi_i \mid \phi_i \in \varphi_l, \omega' \not\models \phi_i\} \mid .$$

The lexmin logic determines the order between outcomes by the number of falsified formulas on each importance level. The discrimin logic looks at the falsified formulas

Table 2.1: An example of lexmin logic

| Outcomes | # of formulas falsified | |
|---|---|---|
| | $\{beef\}$ | $\{soup\}$ |
| $S_1 = \{beef, soup\}$ | 0 | 0 |
| $S_2 = \{beef, salad\}$ | 0 | 1 |
| $S_3 = \{fish, soup\}$ | 1 | 0 |
| $S_4 = \{fish, salad\}$ | 1 | 1 |

themselves. Given two outcomes $\omega$ and $\omega'$, we say $\omega$ is preferred to $\omega'$ in discrimin logic if and only if there is $\varphi_l$ such that

$$\{\phi_i \mid \phi_i \in \varphi_j, \omega \not\models \phi_i\} = \{\phi_i \mid \phi_i \in \varphi_j, \omega' \not\models \phi_i\}$$

for every $j < l$ and

$$\{\phi_i \mid \phi_i \in \varphi_l, \omega \not\models \phi_i\} \subset \{\phi_i \mid \phi_i \in \varphi_l, \omega' \not\models \phi_i\}.$$

Let us consider an example to illustrate these languages. Assume an agent needs to make a decision about what to have for lunch, and she is given four choices:

$$S_1 = \{beef, soup\}, S_2 = \{beef, salad\},$$
$$S_3 = \{fish, soup\}, S_4 = \{fish, salad\}.$$

She has two preferences $\phi_1 : beef$ and $\phi_2 : soup$ where $\phi_1$ is more important than $\phi_2$. The preferences with two importance levels can be represented as $\{beef\} > \{soup\}$. In the possibilistic logic, the complete preference order over outcomes is

$$beef \wedge soup > beef > \neg beef.$$

Thus $S_1 \succ S_2 \succ S_3 \approx S_4$.[1] In the lexmin logic, the number of falsified formulas for each outcome is shown in Table 2.1 inducing a total order of outcomes: $S_1 \succ S_2 \succ S_3 \succ S_4$.

For the discrimin logic, the falsified formulas for each outcome are shown in Table 2.2 inducing a partial order of outcomes: $S_1 \succ S_2 \mid S_3 \succ S_4$.

[1] We denote strict preference by $\succ$, equivalence by $\approx$, and incomparability by $|$.

Table 2.2: An example of discrimin logic

| Outcomes | # of formulas falsified | |
|---|---|---|
| | $\{beef\}$ | $\{soup\}$ |
| $S_1 = \{beef, soup\}$ | $\{\}$ | $\{\}$ |
| $S_2 = \{beef, salad\}$ | $\{\}$ | $\{soup\}$ |
| $S_3 = \{fish, soup\}$ | $\{beef\}$ | $\{\}$ |
| $S_4 = \{fish, salad\}$ | $\{beef\}$ | $\{soup\}$ |

The ASO formalism represents conditional preferences by expressions constructed of propositional formulas. In a way similar to that of possibilistic logic. Preferences are aggregated by the Pareto principle. As ASO programs are the main focus of our work we discuss them in detail in Chapter 3.

## 2.2 Preferences in Answer Set Programming

In recent years, a large number of papers have studied preferences in answer set programming [62]. Some of the existing answer set solvers have (numerical) optimization facilities. For instance, *Smodels* with weight constraints has maximize and minimize statements operating on weights of atoms in answer sets [88] and *dlv* has weak constraints [29]. Other approaches allow for expressing various types of preferences among rules [37, 43]. Unlike in the approaches introducing preferences over rules and literals, Brewka [22, 26] investigated the problem of specifying preference over alternatives in a disjunction by introducing a new connective called *ordered disjunction*. Several generic optimization problems for formalisms extended with the ordered disjunction were discussed in [23].

In another paper [24], Brewka proposed a preference description language which describes complex preferences among answer sets. The language generalizes the rule based preferences of ASO in two respects:

- it allows us to combine qualitative and numerical penalty based preference information within a single framework, and
- it allows us to use different preference aggregation methods.

A preference handling framework *asprin* introduced in [25], allows users to specify ways to aggregate preferences. It provides *#minimize* directives [88], weak constraints [76], ASO [27], and the language for specifying preferences in planning domains [89]. Another framework closely related to ASO programs is that of *qualitative optimization problems* [47]. It provides an abstract view on preference formalisms that model hard constraints and preferences on feasible outcomes.

## 2.3 Preference Learning

With the increased attention on preferences, learning and predicting preferences has become an active research topic in areas such as machine learning, data mining, and recommender systems. Generally speaking, the learning problem is to extract a preference structure by observing the choices the user makes selecting from multiple alternatives. The problems of preference learning can be formalized within different settings based on the preference representation and the type of information provided to the learning process.

Since the preferences are generally represented by utility functions and binary relations, two general approaches to preference learning have been proposed: to learn utility functions [20, 30, 31, 65, 67], and to learn preference relations modeling preference orders [59, 68, 34].

The latter approach typically consists of learning a concise implicit representation of the preference relation. It assumes a specific language for expressing preference orders. The languages for which learning methods were investigated include CP-nets [12] (learning dependencies and conditional preference tables), lexicographic preference model [56], lexicographic preference trees [10] and forests of lexicographic trees. To learn a lexicographic preference model, Dombi et al. [39] proposed an algorithm which guides the user through a sequence of queries involving test examples. Schmitt and Martignon [87] introduced a greedy variable permutation algorithm guaranteeing to find a lexicographic preference model that is compatible with the learning examples, if one exists. Although it

takes polynomial time to determine whether there exists a lexicographic preference model consistent with a set of examples, they also showed that the corresponding optimization problem of minimizing the disagreement is NP-hard. Other work shows results of learning "best guess" lexicographic preference models through approximation algorithms and learning lexicographic preference trees [10, 58, 94, 95]. For CP-nets, Chevaleyre [33] studied both passive and active learning, Koriche and Zanuttini [73, 72] proposed an active learning algorithm from consistent examples, and Liu et al. [77] proposed an approach for learning CP-nets from inconsistent examples. An algorithm for generating an acyclic CP-net entailing all examples is introduced in [38]. Other results show that the problem of learning CP-nets is intractable even under some simplifying assumptions [75, 74, 66].

## 2.4 Strategic Voting in Social Choice Theory

Social choice theory [1, 2] studies how to aggregate individual preferences, opinions or welfares to achieve a group decision or social welfare. For instance, people use elections to select their leader, establish the laws, and decide the policies. In many settings, the preferences from different people are often not weighted equally.

Aggregating preferences means mapping a collection of preferences relations, each representing the preference of a member of a decision-making group, into a collective preference relation. In many cases, we are only interested in the winner (the most preferred alternative of the consensus preference ordering), or a subset of the most preferred alternatives rather than the complete collective preference. Voting is a general method of preference aggregation [16]. Given a set of candidates and a set of voters with preferences, a voting rule is defined as a function from the preference profiles to candidates that specifies the winner of the election. A positional scoring rule computes a numerical value for each candidate on every preference order based on its position on the preference. Let the number of candidates be $n$. The rule can be defined as a vector of integers $(s_1, \ldots, s_n)$ such that $s_1 \geq s_2 \geq \cdots \geq s_n$. For each preference (vote), a candidate receives score $s_k$ if it is

ranked at position $k$. The winner is the candidate with the maximum sum of scores. Some well known positional scoring rules are:

- Plurality: Each voter gives 1 score to his/her most preferred candidate and 0 to all other candidates $((1,0,\ldots,0))$.
- Veto: Each voter gives 0 score to his/her least preferred candidate and 1 to all other candidates $((1,\ldots,1,0))$.
- k-Approval: Each voter gives 1 score to the top $k$ candidates on the preference order $((1,\ldots,1,0,\ldots,0))$.
- Borda: Each voter assigns scores from $n-1$ (with $n$ the number of candidates) down to 0 to the candidates according to the preference order $((n-1,\ldots,0))$.

Some other voting rules include:

- Maximin: Let $N(c_1,c_2)$ be the number of votes that rank candidate $c_1$ before candidate $c_2$. The score of candidate $c$ is defined as $min_{c'\neq c}N(c,c')$. Candidates are ordered by their scores.
- Copeland: The score of a candidate is the number of candidates it beats in pairwise elections. The candidate with the highest score wins.
- Bucklin: For any candidate $c$ and integer $l$, let $N(c,l)$ be the number of votes that rank $c$ among the top $l$ positions. The score of candidate $c$ is defined as $min\{l : N(c,l) > n/2\}$ where $n$ is the number of votes. Candidates are sorted in ascending order of their scores.

Researchers in social choice theory have studied extensively the properties of these and other voting rules. An important topic in computational social choice is to study the resistance of voting systems to manipulative attacks that seek to influence the election result, such as control, manipulation, and bribery.

Here is an example of manipulation in election with Borda count as the voting rule: given three candidates A, B and C,

$$104 \textit{ voters vote for}: \ A > B > C$$

$$98 \textit{ voters vote for}: \ B > A > C$$

$$7 \textit{ voters vote for}: \ C > B > A.$$

According to the Borda count, the candidate which is positioned at the top receives 2 points, the candidate at position 2 gets 1 point and the last candidate has no point. Therefore the candidate B wins with 307 points, the candidate A gets 306 and C gets 14 points. The voters with vote $A > B > C$ do not get their top choice winning. Some of them may realize that by putting B lower in their vote, they will decrease B's score and may in this way push A to the top. If only one of them changes her vote to $A > C > B$, A and B will be tied with 306 points each. If more than one voter in the first group change their votes to $A > C > B$, A wins right away. However, the danger of manipulation is that if there are many voters vote insincerely, the result of the election can be skewed significantly.

In manipulation, a voter decides to cast a vote which is different from his or her true preference in order to obtain a more desirable outcome. The classical work of Gibbard and Satterthwaite [63, 86] established the main impossibility result stating that for three or more candidates, no reasonable voting rule (not dictatorial and every candidate has a chance to be the winner) is robust to manipulation (or *strategy proof*). However, some researchers argued that one of the key desiderata on the class of rules considered in the Gibbard-Satterthwaite result, the requirement that a rule be *resolute* (that is, always returning a single winner) is in many cases unreasonable [61, 70, 90] and at odds with socially desirable requirements of equal treatment of candidates and voters [90]. This critique opened the door to research of strategy proofness of voting rules that are *irresolute*, that is, may return several winners. Early results identifying situations in which multi-winner rules are strategy proof, as well as those when impossibility results similar to that of Gib-

bard and Satterthwaite still hold, were found by Gärdenfors [61] and Kelly [70]. Additional results along these lines are surveyed by Taylor [90] and Barberà [3]. More recently strategy proofness of irresolute rules have been studied by Brandt [17], Brandt and Brill [18], and Brandt and Geist [19]. It turns out that how preferences on candidates are extended to preferences on *sets* is essential for the possibility of strategic behaviors. In particular, Brandt et al. found several strategy proof irresolute rules for the so-called Kelly, Fishburn and Gärdenfors extensions.

Classical social choice theory focuses on the problem to decide whether attacks are possible. However, researchers realized recently that even if elections are vulnerable to attackers, it may still be difficult to find the proper actions for the attackers, because the computational hardness of identifying them could serve as a barrier against such strategic behaviors.

In control, an election chair tries to control the election result by adding/deleting either candidates or voters. The complexity of this problem was studied first by Bartholdi, Tovey, and Trick [6]. Many researchers obtained results for this problem with many different voting rules in varied settings [52, 51].

The study to discover on which voting rule the manipulation is computationally difficult to execute was also started by Bartholdi, Tovey, and Trick [5], continued by many other researchers [35, 82, 46, 98, 21, 93, 36]. Faliszewski provided a overview of the research on manipulation [54].

In bribery, an external agent with a limited budget attempts to affect the outcome of an election by offering payments to some voters for changing their votes. The computational study of this problem was initiated by Faliszewski, Hemaspaandra, and Hemaspaandra [49] and extended in other papers [53, 45].

Another problem which is quite related to manipulation and bribery is named the possible winner problem introduced by Konczak and Lang [71] and further studied by other researchers [4, 92, 91, 9, 7]. An informal definition of this problem is that given an election

with incomplete information (preferences are partial orders), decide whether it is possible to extend the partial preference orders to complete linear orders so that some specific candidate is a winner. This problem models the situation where we have incomplete information and want to know whether some candidate still has a chance to be the winner. Similarly, a problem named necessary winner studies whether a given candidate is guaranteed to be the winner no matter how the votes are completed [71, 92].

The preference aggregation scenario and the problem of strategic misrepresentation of preferences are similar to strategic voting in social choice theory. In social choice, the concern is to determine a *winner* (sometimes, a strict ordering of the candidates) based on the *votes* cast by a group of *voters*. If we think of voters as agents, of candidates as options, and of votes as preferences, the connection is evident and was noted before [32]. However, the distinguishing aspect of the voting problem considered in social choice is that the number of options (candidates in an election) is *small* and preferences (votes) are specified *explicitly*. The main research goals are to design *voting rules* (procedures to determine a winner or winners based on votes), to identify socially desirable properties that voting rules should have, and to determine which voting rules have which properties. In out work, we study the strategic behaviors in preferences representation, the situations when they can occur and how hard to determine that.

# Chapter 3 Preliminaries

## 3.1 Preferences

A *preference* over a set $X$ of *options* is a total preorder on $X$. We will typically denote is by $\succeq$, possibly annotated with superscripts or subscripts. Each total preorder $\succeq$, determines two associated relations: *strict preference*, denoted by $\succ$, where $x \succ y$ if and only if $x \succeq y$ and $y \not\succeq x$, and *indifference*, denoted by $\approx$, where $x \approx y$ if and only if $x \succeq y$ and $y \succeq x$. The indifference relation $\approx$ is an equivalence relation on $X$ and partitions $X$ into equivalence classes, $X_1, \ldots, X_m$, which we always enumerate from the most to the least preferred. Using this notation, we can describe a total preorder $\succeq$ by the expression

$$\succeq: \quad X_1 \succ X_2 \succ \cdots \succ X_m.$$

For example, a total preorder $\succeq$ on $X = \{a,b,c,d,e,f\}$ such that $a \approx d$, $b \approx e \approx f$ and $a \succ b \succ c$ (these identities uniquely determine $\succeq$) is specified by an expression

$$\succeq: \quad a,d \succ b,e,f \succ c.$$

(we omit braces from the notation specifying sets of outcomes to keep the notation simple). For every $x \in X$, we define the *satisfaction degree* of $x$ in $\succeq$, written $d_{\succeq}(x)$, as the unique $i$ such that $x \in X_i$. We denote the total preorder over options on a preference $p$ by $x \succeq_p y$. Let $p$ be a preference, $x \succeq_p y$ if $d_p(x) \leq d_p(y)$.

A *profile* over $X$ is a set of preferences over $X$. A *ranked* profile is a profile in which every preference $p$ is assigned a positive integer, the *rank* $r(p)$ of $p$. The preferences with a lower rank are more important than the preferences with a higher rank. We write $(P,r)$ for a profile $P$ ranked by a *ranking function* $r$ which maps a preference to a positive integer. An unranked profile is a special case of the ranked profile where all preferences have the same rank.

To obtain the collective preorder over options, the preferences are aggregated by the Pareto principle. For a ranked profile $(P,r)$, and a set of options $X$, we define the preoder $\succeq_{P,r}$ on options by setting $x \succeq_{P,r} y$ if

1. $x \approx_p y$ $(d_p(x) = d_p(y))$ for every preference $p \in P$, or

2. there is a preference $p \in P$ such that $x \succ_p y$ $(d_p(x) < d_p(y))$ and for every preference $p' \in P$ such that $r(p') \leq r(p)$, $x \succeq_{p'} y$ $(d_p(x) \leq d_p(y))$.

The relation $\succeq_{P,r}$ is a preorder and we write $\approx$, $|$, and $\succ$ (preserving annotations) for the corresponding *equivalence*, *incomparability*, and *strict* preference relations. For a ranked profile $(P,r)$, $x \in X$ is equivalent to $y \in X$, written $x \approx_{P,r} y$, if $x \succeq_{P,r} y$ and $y \succeq_{P,r} x$; $x$ is incomparable with $y$, written $x \mid_{P,r} y$, if $x \not\succeq_{P,r} y$ and $y \not\succeq_{P,r} x$; and $x$ is strict Pareto-preferred to $y$, written $x \succ_{P,r} y$, if $x \succeq_{P,r} y$ and $y \not\succeq_{P,r} x$. An option $x \in X$ is *Pareto optimal* in $(P,r)$ if there is no $y \in X$ such that $y \succ_{P,r} x$. We denote the set of all options in $X$ that are Pareto-optimal in $(P,r)$ by $Opt(P)$. Virtually all preference aggregation techniques select "group optimal" elements from those that are Pareto-optimal. From now on, we omit the term "Pareto" when speaking about the preference relation $\succeq_{P,r}$ on $X$ and optimal elements of $X$ determined by this relation, as we do not consider any other preference aggregation principles.

We also use $\succeq_P$ to represent the preorder over options on a ranked profile $(P,r)$ where $r(p) = 1$ for every $p \in P$ (also called an unranked profile or an equally ranked profile).

If a preference is a total strict order over a set of options $X$, it is also known as a *vote* and the options are called as *candidates*. The importance of a vote is represented by a non-negative numerical value which is called as *weight*. A *voting profile* is a set of votes.

### 3.2 Answer Set Optimization

Our work is based on the logic-based preference formalism *Answer Set Optimization* (ASO) [27] introduced by Brewka, Niemelä and Truszczyński in 2003 which is a combination of

answer set programming and conditional preferences. An answer set optimization (ASO) program $P$ over a (propositional) vocabulary $\sigma$ consists of two parts: a *generator* $P_{gen}$ and a *selector* $P_{pref}$ [27]. The generator is a propositional non-disjunctive answer set program[1] or a propositional theory over $\sigma$. We consider the propositional case only and assume all programs are finite.

Propositional interpretations of the vocabulary $\sigma$ represent *candidate outcomes*. The generator of an ASO program $P$ over $\sigma$ represents hard constraints on candidate outcomes. It describes the space of *feasible outcomes* of $P$, which we refer to simply as *outcomes* of $P$. These outcomes are answer sets [62] of the generator, if the generator is represented as a program, or models of the generator, if the generator is represented as a propositional theory. We represent outcomes (answer sets or models) and, more generally, candidate outcomes (interpretations) over a vocabulary $\sigma$, as subsets of $\sigma$.

The choice of the formalism for the generator does not affect our results. In particular, the complexity results we present in the following chapters do not depend on the exact form of the generator as the complexity of model generation task is the same for the two types of generators we allow. Therefore, we often remain vague about how generators are represented. However, in the experimentation part of the Chapter 4, for the sake of concreteness we consider as generators propositional CNF theories. Similarly, for the most part, we use the generic term "outcome" rather than the more specific "answer set" and "model."

The selector of an ASO program $P$ over a vocabulary $\sigma$ is a set of *preference rules* or *preferences* of the form:

$$C_1 > \cdots > C_k \leftarrow a_1, \ldots, a_n, \neg\, b_1, \ldots, \neg\, b_m \tag{3.1}$$

where $a_i$'s and $b_i$'s are atoms over $\sigma$ and $C_i$s are propositional formulas over $\sigma$. The expression $a_1, \ldots, a_n, \neg\, b_1, \ldots, \neg\, b_m$ is understood as the conjunction $a_1 \wedge \cdots \wedge a_n \wedge \neg\, b_1 \wedge \cdots \wedge$

[1] Allowing disjunctive programs as generators is possible, but affects the complexity results as well as the effectiveness of computational methods used.

$\neg\, b_m$. The selector represents preferences or soft constraints of the problem. Informally, the preference rule above reads: among outcomes that satisfy $a_1 \wedge \cdots \wedge a_n \wedge \neg\, b_1 \wedge \cdots \wedge \neg\, b_m$ (that is contain $a_1, \ldots, a_n$ and do not contain any of $b_1, \ldots, b_m$), those that satisfy $C_1$ are preferred over those that satisfy $C_2$, etc.

The formal semantics of preference rules is based on the notion of the *satisfaction degree*. Let $p$ be a preference rule of the form (3.1) and $S$ a candidate outcome. As we defined above, the satisfaction degree of $S$ on $p$ is $d_p(S) = min\{i : S \models C_i\}$.[2] If $S$ does not satisfy $a_1 \wedge \cdots \wedge a_n \wedge \neg\, b_1 \wedge \cdots \wedge \neg\, b_m$ or if it does not satisfy any of the options $C_i$, then $p$ is *irrelevant* to $S$ and the satisfaction degree of $S$ on $p$, $d_p(S)$, is set to 1.[3] Preference rules are aggregated by the Pareto principle as described above. Given an ASO program $P$ and two candidate outcomes $S_1$ and $S_2$, we denote $S_1$ is preferred over $S_2$ on $P_{pref}$ by $S_1 \succeq_P S_2$. We denote the set of all optimal outcomes in $P$ by $Opt(P)$.

To illustrate the ASO formalism, we present the same example used before. Let us assume that $P_{gen}$ is any theory generating 4 outcomes:

$$S_1 = \{beef, soup\},\ S_2 = \{beef, salad\},$$

$$S_3 = \{fish, soup\},\ S_4 = \{fish, salad\}.$$

For example, we can take for $P_{gen}$ an answer set program:

$$1\{beef, fish\}1$$

$$1\{soup, salad\}1$$

which means for any answer set of $P_{gen}$, exactly one atom from each set $\{beef, fish\}$ and $\{soup, salad\}$ is true. Or we can represent $P_{gen}$ as a propositional theory:

$$(beef \vee fish) \wedge \neg(beef \wedge fish) \wedge (soup \vee salad) \wedge \neg(soup \wedge salad).$$

[2]The notation $\models$ represents the standard satisfiability relation from propositional logic (we recall that outcomes are interpretations).

[3]Other ways to treat irrelevance are possible. In addition, the user can always overwrite the default treatment. Indeed, irrelevance is described by the formula $I = \neg(a_1 \wedge \cdots \wedge a_n \wedge \neg\, b_1 \wedge \cdots \wedge \neg\, b_m) \vee \neg(C_1 \vee \ldots \vee C_k)$. The user can "attach" this formula to any of the options (say, replace $C_i$ with $C_i \vee I$) or insert it anywhere in the sequence. Therefore, we do not provide any motivation for the default treatment of irrelevance we use in the paper. Instead, we refer to the paper where ASO was introduced [27].

Assuming $P_{pref} = \{p_1, p_2\}$ where

$$p_1 : beef > fish$$

$$p_2 : soup > salad,$$

the vectors of the satisfaction degrees on preferences $r_1$ and $r_2$ for the four outcomes are $V_{S_1} = (1,1)$, $V_{S_2} = (1,2)$, $V_{S_3} = (2,1)$, $V_{S_4} = (2,2)$, respectively. Thus, $S_1$ is the optimal outcome, $S_4$ is the worst outcome, and $S_2$ and $S_3$ are in between $S_1$ and $S_4$, and are incomparable ($S_1 \succ S_2 \sim S_3 \succ S_4$).

The formalism we just presented is rather weak in that it is based on the Pareto Principle. Consequently, it renders many outcomes generated by $P_{gen}$ optimal. To strengthen it one may consider ranked preferences, that is, preferences that differ in importance. A *ranked* ASO program is a tuple $(P_{gen}, P_{pref})$, where $P_{gen}$ is as before and $P_{pref}$ is a collection of *ranked* preference rules, that is, rules of the form

$$C_1 > \cdots > C_k \overset{j}{\leftarrow} a_1, \ldots, a_n, \neg\, b_1, \ldots, \neg\, b_m \tag{3.2}$$

where the notation is as above, the only difference being the presence of a positive integer $j$ indicating the rank (with 1 being the highest rank possible.) For a preference rule $p$, we write $r(p)$ to denote its rank.

If we modify the preferences of the example described above as $P_{pref} = \{p_1, p_2\}$ where

$$p_1 : beef > fish \overset{1}{\leftarrow}$$

$$p_2 : soup > salad \overset{2}{\leftarrow},$$

the preorder over outcomes becomes $S_1 \succ S_2 \succ S_3 \succ S_4$.

### 3.3 Computational Complexity Classes

In our work, we analyze the computational complexity for the problems of computing optimal (near optimal) outcomes, deciding whether manipulation and bribery is possible, and

learning ranks (weights). In this section, we give a brief introduction of the computational complexity classes [84]. Informally, in the computational complexity theory, a *complexity class* is a set of problems of related complexity. A typical complexity class has a definition of the form: the set of problems that can be solved by an abstract machine $M$ using $O(f(n))$ of resource $R$ where $n$ is the size of input. For instance, the class P is a set of decision problems can be solved in polynomial time by a deterministic Turing machine. The class NP is the set of decision problems whose solutions can be determined by a non-deterministic Turing machine in polynomial time. A decision problem is in the class co-NP if its complement is in the class NP. The polynomial hierarchy is a hierarchy of complexity classes that generalize the classes P, NP and co-NP. The class $\Sigma^p_{k+1}$ is the set of decision problems whose solutions can be determined in polynomial time by a non-deterministic Turing machine with a $\Sigma^p_k$ oracle. A decision problem is in the class $\Pi^p_{k+1}$ if its complement is in the class $\Sigma^p_{k+1}$. The class $\Delta^p_{k+1}$ is the set of decision problems that can be solved in polynomial time by a deterministic Turing machine with a $\Sigma^p_k$ oracle. The class PSPACE is the set of decision problems that can be solved by a deterministic Turing machine in polynomial space. A computational problem is complete for a complexity class if it is in this class and every problem in this class can by reduced to it in polynomial time.

## Chapter 4 Optimal Solutions for Answer Set Optimization Problems

In this chapter, we present our work on preference aggregation based on the preference formalism of answer set optimization (ASO). Of main interest to us are computational problems concerning optimal outcomes. We propose, implement and study methods to compute an optimal outcome; to compute an optimal outcome that is different from a given outcome; to compute an optimal outcome that is similar to (or, dissimilar from) a given candidate outcome; and to compute a set of optimal outcomes each significantly different from the others. For the decision versions of several of these problems we establish their computational complexity.

### 4.1 Problems and Complexity

The two fundamental optimization problems for any preference framework concern the existence of an optimal outcome and an optimal outcome different from a given candidate outcome (interpretation). We state them below for ASO programs.

**OPT-OUTCOME** Given an ASO program $P$, decide whether an optimal outcome $S$ for $P$ exists.

**DIFF-OPT-OUTCOME** Given an ASO program $P$ and a candidate outcome (an interpretation) $S$, decide whether $P$ has an optimal outcome $S'$ such that $S' \neq S$.

Clearly, if the generator of an ASO program $P$ has outcomes, $P$ has an optimal outcome. Thus the complexity of *deciding* whether an optimal outcome exists is the same as the complexity of deciding whether the generator $P_{gen}$ is satisfiable (with respect to the semantics of choice). That latter problem is NP-complete for both generator formalisms and therefore, so is the former (we recall that generators are either propositional formulas or *non-disjunctive* logic programs).

**Theorem 1.** *The* OPT-OUTCOME *problem is NP-complete.*

For *computing* an optimal outcome, another problem is fundamental.

**OPT-TEST** Given an ASO program $P$, and an outcome $S$ for $P$, decide whether $S$ is an optimal outcome for $P$.

This problem has been studied by Brewka et al. [27], who proved it to be coNP-complete. The proof consists of showing that the complement of that problem, deciding whether a given outcome $S$ of $P$ is not an optimal outcome of $P$, is NP-complete. The witness used by the non-deterministic polynomial-time algorithm needed for the membership part is an outcome $S'$ of $P$ strictly better than $S$. Computing such better and better witnesses until an optimal outcome is reached forms the basis of an algorithm to compute optimal outcomes (we discuss this matter in more detail in the next section).

The DIFF-OPT-OUTCOME problem is essential to many other problems we consider here as it concerns the existence of optimal outcomes *other* than a given interpretation (in particular, other than a given optimal outcome). We now show that the DIFF-OPT-OUTCOME problem is NP-complete.

**Theorem 2.** *The* DIFF-OPT-OUTCOME *problem is NP-complete.*

*Proof.* Let us consider an ASO program $P$ and a candidate outcome (an interpretation) $S$ (over the vocabulary of $P$). An optimal outcome for $P$ that is different from $S$ exists if and only if $P_{gen}$ has an outcome $M$ such that $S \not\succeq M$. Indeed, if $P$ has an optimal outcome $S'$ different from $S$, then $S \not\succeq S'$ and we can take $S'$ for $M$. Conversely, let $M$ be an outcome of $P$ such that $S \not\succeq M$. Thus, $P$ has an optimal outcome $S'$ such that $S' \succeq M$. It follows that $S \neq S'$.

This property shows that the problem is in NP. To decide the problem, we need to guess an interpretation $M$, check that it is an outcome for $P$ and verify (in polynomial time) that $S \not\succeq M$.

The hardness can be proved by reduction from the outcome existence problem. For the sake of concreteness, we assume that generators are propositional theories (a similar argument can be provided for the case when generators are logic programs). Given a propositional theory $P$, it is NP-hard to decide whether $P$ is satisfiable. We construct an ASO program $P'$ and an interpretation $S$ so that $P'$ has an outcome $M$ satisfying $S \not\succeq M$ if and only if $P$ is satisfiable.

To this end, we introduce a new atom $a$ and construct $P'$ as follows. We set $S = \emptyset$, $P'_{gen} = P$ and $P'_{pref} = \{a > \neg a\}$. There is a correspondence between models of $P$ and outcomes of $P'$. If $M$ is a model of $P$, then $M$ and $M \cup \{a\}$ are outcomes for $P'$. In the other direction, if $S'$ is an outcome for $P'$, $S' \setminus \{a\}$ is a model of $P$.

($\Leftarrow$) If $P$ is satisfiable, let $M$ be a model of $P$. According to the correspondence, $M$ and $M \cup \{a\}$ are outcomes for $P'$. Let $S' = M \cup \{a\}$. Since $S = \emptyset$, $S' \succ S$ and so, $S \not\succeq S'$.

($\Rightarrow$) Let $S'$ be an outcome for $P'$ such that $S' \not\succeq S$. Clearly, $M = S' \setminus \{a\}$ is a model of $P$ and so, $P$ is satisfiable. □

Next, we consider the problems of finding optimal outcomes that are similar to a given interpretation, or dissimilar from a given interpretation. In each problem we assume that the distance between outcomes is determined by some measure of distance between interpretations, say $\Delta$.

**SIM-OPT-OUTCOME** Given an ASP program $P$, an interpretation $S$ and a non-negative integer $d$, decide whether $P$ has an optimal outcome $S'$ such that $\Delta(S, S') \leq d$.

**DISSIM-OPT-OUTCOME** Given an ASP program $P$, an interpretation $S$ and a non-negative integer $d$, decide whether $P$ has an optimal outcome $S'$ such that $\Delta(S, S') \geq d$.

The complexity of these problems depends on the complexity of deciding whether $\Delta(S, S') \leq d$. Assuming that this problem is in P, the problems to find a similar/dissimilar optimal outcome are $\Sigma_2^p$-complete for several natural metrics $\Delta$. The details depend on the definition

of $\Delta$. The theorems we present below assume that the Hamming distance *HD* is used to measure how far from each other are the outcomes.

**Theorem 3.** *The* SIM-OPT-OUTCOME *problem is* $\Sigma_2^p$*-complete when the Hamming distance HD is used for* $\Delta$*.*

*Proof.* (Membership) The problem is in $\Sigma_2^p$ because, given an ASO program $P$, an interpretation $S$ and a non-negative integer $d$, we can guess an interpretation $S'$, verify in polynomial time that it is an outcome of $P$ (an answer set or a model, accordingly), then use a coNP-oracle for the OPT-TEST problem to verify that $S'$ is an optimal outcome and, finally, verify in polynomial time that $HD(S,S') \leq d$.

(Hardness) We provide a detailed argument in the case when generators are logic programs (the case of propositional theories is similar).

The following problem is $\Sigma_2^p$-hard [27]: Given an ASO program $P$ and an atom $l$, decide whether $P$ has an optimal outcome $M$, such that $l \in M$. We will prove the hardness part of the assertion by constructing an ASO program $P'$, an interpretation $S$ and a nonnegative integer $d$ so that $P'$ has an optimal outcome $S'$ with $HD(S,S') \leq d$ if and only if $P$ has an optimal outcome $M$ such that $l \in M$.

To this end, we define $d$ as the number of atoms in $P$, and introduce $d+1$ fresh atoms $a_1, a_2, \ldots, a_{d+1}$. We construct an ASO program $P'$ and an interpretation $S$ as follows. We define

$$P'_{gen} = P_{gen} \cup \{a_1 \leftarrow \neg l, a_2 \leftarrow \neg l, \ldots, a_{d+1} \leftarrow \neg l\},$$

and set $P'_{pref} = P_{pref}$ and $S = \emptyset$. It is clear that the mapping

$$\pi(M) = \begin{cases} M & if\ l \in M \\ M \cup \{a_1, \ldots, a_{d+1}\} & otherwise \end{cases}$$

is a bijection between the outcomes for $P$ and $P'$ (as it is indeed a bijection between answer sets of $P_{gen}$ and $P'_{gen}$). Since no atom $a_i$ appears in the selector $P_{pref}$ $(= P'_{pref})$, $M$ and $\pi(M)$

have the same satisfaction degrees on $P_{pref}$ and $P'_{pref}$. Thus, for every two outcomes $M$ and $M'$ of $P$, $M \succeq_P M'$ if and only if $\pi(M) \succeq_{P'} \pi(M')$. Since $\pi$ is a bijection between outcomes of $P$ and $P'$, it follows that $M \in Opt(P)$ if and only if $\pi(M) \in Opt(P')$.

($\Rightarrow$) Let $M$ be an optimal outcome for $P$ such that $l \in M$. We set $S' = M$. Since $l \in M$, $\pi(M) = M$ By our comments above, $S'$ is an optimal outcome of $P'$. Since $|S'| = |M|$, the number of elements in $S'$ is at most $d$. Since $S = \emptyset$, we have $HD(S,S') \leq d$.

($\Leftarrow$) Let $S'$ be an optimal outcome for $P'$ and $HD(S,S') \leq d$. Let us consider any of the new atoms, say $a_i$. If $a_i \in S'$, then $a_j \in S'$, for every $j$, $1 \leq j \leq d+1$. Thus $|S'| \geq d+1$ and $HD(S,S') \geq d+1$, a contradiction. It follows that $a_i \notin S'$ for any $1 \leq i \leq d+1$, and so, $l \in S'$. Let $M = S'$. It follows that $S' = \pi(M)$ and so, $M$ is an optimal outcome for $P$ and $l \in M$. □

**Theorem 4.** *The* DISSIM-OPT-OUTCOME *problem is $\Sigma_2^p$-complete when the Hamming distance HD is used for* $\Delta$.

*Proof.* (Membership) The problem is in $\Sigma_2^p$ because, given an ASO program $P$, an interpretation $S$ and a non-negative integer $d$, we can guess an interpretation $S'$, verify in polynomial time that it is an outcome of $P$ (an answer set or a model), use a coNP-oracle for the OPT-TEST problem to verify that $S'$ is an optimal outcome of $P$ and, finally, verify in polynomial time that $HD(S,S') \geq d$.

(Hardness) In the reduction we use the same problem as before and reason in a similar way. Specifically, given an ASO program $P$ and an atom $l$, we construct an ASO program $P'$, an interpretation $S$, and a nonnegative integer $d$ so that $P'$ has an optimal outcome $S'$ with $HD(S,S') \geq d$ if and only if $P$ has an optimal outcome $M$ such that $l \in M$.

We define $d$ as the number of atoms in $P$ plus 1. We introduce $d$ new atoms $a_1, a_2, \ldots, a_d$ and construct the program $P'$ as follows. We set

$$P'_{gen} = P_{gen} \cup \{a_1 \leftarrow l, a_2 \leftarrow l, \ldots, a_d \leftarrow l\}$$

and $P'_{pref} = P_{pref}$. Finally, we set $S = \emptyset$. This time, it is clear that the mapping

$$\pi(M) = \begin{cases} M & if\ l \notin M \\ M \cup \{a_1, \ldots, a_d\} & otherwise \end{cases}$$

is a bijection between outcomes of $P$ and $P'$. As before, since no atom $a_i$ appears in the selector $P_{pref}$ ($= P'_{pref}$), $M$ and $\pi(M)$ have the same satisfaction vectors with respect to $P_{pref}$ ($= P'_{pref}$). Thus, for any outcome $M$ of $P$, $M \in Opt(P)$ if and only if $\pi(M) \in Opt(P')$.

($\Rightarrow$) Let $M$ be an optimal outcome of $P$ such that $l \in M$. We define $S' = \pi(M)$. Clearly, $S' = M \cup \{a_1, \ldots, a_d\}$. By our comments above, $S'$ is an optimal outcome for $P'$. Moreover, since $|S'| \geq d$ and $|S| = 0$, we have $HD(S, S') \geq d$.

($\Leftarrow$) Let $S'$ be an optimal outcome for $P'$ and $HD(S, S') \geq d$. Since $|S| = 0$, $|S'| \geq d$. Thus, $S'$ contains at least one atom $a_j$. Consequently, $l \in S'$ and for every $i = 1, \ldots, d$, $a_i \in S'$. Let $M = S' \setminus \{a_1, \ldots, a_d\}$. Clearly, $M$ is an outcome of $P$, $l \in M$ and $S' = \pi(M)$. Moreover, since $S'$ is an optimal outcome of $P'$, $M$ is an optimal outcome of $P$. □

Next, we consider the SIM-OPT-OUTCOME and DISSIM-OPT-OUTCOME problems under the assumption that $d$ is fixed and not a part of input. The resulting problems are concerned with the existence of an optimal outcome very close to a given interpretation (within a fixed distance $d$), and with the existence of an optimal outcome that is not in the close vicinity of a given interpretation (is not within a fixed distance $d$).

**Theorem 5.** *The* SIM-OPT-OUTCOME *problem with a fixed $d$ is in $\Delta_2^p$ (with HD as $\Delta$).*

*Proof.* To show the problem is in $\Delta_2^p$, we present an algorithm to decide it that relies on calls to a coNP-oracle and works in polynomial time. The algorithm considers all interpretations at distance at most $d$ from $S$. For each interpretation it checks whether it is an outcome of $P$ and, if so, whether it is an optimal outcome of $p$. That last task requires a call to a coNP-oracle for the problem OPT-TEST. If an optimal outcome is found in the process, the algorithm outputs YES. Otherwise, it outputs NO. Since the number of interpretations $S'$

such that $HD(S,S') \leq d$ is given by

$$\binom{n}{1} + \cdots + \binom{n}{d} = O(n^d),$$

where $n$ denotes the number of atoms in $P$, the algorithm indeed works in polynomial time (counting the time for oracle calls as 1). □

**Theorem 6.** *The* DISSIM-OPT-OUTCOME *problem with a fixed d is NP-complete (with HD as* $\Delta$*).*

*Proof.* Let $\mathscr{O} = \{O : O \text{ an outcome of } P \text{ and } HD(S,O) < d\}$. We have the following property. The ASO program $P$ has an optimal outcome $S'$ such that $HD(S,S') \geq d$ if and only if $P$ has an outcome $M$ such that for every $O \in \mathscr{O}$, $O \not\succeq M$. Indeed, if $P$ has such an outcome $M$, then any optimal outcome $S'$ of $P$ such that $S' \succeq M$ (such outcomes exist) satisfies $S' \notin \mathscr{O}$, that is, $HD(S',S) \geq d$. Conversely, assume that $P$ has no such outcome $M$, that is, that for every outcome $M$ of $P$ there is an outcome $O \in \mathscr{O}$ such that $O \succeq M$. Then every optimal outcome $S'$ of $P$ belongs to $\mathscr{O}$, that is, every optimal outcome $S'$ of $P$ satisfies $HD(S,S') < d$.

(Membership) First, we note that we can compute $\mathscr{O}$ in polynomial time. Indeed, based on the previous proof, there are $O(n^{d-1})$ interpretations at distance at most $d-1$ from $S$. For each of them, verifying whether it is an outcome of $P$ takes polynomial time and so, the claim follows.

Let us now consider the following algorithm: (1) guess an interpretation $S'$; (2) verify that $S'$ is an outcome of $P$; and (3) for every $O \in \mathscr{O}$, verify the condition $O \not\succeq S'$. Based on the property proved above, this nondeterministic algorithm correctly decides the problem DISSIM-OPT-OUTCOME (with fixed $d$), and the tasks (2) and (3) take polynomial time. Thus, the DISSIM-OPT-OUTCOME problem (with a fixed $d$) is in NP.

(Hardness) We provide a detailed argument in the case when generators are propositional theories. Given a propositional theory $P$, it is NP-hard to decide whether $P$ is satisfiable. We

construct an ASO program $P'$ and an interpretation $S$ such that $P'$ has an optimal outcome $S'$ with $HD(S,S') \geq d$ if and only if $P$ is satisfiable.

To this end, we introduce $d$ new atoms $a_1,\ldots,a_d$ and construct $P'$ by setting $P'_{gen} = P$ and $P'_{pref} = \{a_i > \neg a_i : i = 1,\ldots,d\}$. We also set $S = \emptyset$.

($\Rightarrow$) Let $M$ be a model of $P$. Then $S' = M \cup \{a_1,\ldots,a_d\}$ is a model of $P'_{gen}$ ($a_i$'s do not appear in $P$ and $P'_{gen} = P$). Thus, $S'$ is an outcome of $P'$. Moreover, it is an optimal outcome of $P'$ as its satisfaction degree on each preference in $P'_{pref}$ is 1. Since $S = \emptyset$, $HD(S,S') \geq d$.

($\Leftarrow$) Let $S'$ be an optimal outcome for $P'$. It follows that $S'$ is an outcome of $P'$, that is a model of $P'_{gen}$. Thus, $P$ ($= P'_{gen}$) is satisfiable. □

As an aside, we note that the DISSIM-OPT-OUTCOME problem with $d = 1$ is the same as the DIFF-OPT-OUTCOME problem.

Similarly to questions about the existence of optimal outcomes within (or outside) a small sphere around $S$ given by a fixed distance $d$ that we just studied, it is also of interest to know if an ASO program $P$ has optimal outcomes within (or outside) a large sphere around $S$ given by the distance $n-d$, where $n$ is the size of the vocabulary and $d$ is fixed.

**Theorem 7.** *The* SIM-OPT-OUTCOME *problem with the distance bound given by $n-d$, where $d$ is fixed and $n$ is the number of atoms in the input ASO program is NP-complete.*

*Proof.* (Membership) Let $\mathcal{O} = \{O : O \text{ an outcome of } P \text{ and } HD(S,O) > n-d\}$. As in the earlier proofs, $\mathcal{O}$ can be computed in polynomial time and that $S'$ is an optimal outcome of $P$ such that $HD(S',S) \leq n-d$ if and only if there is an outcome $M$ of $P$ such that for every $O \in \mathcal{O}$, $O \not\geq M$. Similarly as before, these observations yield a non-deterministic polynomial-time algorithm deciding the problem.

(Hardness) We provide a detailed argument in the case when generators are propositional theories. Given a propositional theory $P$, it is NP-hard to decide whether $P$ is satisfiable. We construct an ASO program $P'$ and an interpretation $S$ such that there is an optimal

outcome $S'$ for $P'$, $S' \neq S$ and $HD(S,S') \leq n-d$ if and only if $P$ is satisfiable (here $n$ denotes the number of atoms in $P'$).

To this end, we introduce $d+1$ new atoms $b, a_1, \ldots, a_d$ and construct $P'$ by setting $P'_{gen} = P$ and

$$P'_{pref} = \begin{cases} b > \neg b \\ \\ \neg a_i > a_i \quad i = 1, \ldots, d. \end{cases}.$$

We also set $S = \emptyset$. We denote by $m$ the number of atoms in $P$. Thus, $n = m + d + 1$.

($\Rightarrow$) Let $M$ be a model of $P$ consisting of atoms in $P$. Then, $S' = M \cup \{b\}$ is a model of $P'_{gen}$ and so, an outcome of $P'$. Since the satisfaction degree of $S'$ on any preference rule is 1, $S'$ is an optimal outcome of $P'$. Thus, $HD(S,S') = |M| + 1 \leq m + 1 = n - d$.

($\Leftarrow$) Let $S'$ be an optimal outcome for $P'$ such that $HD(S,S') \leq n-d$. Clearly, $S'$ is a model of $P'_{gen}$ and, consequently, $M = S' \setminus \{b, a_1, \ldots, a_d\}$ is a model of $P$. Thus, $P$ is satisfiable. □

**Theorem 8.** *The* DISSIM-OPT-OUTCOME *problem with the distance bound given by $n-d$, where $d$ is fixed and $n$ is the number of atoms in the input ASO program is in $\Delta_2^P$.*

*Proof.* Let $P$ be an ASO program, say with $n$ atoms, $S$ an interpretation and $d$ a fixed integer. To find an optimal outcome $S'$ such that $HD(S,S') \geq d$, we consider all interpretations $S'$ such that $HD(S,S') \geq d$ (similarly as in the previous proofs, we can show that there are $O(n^d)$ of them). For each of them, we check whether it is an outcome of $P$ (can be done in polynomial time) and if so, whether it is an optimal outcome of $P$ (a call to an oracle for the problem OPT-TEST). This algorithm runs in polynomial time (counting oracle calls as 1). Thus, the problem at hand is in $\Delta_2^p$. □

Finally, we state and discuss the problem of finding a *set* of optimal outcomes that are significantly different from each other. In that problem we are interested in finding sets of optimal outcomes of cardinality $k$, where $k$ is a fixed integer.

$k$-**DIVERSE-OPT-OUTCOMES** Given an ASO program $P$ and a nonnegative integer $d \geq 1$, decide whether there is a set $\mathscr{O}$ of optimal outcomes for $P$ such that $|\mathscr{O}| = k$, and $\Delta(S, S') \geq d$ for any two distinct $S, S' \in \mathscr{O}$.

**Theorem 9.** *The* $k$-DIVERSE-OPT-OUTCOMES *problem is* $\Sigma_2^p$*-complete.*

*Proof.* (Membership) The problem can be decided by (1) guessing $k$ interpretations $S_1, \ldots, S_k$; (2) verifying that they are outcomes for $P$; (3) verifying their optimality in $P$ by calls to an oracle for the OPT-TEST problem; and finally (4) verifying that $HD(S_i, S_j) \geq d$ for every two $S_i, S_j$, $1 \leq i < j \leq k$. The tasks (3) and (4) can be accomplished in polynomial time. Thus, the problem is in $\Sigma_2^p$.

(Hardness) We provide a detailed argument in the case when generators are logic programs. Given an ASO program $P$ and an atom $l$, it is $\Sigma_2^p$-hard to decide whether $P$ has an optimal outcome $M$ such that $l \in M$ [27]. We construct an ASO program $P'$ and a nonnegative integer $d$ so that there is a set $\mathscr{O}$ of $k$ optimal outcomes of $P'$ with $HD(S, S') \geq d$ for every two distinct outcomes $S, S' \in \mathscr{O}$ if and only if $P$ has an optimal outcome $M$ such that $l \in M$. Let $n = m + 3$, where $m$ is the number of atoms in $P$. We start by constructing an answer set program $P''$ which has exactly $k - 1$ answer sets and the distance between any two distinct ones among them is $2n$. To this end, we introduce $(k-1)n$ new atoms $x_{i,j}$, where $i = 1, \ldots, k-1$ and $j = 1, \ldots, n$ and define $P''$ to consist of the rules

$$x_{i,1} \leftarrow \neg x_{1,1}, \ldots, \neg x_{i-1,1}, \neg x_{i+1,1}, \ldots, \neg x_{k-1,1}, \qquad i = 1, \ldots, k-1$$
$$x_{i,j} \leftarrow x_{i,1}, \qquad i = 1, \ldots, k-1, j = 2, \ldots, n.$$

One can show that the program $P''$ has $k - 1$ answer sets $A^1, \ldots, A^{k-1}$, where $A^i = \{x_{i,j} : j = 1, \ldots, n\}$.

To construct the ASO program $P'$ we introduce $n + 2$ fresh atoms $a, b, l_1, l_2, \ldots, l_n$. We then define $P'_{gen}$

$$P'_{gen} = \begin{cases} a \leftarrow \neg b & \\ b \leftarrow \neg a & \\ h \leftarrow B, a & \text{for every rule } h \leftarrow B \in P'' \\ h \leftarrow B, b & \text{for every rule } h \leftarrow B \in P_{gen} \\ l_i \leftarrow b, l & i = 1, \ldots, n \end{cases}$$

and

$$P'_{pref} = P_{pref} \cup \{a > \neg a\} \cup \{b > \neg b\}.$$

Finally, we set $d = 2n$.

Let $M$ be an answer set of $P''$ or of $P_{gen}$. We define

$$\pi(M) = \begin{cases} M \cup \{a\} & \text{if } M \in AS(P'') \\ M \cup \{b\} & \text{if } M \in AS(P_{gen}), l \notin M \\ M \cup \{b, l_1, \ldots, l_n\} & \text{if } M \in AS(P_{gen}), l \in M, \end{cases}$$

where we write $AS(Q)$ for the set of answer sets of an answer set program $Q$. One can show that $\pi(M)$ is an outcome of $P'$ (an answer set of $P'_{gen}$), and that every outcome of $P'$ is of the form $\pi(M)$, where $M$ is an outcome of $P$ or of $P''$.

Moreover, we have two additional properties. First, if $M$ is an outcome (answer set) of $P''$, $\pi(M)$ is an optimal outcome for $P'$. To prove that, let us assume that $P'$ has an outcome $M'$ such that $M' \succ_{P'} \pi(M)$. It follows that the satisfaction degree of $M'$ on the preference $a > \neg a$ is 1 and so, $a \in M'$. Consequently, $M'' = M' \setminus \{a\}$ is an answer set of $P''$. Thus, neither $M$ nor $M'$ contain any atoms that occur in $P$ and so, $M' \approx_{P'} \pi(M)$, a contradiction.

Second, if $M$ is an outcome of $P$ (an answer set of $P_{gen}$), $\pi(M)$ is an optimal outcome of $P'$ if and only if $M$ is an optimal outcome of $P$. To prove it, let us consider an optimal outcome $M$ of $P$ and assume that $M' \succ_{P'} \pi(M)$, for some outcome $M'$ of $P'$. Since $b \in$

$\pi(M)$, $b \in M'$. Let $M'' = M' \setminus \{b, l_1, \ldots, l_n\}$. It follows that $M''$ is an outcome for $P$ and $M'' \succ_P M$, a contradiction. Conversely, if $M$ is not optimal for $P$ then we have $M' \succ_P M$, for some outcome $M'$ for $P$. Clearly, $\pi(M') \succ_{P'} \pi(M)$ and so, $\pi(M)$ is not an optimal outcome of $P'$.

($\Leftarrow$) Let $M$ be an optimal outcome of $P$ such that $l \in M$. Based on the comments above, $\pi(M), \pi(A^1), \ldots, \pi(A^{k-1})$ are optimal outcomes of $P'$. By the definition, $\pi(M) = M \cup \{b, l_1, \ldots, n\}$ and $\pi(A^i) = A^i \cup \{a\}$, $1 \leq ß \leq k-1$. Thus, $HD(\pi(M), \pi(A^i)) \geq 2n = d$ (each atom in $A^i \cup \{l_1 m \ldots, l_n\}$ belongs to exactly one of $\pi(M)$ and $\pi(A^i)$). Moreover, for all $i$ and $j$ such that $1 \leq i \neq j \leq k-1$, we have $HD(A^i, A^j) = 2n = d$ and so, $HD(\pi(A^i), \pi(A^j)) = 2n = d$. Thus, $P'$ has $k$ optimal outcomes, each two distinct ones at distance at lest $d$ from each other.

($\Rightarrow$) Let $\mathscr{O}$ be a set of optimal outcomes for $P'$ such that $|\mathscr{O}| = k$ and $HD(S, S') \geq d$, for any $S, S' \in \mathscr{O}$. Answer sets of $P''$ can only contribute $k-1$ optimal outcomes to $\mathscr{O}$. Thus, at least one outcome in $\mathscr{O}$ is of the form $\pi(M)$ where $M$ is an outcome of $P$. By our observation above, that $M$ is an optimal outcome of $P$. Let us assume that $l \notin M$. Then, $\pi(M) = M \cup \{b\}$. Let $O$ be any other element of $\mathscr{O}$ (we recall that $k \geq 2$). If $O = \pi(M')$, where $M'$ is an outcome of $P$, then $M'$ is an optimal outcome of $P$. Since $HD(\pi(M), \pi(M')) \leq m + n < 2n = d$, a contradiction. Thus, $O = \pi(A^i)$, for some $i = 1, \ldots, k-1$. It follows that $HD(\pi(M), \pi(A^i)) \leq m + 2 + n < 2n = d$, a contradiction. Thus, $l \in M$. □

**Theorem 10.** *The* $k$-DIVERSE-OPT-OUTCOMES *problem with a fixed* $d$ *is in* $\Delta_2^p$.

*Proof.* Let us assume that we have constructed optimal outcomes $S_1, \ldots, S_i$ of $P$, for some $0 \leq i < k$, so that for every $1 \leq j \neq l \leq i$, $HD(S_j, S_l) \geq d$.

1. If $i = k$, we return $S_1, \ldots, S_k$ and terminate.

2. Otherwise, we compute the set $\mathscr{O}$ of all outcomes $S$ of $P$ such $HD(S, S_j) < d$, for some $j = 1, \ldots, i$.

3. If there is an outcome $S'$ of $P$ such that for every $S \in \mathscr{O}$, $S \not\succeq S'$

   a) compute one such $S'$

   b) and then, compute an optimal outcome $S''$ of $P$ such that $S'' \succeq S'$

   c) set $S_{i+1} = S''$ and $i = i+1$, and repeat steps 1, 2 and 3

4. Otherwise (in that case, we have that for every outcome $S'$ of $P$ there is an outcome $S \in \mathscr{O}$ such that $S \succeq S'$; thus, every optimal outcome of $P$ is in $\mathscr{O}$), for every $k$-element subset $\mathscr{O}'$ of $\mathscr{O}$, check if every $S \in \mathscr{O}'$ is an optimal outcome of $P$ and if for every different $S, S' \in \mathscr{O}'$, $HD(S,S') \geq d$; if such a subset $\mathscr{O}'$ is found, return it and terminate, if not, terminate with failure.

The correctness of this algorithm is clear. Next, we note that since $k$ and $d$ are fixed, there are $O(n^{d-1})$ interpretations $S$ such that $HD(S,S_j) < d$, for some $j = 1,\ldots,i$. For each of them, it takes polynomial-time to check if it is an outcome of $P$ and a single call to an coNP-oracle to check if it is optimal (the problem OPT-TEST is coNP-complete). Thus, step 2 can be accomplished by a polynomial-time algorithm with a coNP-oracle (counting each oracle call, here only one, as taking unit time). Moreover, $|\mathscr{O}| = O(n^{d-1})$.

We discuss the computational tasks involved in step 3 in detail in the next section. We show there that they all can be accomplished in polynomial time with the assistance of an NP-oracle.

Step 4 can be accomplished by a polynomial-time algorithm with an NP-oracle for deciding whether an outcome of $P$ is optimal because there are only $O(n^{k(d-1)})$ sets $\mathscr{O}'$ to consider. □

### 4.2 Computational Methods

In this section, we study two methods to solve the four optimization problems we discussed above: finding one optimal solution; given a candidate solution, finding a different

optimal solution; given a candidate solution, finding a similar/dissimilar optimal solution; and finding a diverse set of optimal solutions. We describe them under the assumption that the generator is an answer set program but the discussion extends to the case when we use propositional theories for generators. The first method uses an iterative way to find optimal outcomes for an ASO problem. The second method encodes each optimization problem as a single disjunctive logic program.

### 4.2.1 Iterative Method

The iterative method separates the optimization problem into a series of easier tasks which can be modeled as answer set programs and solves them by ASP solvers. This idea was introduced by Brewka, Niemelä and Truszczyński in 2003 [27] to solve the OPT-OUTCOME problem. To find an optimal outcome for a given ASO program $P$, the iterative method first randomly picks an outcome for $P$, and computes an outcome which is strictly better than it. If such outcome exists, it repeats the process to find a "better" outcome from the achieved one. This process eventually terminates on an optimal outcome for $P$. In our work, we extend this method to apply on ranked preference profiles, and provide computational methods for solving the optimization problems with additional conditions discussed in Section 4.1.

Before introducing the algorithm, we first define two problems which are fundamental in the algorithm and can be modeled as a single answer set program.

**FIND-BETTER** Given an ASO program $P$, and an interpretation $S$, find an outcome for $P$ that is strictly better than $S$.

**FIND-NOTWORSE-DIFF** Given an ASO program $P$, and an interpretation $S$, find an outcome for $P$ that is not worse than and is different from $S$.

The FIND-BETTER problem was discussed in the original paper introduced ASO programs [27]. It can be solved by constructing a certain answer set program $\Pi_1(P,S)$ and, then,

finding its answer sets by means of ASP tools. Given an ASO program $P$ (assume $P_{gen}$ is an answer set program) and an interpretation $S$, the program $\Pi_1(P,S)$ is defined to be the union of $P_{gen}$ and the following rules:

1. Add facts $rule(r)$ for each preference $r \in P_{pref}$.

2. Add facts $v_0(r,d) \leftarrow$ for each preference $r \in P_{pref}$, where $d$ is the satisfaction degree of $r$ in $S$.

3. For each preference $r_i \in P_{pref}$ and $C_j$ in $r_i$, introduce a new atom $c_{i,j}$ and add rules to capture the conditions under which $C_j$ is satisfied.

4. For each preference $r_i \in P_{pref}$ in the form 3.1 in Chapter 3, add rules

$$
\begin{aligned}
body(r_i) &\leftarrow a_1,\ldots,a_n,\neg\, b_1,\ldots,\neg\, b_m \\
head(r_i) &\leftarrow c_{i,1} \\
&\ldots \\
head(r_i) &\leftarrow c_{i,k} \\
v_1(r_i,1) &\leftarrow rule(r_i), not\ body(r_i) \\
v_1(r_i,1) &\leftarrow rule(r_i), not\ head(r_i) \\
v_1(r_i,1) &\leftarrow c_{i,1}, body(r_i) \\
v_1(r_i,2) &\leftarrow not\ c_{i,1}, c_{i,2}, body(r_i) \\
&\ldots \\
v_1(r_i,k) &\leftarrow not\ c_{i,1}, not\ c_{i,2},\ldots, not\ c_{i,k-1}, c_{i,k}, body(r_i).
\end{aligned}
$$

5. Include rules

$$
\begin{aligned}
&\leftarrow not\ better \\
better &\leftarrow v_0(R,V_0), v_1(R,V_1), V_0 > V_1 \\
&\leftarrow v_0(R,V_0), v_1(R,V_1), V_0 < V_1.
\end{aligned}
$$

In $\Pi_1(P,S)$, the facts $v_0$ represents the satisfaction degrees for $S$, $v_1$ represents the satisfaction degree for the outcome $S'$ of $\Pi_1(P,S)$ ($S'$ is also an outcome for $P_{gen}$), and the last three rules restrict that $S' \succ_P S$. Therefore if $\Pi_1(P,S)$ has an answer set $S'$, it is an outcome for $P$ that is strictly better than $S$. If $\Pi_1(P,S)$ is unsatisfiable, such outcome does not exist and so, $S$ is optimal.

With ranked preferences, the $\Pi_1(P,S)$ is constructed in the same way except that we replace the last three rules defined in step 5 with the following rules:

$$
\begin{aligned}
&\leftarrow && \mathit{not\ better} \\
\mathit{better} &\leftarrow && \mathit{not\ failAt}(L),\ \mathit{betterAt}(L) \\
\mathit{betterAt}(L) &\leftarrow && v_0(R,V_0), v_1(R,V_1), V_0 > V_1, \mathit{rank}(R,L) \\
\mathit{failAt}(L) &\leftarrow && \mathit{betterAt}(L), v_0(R,V_0), v_1(R,V_1), V_0 < V_1, \mathit{rank}(R,L_0), L_0 \leq L.
\end{aligned}
$$

and add the fact $\mathit{rank}(r,j)$ for each preference $r \in P_{pref}$ in the form 3.2 in Chapter 3. The fact $\mathit{better}$ means $S'$ (the outcome for $\Pi_1(P,S)$) is strictly better than $S$. It is implemented by defining that there is a rank $l$ such that $S'$ is better than $S$ on some preference with rank $l$ (the fact $\mathit{betterAt}(L)$) and $S'$ is not worse than $S$ on any preference with a rank $l'$, $l' \leq l$. The fact $\mathit{failAt}(L)$ is true if $S'$ is better than $S$ on some preference with rank $L$ and $S'$ is worse than $S$ on some preference with the same rank or a higher rank.

To solve the FIND-NOTWORSE-DIFF problem, we build an answer set program $\Pi_2(P,S)$ in a similar way as $\Pi_1(P,S)$. Let $\mathscr{A}$ be the set of atoms in $P_{gen}$ and assume $S$ is represented by a set of true atoms, the program $\Pi_2(P,S)$ is constructed from $\Pi_1(P,S)$ by:

1. Adding a rule $\leftarrow \{l : l \in S\}, \{\mathit{not}\ l : l \in \mathscr{A}, l \notin S\}$ to restrict that the outcome for $\Pi_2(P,S)$ has to be different from $S$.

2. Replacing the rules defined in step 5 when building $\Pi_1(P,S)$ with the following rules

$$\begin{aligned} worse &\leftarrow v_0(R,V_0), v_1(R,V_1), V_0 < V_1 \\ better &\leftarrow v_0(R,V_0), v_1(R,V_1), V_0 > V_1 \\ &\leftarrow worse, not\ better. \end{aligned}$$

If $\Pi_2(P,S)$ has an outcome $S'$, the new atom *worse* is true if and only if $S'$ is worse than $S$ on some preference rule. Similarly, *better* means $S'$ is strictly better than $S$ on some preference rule. The last rule defines a contradiction if $S'$ is worse than $S$ on $P_{pref}$. Therefore $S'$ is not worst than and different from $S$. The problem has no outcome if $\Pi_2(P,S)$ is unsatisfiable.

Similarly, in the ranked version, we construct $\Pi_2(P,S)$ by replacing the three rules defined in the second step described above by the following rules:

$$\begin{aligned} worseAt(L) &\leftarrow v_0(R,V_0), v_1(R,V_1), V_0 < V_1, level(R,L) \\ failAt(L) &\leftarrow worseAt(L), v_0(R,V_0), v_1(R,V_1), V_0 > V_1, level(R,L_0), L_0 \leq L \\ &\leftarrow worseAt(L), not\ failAt(L). \end{aligned}$$

and adding the fact $level(r,j)$ for each preference $r \in P_{pref}$ in the form 3.2 in Chapter 3. The fact $worseAt(L)$ means $S'$ (the outcome for $\Pi_2(P,S)$) is worse than $S$ on some preference rule with rank $L$. While the fact $failAt(L)$ is true if $S'$ is worse than $S$ on some preference rule with rank $L$, and $S'$ is better than $S$ on some preference rule with the same rank or a higher rank. The last rule defines the contradiction if $S \succ_P S'$.

Understanding how to solve these two basic problems, we can start to describe the iterative method for the optimization problems discussed in Section 4.1.

**Computing One Optimal Solution**

An optimal outcome is a outcome which is not dominated by any other outcome. Given an ASO program $P = (P_{gen}, P_{pref})$ (assume $P_{gen}$ is an answer set program), the OPT-OUTCOME problem can be solved by following the algorithm using an ASP solver:

1. Compute an outcome $S_0$ for $P_{gen}$. If $P_{gen}$is unsatisfiable, return no outcome.

2. Let $S = S_0$.

3. Loop

    a) Compute an outcome $S'$ for $\Pi_1(P,S)$. If $\Pi_1(P,S)$ is unsatisfiable, return $S$ as the solution for the OPT-OUTCOME problem. Otherwise, let $S = S'$.

According to the definition of $\Pi_1$, its output is strictly better than its input. From any start point, this iterative improvement process obtains a better outcome on each loop, and eventually terminates as the space of outcomes is finite. When it does, the last outcome generated is an optimal one. Let $S$ be the output of the algorithm. If $S$ is not optimal for $P$, there exists an outcome $S'$ such that $S' \succ_P S$, then the problem $\Pi_1(P,S)$ is satisfiable and the algorithm will not stop and output $S$ at this point. Therefore $S$ is an optimal outcome for $P$.

**Computing a Different Optimal Solution**

For the DIFF-OPT-OUTCOME problem, according to Theorem 2, we know that given an ASO program $P$ and an interpretation $S$, there is an optimal outcome $S'$ for $P$ such that $S' \neq S$ if and only if there is an outcome $M$ for $P_{gen}$ which is not worse than and is different from $S$. That is because, if such $M$ exists, there is an optimal outcome $M'$ for $P$ dominating $M$ and $M' \neq S$, then $M'$ is a solution for the DIFF-OPT-OUTCOME problem. Given an ASO program $P = (P_{gen}, P_{pref})$ and an interpretation $S$, the algorithm consists of the following steps:

1. Compute an outcome $S_0$ for $\Pi_2(P,S)$. If $\Pi_2(P,S)$ is unsatisfiable, return no solution.

2. Let $S' = S_0$.

3. Loop

a) Compute an outcome $M$ for $\Pi_1(P,S')$. If $\Pi_1(P,S')$ is unsatisfiable, return $S'$ as the solution for the DIFF-OPT-OUTCOME problem. Otherwise, let $S' = M$.

The algorithm is the same as the one solving the OPT-OUTCOME problem, only except using an outcome of $\Pi_2(P,S)$ as the start point for the iterative improvement process. Let $S'$ be the output of the algorithm, it has been shown in section 4.2.1 that it is optimal for $P$. Let us assume that $S' = S$. Based on the algorithm, either $S' = S_0$ or $S' \succ_P S_0$. According to the definition of $\Pi_2$, $S \neq S_0$ and $S \not\succ_P S_0$. If $S' = S_0$, then $S = S_0$, a contradiction. If $S' \succ_P S_0$, $S \succ_P S_0$, a contradiction. Therefore, $S'$ is optimal for $P$ and different from $S$.

**Computing a Similar or Dissimilar Optimal Solution**

To solve the SIM-OPT-OUTCOME and DISSIM-OPT-OUTCOME problems, given an ASO program $P$, an interpretation $S$ and a distance $d$, deciding whether there is an optimal outcome $S'$ such that $S' \neq S$ and $HD(S,S') \leq d$ ($HD(S,S') \geq d$), we introduce two techniques. The basic idea of the first technique is iteratively computing optimal outcomes, until a similar/dissimilar one is found. The second technique computes similar/dissimilar outcomes until an optimal one is found.

**A Straightforward Method** To be specific, the first technique keeps a set $\mathscr{O}$ of outcomes which are not the solution for the problem and initializes $\mathscr{O}$ as $\{S\}$. It computes an optimal outcome $M$ which is different from every $O \in \mathscr{O}$, and checks whether $HD(S,M) \leq d$ (or respectively $HD(S,M) \geq d$). If $M$ satisfies the distance condition, the algorithm returns $M$. Otherwise, it adds $M$ into $\mathscr{O}$ and repeats the process until some similar/dissimilar optimal outcome is found or no more optimal outcome exists. The list $\mathscr{O}$ can get very large.

Given an ASO program $P$, an interpretation $S$ and a distance $d$, the algorithm as follows:

1. Let $\mathscr{O} = \{S\}$.

2. Loop

a) Compute an optimal outcome $M$ for $P$ such that $M$ is different from every $O \in \mathscr{O}$. If no solution, return unsatisfiable.

b) If $HD(S,M) \leq d$ (or respectively $HD(S,M) \geq d$), return $M$ as the solution for the SIM-OPT-OUTCOME (DISSIM-OPT-OUTCOME) problem.

c) Otherwise, add $M$ to $\mathscr{O}$.

The pivotal problem in this algorithm is the first step in the loop and we define it as a new problem.

**FIND-OPT-DIFF-SET** Given an ASO program $P$ and a set $\mathscr{O}$ of interpretations, find an optimal outcome for $P$ which is different from every $O \in \mathscr{O}$.

The FIND-OPT-DIFF-SET problem is similar to the DIFF-OPT-OUTCOME problem and can be solved in a similar way. To solve the DIFF-OPT-OUTCOME problem, given an ASO program $P$ and an interpretation $S$, the algorithm first computes an outcome $S_0$ for $P$ which is not worse than and is different from $S$ (a solution for the FIND-NOTWORSE-DIFF problem), and then start from $S_0$, finds an optimal outcome for $P$ using the iterative improvement process. Inspired by this algorithm, the FIND-OPT-DIFF-SET problem can be solved in the same way only now $S_0$ should be not worse than and different from every $O \in \mathscr{O}$. We introduce a new problem to compute such $S_0$.

**FIND-NOTWORSE-DIFF-SET** Given an ASO program $P$, and a set $\mathscr{O}$ of interpretations, find an outcome for $P$ that is not worse than and is different from every $O \in \mathscr{O}$.

To solve the FIND-NOTWORSE-DIFF-SET problem, we construct an answer set program $\Pi_3(P,\mathscr{O})$ in a similar way as constructing $\Pi_2(P,S)$ for the FIND-NOTWORSE-DIFF problem. Given an ASO program $P$ and a set $\mathscr{O}$ of interpretations, let $\mathscr{A}$ be the set of atoms in $P_{gen}$ and assume every $O \in \mathscr{O}$ is represented by a set of atoms, the complete process to construct the program $\Pi_3(P,\mathscr{O})$ is defined as the following:

1. Include $P_{gen}$.

2. Add facts $rule(r)$ for each preference rule $r \in P_{pref}$.

3. For each $O \in \mathscr{O}$, add $v_0(O,r,d) \leftarrow$ for each preference $r \in P_{pref}$, where $d$ is the satisfaction degree of $r$ in $O$.

4. For each preference $r_i \in P_{pref}$ and $C_j$ in $r_i$, introduce a new atom $c_{i,j}$ and add rules to capture the conditions under which $C_j$ is satisfied.

5. For each preference $r_i \in P_{pref}$ in the form 3.1 in Chapter 3, add rules

$$
\begin{aligned}
body(r_i) &\leftarrow a_1, \ldots, a_n, \neg\, b_1, \ldots, \neg\, b_m \\
head(r_i) &\leftarrow c_{i,1} \\
&\ldots \\
head(r_i) &\leftarrow c_{i,k} \\
v_1(r_i,1) &\leftarrow rule(r_i), not\ body(r_i) \\
v_1(r_i,1) &\leftarrow rule(r_i), not\ head(r_i) \\
v_1(r_i,1) &\leftarrow c_{i,1}, body(r_i) \\
v_1(r_i,2) &\leftarrow not\ c_{i,1}, c_{i,2}, body(r_i) \\
&\ldots \\
v_1(r_i,k) &\leftarrow not\ c_{i,1}, not\ c_{i,2}, \ldots, not\ c_{i,k-1}, c_{i,k}, body(r_i).
\end{aligned}
$$

6. Include rules

$$
\begin{aligned}
worse(O) &\leftarrow v_0(O,R,V_0), v_1(R,V_1), V_0 < V_1 \\
better(O) &\leftarrow v_0(O,R,V_0), v_1(R,V_1), V_0 > V_1 \\
&\leftarrow worse(O), not\ better(O).
\end{aligned}
$$

7. For every $O \in \mathscr{O}$, add a rule $\leftarrow \{l : l \in O\}, \{not\ l : l \in \mathscr{A}, l \notin O\}$.

The difference from $\Pi_2(P,S)$ is that $\Pi_3(P,\mathscr{O})$ includes the facts representing the satisfaction degree for all $O \in \mathscr{O}$ (in step 3), the rules restricting $S'$ (the outcome for $\Pi_3(P,\mathscr{O})$) is

different from every $O \in \mathscr{O}$ (in step 7), and a contradiction if $S'$ is worse than any $O \in \mathscr{O}$ (in step 6). Therefore the outcome for $\Pi_3(P, \mathscr{O})$ is different from and not worse than any $O \in \mathscr{O}$.

Similar to $\Pi_2(P,S)$, in the ranked version, we construct $\Pi_3(P,S)$ by replacing the rules defined in step 6 described above by the following rules:

$$
\begin{aligned}
worseAt(O,L) &\leftarrow v_0(O,R,V_0), v_1(R,V_1), V_0 < V_1, level(R,L) \\
failAt(O,L) &\leftarrow worseAt(O,L), v_0(O,R,V_0), v_1(R,V_1), V_0 > V_1, level(R,L_0), L_0 \leq L \\
&\leftarrow worseAt(O,L), not\ failAt(O,L).
\end{aligned}
$$

and adding the fact $level(r,j)$ for each preference $r \in P_{pref}$ in the form 3.2 in Chapter 3.

Given an ASO program $P = (P_{gen}, P_{pref})$ and a set $\mathscr{O}$ of interpretations, the complete process to solve the FIND-OPT-DIFF-SET problem is the following:

1. Compute an outcome $S_0$ for $\Pi_3(P, \mathscr{O})$. If $\Pi_3(P, \mathscr{O})$ is unsatisfiable, return no solution.
2. Let $S' = S_0$.
3. Loop
    a) Compute an outcome $M$ for $\Pi_1(P, S')$. If $\Pi_1(P, S')$ is unsatisfiable, return $S'$ as the solution for the FIND-OPT-DIFF-SET problem. Otherwise, let $S' = M$.

The other steps in the algorithm for the SIM-OPT-OUTCOME and DISSIM-OPT-OUTCOME problems are trivial.

**An Alternative Method** The first technique computes optimal outcomes one by one and checks the distance condition, while the second technique iteratively finds outcomes satisfying the distance condition and then checks their optimality. Given an ASO program $P$, an interpretation $S$ and a distance $d$, the algorithm is described below:

1. Let $\mathscr{O} = \varnothing$.

2. Loop

   a) Compute an outcome $M$ for $P$ such that $M$ is different from $S$ and every $O \in \mathscr{O}$, and $HD(S,M) \leq d$ (or respectively $HD(S,M) \geq d$). If no solution, return unsatisfiable.

   b) Check whether $M$ is optimal for $P$. If it is, return $M$ as the solution for the SIM-OPT-OUTCOME (DISSIM-OPT-OUTCOME) problem, otherwise, add $M$ to $\mathscr{O}$.

Similar to the straightforward method, the set $\mathscr{O}$ records the outcomes which cannot be the solution. The second step in the loop, checking whether $M$ is optimal for $P$, can be implemented by running $\Pi_1(P,M)$. If $\Pi_1(P,M)$ has a solution, $M$ is not an optimal outcome, otherwise, it is. To implement the first step in the loop, we first need to know how to find an outcome which is similar to (dissimilar from) a given interpretation.

**FIND-SIM** Given an ASO program $P$, an interpretation $S$ and a nonnegative integer $d$, find an outcome $M$ for $P$ such that $M \neq S$ and $HD(S,M) \leq d$.

**FIND-DISSIM** Given an ASO program $P$, an interpretation $S$ and a nonnegative integer $d$, find an outcome $M$ for $P$ such that $M \neq S$ and $HD(S,M) \geq d$.

To solve the FIND-SIM and FIND-DISSIM problems, we construct an answer set program $\Pi_4(P,S,d)$ by modifying $P_{gen}$. Assume $\mathscr{A} = \{l_1,\ldots,l_n\}$ are the atoms in $P_{gen}$ and $S$ is represented by a set of atoms with the truth value. The program $\Pi_4(P,S,d)$ is constructed in the following way:

1. Include $P_{gen}$.

2. Add facts $s_i \leftarrow$ if $l_i \in S$.

3. Add rule $\leftarrow \{l : l \in S\}, \{not\ l : l \in \mathscr{A}, l \notin S\}$.

4. Add rule $\{d_1,\ldots,d_n\}d$ for the FIND-SIM problem, or $d\{d_1,\ldots,d_n\}$ for the FIND-DISSIM problem.

5. For each $i \in [1..n]$, add rules

$$\begin{aligned}
&\leftarrow\ d_i, l_i, s_i\\
&\leftarrow\ d_i, not\ l_i, not\ s_i\\
&\leftarrow\ not\ d_i, l_i, not\ s_i\\
&\leftarrow\ not\ d_i, not\ l_i, s_i.
\end{aligned}$$

In the program $\Pi_4(P,S,d)$, atoms $d_i$ is true if and only if $S$ and $M$ (the outcome for $\Pi_4(P,S,d)$) have different values on atom $l_i$. The number of $d_i$s with the truth value is the Hamming distance between $S$ and $M$. The rule $\{d_1,\ldots,d_n\}d$ (or respectively $d\{d_1,\ldots,d_n\}$) restricts there are at most (least) $d$ of $d_i$s are true. This is equivalent to that the Hamming distance between $S$ and $M$ are at most (least) $d$. Similar to $\Pi_2(P,S)$, the rule $\leftarrow \{l : l \in S\}, \{not\ l : l \in \mathscr{A}, l \notin S\}$ guarantees that the outcome for $\Pi_4(P,S,d)$ is different from $S$.

In the algorithm, we can compute an outcome $M$ for $P$ such that $M$ is different from any $O \in \mathscr{O}$ ($\mathscr{O} = \{S\}$) and $HD(S,M) \leq d$ (or respectively $HD(S,M) \geq d$) by running $\Pi_4(P,S,d)$. If $M$ is not optimal for $P$, we need to compute another outcome $M'$ for $P$ such that $M'$ is different from $S$ and $M$, and $HD(S,M') \leq d$ (or respectively $HD(S,M') \geq d$). Therefore we have the following two problems.

**FIND-SIM-DIFF-SET** Given an ASO program $P$, an interpretation $S$, a set $\mathscr{O}$ of interpretations and a nonnegative integer $d$, find an outcome $M$ for $P$ such that $M \neq S$, $HD(S,M) \leq d$, and $M \neq O$ for every $O \in \mathscr{O}$.

**FIND-DISSIM-DIFF-SET** Given an ASO program $P$, an interpretation $S$, a set $\mathscr{O}$ of interpretations and a nonnegative integer $d$, find an outcome $M$ for $P$ such that $M \neq S$, $HD(S,M) \geq d$, and $M \neq O$ for every $O \in \mathscr{O}$.

The FIND-SIM-DIFF-SET and FIND-DISSIM-DIFF-SET problems can be solved by adding rules to $\Pi_4(P,S,d)$ which restrict the outcome is different from every $O \in \mathscr{O}$. The program $\Pi_5(P,S,\mathscr{O},d)$ for these two problems is a union of $\Pi_4(P,S,d)$ and the following rules:

$$\leftarrow \quad \{l : l \in O\}, \{not\ l : l \in \mathscr{A}, l \notin O\} \quad for\ each\ O \in \mathscr{O}.$$

The program $\Pi_4(P,S,d)$ is a special case of $\Pi_5(P,S,\mathscr{O},d)$ where $\mathscr{O}$ is empty. The alternative method for the SIM-OPT-OUTCOME and DISSIM-OPT-OUTCOME problems can be implemented by $\Pi_5(P,S,\mathscr{O},d)$ ($\mathscr{O}$ can be empty) and $\Pi_1(P,M)$ ($M$ is an intermediate result).

A variant of the alternative method changes the first step in the loop to computing a "local" optimal outcome $M$ for $P$ such that $M$ is different from $S$ and every $O \in \mathscr{O}$ and $HD(S,M) \leq d$ (or respectively $HD(S,M) \geq d$). Given a ASO program $P$, we construct a new ASO program $P'$ where $P'_{gen} = \Pi_4(P,S,d)$ and $P'_{pref} = P_{pref}$. Therefore the outcomes for $P'$ are different from and similar to (dissimilar from) $S$. The variant method is implemented by the following process:

1. Let $\mathscr{O} = \varnothing$.
2. Loop
    a) Compute an optimal outcome $M$ for $P'$ such that $M$ is different from every $O \in \mathscr{O}$. If no solution, return unsatisfiable.
    b) Check whether $M$ is optimal for $P$. If it is, return $M$ as the solution for the SIM-OPT-OUTCOME (DISSIM-OPT-OUTCOME) problem, otherwise, add $M$ to $\mathscr{O}$.

In the loop, the method first computes an optimal outcome $M$ for $P'$, which satisfies the distance condition but may be not optimal for $P$, and then checks whether $M$ is the solution. It uses the set $\mathscr{O}$ to record all failed candidates. The problem to compute an optimal

outcome which is different from a set of interpretations is defined as FIND-OPT-DIFF-SET and has been introduced in the straightforward method.

In the experiments, we implemented both versions for the alternative method.

### Computing a Diverse Set of Optimal Solutions

To solve the $k$-DIVERSE-OPT-OUTCOMES problem, we also provide two techniques. To find $k$ optimal outcomes such that every two of them are far from each other, one technique uses the straightforward backtracking algorithm which iteratively computes optimal outcomes until a diverse set is found. On the other hand, the second technique first computes a diverse set of outcomes, then checks whether every outcome is optimal.

**A Straightforward Method** The straightforward backtracking algorithm works in the following way. Let $\mathscr{O}$ be a set of optimal outcomes found so far, with every two different outcomes in $\mathscr{O}$ at distance at least $d$ from each other. Given that, the algorithm computes an optimal outcome $M$ which is far from every $O \in \mathscr{O}$. If such an $M$ is found, the search for such answer is suspended, $M$ is included in $\mathscr{O}$ and the algorithm repeats the process of trying to expand $\mathscr{O}$. Otherwise, $\mathscr{O}$ is not contained in any diverse set of optimal outcomes of cardinality $k$. Thus, the algorithm pops back the last optimal outcome added into $\mathscr{O}$, and computes a new optimal outcome that could be included in $\mathscr{O}$ (by resuming the search suspended most recently). The algorithm terminates when the size of $\mathscr{O}$ is $k$ or when there are no suspended calls to an ASP solver (the entire search space was covered). The set $\mathscr{O}$ is set to empty initially.

Given an ASO program $P$ and a nonnegative integer $d$, the algorithm is:

1. Let $\mathscr{O}, \mathscr{O}_1, \ldots, \mathscr{O}_k = \varnothing$.

2. Loop

a) If $|\mathscr{O}| = k$, return $\mathscr{O}$ as the solution for the $k$-DIVERSE-OPT-OUTCOMES problem.

b) Let $i = |\mathscr{O}| + 1$. Compute an optimal outcome $M$ for $P$ such that $M$ is different from every $O \in \mathscr{O}_i$, and $HD(M, O') \geq d$ for every $O' \in \mathscr{O}$.

c) If such $M$ exists, add $M$ to $\mathscr{O}$ and $\mathscr{O}_i$.

d) Otherwise, if $\mathscr{O} = \varnothing$, return unsatisfiable; else, pop back the last element in $\mathscr{O}$ and set $\mathscr{O}_i = \varnothing$.

In the algorithm, $\mathscr{O}$ is the set to record the diverse optimal outcomes, and the sets $\mathscr{O}_i$s are used to avoid an endless loop. In every iteration of the loop, we try to compute an optimal outcome $M$ which can be included in $\mathscr{O}$. It is clear that $M$ has to be far from every $O \in \mathscr{O}$. Moreover, we may already tried to include an outcome $M'$ to $\mathscr{O}$ but failed. That means $\mathscr{O} \cup \{M'\}$ cannot be extended to a diverse optimal set. Therefore we do not want to test $M'$ again. To avoid an infinite loop, we use $\mathscr{O}_i$ to record the outcomes already been tested to add into $\mathscr{O}$ where $i = |\mathscr{O}| + 1$. When computing $M$, $M$ has to be different from every $O \in \mathscr{O}_i$. When $\mathscr{O}$ is changed (an element in $\mathscr{O}$ is popped out), the set $\mathscr{O}_i$ is reset to empty.

To implement this algorithm, the key point is how to find an optimal outcome which is different from every $O \in \mathscr{O}$ and far from every $O \in \mathscr{O}'$, where $\mathscr{O}$ and $\mathscr{O}'$ are two given sets of interpretations.

**FIND-OPT-DIFF-DISSIM-SET** Given an ASO program $P$, two sets of interpretations $\mathscr{O}$ and $\mathscr{O}'$ and a nonnegative integer $d$, find an optimal outcome $M$ for $P$ such that $M \neq O$ for every $O \in \mathscr{O}$, and $HD(M, O') \geq d$ for every $O' \in \mathscr{O}'$.

Since the FIND-OPT-DIFF-DISSIM-SET problem is to compute an optimal outcome, it cannot be modeled by a single polynomial-size answer set program. Inspired by the DISSIM-OPT-OUTCOME problem, we introduce two techniques to solve it based on the techniques we discussed in section 4.2.1.

The first technique for the DISSIM-OPT-OUTCOME problem uses a straightforward method which iteratively computes optimal outcomes and checks whether the distance condition is satisfied. We modify that algorithm to solve the FIND-OPT-DIFF-DISSIM-SET problem as iteratively computing optimal outcomes which is different from every $O \in \mathscr{O}$, and checking whether it is far from every $O \in \mathscr{O}'$.

Given an ASO program $P$, two sets of interpretations $\mathscr{O}$ and $\mathscr{O}'$ and a nonnegative integer $d$, the first algorithm for the FIND-OPT-DIFF-DISSIM-SET problem is:

1. Loop

    a) Compute an optimal outcome $M$ for $P$ such that $M$ is different from every $O \in \mathscr{O}$. If no solution, return unsatisfiable.

    b) If $HD(M,O) \geq d$ for every $O \in \mathscr{O}'$, return $M$ as the solution for the FIND-OPT-DIFF-DISSIM-SET problem.

    c) Otherwise, add $M$ to $\mathscr{O}$.

The corresponding algorithm for the DISSIM-OPT-OUTCOME problem is a special case of this one with $\mathscr{O} = \{S\}$ and $\mathscr{O}' = \{S\}$. The problem in the first step of the loop, to find an optimal outcome which is different from each of the given interpretations, has been defined as FIND-OPT-DIFF-SET and solved when we discussed the first technique for the DISSIM-OPT-OUTCOME problem.

The second technique to solve the DISSIM-OPT-OUTCOME problem computes dissimilar outcomes iteratively and checks whether it is optimal. Similarly, to make the algorithm apply to the FIND-OPT-DIFF-DISSIM-SET problem, we modify it as computing the outcomes which is different from every $O \in \mathscr{O}$ and far from every $O \in \mathscr{O}'$, and checks whether it is optimal.

Given an ASO program $P$, two sets of interpretations $\mathscr{O}$ and $\mathscr{O}'$ and a nonnegative integer $d$, the modified algorithm for the FIND-OPT-DIFF-DISSIM-SET problem is:

1. Loop

    a) Compute an outcome $M$ for $P$ such that $M$ is different from every $O \in \mathscr{O}$, and $HD(M,O') \geq d$ for every $O' \in \mathscr{O}'$. If no solution, return unsatisfiable.

    b) Check whether $M$ is optimal for $P$. If it is, return $M$ as the solution for the FIND-OPT-DIFF-DISSIM-SET problem, otherwise, add $M$ to $\mathscr{O}$.

Still, the corresponding algorithm for the DISSIM-OPT-OUTCOME problem is a special case of this one with $\mathscr{O} = \{S\}$ and $\mathscr{O}' = \{S\}$. To implement this algorithm, we need to first solve the problem in the first step of the loop.

**FIND-DIFF-SET-DISSIM-SET** Given an ASO program $P$, two sets of interpretations $\mathscr{O}$ and $\mathscr{O}'$ and a nonnegative integer $d$, find an outcome $M$ for $P$ such that $M \neq O$ for every $O \in \mathscr{O}$, and $HD(M,O') \geq d$ for every $O' \in \mathscr{O}'$.

When introducing the second technique for the DISSIM-OPT-OUTCOME problem, we defined the FIND-DISSIM-DIFF-SET problem which can be solved by an answer set program $\Pi_5(P,S,\mathscr{O},d)$. The FIND-DIFF-SET-DISSIM-SET problem can be solved in a similar way by constructing an answer set program $\Pi_6(P,\mathscr{O},\mathscr{O}',d)$. Given an ASO program $P$, two sets of interpretations $\mathscr{O}$ and $\mathscr{O}'$ and a nonnegative integer $d$, let $\mathscr{A} = \{l_1,\ldots,l_n\}$ be the atoms in $P_{gen}$, the program $\Pi_6(P,\mathscr{O},\mathscr{O}',d)$ is constructed in the following way:

1. Include $P_{gen}$.

2. For every $O \in \mathscr{O}$, add rule $\leftarrow \{l : l \in O\}, \{not\ l : l \in \mathscr{A}, l \notin S\}$.

3. For every $O \in \mathscr{O}'$, add facts $s_{O,i} \leftarrow$ if $l_i \in O$.

4. For every $O \in \mathscr{O}'$, add rule $d\{d_{O,1},\ldots,d_{O,n}\}$.

5. For every $O \in \mathscr{O}'$ and every $i \in [1..n]$, add rules

$$\begin{aligned} \leftarrow \quad & d_{O,i}, l_i, s_{O,i} \\ \leftarrow \quad & d_{O,i}, \mathit{not}\ l_i, \mathit{not}\ s_{O,i} \\ \leftarrow \quad & \mathit{not}\ d_{O,i}, l_i, \mathit{not}\ s_{O,i} \\ \leftarrow \quad & \mathit{not}\ d_{O,i}, \mathit{not}\ l_i, s_{O,i}. \end{aligned}$$

The rules added in step 2 guarantee that the outcomes for $\Pi_6(P, \mathscr{O}, \mathscr{O}', d)$ are different from every $O \in \mathscr{O}$. The fact $s_{O,i}$ is true if the atom $l_i$ is true in the interpretation $O$. The rules added in step 4 restrict that for every $O \in \mathscr{O}'$, at least $d$ of $d_{O,i}$s are true. Similar to the program $\Pi_5(P, S, \mathscr{O}, d)$, the atom $d_{O,i}$ is true if and only if the outcome of $\Pi_6(P, \mathscr{O}, \mathscr{O}', d)$ and $O$ have different values on the atom $l_i$. Therefore these rules guarantee that the Hamming distance between the outcomes for $\Pi_6(P, \mathscr{O}, \mathscr{O}', d)$ and every $O \in \mathscr{O}'$ is at least $d$.

The second technique for the FIND-OPT-DIFF-DISSIM-SET problem can be implemented with the program $\Pi_6(P, \mathscr{O}, \mathscr{O}', d)$. Similar to the situations we discussed for the DISSIM-OPT-OUTCOME problem, this technique also has a variant which computes "local" optimal outcomes as the candidates for the final solution. To implement this variant method, given an instance of the FIND-OPT-DIFF-DISSIM-SET problem, we construct an ASO program $P'$ where $P'_{gen} = \Pi_6(P, \mathscr{O}, \mathscr{O}', d)$ and $P'_{pref} = P_{pref}$. Therefore the outcomes for $P'$ are different from every $O \in \mathscr{O}$ and far from every $O \in \mathscr{O}'$. The variant method is implemented by the following process:

1. Let $\mathscr{O}'' = \varnothing$.
2. Loop
    a) Compute an optimal outcome $M$ for $P'$ such that $M$ is different from every $O \in \mathscr{O}''$. If no solution, return unsatisfiable.
    b) Check whether $M$ is optimal for $P$. If it is, return $M$ as the solution for the FIND-OPT-DIFF-DISSIM-SET problem. Otherwise, add $M$ to $\mathscr{O}''$.

In the loop, the method first computes an optimal outcome $M$ for $P'$, which satisfies other conditions but may be not optimal for $P$, then checks whether $P$ is the solution. If it is not, the method repeats the process to compute a new candidate. The set $\mathscr{O}''$ records all failed candidates to avoid an endless loop.

Once the FIND-OPT-DIFF-DISSIM-SET problem is solved, the algorithm for the $k$-DIVERSE-OPT-OUTCOMES problem can be implemented. However, this algorithm needs to remember all optimal outcomes that have been found and requires a large amount of storage. Moreover, it can be improved since it actually tests all permutations for the $k$ diverse optimal outcomes. For example, let $\{S_1, S_2\}$ be a set of optimal outcomes which cannot be extended to a $k$ diverse set, the algorithm checks both $\{S_1, S_2\}$ and $\{S_2, S_1\}$. We also compare these three implementations in the experiments (the first technique and two variants for the second technique).

**An Alternative Method** To solve the $k$-DIVERSE-OPT-OUTCOMES problem, the idea of the alternative method is first finding diverse sets, then checking whether every outcome in the set is optimal. Given an instance of the $k$-DIVERSE-OPT-OUTCOMES problem, we first see how to compute $k$ diverse outcomes for $P$.

**FIND-DIVERSE** Given an ASO program $P$, an integer $k > 1$, and a nonnegative integer $d$, find a set $\mathscr{O}$ of $k$ outcomes for $P$ such that $HD(S, S') \geq d$ for every different $S, S' \in O$.

The FIND-DIVERSE problem can be solved by constructing an answer set program $\Pi_7(P, k, d)$ in the following way:

1. For each $i \in [1..k]$, include $P^i_{gen}$ obtained from $P_{gen}$ by replacing each $l_j$, $j \in [1..n]$, with $l_{i,j}$.

2. For every different $i, j \in [1..k]$, add rule $d\{d_{i,j,1}, \ldots, d_{i,j,n}\}$.

3. For every $i, j \in [1..k]$, $i < j$, and every $h \in [1..n]$, add rules

$$\begin{aligned} \leftarrow \quad & d_{i,j,h}, l_{i,h}, l_{j,h} \\ \leftarrow \quad & d_{i,j,h}, not\ l_{i,h}, not\ l_{j,h} \\ \leftarrow \quad & not\ d_{i,j,h}, l_{i,h}, not\ l_{j,h} \\ \leftarrow \quad & not\ d_{i,j,h}, not\ l_{i,h}, l_{j,h}. \end{aligned}$$

The program $\Pi_7(P,k,d)$ includes $k$ copies of $P_{gen}$ to generate $k$ outcomes for $P_{gen}$ and the rules representing the distance between every two of them is at least $d$. Therefore the outcome for $\Pi_7(P,k,d)$ is a combination of $k$ diverse outcomes for $P$.

Given an ASO program $P$, and a nonnegative integer $d$, the algorithm for the alternative method is:

1. Let $P'$ be an ASO program where $P'_{gen} = \Pi_7(P,k,d)$ and $P'_{pref} = P_{pref}$.
2. Let $\mathscr{O} = \varnothing$.
3. Loop
    a) Compute an optimal outcome $M$ for $P'$ such that $M$ is different from every $O \in \mathscr{O}$. If no solution, return unsatisfiable.
    b) Check whether all $k$ outcomes in $M$ are optimal for $P$. If yes, return $M$ as the solution for the $k$-DIVERSE-OPT-OUTCOMES problem. Otherwise, add $M$ into $\mathscr{O}$.

The algorithm computes optimal outcomes for $P'$ iteratively, and checks each of them whether it is a set of optimal outcomes for $P$.

### 4.2.2 The Disjunctive Logic Program Encoding

To solve reasoning tasks for ASO programs, the iterative method makes several calls to an ASP solver. Therefore, it is essentially procedural. Moreover, in some cases, it suffers

from large memory demands. Our second method compiles the entire computation into a disjunctive logic program so that a single call to an ASP solver can solve it. The complexity results guarantee that casting the problems as as disjunctive logic programs is possible.

To this end, we first describe the basic problem of computing an optimal outcome of an ASO program in terms of a quantified boolean formula (QBF) and then apply a version of the Eiter-Gottlob translation [44] to produce the program. We provide the details for the problem of computing a single optimal outcome. Other optimization problems can be handled similarly, as they essentially differ only in additional constraints one needs to impose on the generator.

Let $P$ be an ASO program with a generator given as a propositional formula (the case of an answer set program is similar). We denote by $Z$ the set of atoms in $P$ and assume that $Z = \{z_1, \ldots, z_n\}$. We introduce two new sets of atoms $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$. We assume that the sets $Z, X$ and $Y$ are pairwise disjoint. Intuitively, for each $i = 1, \ldots, n$, $x_i$ and $y_i$ are "copies" of $z_i$. We write $P^X_{gen}$ and $P^Y_{gen}$ for the copies of the generator $P_{gen}$ obtained by replacing its atoms with their copies in $X$ and $Y$, respectively. In this way, there are obvious one-to-one correspondence between interpretations over $Z$ that are models of $P_{gen}$, interpretations over $X$ that are models of $P^X_{gen}$, and interpretations over $Y$ that are models of $P^Y_{gen}$.

We will now build a logical representation of the preference relation determined by the selector $P_{pref}$. As before, for a formula $A$ occurring in a preference in $P_{pref}$, we write $A^X$ and $A^Y$ for its copies over $X$ and $Y$, respectively. Let $I$ and $J$ be interpretations over $Z$ and let $p \in P_{pref}$, say

$$p = \quad C_1 > \cdots > C_k \leftarrow B.$$

We will define a formula $\psi_p(X,Y)$ over the set $X \cup Y$ that will capture the condition that $I$ is strictly better than $J$ on $p$. We recall that the interpretation $I$ is strictly better than $J$ on $p$ if and only if one of the following two conditions holds:

1. Both $I$ and $J$ satisfy the condition $R = B \wedge (C_1 \vee \cdots \vee C_k)$, $I$ satisfies $C_j$ for some $j$,

$1 \leq j \leq k-1$, and $J$ does not satisfy any $C_{j'}$, where $1 \leq j' \leq j$;

2. $I$ does not satisfy $R$, $J$ satisfies $R$ and $J$ does not satisfy $C_1$.

We define auxiliary formulas $u_p(X,Y)$ and $v_p(X,Y)$ (over $X \cup Y$) as follows:

$$\begin{aligned} u_p(X,Y) = & R^X \wedge R^Y \wedge \\ & ((C_1^X \wedge \neg C_1^Y) \vee (C_2^X \wedge \neg C_1^Y \wedge \neg C_2^Y) \vee \cdots \vee \\ & (C_{k-1}^X \wedge \neg C_1^Y \wedge \cdots \wedge \neg C_{k-1}^Y)) \end{aligned}$$

and

$$v_p(X,Y) = \neg R^X \wedge R^Y \wedge \neg C_1^Y.$$

Intuitively, these formulas capture the conditions (1) and (2) above, respectively. We now define $\psi_p(X,Y) = u_p(X,Y) \vee v_p(X,Y)$. Formally, we have the following property.

**Proposition 1.** *Let $I$ and $J$ be interpretations over $Z$. Then, $I$ is strictly better than $J$ on a preference $p = C_1 > \cdots > C_k \leftarrow B$ if and only if the interpretation $I^X \cup J^Y$ (over $X \cup Y$) is a model of $\psi_p(X,Y)$.*

*Proof.* Let us notice that for every formulas $A$ over $Z$, and every interpretation $I$ over $Z$, $I \models A$ if and only if $I^X \models A^X$ if and only if $I^Y \models A^Y$. Thus, if $I$ is strictly better than $J$ on $p$ then the condition (1) or the condition (2) holds. Say, the condition (1) holds. It follows that $I^X \models R^X$, $J^Y \models R^Y$ and, for some $i$, $1 \leq i \leq k-1$, $I^X \models C_i^X$ and $J^Y \models \neg C_1^Y \wedge \cdots \wedge \neg C_i^Y$. Since $X$ and $Y$ are disjoint, $I^X \cup J^Y \models u_p(X,Y)$. The arguments for the case when the condition (2) holds and for the converse statement are similar. □

Let $P$ be a set of preferences, we also define

$$\varphi_P(X,Y) = \bigvee \{ \psi_p(X,Y) \colon p \in P \}$$

and

$$F_P(X,Y) = \varphi_P(X,Y) \wedge \neg \varphi_P(Y,X).$$

Directly from the definition of the preference relation determined by an unranked preference profile and from Proposition 1 we have the following property.

**Proposition 2.** *Let $I$ and $J$ be interpretations over $Z$. Then, $I$ is strictly better than $J$ with respect to an unranked preference profile $P$ if and only if the interpretation $I^X \cup J^Y$ (over $X \cup Y$) is a model of $F_P(X,Y)$.*

*Proof.* Based on the Proposition 1, $I$ is strictly better than $J$ on a preference $p$ if and only if the interpretation $I^X \cup J^Y$ is a model of $\psi_p(X,Y)$. Similarly, we can see there is some $p \in P$ such that $I \succ_p J$ if and only if $I^X \cup J^Y$ is a model of $\varphi_P(X,Y)$. According to the Pareto principle, $I \succ_P J$ if and only if $I \succeq_p J$ for every $p \in P$ and $I \succ_{p'} J$ for some $p' \in P$. The condition $I \succeq_p J$ for every $p \in P$ is equivalent to $J \not\succ_p I$ for every $p \in P$ which holds if and only if $I^X \cup J^Y$ is a model of $\neg\varphi_P(Y,X)$. Therefore $I \succ_P J$ if and only if $I^X \cup J^Y$ is a model of $F_P(X,Y)$. □

Let $P = (P_{gen}, P_{pref})$ be an unranked ASO program. We now define

$$\Phi(P) = \exists X \forall Y (P^X_{gen} \wedge (\neg P^Y_{gen} \vee \neg F_{P_{pref}}(Y,X))).$$

From the results obtained above we have the following property.

**Theorem 11.** *Let $P$ be an unranked ASO program. An interpretation $I$ (over $Z$, the set of atoms of $P$) is an optimal outcome for $P$ if and only if $I^X$ is an interpretation over $X$ witnessing the truth of the QBF $\Phi(P)$.*

*Proof.* An interpretation $I$ is an optimal outcome for $P$ if and only if no outcome $J$ for $P$ dominates $I$. Let $I^X$ be an interpretation over $X$ witnessing the truth of the QBF $\Phi(P)$. Then, $I^X \cup J^Y$ is a model of $P^X_{gen} \wedge (\neg P^Y_{gen} \vee \neg F_{P_{pref}}(Y,X))$ for any interpretation $J^Y$ over $Y$. Therefore $I^X \cup J^Y \models P^X_{gen}$ and $I$ is an outcome for $P$. Also since $I^X \cup J^Y \models \neg P^Y_{gen} \vee \neg F_{P_{pref}}(Y,X)$, the interpretation $J^Y$ is not an outcome for $P$ or $J$ is not strictly better than $I$. Therefore $I$ is an optimal outcome for $P$. If $I$ is an optimal outcome for $P$, it is clear that $I^X \models P^X_{gen}$ and $I^X \cup J^Y \models \neg F_{P_{pref}}(Y,X)$ for any interpretation $J^Y$ over $Y$ and $J^Y \models P^Y_{gen}$. Thus

$I$ is an optimal outcome for $P$ if $I^X \cup J^Y$ is a model of $P^X_{gen} \wedge (\neg P^Y_{gen} \vee \neg F_{P_{pref}}(Y,X))$ for any interpretation $J^Y$ over $Y$, which means $I^X$ is an interpretation witnessing the truth of the QBF $\Phi(P)$. □

For a ranked preference profile, let $P = \{P^1, \ldots, P^l\}$ be a sequence of pairwise disjoint preference profiles $P^i$. The rank of a preference rule $p \in P$ is the unique index $i$ for which $p \in P^i$. Let $I$ and $J$ be two interpretations over $Z$ (the set of atoms of $P$), $I \succ_P J$ if and only if there is a preference rule $p \in P$ such that

1. $I \succ_p J$
2. $I \succeq_{p'} J$ for every $p' \in P$ and $rank(p') = rank(p)$
3. $I \approx_{p'} J$ for every $p' \in P$ and $rank(p') < rank(p)$.

Let us denote $I$ is strictly better than $J$ on the preferences of rank $i$, $P^i$, by $I \succ_{P^i} J$. Similarly, we denote $I$ is equivalent to $J$ on the preferences of rank $i$, by $I \approx_{P^i} J$. Therefore, for a ranked profile $P$, $I \succ_P J$ if and only if there is a rank $i$ such that $I \succ_{P^i} J$ and $I \approx_{P^j} J$ for every $j \in [1..i-1]$.

We recall the interpretation $I$ is equivalent to $J$ on a preference $p$ if and only if one of the following conditions holds:

1. Both $I$ and $J$ satisfy the condition $R = B \wedge (C_1 \vee \cdots \vee C_k)$, for some $j$, $1 \leq j \leq k-1$, both $I$ and $J$ satisfy $C_j$, and neither $I$ or $J$ satisfies any $C_{j'}$, where $1 \leq j' < j$;
2. Neither $I$ or $J$ satisfies $R$;
3. $I$ does not satisfy $R$, $J$ satisfies $R$ and $C_1$;
4. $J$ does not satisfy $R$, $I$ satisfies $R$ and $C_1$.

Correspondingly, we define formulas $w_p(X,Y)$, $x_p(X,Y)$, $y_p(X,Y)$ and $z_p(X,Y)$ as follows:

$$
\begin{aligned}
w_p(X,Y) \;=\; & R^X \wedge R^Y \wedge \\
& ((C_1^X \wedge C_1^Y) \vee (C_2^X \wedge C_2^Y \wedge \neg C_1^X \wedge \neg C_1^Y) \vee \cdots \vee \\
& (C_k^X \wedge C_k^Y \wedge \neg C_1^X \wedge \neg C_1^Y \wedge \cdots \wedge \neg C_{k-1}^X \wedge \neg C_{k-1}^Y)),
\end{aligned}
$$

$$
x_p(X,Y) \;=\; \neg R^X \wedge \neg R^Y,
$$

$$
y_p(X,Y) \;=\; \neg R^X \wedge R^Y \wedge C_1^Y,
$$

and

$$
z_p(X,Y) \;=\; \neg R^Y \wedge R^X \wedge C_1^X.
$$

Finally we define $\omega_p(X,Y) = w_p(X,Y) \vee x_p(X,Y) \vee y_p(X,Y) \vee z_p(X,Y)$.

**Proposition 3.** *Let $I$ and $J$ be interpretations over $Z$. Then, $I$ is equivalent to $J$ on a preference $p = C_1 > \cdots > C_k \leftarrow B$ if and only if the interpretation $I^X \cup J^Y$ (over $X \cup Y$) is a model of $\omega_p(X,Y)$.*

*Proof.* Let us notice that for every formulas $A$ over $Z$, and every interpretation $I$ over $Z$, $I \models A$ if and only if $I^X \models A^X$ if and only if $I^Y \models A^Y$. Thus, if $I$ is equivalent to $J$ on $p$ then one of the four conditions listed above holds. Say, the condition (1) holds. It follows that $I^X \models R^X$, $J^Y \models R^Y$ and, for some $i$, $1 \leq i \leq k-1$, $I^X \models C_i^X$, $J^Y \models C_i^Y$, $I^X \models \neg C_1^X \wedge \cdots \wedge \neg C_i^X$ and $J^Y \models \neg C_1^Y \wedge \cdots \wedge \neg C_i^Y$. Since $X$ and $Y$ are disjoint, $I^X \cup J^Y \models w_p(X,Y)$. The arguments for the case when one of the rest three conditions holds and for the converse statement are similar. Thus $I$ is equivalent to $J$ on $p$ if and only if $I^X \cup J^Y$ is a model of $\omega_p(X,Y)$. □

Let $P = \{P^1, \ldots, P^l\}$ be a ranked preference profile. Then we define

$$
F'_P(X,Y) = \bigvee \{F_{P^i}(X,Y) \wedge \bigwedge \{\omega_p(X,Y) \colon p \in P^1 \cup \cdots \cup P^{i-1}\} \colon i \in [1..l]\}.
$$

**Proposition 4.** *Let $I$ and $J$ be interpretations over $Z$. Then, $I$ is strictly better than $J$ with respect to a ranked profile $P = \{P^1, \ldots, P^l\}$ if and only if the interpretation $I^X \cup J^Y$ (over $X \cup Y$) is a model of $F'_P(X,Y)$.*

*Proof.* Based on the Pareto principle, $I$ is strictly better than $J$ with respect to $P$ if and only if there is a rank $i$ such that $I \succ_{P^i} J$ and $I \approx_{P^j} J$ for every $j \in [1..i-1]$. According to the Proposition 2, $I \succ_{P^i} J$ if and only if the interpretation $I^X \cup J^Y$ is a model of $F_{P^i}(X,Y)$. According to the Proposition 3, $I \approx_p J$ if and only if the interpretation $I^X \cup J^Y$ is a model of $\omega_p(X,Y)$. Based on the Pareto principle, $I \approx_{P^j} J$ for every $j \in [1..i-1]$ if $I \approx_p J$ for every $p \in P^1 \cup \cdots \cup P^{i-1}$. Thus $I$ is strictly better than $J$ with respect to $P$ if and only if $I^X \cup J^Y$ is a model of $F'_P(X,Y)$. □

Let $P = (P_{gen}, P_{pref})$ be a ranked ASO program. Similarly, we define

$$\Phi'(P) = \exists X \forall Y (P^X_{gen} \wedge (\neg P^Y_{gen} \vee \neg F'_{P_{pref}}(Y,X))).$$

**Theorem 12.** *Let $P$ be a ranked ASO program. An interpretation $I$ (over $Z$, the set of atoms of $P$) is an optimal outcome for $P$ if and only if $I^X$ is an interpretation over $X$ witnessing the truth of the QBF $\Phi'(P)$.*

*Proof.* The proof follows that of Theorem 11. □

Since an unranked preference profile is a special case of a ranked preference profile, we will not explicitly consider unranked profiles for the problems below.

For the DIFF-OPT-OUTCOME problem to compute an optimal outcome of an ASO program which is different from a given interpretation $S$, we construct an ASP program $P^X_{diff}(P,S)$ generating the answer sets of $P^X_{gen}$ which are different from $S$ by adding the following rule to $P^X_{gen}$:

$$\leftarrow \{x_i : z_i \in S\}, \{not\ x_i : z_i \notin S\}.$$

Then we define

$$\Phi_{diff}(P,S) = \exists X \forall Y (P^X_{diff}(P,S) \wedge (\neg P^Y_{gen} \vee \neg F'_{P_{pref}}(Y,X))).$$

**Theorem 13.** *Let $P$ be an ASO program and $S$ be a candidate outcome (an interpretation). An interpretation $I$ (over $Z$, the set of atoms of $P$) is an optimal outcome for $P$ such that $I \neq S$ if and only if $I^X$ is an interpretation over $X$ witnessing the truth of the QBF $\Phi_{diff}(P)$.*

*Proof.* The proof follows that of Theorem 11. □

For the SIM-OPT-OUTCOME and DISSIM-OPT-OUTCOME problems, given an ASO program $P$, an interpretation $S$ and a nonnegative integer $d$, the target is to find an optimal outcome $S'$ for P such that $HD(S,S') \leq d$ or $HD(S,S') \geq d$ respectively. The FIND-SIM and FIND-DISSIM problems defined in Section 4.2.1 compute a similar or dissimilar outcome for a given ASO program and a given interpretation by an ASP program $\Pi_4(P,S,d)$. The answer sets of $\Pi_4(P,S,d)$ are answer sets for $P_{gen}$ which are similar to or dissimilar from the given interpretation $S$. Let $\Pi_4^X(P,S,d)$ be the copy of $\Pi_4(P,S,d)$ over $X$, we define

$$\Phi_{sim/dis}(P,S,d) = \exists X \forall Y (\Pi_4^X(P,S,d) \wedge (\neg P_{gen}^Y \vee \neg F'_{P_{pref}}(Y,X))).$$

**Theorem 14.** *Let $P$ be an ASO program, $S$ be an interpretation, and $d$ be a non-negative integer. An interpretation $I$ (over $Z$, the set of atoms of $P$) is an optimal outcome for $P$ such that $\Delta(S,I) \leq d$ (or $\Delta(S,I) \geq d$) if and only if $I^X$ is an interpretation over $X$ witnessing the truth of the QBF $\Phi_{sim/dis}(P)$.*

*Proof.* The proof follows that of Theorem 11. □

For the $k$-DIVERSE-OPT-OUTCOMES problem, given an ASO program $P$ and a non-negative integer $d$, we want to find a set of optimal outcomes for $P$ with size $k$ such that the distance between each two outcomes is at least $d$. In Section 4.2.1, we define the FIND-DIVERSE problem to compute a diverse set of outcomes by an ASP program $\Pi_7(P,k,d)$. The program $\Pi_7(P,k,d)$ contains $k$ copies of $P_{gen}$ to generate $k$ outcomes for $P$. Without loss of generality, let us assume the atoms in $\Pi_7(P,k,d)$ are $X^1 \cup \cdots \cup X^k$ where $X^i = \{x_1^i, \ldots, x_n^k\}$. Then we can define

$$\Phi_{div}(P,k,d) = \exists X^1 \ldots X^k \forall Y (\Pi_7(P,k,d) \wedge (\neg P_{gen}^Y \vee \bigwedge \{\neg F'_{P_{pref}}(Y,X^i) : i \in [1..k]\})).$$

**Theorem 15.** *Let $P$ be an ASO program, $S$ be an interpretation, $d$ be a non-negative integer, and $k$ be a positive integer. A set $\mathscr{O} = \{I_1, \ldots, I_k\}$ is a set of optimal outcomes for $P$ such that $\Delta(I_i, I_j) \geq d$ for any two distinct $i, j \in [1..k]$, if and only if $I^{X^1} \cup \cdots \cup I^{X^k}$ is an interpretation over $X^1 \cup \cdots \cup X^k$ witnessing the truth of the QBF $\Phi_{div}(P)$.*

*Proof.* The proof follows that of Theorem 11. □

### 4.3 Experiments and Analysis

All the methods we developed were implemented in C/C++. All experiments were conducted on an Intel processor clocked at 2.30GHz with 4GB memory. We experimented with the two computational methods described in Section 4.2 with the ASP grounder *gringo-3.0.5* and solver *claspD-2.0-R6814*. In the discussion below we simply write Iterative and DLP, respectively to denote the two methods described in Section 4.2.

In experiments we used ASO programs in which generators are represented by 3-CNF formulas. To construct them, we randomly generated 3-CNF formulas with $n$ atoms and $4n$ clauses. The numbers of clauses are below the threshold ratio of $4.25n$ to ensure that with probability close to 1 generators have models (the space of feasible solutions is not empty). This is important as we are interested in studying optimization problems of selecting optimal solutions and not the satisfiability problem.

We generated preference rules for each selector through the following three mechanisms:

1. randomly generating $3n$ preference rules without rank ($n$ being as before the number of atoms)

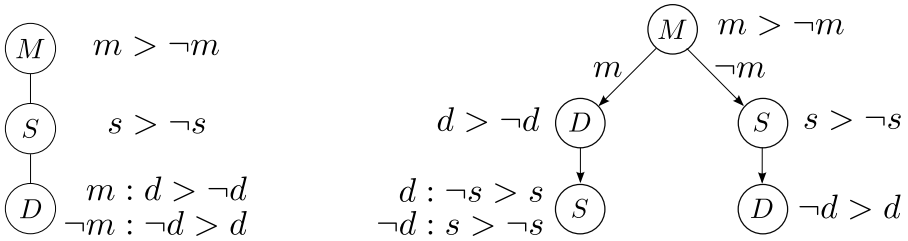
2. randomly generating $3n$ rules with two ranks, half of the rules having rank 1 and half of the rules having rank 2

3. extracting preference rules from two *lexicographic preference trees* (LP-trees, for short) [10].

In the first two cases, we generate one preference rule by randomly choosing a variable, say $x$, and forming the head to be of the form $x > \neg x$ or $\neg x > x$, choosing between them with equal probability. Each rule has no condition or has a condition of at most two literals. With the probability 0.5 it has no condition, having a one literal condition or two literal condition are equally likely. Literals for the conditions are generated uniformly at random.

LP trees are concise representations of strict total orders [10]. A complete LP-tree is a complete binary tree where each node corresponds to a variable and is assigned a preference table representing the preference over this variable conditioned by the values of ancestor nodes. Every variable appears exactly once on every path from the root to a leaf. Each leaf corresponds to an outcome. Preferences encoded by LP trees can be represented as ranked preference rules, with as many ranks as there are atoms in the language. When the two subtrees for some node are identical, they can be collapsed into one subtree [78]. In the third case, we generate collapsed LP-trees and transform them to preference profiles. We subject the LP-trees to some restrictions to make their sizes linear in the number of atoms: each tree may have at most one node where it splits into two branches, and each node's preferences depend on values in at most one ancestor node.

Let us consider the same example we used before to illustrate the LP-trees which generate the preference rules. An agent wants to have lunch in a restaurant and we want to use an LP-tree to represent her preferences on the menu. Given three binary variables: $M$ (*main dish*), with two values $m$ (*beef*) and $\neg m$ (*fish*); $S$ (*side dish*) with two values $s$ (*soup*) and $\neg s$ (*salad*); and $D$ (*drink*) with two values $d$ (*beer*) and $\neg d$ (*white wine*). Figure 4.1 shows two types of LP-trees allowed to be used in the experiment. The LP-tree in Figure 4.1a has no split, and all atoms are ordered strictly. In this LP-tree, the most important variable for the agent is *main dish* with the preference that *beef* is preferred to *fish*. No matter what value the top variable has, the next important variable is *side dish* and *soup* is preferred to *salad*. The last variable considered by the agent is *drink*, with *beer* is better than *white wine* if the *main dish* is *beef*, and *white wine* is better than *beer* otherwise. In Figure 4.1b, the LP-tree




(a) No split (b) Split to two branches

Figure 4.1: LP-trees

is split to two branches from the top node. The most important variable for the agent is still *main dish* with the preference that *beef* is preferred to *fish*. If she has *beef*, the second most important variable is *drink* with the preference to *beer* over *white wine*, and the least important variable is *side dish* where she prefers *salad* to *soup* if she had *beer*, and *soup* to *salad* otherwise. If she has *fish*, *side dish* is more important than *drink* and the preference for *side dish* is that *soup* is preferred to *salad*, and finally she prefers *white wine* to *beer* no matter what she has for *side dish*.

The preferences represented by an LP-tree can be equivalently represented by a set of ranked preference rules. Assuming that node $t$ in level $l$ is labeled with variable $X_i$ with preference $x_i > \neg x_i$, the equivalent preference rule should be $x_i > \neg x_i$ with rank $l$ and conditioned by the values assigned in the path from the root to $t$. In our example, the equivalent preference profile for the LP-tree in Figure 4.1a is:

$$
\begin{aligned}
m &> \neg m \overset{1}{\leftarrow} \\
s &> \neg s \overset{2}{\leftarrow} \\
d &> \neg d \overset{3}{\leftarrow} m \\
\neg d &> d \overset{3}{\leftarrow} \neg m
\end{aligned}
$$

and the profile for the LP-tree in Figure 4.1b is:

$$
\begin{aligned}
m &> \neg m \overset{1}{\leftarrow} \\
d &> \neg d \overset{2}{\leftarrow} m \\
s &> \neg s \overset{2}{\leftarrow} \neg m \\
\neg s &> s \overset{3}{\leftarrow} m, d \\
s &> \neg s \overset{3}{\leftarrow} m, \neg d \\
\neg d &> d \overset{3}{\leftarrow} \neg m
\end{aligned}
$$

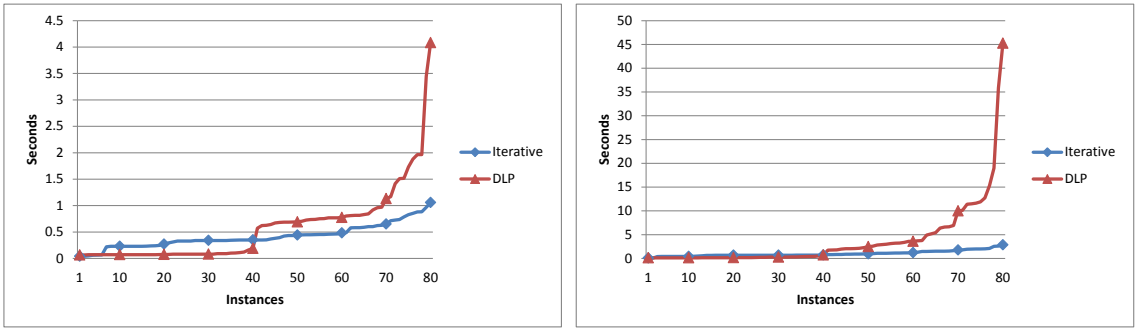
In the third case, the preference profile for each instance consists of the ranked preference rules from two randomly generated LP trees (one LP tree has only one optimal outcome).

We consider five computational problems: OPT-OUTCOME, DIFF-OPT-OUTCOME, SIM-OPT-OUTCOME, DISSIM-OPT-OUTCOME and $k$-DIVERSE-OPT-OUTCOMES, and experiment with the iterative and DLP-based methods we developed for them. For the last three problems, we have several versions of the iterative method. For the SIM-OPT-OUTCOME and DISSIM-OPT-OUTCOME problems, we have three approaches of the Iterative method introduced in Section 4.2.1: a straightforward method, an alternative method and a variant of the alternative method. We denote the straightforward method by OPT-DIST. It first computes optimal outcomes and then checks the distance condition. The alternative method and its variant work in the opposite way by first computing the outcomes satisfying the distance condition and then checking whether they are optimal. To be specific, they construct a new ASO program generating the similar/dissimilar outcomes of the original problem. The alternative method computes outcomes of the new ASO program one by one and checks whether they are optimal for the original problem. It is denoted by DIST(AS)-OPT. Its variant is denoted by DIST(OPT)-OPT. It computes optimal outcomes of the new ASO program and checks their optimality for the original problem. For the $k$-DIVERSE-OPT-OUTCOMES problem, we proposed two techniques for the iterative method in Section

4.2.1. The first one uses a straightforward backtracking algorithm which computes eligible outcomes and adds them to a set of diverse optimal outcomes which have been found. The process to compute an eligible outcome is similar to computing a dissimilar optimal outcome and is implemented similarly as in the three approaches we proposed for the DISSIM-OPT-OUTCOME problem. Therefore the straightforward backtracking technique also has three versions denoted, as before, by OPT-DIST, DIST(AS)-OPT and DIST(OPT)-OPT. The second technique constructs an ASP program to generate a diverse set of outcomes and checks whether each outcome in the set is optimal. We denote this technique by DIV-OPT.

For the problems to find a similar/dissimilar optimal outcome and to find a diverse set of optimal outcomes, we use the Hamming distance to measure the distance between two outcomes. We tested our computational methods on datasets constructed with different values of $d$ ranging from $0.1n$ to $0.7n$, where $n$ is the number of atoms. For instance, when $d$ is $0.1n$, the SIM-OPT-OUTCOME problem asks for an optimal outcome with at most 10% of atoms having different values than they have in the given candidate.

For each dataset, we generated 40 instances for each parameter set to range from $n = 10$ to $n = 120$ with the step of 10. The timeout of solving a problem for each instance is set as 500 seconds. We only show the results for $n = 60$ and $n = 120$. Since the dataset 1 and dataset 2 are similar, we group them together. Each graph below shows the running times of all methods to solve one problem for each instances on datasets 1 and 2 or dataset 3 with $n = 60$ or $n = 120$. Given the number of variables $n$, there are 40 instances in each dataset. Thus, for each problem, we have two datasets. One consists of 80 instances (datasets 1 and 2 combined) and the other consists of 40 instances of the dataset 3. For each problem, we find the running times of all applicable methods, sort them and present in a single "cactus" graph.




(a) $n = 60$ (b) $n = 120$

Figure 4.2: Results for the OPT-OUTCOME problem on datasets 1 and 2

### 4.3.1 Computing One Optimal Solution

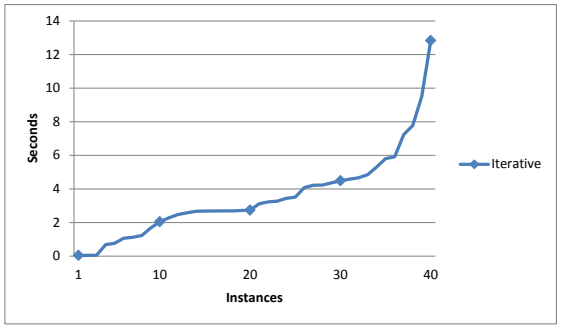
Figure 4.2 shows the results for the Iterative and DLP methods solving the OPT-OUTCOME problem on datasets 1 and 2. The computation time for the both methods increases significantly (one order of magnitude) when $n$ changes from 60 to 120. This is to be expected. The problem is NP-hard and our methods take exponential time in the worst case. However, on the datasets we used, both methods are effective. Moreover, we note that the iterative method scales up much better than the DLP method.

Figure 4.3 shows the results for the dataset 3. The DLP method times out on all instances even when $n$ is small and we do not show any results for it. It is caused primarily by very large sized of programs needed to encode multi-ranked preferences.

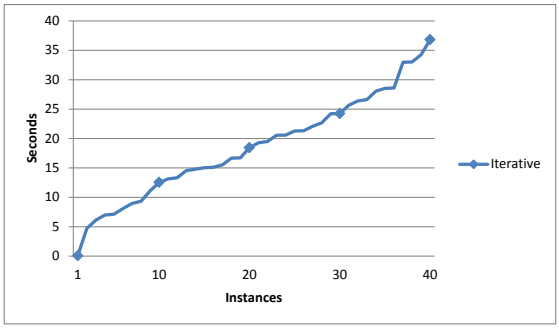
The iterative method is still effective, but requires more time on instances in the dataset 3 than on instances in datasets 1 and 2, again, due to the presence of many ranks.

### 4.3.2 Computing a Different Optimal Solution

The results for the DIFF-OPT-OUTCOME problem are shown in Figure 4.4 and 4.5. The performance of both the Iterative and DLP methods is similar to what we observed on the OPT-OUTCOME problem. The iterative method still has good performance on all datasets. The DLP method works well on datasets 1 and 2, but is not applicable on dataset 3.

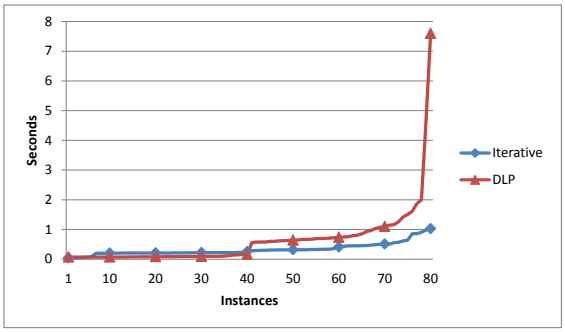



(a) $n = 60$

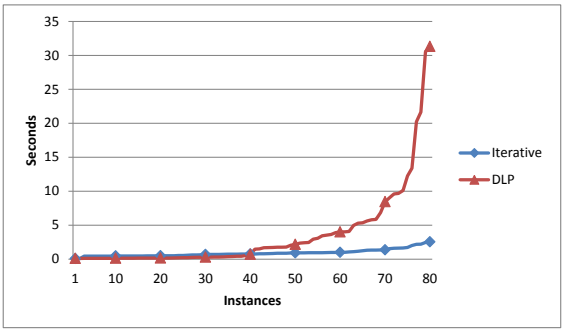



(b) $n = 120$

Figure 4.3: Results for the OPT-OUTCOME problem on dataset 3

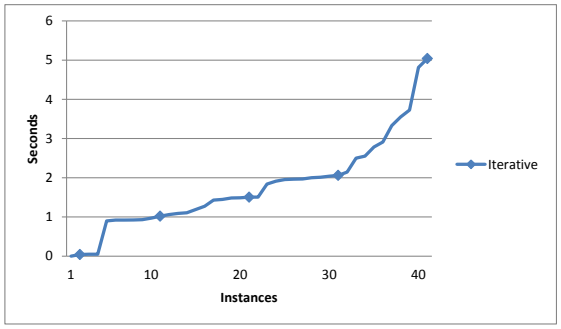



(a) $n = 60$

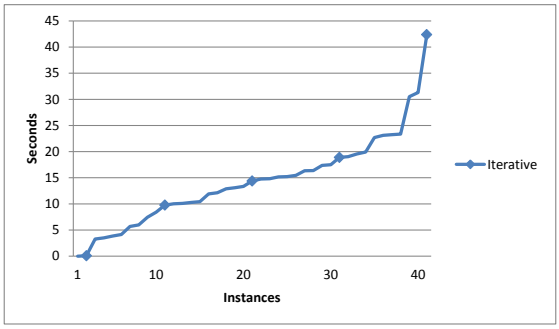



(b) $n = 120$

Figure 4.4: Results for the DIFF-OPT-OUTCOME problem on datasets 1 and 2

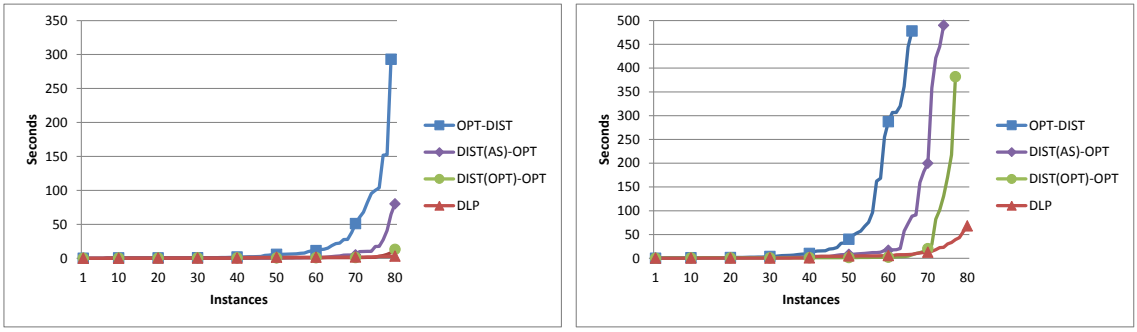



(a) $n = 60$




(b) $n = 120$

Figure 4.5: Results for the DIFF-OPT-OUTCOME problem on dataset 3




(a) $n = 60$ (b) $n = 120$

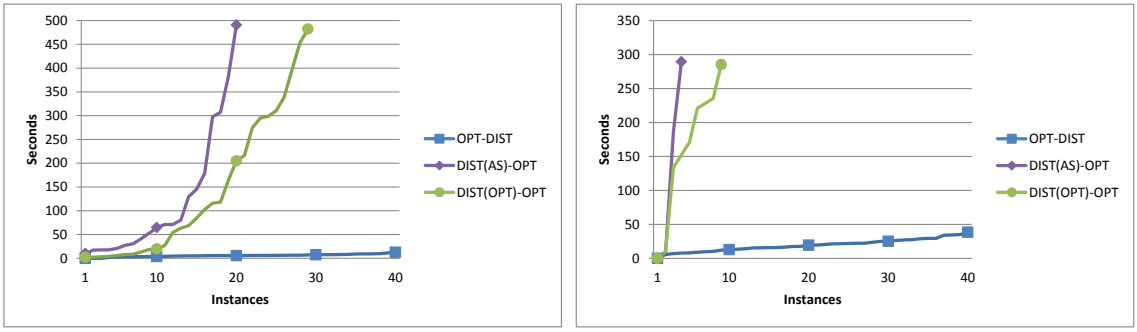
Figure 4.6: Results for the SIM-OPT-OUTCOME problem on datasets 1 and 2

### 4.3.3 Computing a Similar Optimal Solution

Next, we show the results of the Iterative and DLP methods on the SIM-OPT-OUTCOME problem when $d = 0.2n$. Given a candidate solution, the problem is to find an optimal solution with at most 20% of atoms having different values than they have in the given candidate solution. The results presented in Figure 4.6 for datasets 1 and 2 show that all approaches are effective and the DLP method has the best performance. The DIST(OPT)-OPT and DIST(AS)-OPT approaches work better than the OPT-DIST approach, because the former two first filter out the outcomes violating the distance condition using an ASP program and get a much smaller search space when the distance condition is relatively tight.

Not surprisingly, the performance of all methods on instances with $n = 120$ is worse than their performance with $n = 60$. We can observe this phenomenon on every problem and dataset and will not demonstrate it specifically. All methods except the DLP method time out on some instances with $n = 120$.

Figure 4.7 shows the results on dataset 3 and the DLP method is not shown still because of its ineffectiveness on instances with preferences having many different ranks. Interestingly, the OPT-DIST approach, which has the worst performance on datasets 1 and 2, here works best. The DIST(OPT)-OPT and DIST(AS)-OPT approaches perform much worse and become impractical for $n = 120$. For the dataset 3, the preferences for each instances are generated from two LP-trees and have $n$ ranks. Therefore they have very few number of




(a) $n = 60$ (b) $n = 120$

Figure 4.7: Results for the SIM-OPT-OUTCOME problem on dataset 3

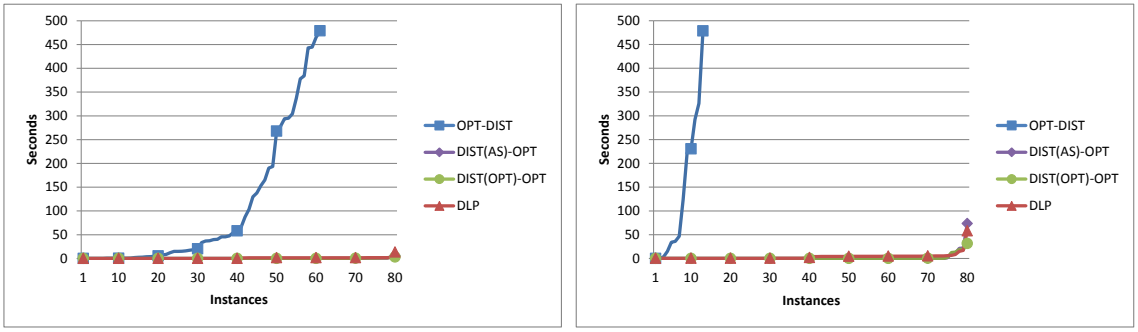
optimal outcomes. In this situation, the OPT-DIST approach has a great advantage because it first computes optimal outcomes and then checks the distance condition. Thus it can get the result quickly after computing the optimal outcomes. The other two approaches need to construct an ASO program generating similar/dissimilar outcomes, and check the optimality for each outcome of the new program until an optimal is found. Since there are only a few optimal outcomes, it is difficult to find one from a large number of candidates.

### 4.3.4 Computing a Dissimilar Optimal Solution

For the DIS-OPT-OUTCOME problem, we show the results for the same methods as for the SIM-OPT-OUTCOME problem. This time we set $d = 0.6n$. Figure 4.8 shows the results for instances from on datasets 1 and 2 (together), and Figure 4.9 shows the results on dataset 3. On datasets 1 and 2, all methods except the OPT-DIST one perform well for both values of $n$. However, on dataset 3, the OPT-DIST approach performs better than the other two iterative ones, similarly to what we observed for the SIM-OPT-OUTCOME problem.

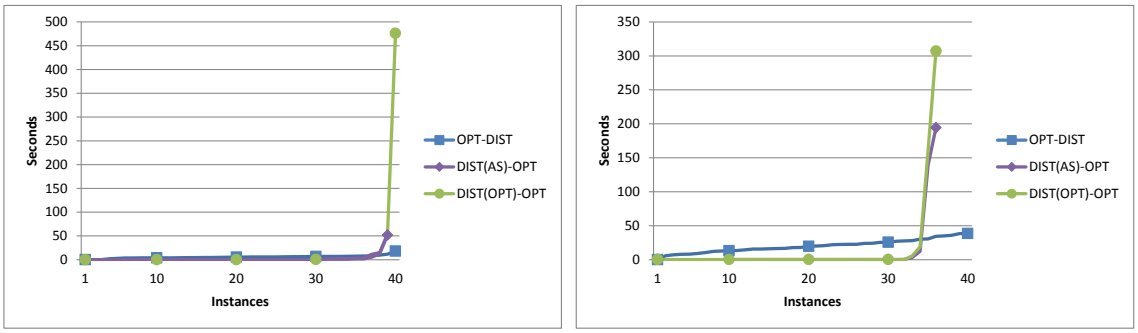
### 4.3.5 Computing a Set of Diverse Optimal Solutions

For the $k$-DIVERSE-OPT-OUTCOMES problem, we tested all four methods (three versions of the iterative approach and the DLP-based method) setting $d = 0.6n$, and $k = 2$ and $k = 3$. That is, we wanted to compute two (three, respectively) optimal outcomes at distance $0.6n$




(a) $n = 60$ (b) $n = 120$

Figure 4.8: Results for the DIS-OPT-OUTCOME problem on datasets 1 and 2



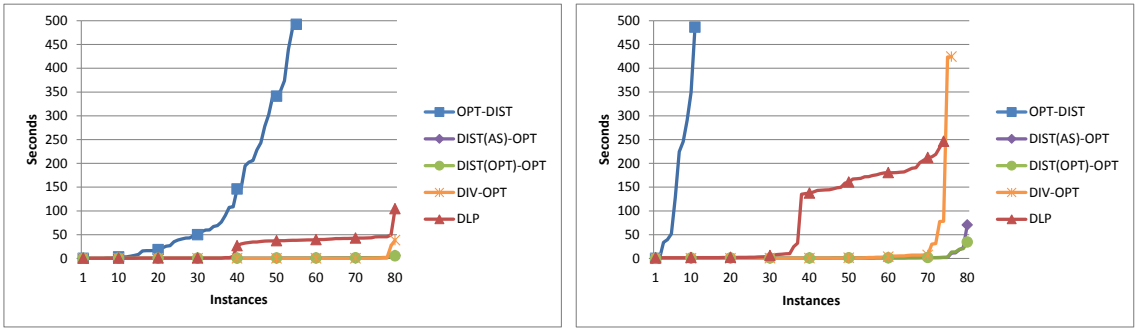

(a) $n = 60$ (b) $n = 120$

Figure 4.9: Results for the DIS-OPT-OUTCOME problem on dataset 3

from each other. Results for $k = 2$ and $k = 3$ are similar. Therefore, we discuss only the results for $k = 3$.

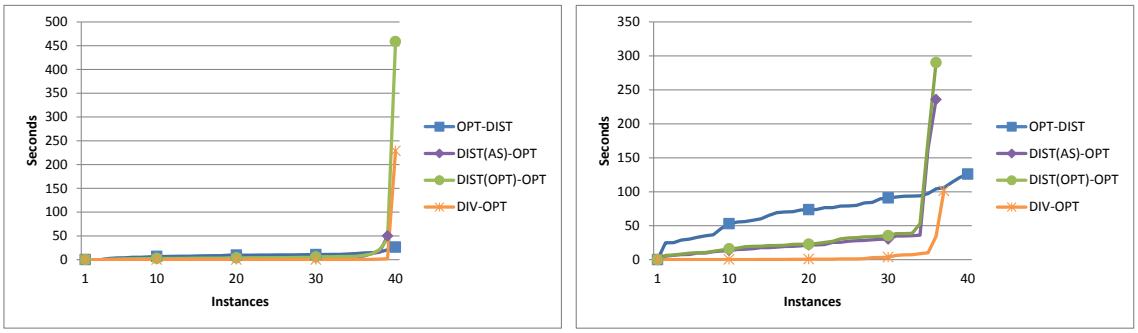
On datasets 1 and 2 (Figure 4.10), similar to the SIM-OPT-OUTCOME and DIS-OPT-OUTCOME problems, the performance of the OPT-DIST algorithm is the worst, and the DIST(OPT)-OPT and DIST(AS)-OPT algorithms are still best. The DLP method is effective but, as the results show, performs worse comparing to the DIS-OPT-OUTCOME problem.

On dataset 3, as we analyzed before, the OPT-DIST approach works best because of the small number of optimal outcomes. Other approaches also work reasonable well with a small number of instances timing out when $n = 120$.




(a) $n = 60$ (b) $n = 120$

Figure 4.10: Results for the $k$-DIVERSE-OPT-OUTCOMES problem on datasets 1 and 2




(a) $n = 60$ (b) $n = 120$

Figure 4.11: Results for the $k$-DIVERSE-OPT-OUTCOMES problem on dataset 3

## 4.4 Discussion and Conclusions

We studied four kinds of computational problems related to reasoning with preferences in the ASO formalism: to find an optimal outcome, to find an optimal outcome different from a given interpretation, to find a similar/dissimilar optimal outcome, and to find a diverse set of optimal outcomes. We extended results known previously by showing that the problems of deciding the existence of a similar (dissimilar, respectively) optimal outcome are $\Sigma_2^p$-complete. In this way, all problems considered are located within the second level of the polynomial hierarchy. Thus, they can be modeled as disjunctive logic programs under the answer-set semantics and disjunctive ASP solvers can be used to solve them. This observation formed the basis of one of the approaches we developed and experimented with in the paper, employing the solvers *claspD2*. We also proposed a mix of imperative and

declarative approaches (the iterative method), which consists of an imperative algorithm that uses ASP solvers for solving some basic optimization tasks that are at the first level of the polynomial hierarchy.

According to the experimental results, the iterative method is effective on all datasets for all problems. The complementary methods we proposed show better performance on different data types. For datasets with no rank or 2 ranks, the method compiling the entire reasoning task into a disjunctive logic program has a good performance. That points to the potential of declarative approaches. For instances with very many ranks, the method based on modeling problems as disjunctive logic programs lags much behind the iterative one and becomes impractical, the reason being much larger sizes of programs needed do model comparisons of interpretations when multiple ranks are present.

To summarize, on the one hand, our work demonstrates the effectiveness of ASP tools in addressing preference optimization problems. On the other hand, it brings up classes of challenging benchmarks based on problems that are $\Sigma_2^p$-complete (ranked ASO programs) that can stimulate further research on solver enhancements.

In the future work, we will study possible improvements to our second method based on disjunctive logic program encodings and other methods for the optimization problems.

## Chapter 5 Manipulation and Bribery in Preference Reasoning

The problem we study in this chapter, misrepresenting preferences by agents to influence preference aggregation to their advantage, has its roots in *strategic voting* studied in social choice [63, 86, 2]. Strategic voting comes in two flavors. *Manipulation* consists of a voter misrepresenting her vote to secure a better outcome for herself [63, 86]. *Bribery* consists of coercing *other* voters to vote against their preferences [48].

The way we view preference reasoning corresponds to the setting of irresolute rules in social choice research. We model agents' preferences as total *preorders* on the space $D$ of outcomes. We assign ranks to preferences as, in real settings, agents will have a hierarchical structure and some will be more important than others. We select a ranked version of *Pareto efficiency* as the principle of preference aggregation. We define the manipulation and bribery in this setting, and establish conditions under which manipulation and bribery are possible.

In each case, the key question is whether misrepresenting preferences can improve for a particular agent the quality of the *collection* of all preferred outcomes resulting from preference aggregation. To be able to decide this question, we have to settle on a way to compare *subsets* of $D$ based on that agent's preference preorder on *elements* of $D$, an issue that underlies all research on strategy proofness of irresolute rules. In this paper, we focus on four natural extensions of a total preorder on $D$ to a total preorder on the power set $\mathscr{P}(D)$. For each of these extensions we characterize a possibility of manipulation or bribery under the Pareto rule (or ranked Pareto rule, for ranked theories). These results apply directly to the setting of social choice as they do not depend on any preference representation language. Since in many cases strategy proofness cannot be assured, following the well established research in computational social choice [5, 50, 48, 57], we turn attention to study the complexity of deciding whether manipulation or bribery are possible. Indeed, the intractability

of computing deviations from true preferences to improve the outcome for an agent may serve as a barrier against strategic behaviors. We look at these questions in the setting of combinatorial domains, the setting not covered by the earlier results. We use our characterizations results as the main tool in this part of our work.

## 5.1 Problem Statements

Let us consider a group $\mathscr{A}$ of $N$ agents each with her own preference on $D$ and with its rank in the set. We denote these agents by integers from $\{1,\ldots,N\}$, their ranks by $r_1,\ldots,r_N$ (lower rank values imply higher importance), and their preferences by $\succeq_1,\ldots,\succeq_N$. In this chapter, sometimes, we write $\succeq_{i,r_i}$ for the preference of an agent $i$, indicating at the same time, the rank of the agent (the rank of her preference). We write $D^i_1,\ldots,D^i_{m_i}$ for the equivalence classes of the relation $\approx_i$ enumerated, as above, from the most to the least preferred with respect to $\succeq_i$. We call the sequence $(\succeq_{1,r_1},\ldots,\succeq_{N,r_N})$ of preferences of agents in $\mathscr{A}$ a (preference) *profile* of $\mathscr{A}$. For instance,

$$\begin{aligned}
&\succeq_{1,1}: \quad f \succ a,c,e \succ b,d\\
&\succeq_{2,1}: \quad a,c \succ d,e,f \succ b\\
&\succeq_{3,2}: \quad a \succ b,c \succ d \succ e,f.
\end{aligned}$$

is a profile of agents $1,2$ and 3. The preferences of agents 1 and 2 are equally ranked and more important than the preferences of agent 3.

Let $P$ be the profile given above. Considering the preferences of agents 1 and 2, $a$ and $c$ are indifferent, no outcome can strictly dominate $a$, $c$ or $f$, and outcomes $b,d,e$ are strictly dominated by $a$ and $c$. According to the preference of agent 3, $a$ is strictly better than $c$. Thus, $Opt(P) = \{a,f\}$. It is interesting to note that for each of the first two agents, the set $Opt(P)$ contains at least one of her "top-rated" outcomes. This is an instance of a general *fairness* property of the Pareto principle.

**Theorem 16.** *For every profile $P$ of a set $\mathscr{A}$ of agents, and for every top-ranked agent $i \in \mathscr{A}$, the set $Opt(P)$ of optimal outcomes for $P$ contains at least one outcome most preferred by $i$.*

*Proof.* Let us pick any outcome $w \in D$ that is optimal for $i$ (that is, $w \in D_1^i$). Clearly, there is $v \in Opt(P)$ such that $v \succeq_P w$. In particular, $v \succeq_i w$. Thus, $v \in D_1^i$ and $v \in Opt(P)$. □

Coming back to our example, it is natural to ask how satisfied agent 3 is with the result of preference aggregation and what means might she have to influence the result. If she submits a different ("dishonest") preference, say

$$\succeq'_{3,2}: \quad a, c \succ b \succ d \succ e, f$$

then, writing $P'$ for the profile $(\succeq_{1,1}, \succeq_{2,1}, \succeq'_{3,2})$, $Opt(P') = \{a, c, f\}$. It may be that agent 3 would prefer $\{a, c, f\}$ to $\{a, f\}$, for instance, because the new set contains an additional highly preferred outcome for her. Thus, agent 3 may have an incentive to misrepresent her preference to the group. We will call such behavior *manipulation*. Similarly, agent 3 might keep her preference unchanged but convince agent 1 to replace his preference with

$$\succeq'_{1,1}: \quad b \succ f \succ a, c, e \succ d.$$

Denoting the resulting profile $(\succeq'_{1,1}, \succeq_{2,1}, \succeq_{3,2})$ by $P''$, $Opt(P'') = \{a, b, f\}$ and, because of the same reason as above, this collection of outcomes may also be preferred to $\{a, f\}$ by agent 3. Thus, agent 3 may have an incentive to try to coerce other agents to change their preference. We will call such behavior *simple bribery*.

We now formally define *manipulation* and *simple bribery*. For a profile $P = (\succeq_{1,r_1}, \ldots, \succeq_{N,r_N})$ and a preference $\succeq'_{i,r_i}$, we write $P_{\succeq_{i,r_i}/\succeq'_{i,r_i}}$ for the profile obtained from $P$ by replacing the preference $\succeq_{i,r_i}$ of the agent $i$ with the preference $\succeq'_{i,r_i}$. Let now $\mathscr{A}$ be a group of $N$ agents with a profile $P = (\succeq_{1,r_1}, \ldots, \succeq_{N,r_N})$, and let $\succeq'_{i,r_i}$ be a preference of agent $i$ on *subsets* of $D$.

**Manipulation:** An agent $i$ can *manipulate* preference aggregation if there is a preference $\succeq'_{i,r_i}$ such that $Opt(P_{\succeq_{i,r_i}/\succeq'_{i,r_i}}) \succ'_i Opt(P)$.

**Simple Bribery:** An agent $t$ is a target for *bribery* by an agent $i$, if there is a preference $\succeq'_{t,r_t}$ such that $Opt(P_{\succeq_{t,r_t}/\succeq'_{t,r_t}}) \succ'_i Opt(P)$.[1]

Clearly, when deciding whether to manipulate (or bribe), agents must be able to compare sets of outcomes and not just single outcomes. This is why we assumed that the agent $i$ has a preorder $\succeq'_i$ on $\mathscr{P}(D)$. However, even when $D$ itself is not a combinatorial domain, $\mathscr{P}(D)$ is. Thus, explicit representations of that preorder may be infeasible.

The question then is whether the preorder $\succeq'_i$ of $\mathscr{P}(D)$, which parameterizes the definitions of manipulation and bribery, can be expressed in terms of the preorder $\succeq_i$ on $D$, as the latter clearly imposes some strong constraints on the former. This problem has received attention from the social choice and AI communities [55, 61, 70, 3, 28] and it turns out to be far from trivial. The difficulty comes from the fact that there are several ways to "lift" a preorder from $D$ to the power set of $D$, none of them fully satisfactory (cf. impossibility theorems [3]). In this paper, we sidestep this issue and simply select and study several most direct and natural "liftings" of preorders on sets to preorders on power sets. We introduce them below. We write $X$ and $Y$ for subsets of $D$ and $\succeq$ for a total preorder on $D$ that we seek to extend to a total preorder on $\mathscr{P}(D)$.

**Compare best:** $X \succeq^{cb} Y$ if there is $x \in X$ such that for every $y \in Y$, $x \succeq y$.

**Compare worst:** $X \succeq^{cw} Y$ if there is $y \in Y$ such that for every $x \in X$, $x \succeq y$.

For the next two definitions, we assume that $\succeq$ partitions $D$ into strata $D_1, \ldots, D_m$, as discussed above.

**Lexmin:** $X \succeq^{lmin} Y$ if for every $i$, $1 \leq i \leq m$, $|X \cap D_i| = |Y \cap D_i|$, or if for some $i$, $1 \leq i \leq m$, $|X \cap D_i| > |Y \cap D_i|$ and, for every $j \leq i-1$, $|X \cap D_j| = |Y \cap D_j|$.

[1] Bribery is traditionally understood as an effort by an *external* agent to bribe a group of voters to obtain a more satisfying result. To stress the difference between this notion and the notion we consider here, we use the term simple bribery.

**Average-rank:**[2] $X \succeq^{ar} Y$ if $ar_{\succeq}(X) \leq ar_{\succeq}(Y)$, where for a set $Z \subseteq D$, $ar_{\succeq}(Z)$ denotes the average rank of an element in $Z$ and is defined by $ar_{\succeq}(Z) = \sum_{i=1}^{m} i\frac{|Z \cap D_i|}{|Z|}$.

Finally, we describe the classes of profiles that we focus on here. Namely, as the setting of ranked preferences is rich, we restrict attention to the two "extreme" cases. In the first one, all agents are equally ranked. In such case, the Pareto principle makes many outcomes optimal as pairs of outcomes are often incomparable. Nevertheless, all practical aggregation techniques, can be understood as simply refining the set of Pareto-optimal outcomes. Thus, improving the quality of the Pareto-optimal set is a desirable objective as it increases a chance of a more favorable outcome once a refinement is applied. In the second setting, we assume all agents have distinct ranks. In such case, the Pareto principle is natural and quite effective, resulting in a total preorder refining the one of the most important agent by breaking ties based on preferences of lower ranked agents.

### 5.2 Equally Ranked Preferences

In this section, we discuss the manipulation and simple bribery problems in the case where all preferences are equally ranked, and study them with respect to each of the four extensions of total preorders on $D$ to $\mathscr{P}(D)$ defined above. An *equally ranked preference profile* is a profile $P = (\succeq_{1,r_1}, \ldots, \succeq_{N,r_N})$, where $r_1 = \cdots = r_N$. To simplify the notation, we write it as $P = (\succeq_1, \ldots, \succeq_N)$.

#### 5.2.1 Manipulation

Given a set $\mathscr{A}$ of $N$ agents and a profile $P = (\succeq_1, \ldots, \succeq_N)$, the manipulation problem is to determine whether an agent $i$ can find a total preorder $\succeq$ such that $Opt(P_{\succeq_i/\succeq}) \succ_i' Opt(P)$ where $\succeq_i'$ is the total preorder agent $i$ uses to compare subsets of $D$.

[2]This method is well defined only if both sets to compare are non-empty. This is not a strong restriction because our aggregation method returns only non-empty sets of optimal outcomes.

**Theorem 17.** *Manipulation is impossible for* compare best *and* compare worst *on profiles of equally ranked preferences.*

*Proof.* Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$. We want to show that for every $i \in \mathscr{A}$ and every total preorder $\succeq$, $Opt(P) \succeq_i^{cb} Opt(P_{\succeq_i/\succeq})$ and $Opt(P) \succeq_i^{cw} Opt(P_{\succeq_i/\succeq})$.

For *compare best*, let $v \in Opt(P)$ be an outcome that is also optimal for $i$ (such a $v$ exists by Theorem 16). It follows that for every $w \in D$, $v \succeq_i w$. Thus, $v \succeq_i w$, for every $w \in Opt(P_{\succeq_i/\succeq})$. By the definition of $\succeq_i^{cb}$, $Opt(P) \succeq_i^{cb} Opt(P_{\succeq_i/\succeq})$.

For *compare worst*, let us assume that there is a total preorder $\succeq$ such that $Opt(P_{\succeq_i/\succeq}) \succ_i^{cw} Opt(P)$. It follows from the definition of $\succeq_i^{cw}$ that there is $w' \in Opt(P)$ such that for every $w \in Opt(P_{\succeq_i/\succeq})$, $w \succ_i w'$. Thus, $w' \notin Opt(P_{\succeq_i/\succeq})$ and, consequently, there is $v \in Opt(P_{\succeq_i/\succeq})$ such that $v \succ_{P_{\succeq_i/\succeq}} w'$. It follows that $v \succeq_j w'$, for every agent $j \neq i$. Since by an earlier observation, $v \succ_i w'$, we obtain $v \succ_P w'$, a contradiction with $w' \in Opt(P)$. □

On the other hand, manipulation is possible for every agent using the *lexmin* comparison rule precisely when not every outcome in $D$ is optimal. The reason is that by changing her preference an agent can cause a Pareto-nonoptimal outcome become Pareto-optimal, while keeping the optimality status of every other outcome unchanged.

**Theorem 18.** *Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$ and let $i \in \mathscr{A}$. There exists a total preorder $\succeq$ such that $Opt(P_{\succeq_i/\succeq}) \succ_i^{lmin} Opt(P)$ if and only if $Opt(P) \neq D$.*

*Proof.* ($\Leftarrow$) Let us assume that $\succeq_i$ is given by

$$\succeq_i: \quad D_1^i \succ_i \ldots \succ_i D_{m_i}^i.$$

Let $\ell$ be the smallest $k$ such that $D_k^i \setminus Opt(P) \neq \emptyset$ and let $a \in D_\ell^i \setminus Opt(P)$. We will now construct a preference $\succeq$ for agent $i$ so that $Opt(P) \cup \{a\} = Opt(P_{\succeq_i/\succeq})$. For that preference,

we have $Opt(P_{\succeq_i/\succeq}) \succ_i^{lmin} Opt(P)$, which demonstrates that $i$ can manipulate preference aggregation in $P$.

To construct $\succeq$, we first note that since $a \notin Opt(P)$, there is $w \in Opt(P)$ such that $w \succ_P a$. Since $w \succeq_i a$ and $a \in D_\ell^i$, $w \in D_j^i$, for some $j \leq \ell$. Without loss of generality, we may assume that this $w$ is chosen so that to minimize $j$.

In the remainder of the proof, we write $P^{-i}$ for the profile obtained from $P$ by removing the preference of agent $i$. To simplify the notation, we also write $P'$ for $P_{\succeq_i/\succeq}$.

*Case* 1: $w \approx_{P^{-i}} a$. Since $w \succ_P a$, we have $w \succ_i a$, that is, $j < \ell$. Let us define $\succeq$ as follows:

$$\succeq: \quad D'_1 \succ \ldots \succ D'_{m_i},$$

where $D'_j = D_j^i \cup \{a\}$, $D'_\ell = D_\ell^i \setminus \{a\}$, and $D'_k = D_k^i$, for the remaining $k \in [1..m_i]$. Thus $a \approx_{P'} w$. We also have that for every $w', w'' \in D \setminus \{a\}$, $w' \succeq_{P'} w''$ if and only if $w' \succeq_P w''$ (the degrees of quality of outcomes other than $a$ remain the same when we move from $P$ to $P'$). Finally, for every $w' \in D$, $a \succeq_{P'} w'$ if and only if $w \succeq_{P'} w'$. These observations imply that $Opt(P') = Opt(P) \cup \{a\}$.

*Case* 2: $w \succ_{P^{-i}} a$. Let us define $\succeq$ as follows:

$$\succeq: \quad D'_1 \succ \ldots \succ D'_{m_i+1},$$

where $D'_k = D_k^i$, for $k < j$, $D'_j = \{a\}$, $D'_{\ell+1} = D_\ell^i \setminus \{a\}$, and $D'_k = D_{k-1}^i$, for every $k \in \{j+1, \ldots, m_i+1\}$ such that $k \neq \ell+1$. Informally, $\succeq$ is obtained by pulling $a$ from $D_\ell^i$, and inserting it as a singleton cluster directly before $D_j^i$. Since $a$ is the only outcome moved, for every $w', w'' \in D \setminus \{a\}$, $w' \succeq_{P'} w''$ if and only if $w' \succeq_P w''$ (and similarly, for the derived relation $\succ_{P'}$).

Let us observe that $a \in Opt(P')$. Indeed, if for some $w' \in D$, $w' \succeq_{P'} a$, then $w' \in D_k^i$, for some $k < j$. It follows that $w' \in Opt(P)$ and $w' \succ_i a$. Consequently, $w' \succ_P a$, contrary to our choice of $w$.

Let $w' \in Opt(P)$ and let us assume that $w'' \succ_{P'} w'$ for some $w'' \in D$. Since $a \notin Opt(P)$, $w' \neq a$. If $w'' \neq a$, then $w'' \succ_P w'$ (indeed, $a$ is the only outcome whose relation to other

outcomes changes when we move from $P$ to $P'$). This is a contradiction with $w' \in Opt(P)$. Thus, $w'' = a$. Consequently, $w'' \succ_{P'} w'$ implies $a \succeq_{P^{-i}} w'$ and $a \succeq w'$. By the construction of $\succeq$, the latter property implies that $w \succeq_i w'$. Since $w \succ_{P^{-i}} a \succeq_{P^{-i}} w'$, $w \succ_P w'$, a contradiction. It follows that $w' \in Opt(P')$ and, consequently, we have $Opt(P) \cup \{a\} \subseteq Opt(P')$.

Conversely, let us consider $w' \in Opt(P')$ such that $w' \neq a$. Let us assume that for some $w'' \in Opt(P)$, $w'' \succ_P w'$. If $w'' \neq a$, we can get $w'' \succ_{P'} w'$, a contradiction. If $w'' = a$, we can get $a \succeq_k w'$ for every $k \in \mathscr{A}$ and $k \neq i$ from $a \succ_P w'$ and $a \succ w'$. Thus $a \succ_{P'} w'$ which contradicts the property that $w' \in Opt(P')$. It follows that $w' \in Opt(P)$. Thus, $Opt(P') \subseteq Opt(P) \cup \{a\}$. Consequently, $Opt(P) \cup \{a\} = Opt(P')$.

($\Rightarrow$) If $Opt(P) = D$, then there is no set $S$ such that $S \succ_i^{lmin} Opt(P)$. □

For the *average-rank* preorder for comparing sets, an agent can manipulate the result to her advantage if there are nonoptimal outcomes that are highly preferred by the agent, or when there are optimal outcomes that are low in the preference of that agent, as the former can be made optimal and the latter made non-optimal without changing the optimality status of other outcomes.

**Theorem 19.** *Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$ and let $i \in \mathscr{A}$. There exists a total preorder $\succeq$ such that $Opt(P_{\succeq_i/\succeq}) \succ_i^{ar} Opt(P)$ if and only if:*

1. *For some $j < ar_{\succeq_i}(Opt(P))$, there exists $a' \in D_j^i$ such that $a' \notin Opt(P)$; or*
2. *For some $j > ar_{\succeq_i}(Opt(P))$, there are $a' \in Opt(P) \cap D_j^i$ and $a'' \in Opt(P)$ such that $a' \neq a''$, and $a'' \succeq_k a'$, for every $k \in \mathscr{A}$, $k \neq i$.*

*Proof.* ($\Leftarrow$) Let us assume that the first condition holds. Let $\ell$ be the smallest $k$ such that $D_k^i \setminus Opt(P) \neq \emptyset$, and let $a' \in D_\ell^i \setminus Opt(P)$. Reasoning as in the proof of the previous theorem, we can construct a total preorder $\succeq$ such that $Opt(P') = Opt(P) \cup \{a'\}$ (where $P'$ denotes $P_{\succeq_i/\succeq}$). Clearly, $ar_{\succeq_i}(Opt(P')) < ar_{\succeq_i}(Opt(P))$ and so, $Opt(P') \succ_i^{ar} Opt(P)$ ($i$ can manipulate).

If the second condition is satisfied then, let us assume that $a'' \in D^i_{j'}$. Then, we have $j' \geq j$ (otherwise, $a'' \succ_P a'$, contradicting optimality of $a'$ in $P$). Let us construct $\succeq$ as in the previous argument, but substituting $a''$ for $a'$ (and, as before, we write $P'$ for $P_{\succeq_i/\succeq}$). Without loss of generality, we may select $a''$ so that $j'$ be minimized.

We know that $a'' \in Opt(P')$. Moreover, by the definition, $a'' \succ a'$. Thus, $a'' \succ_{P'} a'$ and so, $a' \notin Opt(P')$.

Next, if $w \in Opt(P)$ and $w \succ_i a'$, then $w \in Opt(P')$. To show this, let us assume that there is $w' \in Opt(P')$ such that $w' \succ_{P'} w$. Since $w \succ_i a'$, $w \neq a''$ and $w \succ a''$. The latter implies that $w' \neq a'$. Thus, $w' \succ_P w$, a contradiction.

Finally, if $w \notin Opt(P)$ and $a' \succeq_i w$, $w \notin Opt(P')$. Indeed, if $w' \succ_P w$ then $w' \succ_{P'} w$.

Since $j > ar_{\succeq_i}(Opt(P))$, these observations imply that $ar_{\succeq_i}(Opt(P')) < ar_{\succeq_i}(Opt(P))$.

($\Rightarrow$) We set $x = ar_{\succeq_i}(Opt(P))$. By the assumption, there is a total preorder $\succeq$ on $D$ such that $ar_{\succeq_i}(Opt(P_{\succeq_i/\succeq})) < x$. Let us set $O = Opt(P_{\succeq_i/\succeq})$ and let $D_1$ be the set of all elements $w \in D$ such that $q_{\succeq_i}(w) < x$. If $D_1 \setminus Opt(P) \neq \emptyset$, then the condition (1) holds. Thus, let us assume that $D_1 \subseteq Opt(P)$. We denote by $O'$ the set obtained by

1. removing from $Opt(P)$ every element $w \in D_1 \setminus O$
2. removing from $Opt(P)$ every element $w \notin O$ such that $q_{\succeq_i}(w) = x$
3. including every element $w \in O \setminus Opt(P)$ such that $q_{\succeq_i}(w) = x$.

We have $ar_{\succeq_i}(O') \geq x$. Moreover, $O'$ differs from $O$ (if at all) only on elements $w$ such that $q_{\succeq_i}(w) > x$. If $O$ contains every element $w \in Opt(P)$ such that $q_{\succeq_i}(w) > x$, then $ar_{\succeq_i}(O) \geq ar_{\succeq_i}(O')$ and so, $ar_{\succeq_i}(O) \geq x$, a contradiction. Thus, there is an element $w \in Opt(P)$ such that $q_{\succeq_i}(w) > x$ and $w \notin O$. Since $O = Opt(P_{\succeq_i/\succeq})$, it is only possible if the condition (2) holds. □

We consider this problem in the profile with two-level preferences which have exactly two levels of choices, good or bad. The preference for agent $i$ is represented by $\succeq_i$: $D^i_1 \succ_i$

$D_2^i$.

**Corollary 1.** *Let $\mathscr{A}$ be a set of N agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$, and let $i \in \mathscr{A}$ where $\succeq_i$: $D_1^i \succ_i D_2^i$. There exists a total preorder $\succeq$ such that $Opt(P_{\succeq_i/\succeq}) \succ_i^{ar} Opt(P)$ if and only if there are outcomes $a', a'' \in D$ such that*

1. *$a' \in D_1^i \setminus Opt(P)$, and $a'' \in D_2^i \cap Opt(P)$; or*
2. *$a', a'' \in Opt(P)$, $a' \neq a''$, $a' \approx_P a''$, and $a' \in D_2^i$.*

*Proof.* ($\Rightarrow$) By Theorem 19, one of its conditions (1) and (2) holds. Let us assume that the condition (1) of Theorem 19 holds. It follows that $j = 1$, and that $D_1^i$ contains an non-optimal outcome for $P$, say $a'$. Moreover, $D_2^i$ must also contain an optimal outcome, say $a''$ (otherwise, $ar_{\succeq_i}(Opt(P)) = 1$). Thus, the condition (1) holds. Next, let us assume that the condition (2) of Theorem 19 holds. It follows that $j = 2$, and that there are outcomes $a', a'' \in Opt(P)$ such that $a' \neq a''$, $a' \in D_2^i$, and $a'' \succeq_k a'$, for every $k \in \mathscr{A}$, $k \neq i$. Since $a' \in D_2^i$, $a'' \succeq_i a'$ ($\succeq_i$ has only two clusters, $D_1^i$ and $D_2^i$, and $a' \in D_2^i$). It follows that $a'' \succeq_P a'$. By the optimality of $a'$ for $P$, $a' \approx_P a''$ follows. Thus, the condition (2) holds.

($\Leftarrow$) Let us assume that the condition (1) holds. Since $a' \in D_1^i \setminus Opt(P)$, $1 < ar_{\succeq_i}(Opt(P))$ and the condition (1) of Theorem 19 holds (with $j = 1$). Next, let us assume that the condition (1) does not hold but the condition (2) does. It follows that every element in $D_1^i$ is optimal for $P$. Since $D_1^i \neq \emptyset$ and $a' \in Opt(P) \cap D_2^i$, $2 > ar_{\succeq_i}(Opt(P))$. Thus, the condition (2) of Theorem 19 holds (with $j = 2$). □

The main message of these theorems is that when the result of preference aggregation is a *set* of optimal outcomes, then even the most elementary aggregation rule, Pareto principle, may be susceptible to manipulation. Whether it is or is not depends on how agents measure the quality of a set. If the comparison is based on the best or worst outcomes, manipulation is not possible (a positive result). However, under less simplistic rules such as *lexmin* or *average-rank* the possibility for manipulation emerges (a negative result that, in some settings, we later moderate by means of the complexity barrier).

### 5.2.2 Simple Bribery

In the same setting, the simple bribery problem is to decide whether an agent $i$ can find an agent $t$, $t \neq i$, and a total preorder $\succeq$ such that $Opt(P_{\succeq_t/\succeq}) \succ'_i Opt(P)$. Our results on bribery are similar to those we obtained for manipulation, with one notable exception, and show that whether bribery is possible depends on how agents measure the quality of sets of outcomes.

**Theorem 20.** *Simple bribery is impossible for* compare best *on profiles of equally ranked preferences.*

*Proof.* Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$. We need to prove that for every $i,t \in \mathscr{A}$, $t \neq i$, and every total preorder $\succeq$, $Opt(P) \succeq_i^{cb} Opt(P_{\succeq_t/\succeq})$.

Let $v \in Opt(P)$ be optimal for $i$ (such a $v$ exists by Theorem 16). It follows that for every $w \in D$, $v \succeq_i w$. Thus, $v \succeq_i w$, for every $w \in Opt(P_{\succeq_t/\succeq})$. By the definition of $\succeq_i^{cb}$, $Opt(P) \succeq_i^{cb} Opt(P_{\succeq_t/\succeq})$. □

The result is proved in the same way as Theorem 17. We stress that it states that no agent using *compare best* preorder on sets can successfully bribe *any other* agent.

The situation changes if agents are interested in the worst outcomes in a set. Unlike in the case of manipulation, simple bribery may now be possible. Given a set $X \subseteq D$ and a total preorder $\succeq$, by $Min_{\succeq}(X)$ we denote the set of all "worst" elements in $X$, that is the set that contains every element $x \in X$ such that for every $y \in X$, $y \succeq x$.

**Theorem 21.** *Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$ and let $i \in \mathscr{A}$. There exist $t \in \mathscr{A}$, $t \neq i$, and a total preorder $\succeq$ such that $Opt(P_{\succeq_t/\succeq}) \succ_i^{cw} Opt(P)$ if and only if for every $a \in Min_{\succeq_i}(Opt(P))$, there is $a' \in D$ such that $a' \succ_i a$, and $a' \succeq_k a$, for every $k \in \mathscr{A}$, $k \neq t$.*

*Proof.* ($\Leftarrow$) To define $\succeq$, we modify the total preorder $\succeq_t$ as follows. For every $a \in Min_{\succeq_i}(Opt(P))$, we move $a'$ (the element satisfying $a' \succ_i a$, and $a' \succeq_k a$, for every $k \in \mathscr{A}$, $k \neq t$, whose existence is given by the assumption) from its cluster in $\succeq_t$ to the cluster of $\succeq_t$ containing $a$.

First, we note that for every $a \in Min_{\succeq_i}(Opt(P))$, $a' \succ_{P_{\succeq_t/\succeq}} a$. Second, the only change when moving from $P$ to $P_{\succeq_t/\succeq}$ is in the profile of agent $t$, and that profile changes by *promoting* elements $a'$ (indeed, for every $a \in Min_{\succeq_i}(Opt(P))$, $a \succ_t a'$; otherwise, we would have $a' \succ_P a$, contrary to $a \in Opt(P)$). Thus, some of these elements might become optimal but their degrees of quality in $\succeq_i$ are better than those of their corresponding elements $a$. Finally, other elements than the $a'$s cannot become optimal. These three observations imply that $Opt(P_{\succeq_t/\succeq}) \succ_i^{cw} Opt(P)$.

($\Rightarrow$) Let an agent $t \neq i$ and a total preorder $\succeq$ satisfy $Opt(P_{\succeq_t/\succeq}) \succ_i^{cw} Opt(P)$. To simplify notation, we set $Q = P_{\succeq_t/\succeq}$.

Let us consider $a \in Min_{\succeq_i}(Opt(P))$. Since $Opt(Q) \succ_i^{cw} Opt(P)$, $a \notin Opt(Q)$. It follows that there is $a' \in Opt(Q)$ such that $a' \succ_Q a$. Thus $a' \succ_i a$ (otherwise, we would have $Opt(P) \succeq_i Opt(Q)$, a contradiction). Moreover, for every $k \in \mathscr{A}$, $k \neq t$, $a' \succeq_k a$. □

Simple bribery is also possible when *lexmin* or *average-rank* methods are used by agents to extend a preorder on $D$ to a preorder on $\mathscr{P}(D)$. Similarly to Theorem 20, the following two theorems are literal generalizations of the earlier results on manipulation.

**Theorem 22.** *Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$ and let $i,t \in \mathscr{A}$, $t \neq i$. There exists a total preorder $\succeq$ such that $Opt(P_{\succeq_t/\succeq}) \succ_i^{lmin} Opt(P)$ if and only if $Opt(P) \neq D$.*

*Proof.* ($\Leftarrow$) Let us assume that $\succeq_i$ is given by

$$\succeq_i: \quad D_1^i \succ_i \ldots \succ_i D_{m_i}^i,$$

and $\succeq_t$ is given by

$$\succeq_t: \quad D_1^t \succ_t \ldots \succ_t D_{m_t}^t.$$

Let $\ell$ be the smallest $k$ such that $D^i_k \setminus Opt(P) \neq \emptyset$ and let $a \in D^i_\ell \setminus Opt(P)$. We will now construct a preference $\succeq$ for agent $i$ so that $Opt(P) \cup \{a\} = Opt(P_{\succeq_t/\succeq})$ in the way introduced in Theorem 18. For that preference, we have $Opt(P_{\succeq_t/\succeq}) \succ_i^{lmin} Opt(P)$, which demonstrates that $i$ can manipulate preference aggregation in $P$.

($\Rightarrow$) Obviously, if $Opt(P) = D$, there is no set $S$ such that $S \succ_i^{lmin} Opt(P)$. □

**Theorem 23.** *Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$ and let $i \in \mathscr{A}$. There exist $t \in \mathscr{A}$, $t \neq i$, and a total preorder $\succeq$ such that $Opt(P_{\succeq_t/\succeq}) \succ_i^{ar} Opt(P)$ if and only if:*

1. *For some $j < ar_{\succeq_i}(Opt(P))$, there exists $a' \in D^i_j$ such that $a' \notin Opt(P)$; or*
2. *For some $j > ar_{\succeq_i}(Opt(P))$, there are $a' \in Opt(P) \cap D^i_j$, and $a'' \in Opt(P)$ such that $a' \neq a''$ and $a'' \succeq_k a'$, for every $k \in \mathscr{A}$, $k \neq t$.*

*Proof.* ($\Leftarrow$) Let us assume that the first condition holds. Let $\ell$ be the smallest $k$ such that $D^i_k \setminus Opt(P) \neq \emptyset$, and let $a' \in D^i_\ell \setminus Opt(P)$. Reasoning as in the proof of the previous theorem, we can construct a total preorder $\succeq$ such that $Opt(P') = Opt(P) \cup \{a'\}$ (where $P'$ denotes $P_{\succeq_t/\succeq}$). Clearly, $ar_{\succeq_i}(Opt(P')) < ar_{\succeq_i}(Opt(P))$ and so, $Opt(P') \succ_i^{ar} Opt(P)$.

If the second condition is satisfied, we have $a' \succeq_t a''$ (otherwise, $a'' \succ_P a'$, contradicting optimality of $a'$ in $P$). Let us assume that $a'' \in D^i_{j'}$. Without loss of generality, we may select $a''$ so that $j'$ be maximized.

If $j' > ar_{\succeq_i}(Opt(P))$, Let us construct $\succeq$ as in the theorem 19 argument, but replace $\succeq_t$ with $\succeq$ (and, as before, write $P'$ for $P_{\succeq_t/\succeq}$).

We know that $a'' \in Opt(P')$. Moreover, by the definition, $a'' \succ a'$. Thus, $a'' \succ_{P'} a'$ and so, $a' \notin Opt(P')$.

Next, if $w \in Opt(P)$ and $w \succ_i a''$, then $w \in Opt(P')$. To show this, let us assume that there is $w' \in Opt(P')$ such that $w' \succ_{P'} w$. Since $w \succ_i a'' \succeq_i a'$, $w' \neq a'$ and $w' \neq a''$. Thus, $w' \succ_P w$, a contradiction.

Finally, if $w \notin Opt(P)$, $w \notin Opt(P')$. Since $w \notin Opt(P)$, $w \neq a'$ and $w \neq a''$. Indeed, it is clear that if $w' \succ_P w$ then $w' \succ_{P'} w$.

Since $j' > ar_{\succeq_i}(Opt(P))$, these observations imply that $ar_{\succeq_i}(Opt(P')) < ar_{\succeq_i}(Opt(P))$.

If $j' < ar_{\succeq_i}(Opt(P))$, let $X$ be a set of outcomes $a$ such that $a \in Opt(P)$, $d_{\succeq_i}(a) < ar_{\succeq_i}(Opt(P))$, $a' \succeq_t a \succeq_t a''$ and $a'' \succeq_k a$ for every $k \in \mathscr{A}$, $k \neq t$. We construct a total preorder $\succeq$ by moving every $a \in X$ and $a''$ before $a'$ and keeping the relative order among all $a \in X$ and $a''$.

By the definition, $a'' \succ a'$. Thus, $a'' \succ_{P'} a'$ and so, $a' \notin Opt(P')$.

Next, we want to prove that if $w \in Opt(P)$ and $d_{\succeq_i}(w) < ar_{\succeq_i}(Opt(P))$, then $w \in Opt(P')$. If $w \succ_i a''$, similar to the previous argument, $w \in Opt(P')$. If $a'' \succeq_i w$, let us assume that there is $w' \in Opt(P')$ such that $w' \succ_{P'} w$. If $w' \notin X$ and $w' \neq a''$, we can get $w' \succ_P w$ contradicting $w \in Opt(P)$. Thus $w' \in X$ or $w' = a''$. If $a'' \succ_t w$, we can get $a \succ_t w$ for every $a \in X$. Thus $w' \succ_t w$ and $w' \succ_P w$, a contradiction. If $w \succ_t a'$, we can get $w \succ a''$ and $w \succ a$ for every $a \in X$. Thus $w \succ w'$ contradicting $w' \succ_{P'} w$. Thus $a' \succeq_t w \succeq_t a''$. Since $w' \succ_{P'} w$, $w' \succeq_k w$ for every $k \in \mathscr{A}$, $k \neq t$. We already know that $w' \in X$ or $w' = a''$, and for every $a \in X$, $a'' \succeq_k a$ for every $k \in \mathscr{A}$, $k \neq t$. Thus $a'' \succeq_k w$ for every $k \in \mathscr{A}$, $k \neq t$. According to all these, we can get that $w \in X$. If $w \in X$, according to the definition of $\succeq$, $w' \succ_{P'} w$ if and only if $w' \succ_P w$ contradicting to $w \in Opt(P)$. Thus for every $w \in Opt(P)$ and $d_{\succeq_i}(w) < ar_{\succeq_i}(Opt(P))$, $w \in Opt(P')$.

Finally, if $w \notin Opt(P)$, $w \notin Opt(P')$. Since $w \notin Opt(P)$, $w \neq a'$, $w \neq a''$ and $w \notin X$. Indeed, it is clear that if $w' \succ_P w$ then $w' \succ_{P'} w$.

These observations imply that $ar_{\succeq_i}(Opt(P')) < ar_{\succeq_i}(Opt(P))$.

($\Rightarrow$) We set $x = ar_{\succeq_i}(Opt(P))$. By the assumption, there is a total preorder $\succeq$ on $D$ such that $ar_{\succeq_i}(Opt(P_{\succeq_t/\succeq})) < x$. Let us set $O = Opt(P_{\succeq_t/\succeq})$ and let $D_1$ be the set of all elements $w \in D$ such that $d_{\succeq_i}(w) < x$. If $D_1 \setminus Opt(P) \neq \emptyset$, then the condition (1) holds. Thus, let us assume that $D_1 \subseteq Opt(P)$.

Similar to the proof for Theorem 19, we can construct the set $O'$ and show that if $O$

contains every element $w \in Opt(P)$ such that $d_{\succeq_i}(w) > x$, we can get a contradiction. Thus, there is an element $w \in Opt(P)$ such that $d_{\succeq_i}(w) > x$ and $w \notin O$. Since $O = Opt(P_{\succeq_t/\succeq})$, it is only possible if the condition (2) holds. □

Theorems 21, 22 and 23 show that a possibility for simple bribery may arise when *compare worst*, *lexmin* and *average-rank* are used to compare sets of outcomes. There is, however, a difference between *lexmin* and the other two methods. For the former, if simple bribery is possible, then every agent can be the target (can be used as $t$ in the theorem). This is not the case for the other two methods.

We also provide the possibility for simple bribery with two-level preferences which is used for later proofs.

**Corollary 2.** *Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$ and let $i \in \mathscr{A}$ where $\succeq_i$: $D_1^i \succ_i D_2^i$. There exist $t \in \mathscr{A}$ and a total preorder $\succeq$ such that $Opt(P_{\succeq_t/\succeq}) \succ_i^{ar} Opt(P)$ if and only if there are outcomes $a', a'' \in D$ such that:*

1. *$a' \in D_1^i \setminus Opt(P)$ and $a'' \in D_2^i \cap Opt(P)$; or*
2. *$a', a'' \in Opt(P)$, $a' \in D_2^i$, and $a'' \succeq_k a'$, for every $k \in \mathscr{A}$, $k \neq t$.*

*Proof.* Similar to Corollary 1, agent $i$ can improve the quality of the optimal outcomes if and only if $ar_{\succeq_i}(Opt(P)) > 1$ (there is $a'' \in D_2^i \cap Opt(P)$), and there is $a' \in D_1^i \setminus Opt(P)$ which can become optimal for $P_{\succeq_t/\succeq}$ or there is $a' \in D_2^i \cap Opt(P)$ which can become non-optimal for $P_{\succeq_t/\succeq}$. □

### 5.3 Strictly Ranked Preferences

In this section, we discuss the manipulation and simple bribery problems in the setting in which all agents have distinct ranks and so, can be seen as strictly ranked. A strictly ranked preference profile can be written as $P = (\succeq_{1,1},\ldots,\succeq_{N,N})$ (after possibly relabeling

agents). In this section, we will write such profiles as $P = (\succeq_1, \ldots, \succeq_N)$. Such a preference formalism generates a total preorder over outcomes. Moreover, all optimal outcomes are indifferent and share the same quality degree for every preference. In general, proceeding from the most important preference to the least, the relation between two outcomes is decided by the first preference, where they have different quality degrees.

Our first two results in this section concern the manipulation problem.

**Theorem 24.** *Manipulation is impossible for* compare best*,* compare worst *and* average-rank *on profiles of strictly ranked preferences.*

*Proof.* Let $\mathscr{A}$ be a set of $N$ agents $1, \ldots, N$ with a profile $P = (\succeq_1, \ldots, \succeq_N)$ and $i \in \mathscr{A}$. Let us assume the preference of an agent $i \in \mathscr{A}$ is

$$\succeq_i: \quad D_1^i \succ_i D_2^i \succ_i \cdots \succ_i D_{m_i}^i.$$

Since all optimal outcomes are indifferent, we can assume $a \in D_j^i$ for every $a \in Opt(P)$. Let $a$ be any optimal outcome. According to the definitions of *compare-best*, *compare-worst* and *average-rank*, if $Opt(P')$ is better than $Opt(P)$ based on the corresponding extension of $\succeq_i$, then there exists $a' \in Opt(P')$ such that $a' \succ_i a$. Since $a' \notin Opt(P)$, $a \succ_P a'$. And because of $a' \succ_i a$, $a \succ_j a'$ for some $j < i$. This can not be changed no matter how $i$ changes her preference. Thus $a \succ_{P'} a'$ and $a' \notin Opt(P')$. □

For *lexmin*, agent $i$ can get a better result by making non-optimal outcomes equivalent or worse to currently optimal outcomes become optimal. The precise description of the conditions when it is possible is given below.

**Theorem 25.** *Let $\mathscr{A}$ be a set of $N$ agents $1, \ldots, N$ with a profile of strictly ranked preferences $P = (\succeq_1, \ldots, \succeq_N)$. For every $i \in \mathscr{A}$, manipulation is possible for* lexmin *if and only if at least one of the following conditions holds:*

1. *There exists $a' \in D \setminus Opt(P)$, such that for every $a'' \in Opt(P)$, $a'' \approx_{P/i} a'$*[3]

[3] $a'' \approx_{P/i} a'$ means $a'' \approx_P a'$ except for $\succeq_i$.

2. *There exists* $a' \in D \setminus Opt(P)$*, such that*

$$|\{a : a \in D, a \approx_P a'\}| > |Opt(P)|,$$

*and for every* $a'' \in Opt(P)$ *and for every* $l \leq i$*,* $a'' \approx_l a'$

3. *There exists* $a' \in D \setminus Opt(P)$*, such that*

$$|\{a : a \in D, a \approx_P a'\}| = |Opt(P)|,$$

*for every* $a'' \in Opt(P)$ *and for every* $l \leq i$*,* $a'' \approx_l a'$*, and for some* $w \in D$*,* $w \approx_{P/i} a'$ *and* $w' \not\approx_i a'$*.*

*Proof.* Let $P'$ be $P_{\succeq_i/\succeq}$.

($\Leftarrow$) In case 1, since $a'' \succ_P a'$ and $a'' \approx_{P/i} a'$, $a'' \succ_i a'$. Let us assume that $\succeq_i$ is given by

$$\succeq_i: \quad D^i_1 \succ_i \cdots \succ_i D^i_{m_i},$$

and $a'' \in D^i_j$ . Let us define $\succeq$ as follows:

$$\succeq: \quad D'_1 \succ \cdots \succ D'_{m_i},$$

where $D'_j = D^i_j \cup \{a'\}$, and $D'_k = D^i_k / \{a'\}$ for every $k \in [1..m_i]$, $k \neq j$.

We can get $a'' \approx_{P'} a'$, $a' \in Opt(P')$ and $Opt(P') = Opt(P) \cup \{a'\}$. Thus $Opt(P') \succ_i^{lmin} Opt(P)$.

In case 2, we can construct a total preorder $\succeq$ such that $Opt(P') = \{a : a \in D, a \approx_P a'\}$. Let us assume that $\succeq_i$ is given by

$$\succeq_i: \quad D^i_1 \succ_i \cdots \succ_i D^i_{m_i},$$

and $a', a'' \in D^i_j$ . Let us define $\succeq$ as follows:

$$\succeq: \quad D'_1 \succ \cdots \succ D'_{m_i+1},$$

where $D'_j = \{a : a \in D, a \approx_P a'\}$, $D'_{j+1} = D^i_j \setminus \{a : a \in D, a \approx_P a'\}$, $D'_k = D^i_k$ for every $k \in [1..j-1]$, and $D'_k = D^i_{k-1}$ for every $k \in [j+2..m_i+1]$.

Since $a' \approx_l a''$ for every $l < i$, $a' \succ_i a''$, we can get $a' \succ_{P'} a''$. Thus $Opt(P') = \{a : a \in D, a \approx_P a'\}$ and since $|\{a : a \in D, a \approx_P a'\}| > |Opt(P)|$, $Opt(P') \succ_i^{lmin} Opt(P)$.

In case 3, similarly, $Opt(P') = \{a : a \in D, a \approx_P a'\} \cup w$, since $|\{a : a \in D, a \approx_P a'\}| = |Opt(P)|$, $Opt(P') \succ_i^{lmin} Opt(P)$.

($\Rightarrow$) Assume there are $i \in \mathscr{A}$ and a total preorder $\succeq$, such that $Opt(P') \succ_i^{lmin} Opt(P)$ and none of the conditions is satisfied. We know that there is $a' \in Opt(P')$ but $a' \notin Opt(P)$, and for any $a'' \in Opt(P)$, $a'' \succ_P a'$.

1. If $a'' \succ_j a'$ where $j < i$, then $a'' \succ_{P'} a'$, a contradiction to $a' \in Opt(P')$.

2. If $a'' \succ_i a'$, since case 1 is not satisfied, $a' \not\approx_{P'} a''$ for any $a'' \in Opt(P)$. Since $a' \in Opt(P')$, $a'' \notin Opt(P')$ for any $a'' \in Opt(P)$. According to the definition of lexmin, since $a'' \succ_i a'$, $Opt(P') \not\succ_i^{lmin} Opt(P)$, a contradiction.

3. If $a'' \succ_j a'$ where $j > i$, we can get $a' \not\approx_i a''$ and $a' \not\approx_{P'} a''$. Since $a' \in Opt(P')$, $a'' \notin Opt(P')$ for any $a'' \in Opt(P)$. According to the definition of lexmin, let $a'' \in D_j^i$, $|Opt(P') \cap D_j^i| \geq |Opt(P) \cap D_j^i|$. We can get $Opt(P') \cap D_j^i = \{a : a \in D, a \approx_P a'\}$. Thus if $|Opt(P') \cap D_j^i| > |Opt(P) \cap D_j^i|$, it satisfies condition 2. If $|Opt(P') \cap D_j^i| = |Opt(P) \cap D_j^i|$, there must exists $w \in D$ and $w \in Opt(P') \cap D_{j'}^i$ where $j' \neq j$. Thus $w \approx_{P-i} a'$ and $w' \not\approx_i a'$. This satisfies the condition 3.

□

We will now consider simple bribery. There are rather intuitive conditions describing when simple bribery is possible for the *compare best*, *compare worst* and *average-rank* set comparison methods, and somewhat more complicated ones for *lexmin*.

**Theorem 26.** *Let $\mathscr{A}$ be a set of $N$ agents $1, \ldots, N$ with a profile of strictly ranked preferences $P = (\succeq_1, \ldots, \succeq_N)$. For every $i \in \mathscr{A}$, simple bribery is possible for* compare best, compare worst *and* average-rank *if and only if there exists $a' \in D$ such that $a' \succ_i a$ for every $a \in Opt(P)$.*

*Proof.* Since all optimal outcomes are indifferent, we can assume $a \in D^i_j$ for every $a \in Opt(P)$.

($\Leftarrow$) If such $a'$ exists, according to Theorem 16, $i \neq 1$. Let $t = 1$ and assume that $\succeq_1$ is given by

$$\succeq_1: \quad D^1_1 \succ_1 \ldots \succ_1 D^1_{m_1}.$$

We define $\succeq$ as follows:

$$\succeq: \quad D'_1 \succ \ldots \succ D'_{m_1+1},$$

where $D'_1 = \{a'\}$, and $D'_k = D^t_{k-1} \setminus \{a'\}$ for every $k \in \{2, \ldots, m_1+1\}$. Thus $Opt(P_{\succeq_1/\succeq}) = \{a'\}$. Since $a' \succ_i a$ for every $a \in Opt(P)$, we have $Opt(P_{\succeq_t/\succeq}) \succ_i^{cb/cw/ar} Opt(P)$.

($\Rightarrow$) If such $a'$ does not exist, $a \in D^i_1$ for every $a \in Opt(P)$. Thus there is no $P \subseteq D$ such that $P \succ_i^{cb/cw/ar} Opt(P)$. □

**Theorem 27.** *Let $\mathscr{A}$ be a set of $N$ agents $1, \ldots, N$ with a profile of strictly ranked preferences $P = (\succeq_1, \ldots, \succeq_N)$. For every $i \in \mathscr{A}$, simple bribery is possible for* lexmin *if and only if at least one of the following three conditions holds:*

1. *There is $a' \in D$ such that for all $a'' \in Opt(P)$, $a' \succ_i a''$*
2. *There is $a' \in D \setminus Opt(P)$ such that for some $t \in \mathscr{A}$, $t \neq i$, and for every $a'' \in Opt(P)$, $a' \approx_{P/t} a''$*
3. *There is $a' \in D$ and $t, j \in \mathscr{A}$ such that $t \leq j$, $t \neq i$,*

$$|\{a : a \in D, a \approx_{P/t} a'\}| > |Opt(P)|,$$

*for every $a'' \in Opt(P)$, $a' \approx_i a''$, $a'' \succ_j a'$, and $a'' \approx_l a'$, for every $l < j$.*

*Proof.* Let $P'$ be $P_{\succeq_t/\succeq}$.

($\Leftarrow$) In case 1, similar to Theorem 26, we can make $a' \in Opt(P')$ and thus $Opt(P') \succ_i^{lmin} Opt(P)$.

In case 2, we can get $a'' \succ_t a$. Let us assume that $\succeq_t$ is given by

$$\succeq_t: \quad D_1^t \succ_t \cdots \succ_t D_{m_t}^t,$$

and $a'' \in D_j^t$ . Let us define $\succeq$ as follows:

$$\succeq: \quad D_1' \succ \cdots \succ D_{m_t}',$$

where $D_j' = D_j^t \cup \{a'\}$, and $D_k' = D_k^t / \{a'\}$ for every $k \in [1..m_t]$, $k \neq j$. Then $a' \approx_{P'} a''$ and $Opt(P') = Opt(P) \cup \{a'\}$. Thus $Opt(P') \succ_i^{lmin} Opt(P)$.

In case 3, we can construct a total preorder $\succeq$ such that $Opt(P') = \{a : a \in D, a \approx_{P/t} a'\}$. Let us assume that $\succeq_t$ is given by

$$\succeq_t: \quad D_1^t \succ_t \cdots \succ_t D_{m_t}^t,$$

and $a' \in D_j^t$, $a'' \in D_{j'}^t$ where $j' \leq j$. Let us define $\succeq$ as follows:

$$\succeq: \quad D_1' \succ \cdots \succ D_{m_t+1}',$$

where $D_{j'}' = \{a : a \in D, a \approx_{P/t} a'\}$, $D_k' = D_k^t$ for every $k \in [1..j'-1]$, and $D_k' = D_{k-1}^t / \{a : a \in D, a \approx_{P/t} a'\}$ for every $k \in [j'+1..m_t+1]$. Since $|\{a : a \in D, a \approx_{P/t} a'\}| > |Opt(P)|$, $Opt(P') \succ_i^{lmin} Opt(P)$.

($\Rightarrow$) Assume there are $t \in \mathscr{A}$ and a total preorder $\succeq$, such that $Opt(P') \succ_i^{lmin} Opt(P)$ and none of the conditions is satisfied. We know that there is $a' \in Opt(P')$ but $a' \notin Opt(P)$, and for any $a'' \in Opt(P)$, $a'' \succ_P a'$.

1. If $a' \succ_i a''$, it satisfies the condition 1.

2. If $a'' \succ_i a'$, $a' \not\approx_{P'} a''$ for any $a'' \in Opt(P)$. Since $a' \in Opt(P')$, $a'' \notin Opt(P')$ for any $a'' \in Opt(P)$. According to the definition of Lexmin, since $a'' \succ_i a'$, $Opt(P') \not\succ_i^{lmin} Opt(P)$, a contradiction.

3. If $a' \approx_i a''$, since case 2 is not satisfied, $a' \not\approx_{P'} a''$ for any $a'' \in Opt(P)$. Similarly, $a'' \notin Opt(P')$ for any $a'' \in Opt(P)$. Let $a'' \in D_j^i$, we can get $|Opt(P') \cap D_j^i| \geq |Opt(P) \cap D_j^i|$.

If $|Opt(P') \cap D^i_j| = |Opt(P) \cap D^i_j|$, there must exists $w \in D$ and $w \in Opt(P') \cap D^i_{j'}$ where $j' \neq j$. Since we already know $w \not\succ_i a''$ and $a'' \not\succ_i w$, $w \approx_i a''$, a contradiction with $j' \neq j$. Thus $|Opt(P') \cap D^i_j| > |Opt(P) \cap D^i_j|$. Since all optimal outcomes are indifferent, $Opt(P') \cap D^i_j = \{a : a \in D, a \approx_{P/t} a'\}$. Thus $|\{a : a \in D, a \approx_{P/t} a'\}| > |Opt(P)|$ and it satisfies the condition 3.

□

## 5.4 Complexity

So far we studied the problems of manipulation and simple bribery ignoring the issue of how preferences (total preorders) on $D$ are represented. In this section, we will establish the complexity of deciding whether manipulation or simple bribery are possible. For this study, we have to fix a preference representation schema.

First, let us assume that preference orders on elements of $D$ are represented explicitly as sequences $D_1, \ldots, D_m$ of the indifference strata, enumerating them from the most preferred to the least preferred. For this representation, the characterizations we presented in the previous section imply that the problems of the existence of manipulation and bribery can be solved in polynomial time. Thus, in the "explicit representation" setting, computational complexity cannot serve as a barrier against them.

However, for combinatorial domains explicit representations are not feasible. We now take for $D$ a common combinatorial domain given by a set $U$ of binary attributes. We view elements of $U$ as propositional variables and assume that each element of $U$ can take a value from the domain $\{true, false\}$. In this way, we can view $D$ as the set of all truth assignments on $U$. Following a common convention, we identify a truth assignment on $U$ with the subset of $U$ consisting of elements that are true under the assignment. Thus, we can think of $D$ as the power set $\mathscr{P}(U)$ of $U$.

By taking this perspective, we can use a formula $\varphi$ over $U$ as a concise implicit representation of the set $M(\varphi) = \{X \subseteq U : X \models \varphi\}$ of all interpretations of $U$ (subsets of $U$) that

satisfy $\varphi$, and we can use sequences of formulas to define total preorders on $\mathscr{P}(U)$ $(= D)$.

A *preference statement* over $U$ is an expression

$$\varphi_1 > \varphi_2 > \cdots > \varphi_m, \tag{5.1}$$

where all $\varphi_i$s are formulas over $U$ and $\varphi_1 \vee \cdots \vee \varphi_m$ is a tautology. A preference statement $p = \varphi_1 > \varphi_2 > \cdots > \varphi_m$ determines a sequence $(D_1, \ldots, D_m)$ of subsets of $\mathscr{P}(U)$, where, for every $i = 1, \ldots, m$,

$$D_i = \{X \subseteq U : X \models \varphi_i\} \setminus (D_1 \cup \cdots \cup D_{i-1}).$$

These subsets are disjoint and cover the entire domain $\mathscr{P}(U)$ (the latter by the fact that $\varphi_1 \vee \cdots \vee \varphi_m$ is a tautology). It follows that if $X \subseteq U$, then there is a unique $i_X$ such that $X \in D_{i_X}$. The relation $\succeq_p$ defined so that $X \succeq_p Y$ precisely when $i_X \leq i_Y$ is a total preorder on $\mathscr{P}(U)$. We say that the preference expression $p$ *represents* the preorder $\succeq_p$.[4]

We will now study the complexity of the existence of manipulation and simple bribery when preferences are given in terms of preference statements. That is, we assume that the input to these problems consists of $N$ ranked preferences $\succeq_{1,r_1}, \ldots, \succeq_{N,r_N}$. We will denote by $(D^i_1, \ldots, D^i_{m_i})$ the sequence of indifference strata determined by $\succeq_{i,r_i}$, as defined above. We refer to these two problems as the *existence-of-manipulation* (EM) problem and the *existence-of-simple-bribery* (ESB) problem, respectively. These problems are parameterized by the method used to compare sets. We denote the methods by *cb* (*compare best*), *cw* (*compare worst*), *lmin* (*lexmin*) and *ar* (*average-rank*).

For equally ranked preference statements, since manipulation is impossible for the *compare best* and *compare worst* methods for comparing sets, the problems are (trivially) in P. Similarly, the problem of deciding whether simple bribery is possible for *compare best* is in P, too. The summary of the complexity results for all the cases is given in Table 5.1 and proved in Theorem 28.

[4]The partition of $D$ into strata that is determined by $\succeq_p$ is not always $(D_1, \ldots, D_m)$ as some sets $D_i$ may be empty.

Table 5.1: Complexity results of manipulation and bribery for equally ranked preferences

| | EM | ESB |
|---|---|---|
| *cb* | P | P |
| *cw* | P | $\Sigma_2^P$-hard, $\Pi_2^P$-hard, in $\Delta_3^P$ |
| *lmin* | NP-complete | NP-complete |
| *ar* | $\Sigma_2^P$-hard, in PSPACE | $\Sigma_2^P$-hard, in PSPACE |

**Lemma 1.** *The EM$^{ar}$ and EB$^{ar}$ problems for equally ranked preferences restricted to the case when the agent seeking manipulation or bribery, respectively, has a two-level preference are $\Sigma_2^P$-complete.*

*Proof.* For the problem EM$^{ar}$, Corollary 1 states that if $P$ is a profile in which agent $i$ has a two-cluster preference

$$\succeq_i: \quad \varphi_1 \succ_i \varphi_2$$

then agent $i$ can manipulate preference aggregation if an only if there are outcomes $M', M'' \subseteq U$ such that

1. $M' \notin Opt(P)$, $M'' \in Opt(P)$, $d_{\succeq_i}(M') = 1$, and $d_{\succeq_i}(M'') = 2$; or
2. $M', M'' \in Opt(P)$, $M' \neq M''$, $M' \approx_P M''$, and $d_{\succeq_i}(M') = 2$.

Let us consider an algorithm that non-deterministically selects two outcomes $M', M'' \subseteq U$ and then, with the help of an oracle for the problem to decide whether an outcome is optimal for a profile, verifies that $M' \notin Opt(P)$, $M'' \in Opt(P)$, $d_{\succeq_i}(M') = 1$, and $d_{\succeq_i}(M'') = 2$; or $M' \in Opt(P)$, $M'' \in Opt(P)$, $M' \neq M''$, $M' \approx_P M''$, and $d_{\succeq_i}(M') = 2$. From the comment above it follows that this non-deterministic algorithm correctly decides whether $i$ can manipulate. Moreover, it runs in polynomial time (assuming that we count each call to the oracle as taking constant time). Thus, the membership follows.

For the hardness, we reduce to our problem the problem to decide for a profile $P$ over a set $U$ and an atom $a \in U$ whether there is an outcome $M \in Opt(P)$ such that $a \in M$. That problem is $\Sigma_2^P$-complete [27].

Thus, let us consider a profile $P$ over $U$. We define the profile $P' = (\succeq'_1, \ldots, \succeq'_N)$ as follows. Assuming that $\succeq_1$ is of the form

$$\succeq_1 \colon\ \varphi_1 \succ_1 \ldots \succ_1 \varphi_m,$$

we set

$$\succeq'_1 \colon\ \varphi_1 \wedge a \succ'_1 \ldots \succ'_1 \varphi_m \wedge a \succ'_1 \neg a.$$

For $i = 2, \ldots, N$, we set $\succeq'_i = \succeq_i$. Then, for every $M \subseteq U$ such that $a \in M$, $M \in Opt(P)$ if and only if $M \in Opt(P')$. Moreover, $P'$ has the following property: if $M \approx_{P'} M'$ and $a \in M$, then $a \in M'$.

Let us assume that $U = \{a, x_1, \ldots, x_{n-1}\}$. We introduce agents $0, N+1, \ldots N+2(n-1)$. We define a profile $P''$ of the extended set of agents by setting $\succeq''_i = \succeq'_i$, for $i = 1, \ldots, N$ and by defining preferences of the new agents as follows:

$$\begin{aligned}
&\succeq''_0 \colon\ \neg a \succ a \\
&\succeq''_{N+2i-1} \colon\ \neg a \wedge x_i \succ a \vee \neg x_i \quad \textit{for each } i \in [1..n-1] \\
&\succeq''_{N+2i} \colon\ \neg a \wedge \neg x_i \succ a \vee x_i \quad \textit{for each } i \in [1..n-1].
\end{aligned}$$

We will now show that every set $Y \subseteq U$ such that $a \notin Y$ is optimal in $P''$. Indeed, let us consider $Y' \subseteq U$ such that $Y' \succeq_{P''} Y$. This implies that $Y' \succeq''_0 Y$, and so $a \notin Y'$. Since for every $j = 1, \ldots, n-1$, $Y' \succeq''_{N+2j-1} Y$ and $Y' \succeq''_{N+2j} Y$, $x_j \in Y$ if and only if $x_j \in Y'$. Thus, $Y = Y'$, which proves optimality of $Y$ in $P''$.

We now introduce a fresh element (atom) $b$ and define $U' = U \cup \{b\}$. We view $P''$ as a profile over $U' = U \cup \{b\}$. Clearly, for every $M \subseteq U$, $M \approx_{P''} M \cup \{b\}$.

Let us assume that $M \subseteq U$, $a \in M$ and $M \in Opt(P)$. We will show that $M \in Opt(P'')$. To this end, let us consider $M' \subseteq U'$ such that $M' \succeq_{P''} M$ and let us assume first that $b \notin M'$. Thus, $M' \subseteq U$. Clearly, $M' \succeq_{P'} M$. Since $M \in Opt(P')$, $M' \approx_{P'} M$. Consequently, $a \in$

$M'$ (otherwise, the degrees of quality of $M$ and $M'$ on preference $\succeq'_1$ would be different) and it follows that $M' \approx_{P''} M$. Next, let us assume that $b \in M'$ and set $M'' = M' \setminus \{b\}$. Then $M'' \succeq_{P''} M$ (the absence or presence of $b$ does not affect the degrees of quality). Consequently, $M'' \approx_{P''} M$ and so, also $M' \approx_{P''} M$. It follows that $M$ is optimal in $P''$. Moreover, $M \cup \{b\}$ is optimal in $P''$, too. Since $M \approx_{P''} M \cup \{b\}$, agent 0 can manipulate preference aggregation in $P''$.

Conversely, let us assume that agent 0 can manipulate preference aggregation in profile $P''$. Since all outcomes that do not contain $a$ are optimal in $P''$, it follows that there is an outcome $M \subseteq U'$ such that $a \in M$ and $M \in Opt(P'')$. Without loss of generality, we can assume that $b \notin M$. Therefore, $M \subseteq U$. We will prove that $M \in Opt(P')$. Since $a \in M$, that will imply that $P$ has an optimal outcome containing $a$.

To show that $M \in Opt(P')$, let us consider any $M' \subseteq U$ such that $M' \succeq_{P'} M$. If $a \notin M'$, then $M' \succ''_0 M$ and $M' \succeq''_j M$, for $j = N+1, \ldots, N+2(n-1)$. Thus, $M' \succ_{P''} M$, a contradiction. Thus, $a \in M'$. Since $M'$ and $M$ have the same degrees of quality on all preferences $\succeq''_j$, $j = 0, N+1, \ldots, N+2(n-1)$, $M' \succeq_{P''} M$. Since $M \in Opt(P'')$, $M' \approx_{P''} M$ and so, $M' \approx_{P'} M$. Thus, $M \in Opt(P')$, as claimed.

The problem EB$^{ar}$ is similar to EM$^{ar}$. Corollary 2 states that if $P$ is a profile in which agent $i$ has a two-cluster preference

$$\succeq_i: \quad \varphi_1 \succ_i \varphi_2$$

then agent $i$ can manipulate preference aggregation if an only if there are outcomes $M', M'' \subseteq U$ such that

1. $M' \notin Opt(P)$, $M'' \in Opt(P)$, $d_{\succeq_i}(M') = 1$, and $d_{\succeq_i}(M'') = 2$; or

2. $M', M'' \in Opt(P)$, $d_{\succeq_i}(M') = 2$, and $M'' \succeq_k M'$, for every $k \in \mathscr{A}$, $k \neq t$.

Thus we can guess two outcomes $M', M'' \subseteq U$ as we did for the problem EM$^{ar}$. The only difference is in EB$^{ar}$ we checked whether $M' \approx_P M''$ and here we guess $t$ and check whether

$M'' \succeq_k M'$, for every $k \in \mathscr{A}$, $k \neq t$. We can also do this in polynomial time assuming each call to the oracle taking constant time.

For the hardness, we can reduce the problem EM$^{ar}$ to EB$^{ar}$. Consider a profile $P = (\succeq_1 , \ldots, \succeq_m)$ over $U$ where

$$\succeq_i: \quad \varphi_1 \succ_i \varphi_2.$$

We introduce a new agent 0 and construct a profile $P' = (\succeq'_0, \succeq'_i)$ as follows. We set

$$\succeq'_0: \quad \varphi_1 \succ'_0 \varphi_2$$

and $\succeq'_i = \succeq_i$. Since agent 0 and agent $i$ have the same preference, agent 0 can bribe some other agent (in our case, it can only be agent $i$) to misrepresent her preference if and only if agent $i$ in profile $P$ can manipulate. Thus problem EB$^{ar}$ is also $\Sigma_2^P$-complete. □

**Theorem 28.** *The complexity of deciding whether manipulation and simple bribery is possible for equally ranked preferences with four ways to lift the preorder over outcomes to a preorder over sets of outcomes is given in the following table:*

| | *EM* | *ESB* |
|---|---|---|
| *cb* | *P* | *P* |
| *cw* | *P* | $\Sigma_2^P$*-hard,* $\Pi_2^P$*-hard, in* $\Delta_3^P$ |
| *lmin* | *NP-complete* | *NP-complete* |
| *ar* | $\Sigma_2^P$*-hard, in PSPACE* | $\Sigma_2^P$*-hard, in PSPACE* |

*Proof.* According to Theorem 17 and 20, manipulation is impossible for *compare best* and *compare worst*, and bribery is possible for *compare best,* these problems are (trivially) in P.

Bribery for *compare worst*:

We want to show that this problem is in $\Delta_3^P$ and is both $\Sigma_2^P$- and $\Pi_2^P$-hard.

Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of equally ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$ and let $i \in \mathscr{A}$, Theorem 21 states that there is another agent $t$ such that $Opt(P_{\succeq_t/\succeq}) \succ_i^{cw} Opt(P)$ if and only if for every $a \in Min_{\succeq_i}(Opt(P))$, there is $a' \in D$ such that $a' \succ_i a$, and $a' \succeq_k a$, for every agent $k$, $k \neq t$. Let us assume

$$\succeq_i:\ \varphi_1 \succ_i \cdots \succ_i \varphi_m,$$

and $Min_{\succeq_i}(Opt(P)) = \{X : X \in Opt(P), X \models \neg\varphi_1 \wedge \cdots \wedge \neg\varphi_{j-1} \wedge \varphi_j\}$. To find out $j$, we can first check whether there exists $M \in Opt(P)$ such that $M \models \neg\varphi_1 \wedge \cdots \wedge \neg\varphi_{m-1} \wedge \varphi_m$ using a $\Sigma_2^P$ oracle [27]. If there exists such outcome, $j = m$. Otherwise, we check whether there exists $M \in Opt(P)$ such that $M \models \neg\varphi_1 \wedge \cdots \wedge \neg\varphi_{m-2} \wedge \varphi_{m-1}$ also using a $\Sigma_2^P$ oracle. We will iteratively check the existence of optimal outcome until we find $M \in Opt(P)$ such that for every $M' \in Opt(P)$, $M' \succeq_i M$. This can be achieved by an algorithm using a $\Sigma_2^P$ oracle polynomial times. Then we need to check whether for every $a \in Min_{\succeq_i}(Opt(P))$, there is $a' \in D$ such that $a' \succ_i a$, and $a' \succeq_k a$, for every agent $k$, $k \neq t$. To do that, we can check whether there exists $a \in Opt(P)$ and $a \models \neg\varphi_1 \wedge \cdots \wedge \neg\varphi_{j-1} \wedge \varphi_j$ such that for every $a' \in D$, $a \succeq_i a'$ or $a \succ_k a'$ for some agent $k$, $k \neq t$ with a $\Sigma_2^P$ oracle. Thus our problem can be solved by an algorithm using a $\Sigma_2^P$ oracle polynomial times, and it is in $\Delta_3^P$.

For the $\Sigma_2^P$-hardness, we reduce to our problem the problem to decide for a profile $P$, over a set $U$, and an atom $a \in U$ whether there is an outcome $M \in Opt(P)$ such that $a \in M$. This problem is $\Sigma_2^P$-complete [27].

Let us consider a profile $P = (\succeq_1,\ldots,\succeq_N)$ over $U$ where

$$\succeq_1: \varphi_1 \succ_1 \ldots \succ_1 \varphi_m.$$

We define the profile $P' = (\succeq'_1,\ldots,\succeq'_N)$ as follows. We set

$$\succeq'_1: \varphi_1 \wedge a \succ'_1 \ldots \succ'_1 \varphi_m \wedge a \succ'_1 \neg a.$$

For $i = 2,\ldots,N$, we set $\succeq'_i = \succeq_i$. Clearly, for every $M \subseteq U$ such that $a \in M$, $M \in Opt(P)$ if and only if $M \in Opt(P')$.

We introduce agents 0 and $N+1$, a new atom $b$ and define $U' = U \cup \{b\}$. We define a profile $P''$ of the extended set of agents by setting $\succeq''_i = \succeq'_i$, for $i = 1, \ldots, N$ and by defining preferences of the new agents as follows:

$$\succeq''_0 : \neg a \succ'_0 a \wedge \neg b \succ'_0 a \wedge b$$
$$\succeq''_{N+1} : b \succ'_{N+1} \neg b.$$

Let $M \in Opt(P)$ and $a \in M$. We will show that $M, M \cup \{b\} \in Opt(P'')$. To simplify, we write $M'$ for $M \cup \{b\}$. Let us consider $M'' \subseteq U'$ and $M'' \succeq_{P''} M$. This implies that $M'' \succeq''_1 M$, and so $a \in M''$. Since $M'' \succeq''_0 M$, $M'' \approx''_0 M$. Since $M \in Opt(P)$, we can get that $M \in Opt(P')$, and $M'' \approx_{P'} M$. If $M'' \succ''_{N+1} M$, we can get that $b \in M''$, and $M \succ''_0 M''$ contradicts to $M'' \succeq_{P''} M$. Thus $M'' \approx''_{N+1} M$ and $M'' \approx_{P''} M$. This implies $M \in Opt(P'')$. Similarly, let us consider $M'' \subseteq U'$ and $M'' \succeq_{P''} M'$. This implies that $M'' \succeq''_{N+1} M'$, and so $b \in M''$. Since $M \approx_P M'$, we can get $M' \in Opt(P)$, $M' \in Opt(P')$ and $M'' \approx_{P'} M'$. If $M'' \succ''_0 M'$, we can get that $a \notin M''$, and $M' \succ''_1 M''$ contradicts to $M'' \succeq_{P''} M'$. Thus $M'' \approx_{P''} M'$ and $M' \in Opt(P'')$.

Since $M, M' \in Opt(P'')$, $Min_{\succeq''_0}(Opt(P'')) = \{X : X \in Opt(P''), a \in X, b \in X\}$. We want to show that there exist $t \in [0, \ldots, N+1]$, for every $X \in Min_{\succeq''_0}(Opt(P''))$, there is a $X' \subseteq U'$ such that $X' \succ''_0 X$, and $X' \succeq''_k X$ for every $k \in [0, \ldots, N+1]$, $k \neq t$. Let $t$ be $N+1$. For every $X \in Min_{\succeq''_0}(Opt(P''))$, let $X'$ be $X \setminus \{b\}$. Obviously, $X' \succ''_0 X$, and $X' \approx_{P'} X$. Thus if agent $N+1$ change her preference to $\succeq$: $\neg b \succ b$, for every $X \in Min_{\succeq''_0}(Opt(P''))$, $X \notin Opt(P''_{\succeq''_{N+1}/\succeq})$.

Conversely, let us assume that agent 0 can find another agent to misrepresent her preference. If for every $M \in Opt(P'')$, $a \notin M$, agent 0 cannot improve the quality of optimal outcomes. Thus there exist some $M \in Opt(P'')$ and $a \in M$. Without loss of generality, we can assume that $b \notin M$. We want to show that $M \in Opt(P)$. Let us consider $M' \subseteq U$ and $M' \succeq_P M$. Thus $a \in M'$, $M' \approx''_0 M$ and $M' \approx''_{N+1} M$. Since $M' \succeq_P M$ and $a \in M$, $M' \succeq_{P'} M$.

If $M' \succ_{P'} M$, then $M' \succ_{P''} M$ contradicts to $M \in Opt(P'')$. Thus $M' \approx_{P'} M$, and $M' \approx_P M$. That means $M \in Opt(P)$.

We have proved that our problem is $\Sigma_2^P$-hard. It is easy to see that the complement of our problem is $\Pi_2^P$-hard. Since our problem is in $\Delta_3^P$, the complement problem is as hard as the original one. Thus our problem is $\Pi_2^P$-hard.

Manipulation and bribery for *lexmin*:

We want to show that these two problems are NP-complete.

We recall that both problems concern profiles of preference statements over a set of atoms $U$, with subsets of $U$ (viewed as truth assignments) as the set of outcomes. According to Theorems 18 and 22, manipulation (bribery) for a profile $P$ is possible if and only if $Opt(P) \neq \mathscr{P}(U)$.

Clearly, we can decide whether $Opt(P) \neq \mathscr{P}(U)$ by the following non-deterministic algorithm: guess two outcomes $M, M' \subseteq U$, and check that $M' \succ_P M$. That latter task can be executed in polynomial time. Indeed, for a given set $M \subseteq U$ and a propositional formula $\varphi$ over $U$, checking $M \models \varphi$ takes polynomial time. This allows us to compute the quality degrees of $M$ and $M'$ for all preference statements in $P$ and, consequently, to compare them with respect to the profile $P$, in polynomial time. It follows that the manipulation (bribery) problem is in NP.

For the NP-hardness, we show that the SAT problem can be reduced to the problem to decide whether $Opt(P) \neq \mathscr{P}(U)$. To this end, let us consider a SAT instance $\varphi$ over a set $U$ of atoms. We introduce a new atom $a$ and define $U' = U \cup \{a\}$. We define a profile $P = (\succeq)$ over $U'$ (a one-agent profile) as follows:

$$\succeq: \quad a \vee \neg\varphi \succ \varphi \wedge \neg a.$$

To complete the argument, we show that $Opt(P) \neq \mathscr{P}(U)$ if and only if $\varphi$ is satisfiable. Let us assume that $M$ is a model of $\varphi$ and let $M' = \{a\}$. Clearly, $M' \succ_P M$, that is $M \notin Opt(P)$. Conversely, if there is an outcome $M \subseteq U'$ that is not optimal in $P'$, then $d_{\succeq}(M') = 2$, that is $M$ is a model of $\varphi \wedge \neg a$. It follows that $\varphi$ is satisfiable.

Table 5.2: Complexity results of manipulation and bribery for strictly ranked preferences

| | EM | ESB |
|---|---|---|
| *cb* | P | $\Delta_2^P$-complete |
| *cw* | P | $\Delta_2^P$-complete |
| *lmin* | $\Delta_2^P$-hard | $\Delta_2^P$-hard |
| *ar* | P | $\Delta_2^P$-complete |

Manipulation and bribery for *average-rank*:

We want to show that these two problems are $\Sigma_2^P$-hard and in PSPACE.

Lemma 1 provides a lower bound for the complexity for the the general case. Since both problems are clearly in PSPACE, we obtain the result. □

For strictly ranked preferences, for the *compare best*, *compare worst* and *average-rank* methods, the manipulation is impossible and so, deciding its existence is trivially in P. On the other hand, simple bribery is possible for all set comparison methods. The complete complexity results are given in Table 5.2 and proved by Theorem 29.

**Lemma 2.** *Given a Boolean formula $\varphi(x_1,\ldots,x_n)$ such that $\varphi \wedge x_n$, $\varphi \wedge \neg x_n$ are satisfiable, deciding whether $x_n = 1$ is in $\varphi$'s maximum satisfying assignment is $\Delta_2^P$-complete.*

*Proof.* To show this problem is in $\Delta_2^P$, we can first check whether $\varphi \wedge x_1$ is satisfiable. If it is, we know that $x_1 = 1$ is in $\varphi$'s MSA, and we can check whether $\varphi \wedge x_1 \wedge x_2$ is satisfiable; otherwise, $x_1 = 0$ is in $\varphi$'s MSA and we check whether $\varphi \wedge \neg x_1 \wedge x_2$ is satisfiable. In a similar way, we can get the values of $x_2,\ldots,x_n$ in $\varphi$'s MSA by $n$ SAT checks. Then we can see whether $x_n = 1$ is in $\varphi$'s MSA.

To prove the hardness, we reduce a $\Delta_2^P$-complete problem to this one. Given a Boolean formula $\varphi(x_1,\ldots,x_n)$, it is $\Delta_2^P$-complete to decide whether $x_n = 1$ is in $\varphi$'s MSA. We construct a Boolean formula $\psi = (\varphi \wedge (x_n \vee \neg x_{n+1})) \vee (\neg x_1 \wedge \cdots \wedge \neg x_{n-1} \wedge \neg x_n \wedge x_{n+1}) \vee (\neg x_1 \wedge \cdots \wedge \neg x_{n-1} \wedge x_n \wedge \neg x_{n+1})$. Formula $\psi$ is satisfiable and $M_1, M_2 \models \psi$ where $M_1 = \{\neg x_1,\ldots,\neg x_{n-1},\neg x_n,x_{n+1}\}$ and $M_2 = \{\neg x_1,\ldots,\neg x_{n-1},x_n,\neg x_{n+1}\}$.

($\Leftarrow$) We show that if $x_n = 1$ is in $\varphi$'s MSA, then $x_{n+1} = 1$ is in $\psi$'s MSA.

Let $M$ be $\varphi$'s MSA. Then $M'$ is a model for $\psi$ where $M' = M \cup \{x_{n+1}\}$. Assume $M''$ is the MSA of $\psi$ and $M'' \neq M'$, then $M'' \not\models \varphi \wedge (x_n \vee \neg x_{n+1})$ and $M'' = M_1$ or $M'' = M_2$. Since $M_2$ is lexicographically larger than $M_1$, $M'' = M_2$. Since $M' \models x_n \wedge x_{n+1}$, $M'$ is lexicographically larger than $M''$ and $M''$ cannot be the MSA. Thus $M'$ is the MSA of $\psi$ and $x_{n+1} = 1$ in $M'$.

($\Rightarrow$) We show that if $x_n = 1$ is not in $\varphi$'s MSA, then $x_{n+1} = 1$ is not in $\psi$'s MSA.

There are two possibilities. One is $\varphi$ is unsatisfiable. In this case, $\psi$ has two models $M_1$ and $M_2$, and $M_2$ is the MSA of $\psi$. Since $M_2 \models \neg x_{n+1}$, $x_{n+1} = 1$ is not in $\psi$'s MSA. The other possibility is $x_n = 0$ is in $\varphi$'s MSA. Let $M$ be $\varphi$'s MSA. Then $M'$ is a model for $\psi$ where $M' = M \cup \{\neg x_{n+1}\}$. Let $M''$ be the MSA of $\psi$. If $M'' = M'$, $x_{n+1} = 1$ is not in $\psi$'s MSA. If $M'' \neq M'$, then because of the similar reason $M'' = M_2$. Since $M_2 \models \neg x_{n+1}$, $x_{n+1} = 1$ is not in $\psi$'s MSA. □

**Theorem 29.** *The complexity of deciding whether manipulation and simple bribery is possible for strictly ranked preferences with four ways to lift the preorder over outcomes to a preorder over sets of outcomes is given in the following table:*

| | *EM* | *ESB* |
|---|---|---|
| *cb* | *P* | $\Delta_2^P$*-complete* |
| *cw* | *P* | $\Delta_2^P$*-complete* |
| *lmin* | $\Delta_2^P$*-hard* | $\Delta_2^P$*-hard* |
| *ar* | *P* | $\Delta_2^P$*-complete* |

*Proof.* According to Theorem 24, manipulation is impossible for *compare best*, *compare worst* and *average-rank*. Thus these problems are in P.

Manipulation for *lexmin*:

Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of strictly ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$. According to Theorem 25, we want to show that for any $i \in \mathscr{A}$, deciding whether at least one of the following conditions holds:

1. There exists $a' \in D \setminus Opt(P)$, such that for every $a'' \in Opt(P)$, $a'' \approx_{P/i} a'$

2. There exists $a' \in D \setminus Opt(P)$, such that

$$|\{a : a \in D, a \approx_P a'\}| > |Opt(P)|,$$

   and for every $a'' \in Opt(P)$ and for every $l \leq i$, $a'' \approx_l a'$

3. There exists $a' \in D \setminus Opt(P)$, such that

$$|\{a : a \in D, a \approx_P a'\}| = |Opt(P)|,$$

   for every $a'' \in Opt(P)$ and for every $l \leq i$, $a'' \approx_l a'$, and for some $w \in D$, $w \approx_{P/i} a'$ and $w' \not\approx_i a'$.

is $\Delta_2^P$-hard.

Given a Boolean formula $\varphi(x_1, \ldots, x_n)$ and $\varphi \wedge x_n$, $\varphi \wedge \neg x_n$ are satisfiable, according to Lemma 2, it is $\Delta_2^P$-complete to decide whether $x_n = 1$ is in $\varphi$'s MSA. We introduce a new atom $x_{n+1}$ and construct the profile $P = (\succeq_{1,1}, \ldots, \succeq_{n,n}, \succeq_{n+1,n+1})$ as follows. We define

$$\succeq_{i,i}: \quad \psi \wedge x_i > \psi \wedge \neg x_i > \neg \psi$$

for $i \in (1..n)$, and

$$\succeq_{n+1,n+1}: \quad x_{n+1} > \neg x_{n+1}$$

where $\psi = \varphi \wedge (x_n \vee \neg x_{n+1})$. Let $i = n+1$.

($\Leftarrow$) We show that if $x_n = 1$ is in $\varphi$'s MSA, then there exists $a' \in D$, $a' \notin Opt(P)$, and $a'' \approx_{P/i} a'$ for every $a'' \in Opt(P)$.

Let $M$ be the MSA of $\varphi$, $M_1$ be $M \cup \{x_{n+1}\}$ and $M_2$ be $M \cup \{\neg x_{n+1}\}$. Since $M \models x_n$, $M_1, M_2 \models \psi$. We want to show that $M_1$ is the optimal outcome for $P$. If it is not, there is $M' \in D$ such that $M' \succ_P M_1$. Then $M'/\{x_{n+1}\}$ is lexicographical larger than $M$, a contradiction with $M$ is the MSA of $\varphi$. Thus $M_1$ is the optimal outcome for $P$, $M_1 \succ_P M_2$, and $M_2 \approx_{P/i} M_1$.

($\Rightarrow$) We show that if $x_n = 1$ is not in $\varphi$'s MSA, then none of the three conditions is satisfied.

Since $\varphi$ is satisfiable, let $M$ be the MSA. Thus $x_n = 0$ is in $M$. Let $M'$ be $M \cup \{\neg x_{n+1}\}$, $M'$ is an outcome for $P$. We want to prove that $M'$ is the optimal outcome for $P$. Assume there is $M''$ such that $M'' \succ_P M'$. If $M'' \succ_{P/n+1} M'$, $M''/\{x_{n+1}\}$ is lexicographical larger than $M$, a contradiction with $M$ is the MSA of $\varphi$. If $M'' \approx_{P/n+1} M'$ and $M'' \succ_{n+1} M'$, then $M'' = M \cup \{x_{n+1}\}$, $M'' \not\models \psi$ and $M' \succ_{P/n+1} M''$. Thus $M'$ is the optimal outcome for $P$.

1. For any $a \approx_{P/n+1} M'$, $a = M'$. Thus the first condition is not satisfied;

2. For any $a \approx_l M'$ for every $l \leq n+1$, $a = M'$. Thus the second condition is not satisfied;

3. Similarly, the third condition is not satisfied.

Bribery for *compare best*, *compare worst* and *average-rank*:

Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of strictly ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$. According to Theorem 26, we want to show that for any $i \in \mathscr{A}$, deciding whether there exists $a' \in D$ such that $a' \succ_i a$ for every $a \in Opt(P)$ is $\Delta_2^P$-complete.

To show this problem is in $\Delta_2^P$, assume preference $\succeq_{i,i}$ is in the form $\varphi_1^i > \varphi_2^i > \cdots > \varphi_{m_i}^i$ and $\varphi_1^i \vee \varphi_2^i \vee \cdots \vee \varphi_{m_i}^i$ is a tautology. We first check whether $\varphi_1^1$ is satisfiable, if it is not, continue to check whether $\varphi_2^1$ is satisfiable until find the first $\varphi^1$ which is satisfiable and denote it by $\varphi_{j_1}^1$. Then we check whether $\varphi_{j_1}^1 \wedge \varphi_1^2$ is satisfiable and in a similar way to find $\varphi_{j_2}^2$. By polynomial number of SAT checks, we can get $\varphi_{j_1}^1, \varphi_{j_2}^2, \ldots, \varphi_{j_N}^N$, and for any $a \in D$, $a \in Opt(P)$ if and only if $a \models \varphi_{j_1}^1 \wedge \varphi_{j_2}^2 \wedge \cdots \wedge \varphi_{j_N}^N$. By checking whether $\varphi_1^i \wedge \cdots \wedge \varphi_{j_i-1}^i$ is satisfiable, we can decide whether there exists $a' \in D$ such that $a' \succ_i a$ for every $a \in Opt(P)$.

For hardness, given a Boolean formula $\varphi(x_1,\ldots,x_n)$ and $\varphi \wedge x_n$, $\varphi \wedge \neg x_n$ are satisfiable, according to Lemma 2, it is $\Delta_2^P$-complete to decide whether $x_n = 1$ is in $\varphi$'s MSA. We construct the profile $P = (\succeq_{1,1},\ldots,\succeq_{n,n},\succeq_{n+1,n+1})$ as follows:

$$\succeq_{i,i}: \quad \varphi \wedge x_i > \varphi \wedge \neg x_i > \neg\varphi$$

for $i \in (1..n)$, and

$$\succeq_{n+1,n+1}: \quad \neg x_n > x_n.$$

Let $i = n+1$.

($\Leftarrow$) We show that if $x_n = 1$ is in $\varphi$'s MSA, then there exists $a' \in D$ such that $a' \succ_i a$ for every $a \in Opt(P)$.

Since $\varphi$is satisfiable, any outcome which is not a model of $\varphi$ is not optimal for $P$. Let $M$ be the MSA of $\varphi$, $M$ is the unique optimal outcome for $P$ because for any $M' \models \varphi$, $M \succ_{P/n+1} M'$ and also $M \succ_P M'$. Since $M \models x_n$ and there exists $M'' \neq M$ and $M'' \models \varphi \wedge \neg x_n$, we can get $M'' \succ_i M$.

($\Rightarrow$) We show that if there exists $a' \in D$ such that $a' \succ_i a$ for every $a \in Opt(P)$, then $x_n = 1$ is in $\varphi$'s MSA.

If there exists $a' \in D$ such that $a' \succ_i a$ for every $a \in Opt(P)$, it can only be the case $a' \models \neg x_n$ and $a \models x_n$ for every $a \in Opt(P)$. It is easy to see that $a \models \varphi$. We want to show that $a$ is the MSA for $\varphi$. Let $M$ be the MSA. Assume $M \neq a$, then $M \succ_{P/n+1} a$ and also $M \succ_P a$, a contradiction with $a \in Opt(P)$. Thus $M = a$ and $x_n = 1$ is in $M$.

Bribery for *lexmin*:

Let $\mathscr{A}$ be a set of $N$ agents $1,\ldots,N$ with a profile of strictly ranked preferences $P = (\succeq_1,\ldots,\succeq_N)$. According to Theorem 27, we want to prove that for any $i \in \mathscr{A}$, deciding whether at least one of the following three conditions holds:

1. There is $a' \in D$ such that for all $a'' \in Opt(P)$, $a' \succ_i a''$
2. There is $a' \in D \setminus Opt(P)$ such that for some $t \in \mathscr{A}$, $t \neq i$, and for every $a'' \in Opt(P)$, $a' \approx_{P/t} a''$
3. There is $a' \in D$ and $t, j \in \mathscr{A}$ such that $t \leq j$, $t \neq i$,

$$|\{a : a \in D, a \approx_{P/t} a'\}| > |Opt(P)|,$$

for every $a'' \in Opt(P)$, $a' \approx_i a''$, $a'' \succ_j a'$, and $a'' \approx_l a'$, for every $l < j$.

is $\Delta_2^P$-hard.

Given a Boolean formula $\varphi(x_1,\ldots,x_n)$ and $\varphi \wedge x_n$, $\varphi \wedge \neg x_n$ are satisfiable, according to Lemma 2, it is $\Delta_2^P$-complete to decide whether $x_n = 1$ is in $\varphi$'s MSA. We introduce a new atom $x_{n+1}$ and construct the profile $P = (\succeq_{1,1},\ldots,\succeq_{2n,2n},\succeq_{2n+1,2n+1})$ as follows. We define

$$\succeq_{2i-1,2i-1}: \quad \varphi \wedge x_i > \varphi \wedge \neg x_i > \neg\varphi$$

$$\succeq_{2i,2i}: \quad \varphi \wedge x_i > \varphi \wedge \neg x_i > \neg\varphi$$

for $i \in (1..n)$, and

$$\succeq_{2n+1,2n+1}: \quad \neg x_n > x_n.$$

Let $i = 2n+1$.

($\Leftarrow$) We show that if $x_n = 1$ is in $\varphi$'s MSA, then there exists $a' \in D$, $a' \succ_i a''$ for any $a'' \in Opt(P)$.

Let $M$ be the MSA of $\varphi$. Since any $M' \in D$, $M \succ_{P/2n+1} M'$. Thus $M$ is the optimal outcome for $P$. Since $\varphi \wedge \neg x_n$ is satisfiable, let $M'' \models \varphi \wedge \neg x_n$. Thus $M'' \succ_{2n+1} M$.

($\Rightarrow$) We show that if $x_n = 1$ is not in $\varphi$'s MSA, then none of the three conditions is satisfied.

Since $\varphi$ is satisfiable, let $M$ be the MSA. Thus $x_n = 0$ is in $M$. We know that $M$ is the optimal outcome for $P$.

1. Since $M \models \neg x_n$, there is no $M' \succ_{2n+1} M$ and the first condition is not satisfied.

2. If there is $a \approx_{P/t} M$, $t \neq i$, we can get $a = M$. Thus the second condition is not satisfied.

3. Since $|Opt(P)| = 1$, $|O'| > 1$. There does not exist $a$ and $a'$ such that $a \neq a'$ and $a \approx_{P/t} a'$. Thus $|O'| \not> 1$ and the third condition is not satisfied.

□

## 5.5 Conclusions and Future Work

We studied manipulation and simple bribery problems arising when one aggregates sets of ranked preferences. As a preference aggregation method we used the Pareto rule. We considered two extreme cases of that general setting. In one of them, all preferences are equally ranked, in the other one the preferences are strictly ranked. In the scenario we investigated, agents submit preferences on elements of the space of outcomes but, when considering manipulation and simple bribery, they need to assess the quality of *sets* of such elements. In the work, we considered several natural ways in which a total preorder on a space of outcomes can be lifted to a total preorder on the space of sets of outcomes. For each of these "liftings", we found conditions characterizing situations when manipulation (simple bribery) are possible. These characterizations show that in many cases it is impossible for any agent to strategically misrepresent preferences (*compare best* and *compare worst* for manipulation, in both equally ranked and strictly ranked settings; *compare best* for simple bribery in the equally ranked setting; and, somewhat surprisingly, *average-rank* for manipulation in the strictly ranked setting). In those cases, the Pareto principle is "strategy-proof".

However, in all other cases, it is no longer the case. Manipulation and simple bribery cannot be *a priori* excluded. To study whether computational complexity may provide a barrier against strategic misrepresentation of preferences, we considered a simple logical preference representation language closely related to possibilistic logic and answer-set optimization. For sets of preferences given in this language (in the settings of equally ranked or strictly ranked preferences) and for each way of lifting preorders from sets to power sets for which manipulation and simple bribery are possible, we proved that deciding the existence of manipulation or simple bribery is intractable.

Our work leaves several interesting open problems. First, methods to lift preorders from sets to power sets can be defined axiomatically in terms of properties for the lifted preorders to satisfy. Are there general results characterizing the existence of manipulation (simple

bribery) for lifted preorders specified only by axioms they satisfy? Second, we do not know the exact complexity of the problems $EB^{cw}$, $EM^{ar}$ and $EB^{ar}$ for the equally ranked preferences, nor for $EM^{lmin}$ and $EB^{lmin}$ for the strictly ranked preferences (the superscript indicates the set comparison method used). Finally, in the setting of equally ranked preferences, most aggregation rules of practical significance properly extend the Pareto one. We conjecture that at least for some of these rules, one can derive results on existence of manipulation and simple bribery from our results concerning the Pareto rule.

## Chapter 6 Importance Learning for Preferences

The general objective of preference learning is to predict unknown preferences from observed user's behavior. The preference learning problem attracts significant attention in the study of preferences. However, in many situations where we aggregate preferences, not only preferences but also their importance matters. The problem of learning the importance of preferences has not yet been considered. In this chapter, we study this problem. Given a preference profile over a set of outcomes, and a set of examples that represent a collective preorder on the outcomes, we want to determine whether there is an assignment of ranks (or weights) for which the resulting ranked (weighted) profile correctly orders all examples (or possibly many examples). In other words, we want to discover the importance of preferences in the profile that would explain observations. We consider the learning problem with ranked preference profiles and weighted voting profiles.

### 6.1 Problem Statements

We start by formulating precise terms the two versions of the problem we will study. First, we deal with the fully qualitative case, when preferences are ranked. Let us recall (cf. Chapter 3) that a ranked profile is a pair $(P,r)$, where $P$ is a set of preferences and $r$ is a ranking function assigning positive integers to preferences in $P$. The *number of ranks* in a ranked profile $(P,r)$ is denoted by $range(P,r) = max_{p\in P} r(p)$. We say a ranked profile $(P,r)$ is *gap-free* if for every $i = 1,\ldots,range(P,r)$, there exists $p \in P$ such that $r(p) = i$. For a gap-free ranked profile $(P,r)$, $range(P,r) \leq |P|$.

**Theorem 30.** *Let $P$ be a preference profile over a set of outcomes $O$. For every ranking function $r$, there is a ranking function $r'$ such that $(P,r)$ and $(P,r')$ define the same total order over $O$ and $(P,r')$ is gap-free.*

*Proof.* Let $(P,r)$ be a ranked profile, and $(P,r')$ be a ranked profile with minimal number of ranks for the ranked profiles which define the same total order over $O$ with $(P,r)$. If $(P,r')$ is not gap-free, there exists $i$, $i \leq |P|$, such that $r'(p) \neq i$ for every $p \in P$. Let $r''$ be a ranking function such that $r''(p) = r'(p)$ for each $p \in P$ such that $r'(p) < i$, and $r''(p) = r'(p) - 1$ for each $p \in P$ such that $r'(p) > i$. It is clear that $(P,r)$ and $(P,r'')$ define the same total order over $O$, and $range(P,r'') < range(P,r')$, a contradiction with $(P,r')$ is a ranked profile with minimal number of ranks that defines the same total order over $O$ as $(P,r)$. □

Given a set $O$ of outcomes, an *example* is an expression $\langle x,y,R\rangle$, where $x,y \in O$ and $R \in \{\succ, \approx, |\}$. Given a ranked profile $(P,r)$ and an example $\langle x,y,R\rangle$, we say the example is *decided* by $(P,r)$ if $xR_{P,r}y$. If $R \in \{\succ, |\}$, we also say the example is *decided* on rank $i$ by $(P,r)$ if $x \approx_p y$ for every $p \in P$, $r(p) < i$ and

1. If $R = \succ$, $x \succ_p y$ for some $p \in P$ such that $r(p) = i$, and $x \succeq_p y$ for every $p \in P$ such that $r(p) = i$;

2. If $R = |$, $x \succ_p y$ and $y \succ_q x$ for some $p,q \in P$ such that $r(p) = r(q) = i$.

Given a preference profile $P$, a set $E$ of examples (we assume $P$ and $E$ are over a set of outcomes $O$) and a ranking function $r$, we say the ranked profile $(P,r)$ is *consistent* with $E$ if $(P,r)$ decides all examples in $E$. Similarly, we say the profile $P$ is *consistent* with $E$ if there exists a ranking function $r$ such that $(P,r)$ is consistent with $E$.

**Theorem 31.** *Let $P$ be a preference profile and $E$ be a set of examples. If $P$ is consistent with $E$, there exists a ranking function $r$ such that $(P,r)$ is consistent with $E$ and $range(P,r) \leq min(|P|, |E|+1)$.*

*Proof.* Let $r$ be a ranking function for $P$ such that $(P,r)$ is consistent with $E$. According to Theorem 30, there exists a ranking function $r'$ such that $(P,r')$ is consistent with $E$ and $(P,r')$ is gap-free. Thus $range(P,r') \leq |P|$.

Let $r$ be a ranking function for $P$ such that $(P,r)$ is consistent with $E$, and $(P,r)$ has a minimum possible range. We want to prove that every rank in $(P,r)$ decides some example in $E$ except the lowest rank. Assume $i$ is a rank in $(P,r)$ which is not the lowest rank and does not decide any example in $E$. Then every example is decided above or below rank $i$. For every example $\langle x,y,R\rangle$ which is decided below rank $i$, $x \approx_p y$ for every $p \in P$ such that $r(p) = i$. Let $r'$ be a ranking function such that $r'(p) = r(p)$ for every $p \in P$ and $r(p) \neq i+1$ and $r'(p) = i$ for every $p \in P$ and $r(p) = i+1$. Then $(P,r')$ is consistent with $E$. If $range(P,r) = i+1$, then $range(P,r') = i$ and $range(P,r') < range(P,r)$, a contradiction. If $range(P,r) > i+1$, $(P,r')$ is not gap-free and so, not of minimum range. Since $range(P,r) = range(P,r')$, $(P,r)$ does not have a minimum range, a contradiction.

Then every rank in $(P,r)$ except the lowest rank decides some example in $E$. Therefore $range(P,r) \leq min(|P|,|E|+1)$. □

The two problems we will study in the ranked profile setting can be stated as follows.

**RANK-CONSISTENCY** Given a preference profile $P$ and a set $E$ of examples, decide whether $P$ is consistent with $E$.

A more practical version of the problem, its approximation version, asks for a ranking function such that the corresponding ranked profile decides at least $k$ examples.

**RANK-MATCH-MANY** Given a preference profile $P$, a set $E$ of examples, and a positive integer $k$, decide whether there is a ranking function $r$ such that $(P,r)$ decides at least $k$ examples from $E$.

The RANK-MATCH-MANY problem can be used to compute the best ranking function, that is, the ranking function such that the corresponding ranked profile decides the maximum number of examples.

Next, we move on to the quantitative importance representation, where the importance of preferences are represented by weights rather than ranks. A *weighted* profile $(P,w)$ is

a profile in which every preference $p \in P$ is assigned a non-negative number, the *weight* $w(p)$ of $p$, by the *weight assignment* $w$. The preferences with a larger weight are more important than the preferences with a smaller weight. In our study of this setting, we further restrict preferences to be strict total orders, that is, votes, and call a preference profile a voting profile. We use positional scoring rules studied in social choice as the mechanism to aggregate preferences (votes).

**Definition 1.** An effective positional scoring rule $F$ is determined by a function $f : \{1,2,\ldots\} \times \{1,2,\ldots\} \to \{0,1,\ldots\}$ such that for every $n \geq 1$ and $m > n$, $f(n,m) = 0$, for every $n \geq 1$ and $1 \leq m < m' \leq n$, $f(n,m) \geq f(n,m')$, and $f$ can be computed in polynomial time in its first argument.

Given a vote $p$ (over the set $O$, say, of $n$ outcomes which are called *candidates* in this setting), the score $s_p^F(c)$ of a candidate $c$ in $p$ is given by $f(n, r_p(c))$, where $r_p(c)$ is the rank of $c$ in $p$. The score of a candidate in a weighted profile $(P,w)$, $s_{P,w}^F(c)$ is given by

$$s_{P,w}^F(c) = \sum_{p \in P} s_p^F(c) \cdot w(p).$$

The scores in a profile determine a total preorder on candidates (the higher the score, the more preferred the candidate). We denote this preorder by $\succ_{P,w}^F$ and we write $r_{P,w}^F(c)$ for the rank of a candidate $c$ in that preorder. Whenever the rule $F$ is clear from the context, we drop it from the notation.

Given an effective positional scoring rule $F$ and a weighted profile $(P,w)$, the preorder $\succ_{P,w}^F$ and the corresponding rank function $r_{P,w}^F$ can be computed in polynomial time (in the size of the profile).

For a voting profile over a set $O$ of candidates (outcomes), an *example* is an expression $\langle x,y,R \rangle$, where $x,y \in O$ and $R \in \{\succ, \approx\}$. Since $\succ_{P,w}^F$ is a total preorder for any effective positional scoring rule $F$, we do not allow incomparability examples, as they will not be observed. Given a weighted profile $(P,w)$ and an effective positional scoring rule $F$, we

say a strict example $\langle x, y, \succ \rangle$ is *decided* by $(P, w)$ under $F$ if $s^F_{P,w}(x) > s^F_{P,w}(y)$, and an equivalence example $\langle x, y, \approx \rangle$ is *decided* by $(P, w)$ under $F$ if $s^F_{P,w}(x) = s^F_{P,w}(y)$.

Given a voting profile $P$, a set of strict and equivalence examples $E$ (where we assume $P$ and $E$ are over a set of candidates $O$), an effective positional scoring rule $F$ and a weight assignment $w$, we say the weighted profile $(P, w)$ is *consistent with E under F* if all examples in $E$ are decided by $(P, w)$ under $F$. Also, we say that $P$ is *consistent with E under F* if there exists a weight assignment $w$ such that $(P, w)$ is *consistent with E under F*.

Let $F$ be an effective positional scoring rule. The two problems discussed above in the ranked profile setting can be stated as follows for the setting of weighted profiles of votes.

**WEIGHT-CONSISTENCY-$F$** Given a voting profile $P$ and a set of strict and equivalence examples $E$, decide whether $P$ is consistent with $E$ under $F$.

**WEIGHT-MATCH-MANY-$F$** Given a voting profile $P$, a set of strict and equivalence examples $E$, and a positive integer $k$, decide whether there is a weight assignment $w$ such that $(P, w)$ decides at least $k$ examples from $E$ under $F$.

## 6.2 Learning Ranks for Preferences

### 6.2.1 The Consistency Problem

For the RANK-CONSISTENCY problem, we show that it can be solved by a polynomial time algorithm. If the profile $P$ is consistent with the examples $E$, the algorithm finds a ranking function so that all examples are decided by the ranked profile. Moreover, the number of ranks is minimized.

**Proposition 5.** *Let $(P, r)$ be a ranked profile and $\langle x, y, \approx \rangle$ be an example. Then $\langle x, y, \approx \rangle$ is decided by $(P, r)$ if and only if $x \approx_p y$ for every $p \in P$.*

*Proof.* Directly from the definition. □

Given a preference profile $P$ and a set $E$ of examples, an algorithm to decide the consistency of $P$ with $E$ could first check all equivalence examples. If there is an equivalence example $\langle x, y, \approx \rangle \in E$ and a preference $p \in P$ such that $x \not\approx_p y$, then (by Proposition 5), $P$ is inconsistent with $E$. Otherwise (also by Proposition 5), for any ranking function $r$, all equivalence examples are decided by $(P, r)$ and so the consistency of $P$ with $E$ depends only on the consistency of $P$ with the set of non-equivalence examples in $E$.

Thus, from now on we will assume that all examples in $E$ involve either $\succ$ or $|$ relations. To solve the consistency problem under this assumption about the example set, we first describe an algorithm to determine the preferences that cannot have rank 1 in any ranked profile consistent with the examples. Given a preference profile $P$ and a set $E$ of examples, we say a preference $p$ is *non-rank-1* if for any ranking function $r$ such that $r(p) = 1$, $(P, r)$ is not consistent with $E$.

Before introducing the algorithm, we establish a few auxiliary facts.

**Proposition 6.** *Let $(P, r)$ be a ranked profile and $\langle x, y, \succ \rangle$ be an example. If $y \succ_p x$ for some $p \in P$ and $r(p) = 1$, then $\langle x, y, \succ \rangle$ is not decided by $(P, r)$.*

*Proof.* Directly from the definition. □

**Proposition 7.** *Let $(P, r)$ be a ranked profile and $\langle x, y, | \rangle$ be an example. If $x \succeq_p y$, for every $p \in P$ such that $r(p) = 1$, and $x \succ_q y$, for some $q \in P$ such that $r(q) = 1$, then $\langle x, y, | \rangle$ is not decided by $(P, r)$. Similarly, if $y \succeq_p x$, for every $p \in P$ such that $r(p) = 1$, and $y \succ_q x$, for some $q \in P$ such that $r(q) = 1$, then $\langle x, y, | \rangle$ is not decided by $(P, r)$.*

*Proof.* If $x \succeq_p y$, for every $p \in P$ such that $r(p) = 1$, and $x \succ_q y$, for some $q \in P$ such that $r(q) = 1$, then the definition implies that $x \succ_{P,r} y$, and so, the example $\langle x, y, | \rangle$ is not decided by $(P, r)$. The other part of the assertion can be proved in a similar manner. □

Inspired by the two propositions above, non-rank-1 preferences could be determined in the following way. Given a preference profile $P$ and a set $E$ of examples, let $U$ denote the

set of all non-rank-1 preferences identified so far (initially empty). First, let us consider an example $\langle x, y, \succ \rangle$. For every ranked profile $(P, r)$ that is consistent with $E$, we have $x \succ_{P,r} y$. Thus, no preference $p$ such that $y \succ_p x$ holds can have rank 1 according to Proposition 6. Each such preference is added to $U$. After checking all strict examples, we move on to incomparability examples. For an incomparability example $\langle x, y, | \rangle$, if $x \succeq_p y$ for every $p \in P \backslash U$, any preference $q \in P \backslash U$ with $x \succ_q y$ cannot have rank 1 according to Proposition 7. Similarly, if $y \succeq_p x$ for every $p \in P \backslash U$, any preference $q \in P \backslash U$ with $y \succ_q x$ cannot have rank 1 also by Proposition 7. Therefore such preferences are included in $U$ and we move on to the next incomparability example. After checking all incomparability examples, we repeat this process checking all incomparability examples again. In every iteration, all incomparability examples are checked and the process terminates when there is no preference added to $U$ in some iteration. The loop is necessary because when $U$ is changed, an incomparability example that has not "pushed" a preference out of rank 1 before, may do so now.

We will now describe formally an algorithm $NotTopRanked(P, E)$ where $P$ is a preference profile and $E$ is a set of examples. The algorithm returns the set of all non-rank-1 preferences.

1. $U = \varnothing$.
2. For each $\langle x, y, \succ \rangle \in E$,
    a) if $y \succ_p x$, for some $p \in P$, $U = U \cup \{p\}$.
3. Loop until no change in $U$
    a) For each $\langle x, y, | \rangle \in E$,
        i. if $x \succeq_p y$ for every $p \in P \backslash U$, $U = U \cup \{q : q \in P, x \succ_q y\}$.
        ii. if $y \succeq_p x$ for every $p \in P \backslash U$, $U = U \cup \{q : q \in P, y \succ_q x\}$.

4. Return $U$.

Since $U$ is expanded by at least one preference in each iteration, there are at most $m$ iterations where $m$ is the number of preferences. Thus, the algorithm terminates and runs in polynomial time.

**Theorem 32.** *Let $P$ be a preference profile, $E$ be a set of strict and incomparability examples, and $U = NotTopRanked(P,E)$. For every ranking function $r$ such that $(P,r)$ is consistent with $E$, $r(p) > 1$ for every $p \in U$.*

*Proof.* By Proposition 6, after Step 2, for every $q \in U$, $r(q) > 1$. We will now prove that after ever iteration of loop 3(a), for every $q \in U$, $r(q) > 1$. Our comment above shows that the condition holds before any iterations of the loop 3(a) take place. Let us assume a particular iteration of the loop 3(a), and assume that just before it, the condition holds. That is, for every $q \in U$, $r(q) > 1$. Consider a preference $q$ moved to $U$ in this iteration. There must be an example $\langle x, y, | \rangle$ such that $x \succeq_p y$, for every $p \in P \backslash U$, and $x \succ_q y$ (or $y \succeq_p x$, for every $p \in P \backslash U$, and $y \succ_q x$). Since $(P,r)$ is consistent with $E$, $(P,r)$ decides $\langle x, y, | \rangle$. Thus, by Proposition 7, $r(q) > 1$. □

Next, we prove if $NotTopRanked(P,E) = P$, $P$ is inconsistent with $E$.

**Theorem 33.** *Let $P$ be a preference profile, $E$ be a set of strict and incomparability examples, and $U = NotTopRanked(P,E)$. If $U = P$, $P$ is inconsistent with $E$.*

*Proof.* Assume $P$ is consistent with $E$. Then there is a gap-free ranking function $r$ such that $(P,r)$ is consistent with $E$ (by Theorem 30). According to Theorem 32, for each $p \in P$ such that $r(p) = 1$, $p \notin U$, a contradiction with $U = P$. □

Next, we show that for every example $e = \langle x, y, R \rangle$ in $E$, $x$ and $y$ are indifferent on $P \backslash U$ or $xR_{P \backslash U} y$.

**Lemma 3.** *Let $P$ be a preference profile, $E$ be a set of strict and incomparability examples, and $U = NotTopRanked(P,E)$. Then for every example $e = \langle x,y,R\rangle$ in $E$, either $x \approx_{P\backslash U} y$ or $xR_{P\backslash U}y$.*

*Proof.* Let $e = \langle x,y,\succ\rangle$ and let us assume that $x \not\succ_{P\backslash U} y$. From the definition of $\succ_{P\backslash U}$, it follows that for every $p \in P\backslash U$, $x \approx_p y$, or that there is $p \in P\backslash U$ such that $y \succ_p x$. In the former case, the assertion holds. In the latter, by the way the algorithm works we have $p \in U$, a contradiction.

Next, let $e = \langle x,y,|\rangle$, and let us assume that $x \not\approx_p y$ for some $p \in P\backslash U$, and that $x$ and $y$ are comparable on $P\backslash U$. If $x \succ_p y$, there is no $q \in P\backslash U$ such that $y \succ_q x$ since $x$ and $y$ are comparable on $P\backslash U$. Based on the algorithm, $p \in U$, a contradiction. Thus $y \succ_p x$, and we get a contradiction in the same way. Thus, x and y are incomparable on $P\backslash U$. □

If $U \neq P$, we assign rank 1 to the preferences in $P\backslash U$. If $xR_{P\backslash U}y$ holds for every example $\langle x,y,R\rangle$ in $E$, $(P,r)$ is consistent with $E$, where $r$ is a ranking function such that $r(p) = 1$ for every $p \in P\backslash U$ and $r(p) = 2$ for every $p \in U$. Otherwise, we need to decide whether the preferences in $U$ can be ranked such that the remaining examples are decided. If $U = \varnothing$, the remaining examples cannot be decided and the problem is inconsistent. Otherwise, we define $E'$ to be the set of examples $\langle x,y,R\rangle$ in $E$ such that $x \approx_{P\backslash U} y$, and proceed recursively.

The formal description of the algorithm follows. We call it $Rank(P,E,r)$. It returns false if $P$ is not consistent with $E$, and returns true and a ranking function $r$ such that $(P,r)$ is consistent with $E$, otherwise.

1. If $E = \varnothing$, set $r(p) = 1$ for each $p \in P$ and return true.

2. If $P = \varnothing$, return false.

3. Set $U = NotTopRanked(P,E)$, and $E' = \{\langle x,y,R\rangle : \langle x,y,R\rangle \in E, x \approx_{P\backslash U} y\}$.

4. If $U = P$, return false.

5. If $Rank(U, E', r')$ returns false, return false; otherwise, set $r(p) = 1$ for each $p \in P \backslash U$ and $r(p) = r'(p) + 1$ for each $p \in U$, and return true.

To show the correctness of the algorithm $Rank(P, E, r)$, we first prove the following two results.

**Lemma 4.** *Let $P$ be a preference profile, $E$ be a set of strict and incomparability examples, $U = NotTopRanked(P,E)$ and $E' = \{e = \langle x, y, R\rangle : e \in E, x \approx_{P\backslash U} y\}$. If $r'$ is a ranking function such that $(U, r')$ is consistent with $E'$, then $(P, r)$ is consistent with $E$ where $r$ is a ranking function such that $r(p) = 1$ for every $p \in P\backslash U$ and $r(p) = r'(p) + 1$ for every $p \in U$.*

*Proof.* Let $r$ be a ranking function for $U$ such that $(U, r)$ is consistent with $E'$, and $r'$ be a ranking function for $P$ such that $r'(p) = 1$ for every $p \in P\backslash U$ and $r'(p) = r(p) + 1$ for every $p \in U$. For each example $e = \langle x, y, R\rangle$ in $E'$, since $x \approx_{P\backslash U} y$ and $e$ is decided by $(U, r)$, $e$ is also decided by $(P, r')$. For every example $e = \langle x, y, R\rangle$ in $E\backslash E'$, based on Lemma 3, $xR_{P\backslash U}y$. Then every example $e \in E\backslash E'$ is decided by $(P, r')$ on rank 1. Thus $(P, r')$ is consistent with $E$. □

**Theorem 34.** *Let $P$ be a preference profile, $E$ be a set of strict and incomparability examples, $U = NotTopRanked(P,E)$ and $E' = \{e = \langle x, y, R\rangle : e \in E, x \approx_{P\backslash U} y\}$. Then $P$ is consistent with $E$ if and only if $U$ is consistent with $E'$.*

*Proof.* ($\Rightarrow$) Let $r$ be a ranking function for $P$ such that $(P, r)$ is consistent with $E$, and $r'$ be a ranking function for $U$ such that $r'(p) = r(p)$ for every $p \in U$. Assume that an example $e = \langle x, y, R\rangle$ in $E'$ is not decided by $(U, r')$. Since $x \approx_{P\backslash U} y$, $e$ is not decided by $(P, r)$, a contradiction. Thus $(U, r')$ is consistent with $E'$.

($\Leftarrow$) Let $r'$ be a ranking function for $U$ such that $(U, r')$ is consistent with $E'$, and $r$ be a ranking function for $P$ such that $r(p) = 1$ for every $p \in P\backslash U$ and $r(p) = r'(p) + 1$ for every $p \in U$. According to Lemma 4, $(P, r)$ is consistent with $E$. □

Now we provide a formal prove of the correctness of the algorithm *Rank*.

**Theorem 35.** *Let $P$ be a preference profile and $E$ be a set of strict and incomparability examples. If $P$ is consistent with $E$, then $Rank(P,E,r)$ returns true and $(P,r)$ is consistent with $E$. If $P$ is inconsistent with $E$, then $Rank(P,E,r)$ returns false.*

*Proof.* Let $n$ be the depth of recursion for $Rank(P,E,r)$. We prove by induction that the theorem holds for all possible $n$'s.

First, let $n = 0$. That means that $Rank(P,E,r)$ terminates without making any recursive calls. Thus, one of the following three conditions holds:

1. $E = \varnothing$. In this case, the algorithm returns true. This is clearly correct. Moreover, $(P,r)$ is consistent with $E$ for any ranking function $r$, in particular, on the function returned by the algorithm.

2. $P = \varnothing$ and $E \neq \varnothing$. The algorithm returns false. This is correct. Since $E$ contains strict and incomparability examples, it is clear that $E$ cannot be decided by $(P,r)$ for any ranking function $r$, and $P$ is inconsistent with $E$.

3. $NotTopRanked(P,E) = P$. The algorithm returns false. According to Theorem 33, $P$ is inconsistent with $E$. Thus, what the algorithm returns is correct.

Next we prove that if the theorem holds whenever the depth of the recursion is $k$, then it holds when the depth of the recursion is $k+1$. Let us consider an execution of $Rank(P,E,r)$, where the depth of recursion is $k+1$. Let $U = NotTopRanked(P,E)$ and $E' = \{e = \langle x,y,R\rangle : e \in E, x \approx_{P\setminus U} y\}$. Since the algorithm does not terminate earlier (the depth of the recursion is $k+1 > 0$), we are at step 5. There are two possibilities.

1. $Rank(U,E',r')$ returns false. The depth of recursion for $Rank(U,E',r')$ is $k$. Based on the induction hypothesis, $U$ is inconsistent with $E'$. According to Theorem 34, $P$ is inconsistent with $E$, and that is precisely what $Rank(P,E,r)$ returns.

2. $Rank(U, E', r')$ returns true. Again by induction, it follows that $U$ is consistent with $E'$ and that for $r'$ computed by $Rank(U, E', r')$, $(U, r')$ is consistent with $E'$. By Theorem 34, $P$ is consistent with $E$ and, accordingly, $Rank(P, E, r)$ returns true. Moreover, by Lemma 4, the function $r$ computed and returned by $Rank(P, E, r)$ has the property that $(P, r)$ is consistent with $E$.

□

If a preference profile $P$ is consistent with a set $E$ of examples, the ranking function computed by the algorithm *Rank* minimizes the number of ranks.

**Theorem 36.** *Let $P$ be a preference profile and $E$ be a set of strict and incomparability examples. If $Rank(P, E, r)$ returns true, $range(P, r) \leq range(P, r')$ for any ranking function $r'$ such that $(P, r')$ is consistent with $E$.*

*Proof.* Let $n$ be the depth of recursion for $Rank(P, E, r)$. We prove by induction that the theorem holds for all possible $n$'s.

We first prove the theorem holds when $n = 0$. Since $Rank(P, E, r)$ returns true, $E = \varnothing$ and $r(p) = 1$ for every $p \in P$. Since $range(P, r) = 1$, it is clear $range(P, r) \leq range(P, r')$ for any ranking function $r'$.

Next we prove if the theorem holds when $n = k$, then it holds when $n = k + 1$. Let $U = NotTopRanked(P, E)$, $E' = \{e = \langle x, y, R \rangle : e \in E, x \approx_{P \setminus U} y\}$. Let $r$ be the ranking function returned by $Rank(P, E, r)$ and $r'$ be the ranking function returned by $Rank(U, E', r')$. According to the algorithm, the depth of recursion for $Rank(U, E', r')$ is $k$ and $range(P, r) = range(U, r') + 1$. Let $s$ be a ranking function for $P$ such that $(P, s)$ is consistent with $E$, and $s'$ be the ranking function on $U$ induced by $s$. Then $range(P, s) \geq range(U, s') + 1$ (by Theorem 32). Since $x \approx_{P \setminus U} y$ for every $\langle x, y, R \rangle \in E'$, $(U, s')$ is consistent with $E'$. Then based on the induction hypothesis, $range(U, r') \leq range(U, s')$. Thus $range(P, s) \geq range(P, r)$. □

Based on Theorem 36 and 31, the ranking function $r$ computed by the algorithm *Rank* has at most $min(|P|, |E|+1)$ ranks and each rank except the lowest one decides at least one example.

### 6.2.2 The Optimization Problem

If a preference profile is inconsistent with a set of examples, that is, if there does not exist a ranking function such that the corresponding ranked profile decides all examples, an important problem is to find a ranking function so that the corresponding ranked profile decides as many examples as possible. This problem is an optimization version of the RANK-MATCH-MANY problem which we will discuss now.

**Theorem 37.** *The* RANK-MATCH-MANY *problem is NP-complete.*

*Proof.* The problem is in NP as for every preference $p \in P$, we can guess its rank $r(p)$ (and all these ranks can be chosen from the set $\{1, \ldots, |P|\}$ according to Theorem 30), and then verify in polynomial time whether the ranked profile $(P, r)$ decides at least $k$ examples from $E$ (exploiting the fact that dominance testing in ranked profiles is a polynomial-time task).

For the hardness part, we provide a reduction from the *vertex cover* problem in graphs. Let $G = (V, E)$ be an undirected graph. A set $V' \subseteq V$ is a *vertex cover* of $G$ if every edge in $E$ is incident to at least one vertex in $V'$. Given a graph $G$ and an integer $k \leq |V|$, the *vertex cover* problem is to decide whether $G$ has a vertex cover of size at most $k$. The vertex cover problem is known to be NP-complete [79].

Let $G = (V, E)$ be an undirected graph with $n$ vertices and $m$ edges (that is, $n = |V|$ and $m = |E|$). We denote by $E_v$ the set of edges in $E$ incident to $v$ in $G$. For every $e \in E$, we introduce $n+1$ copies of $e$, say $e_1, \ldots, e_{n+1}$. We denote that set by $C_e$. Finally, we also introduce one more object, say $c$, different from all elements in $V \cup \bigcup_{e \in E} C_e$.

We now define a set $O$ of outcomes, a profile $P$ over $O$, and a set $E$ of examples over $O$

by setting

$$O = V \cup \bigcup_{e \in E} C_e \cup \{c\}$$

$$P = \{ \bigcup_{e \in E_v} C_e \succ v \succ \mathit{REM} \colon v \in V \} \cup \{ c \succ \mathit{REM} \}.$$

and

$$E = \{ \langle c, v, \succ \rangle \colon v \in V \} \cup \{ \langle x, c, \succ \rangle \colon x \in C_e, e \in E \}.$$

We adopt the convention to write *REM* as the last option in a preference. By default, *REM* stands for the set of all outcomes not included in the more preferred options. We will denote each preference $\bigcup_{e \in E_v} C_e \succ v \succ \mathit{REM}$, $v \in V$, by $p_v$, and the preference $c \succ \mathit{REM}$ by $p_c$.

Clearly, $E$ consists of $n + m(n+1)$ examples. Let $k \leq n$. We claim that $G$ contains a vertex cover $V'$ such that $|V'| \leq k$ if and only if there exists a ranking function $r$ to preferences in $P$ such that the ranked profile $(P, r)$ decides at least $m(n+1) + n - k$ examples.

($\Rightarrow$) Let $V'$ be a vertex cover for $G$ and $|V'| \leq k$. We define the ranking function $r$ on $P$ by setting

$$r(p_v) = \begin{cases} 1 & if\ v \in V' \\ 3 & if\ v \in V \backslash V' \end{cases}$$

and $r(p_c) = 2$.

Let us consider an example $\langle x, c, \succ \rangle$, for some $x \in C_e$ and $e \in E$. Since $V'$ is a vertex cover of $G$, there is a vertex $v \in V'$ such that $e \in E_v$. Thus, $d_{p_v}(x) = 1$ and $r(p_v) = 1$. Since for every $p \in P$ such that $r(p) = 1$, $d_p(c) = 3$ and $d_p(x) \leq 3$, the example $\langle x, c, \succ \rangle$ is decided by $(P, r)$.

Next, let us consider an example $\langle c, v, \succ \rangle$, where $v \in V$. If $v \in V'$, the ranked profile $(P, r)$ does not decide the example. Indeed, $r(p_v) = 1$, $d_{p_v}(v) = 2$ and $d_{p_v}(c) = 3$. Thus, $c \not\succ_{(P,r)} v$. On the other hand, if $v \notin V'$, then $r(p_v) = 3$. Moreover, $r(p_c) = 2$, $d_{p_c}(c) = 1$, $d_{p_c}(v) = 2$, and for every other preference $p$ (it is of the form $p_w$ for some $w \in V$, $w \neq v$), $d_p(v) = d_p(c)$. Thus, the ranked profile $(P, r)$ decides $\langle c, v, \succ \rangle$.

It follows that the ranked profile $(P,r)$ decides $n+m(n+1)-|V'|$ examples. Since $|V'| \leq k$, $(P,r)$ decides at least $n+m(n+1)-k$ examples, as claimed.

($\Leftarrow$) For this direction, let us assume that $r$ is a ranking function such that $(P,r)$ decides at least $m(n+1)+n-k$ examples. If there is $e \in E$ and $x \in C_e$ such that $(P,r)$ does not decide $\langle x,c,\succ\rangle$, then $(P,r)$ does not decide any of the examples $\langle y,c,\succ\rangle$, where $y \in C_e$. Thus the number of examples decided by $(P,r)$ is at most $n+m(n+1)-(n+1) < n+m(n+1)-k$, a contradiction.

It follows that $(P,r)$ decides all examples $\langle x,c,\succ\rangle$, where $e \in E$, $x \in C_e$. Moreover, $(P,r)$ decides at least $n-k$ examples $\langle c,v,\succ\rangle$, where $v \in V$. Let $V'$ consists of all elements $v \in V$ such that $(P,r)$ does not decide $\langle c,v,\succ\rangle$. By our observation, $|V'| \leq k$. We will show that $V'$ is a vertex cover for $G$.

Let us consider an edge $e \in E$ and $x \in C_e$, and let $v$ and $w$ be the two endpoints of $e$. Since $(P,r)$ decides $\langle x,c \succ\rangle$, and $c \succ_{p_c} x$, it follows that there is a preference $p_u$, for some $u \in V$, such that $x \succ_{p_u} c$ and $r(p_u) < r(p_c)$. It follows that $u = v$ or $u = w$. Without loss of generality, we may assume that $u = v$. Thus, $r(p_v) < r(p_c)$ and $x \succ_{p_v} v \succ_{p_v} c$. Since for all preferences $p$ other that $p_v$ and $p_c$, $v \approx_p c$, $v \succ_{(P,r)} c$, that is, $(P,r)$ does not decide the example $\langle c,v,\succ\rangle$. Thus, $v \in V'$. It follows that $V'$ is a vertex cover for $G$. □

This result shows that, given an integer $k$, it is hard to decide whether a preference profile can be ranked to decide at least $k$ examples. If $k$ is fixed, the problem can be decided in polynomial time by solving the RANK-CONSISTENCY problem for every set of examples with size $k$.

### 6.3 Learning Weights for Voting

Next we consider the learning problems for voting profiles with an effective positional scoring rule as the aggregation method. In a weighted voting scenario, each vote is a strict total order, and a candidate receives scores from every vote based on its position in the vote

and on the weight of the vote. The weights are represented by numerical values and we consider both real and integer numbers in our work.

### 6.3.1 The Consistency Problem

**Theorem 38.** *The* WEIGHT-CONSISTENCY-$F$ *problem is in P (both when weights are allowed to be reals and when they are restricted to be integers).*

*Proof.* This problem can be modeled as a linear programming problem. We construct a linear program over variables $w_p$, $p \in P$, and $\varepsilon$. Variables $w_p$, $p \in P$, are meant to represent the weights we seek.

For every example $\langle a, b, \succ \rangle$, we include in the program the linear inequality

$$\Sigma_{p \in P} w_p s_p(a) \geq \Sigma_{p \in P} w_p s_p(b) + \varepsilon,$$

and for every example $\langle a, b \approx \rangle$, similarly, we include the inequalities

$$\Sigma_{p \in P} w_p s_p(a) \geq \Sigma_{p \in P} w_p s_p(b)$$

$$\Sigma_{p \in P} w_p s_p(b) \geq \Sigma_{p \in P} w_p s_p(a).$$

We also include in the program the inequalities $w_p \geq 0$, for every $p \in P$, $0 \leq \varepsilon$ and $\varepsilon \leq 1$. We take $\varepsilon$ as the objective function and we want to maximize it.

If $P$ is consistent with $E$ under $F$, there is a solution for this linear program with $\varepsilon$ satisfying $0 < \varepsilon \leq 1$. Thus, an optimal solution also has a positive $\varepsilon$. Conversely, if an optimal solution for the linear programming problem has a positive $\varepsilon$, the profile $P$ is consistent with examples in $E$ under $F$. It is also clear that if the linear programming problem has no solutions or if the optimal solution has $\varepsilon = 0$, $P$ is inconsistent with $E$ under $F$. Thus, the consistency of $P$ with $E$ can be decided by solving the linear programming problem we constructed.

The linear programming problem can be solved in polynomial time. Therefore, the consistency problem can be decided in polynomial time. Moreover, if the answer is yes, there

is a rational solution to the linear programming problem (as the coefficients are integers) and so, a rational weight assignment showing the consistency of $P$. Thus, also an integer weight assignment showing the consistency of $P$ exists (it can be obtained by scaling up the rational weight assignment). □

This result has a corollary limiting the size of the weights, in the case when a profile is consistent with a set of examples.

**Corollary 3.** *Given a set $P$ of votes, a set $E$ of strict and equivalence examples, and an effective positional scoring rule $F$, if $P$ is consistent with $E$ under $F$, then there exists an integer weight assignment $w$ such that $(P,w)$ decides $E$ under $F$ and the size of the representation of all weights $w(p)$ is polynomial in the size of the input.*

*Proof.* From Theorem 38, we know that we can decide whether $P$ is consistent with $E$, and compute an integer weight assignment $w$ in polynomial time in the size of the input. Thus, the size of the representation of all weights $w(p)$ is polynomial in the size of the input. □

### 6.3.2 The Optimization Problem

In this section, we consider the optimization problem for voting profiles, which decides whether a voting profile can be weighted to decide at least $k$ examples.

We first discuss a simple instance of the optimization problem where the positional scoring rule is veto, the number of votes is equal to the number of candidates and every candidate is ranked at the last position by exactly one vote. Under veto, every candidate gets one point from a vote except the one placed at the very end of that vote. The *veto* rule can be formally defined by the function $f$ such that $f(n,m) = 1$ for every $n \geq 1$ and $1 \leq m < n$, and $f(n,m) = 0$ for every $n \geq 1$ and $m \geq n$. Given a set $O$ of candidates, We use $O \setminus \{c\} \succ c$ to denote the vote over $O$ that positions the candidate $c$ at the end and the remaining candidates in an arbitrary order.

To prove the complexity of the optimization problem for this special voting profile, we need a lemma to bound the size of weights for the profile.

**Lemma 5.** *Let $O$ be a set of candidates and $P = \{O \setminus \{c\} \succ c : c \in O\}$ a profile over $O$. If $E$ is a set of strict and equivalence examples over $O$ such that some total preorder on $O$ is consistent with $E$, then there is a weight function $w$ such that $(P,w)$ is consistent with $E$ under the veto rule, and for every $p \in P$, $w(p) < |O|$.*

*Proof.* Let $\succeq$ be a total preorder on $O$. We will write $d(x)$ for the satisfaction degree of a candidate $x$ in $\succeq$. We note that for every $x \in O$, $d(x) \leq |O|$. Moreover, directly from the definition we have that for every $x, y \in C$,

$$x \succ y \quad \text{if and only if} \quad d(x) < d(y),$$

and

$$x \approx y \quad \text{if and only if} \quad d(x) = d(y).$$

For every $x \in O$ we denote the preference $O \setminus \{x\} \succ x$ by $p_x$. We now define $w(p_x) = d(x)$. It follows that for every $x \in O$,

$$s^{veto}_{P,w}(x) = W - d(x), \tag{6.1}$$

where $W = \sum_{c \in C} d(c)$. It follows that $x \succ^{veto}_{P,w} y$ if and only if $d(x) < d(y)$ and $x \approx^{veto}_{P,w} y$ if and only if $d(x) = d(y)$. Thus, $x \succ^{veto}_{P,w} y$ if and only if $x \succ y$ and $x \approx^{veto}_{P,w} y$ if and only if $x \approx y$.

In particular, if the total preorder $\succeq$ is consistent with $E$, the weighted profile $(P,w)$ is consistent with $E$ under the veto rule. Since for every $p \in P$, $w(p) \leq |O|$, the result follows. □

We will use this result to prove that the optimization problem for this special voting profile under veto is NP-complete.

**Theorem 39.** *The following problem is NP-complete: Given a set of candidates $O$, a set of votes $P = \{O \setminus \{c\} \succ c : c \in O\}$, a set of strict and equivalence examples $E$ over $O$, and an integer $k$, $1 \leq k \leq |E|$, decide whether there is a subset $E'$ of $E$ such that $|E'| \geq k$ and $P$ is consistent with $E'$ under the veto rule.*

*Proof.* The problem is in NP because we can guess a subset $E'$ of $E$ such that $E' \geq k$ and a weight assignment $w$ with each weight smaller than $|O|$ and verify in polynomial time whether the weighted profile $(P,w)$ decides at least $k$ examples from $E$ (cf. Lemma 5).

To prove the NP-hardness, we use a reduction from the *feedback arc set* (FAS) problem, which is known to be NP-complete [79]. In the FAS problem we are given a directed graph $G = (V,A)$ and an integer $k$. The objective is to decide whether there is a set of edges $A' \subseteq A$ such that $|A'| \leq k$ and $G' = (V, A \setminus A')$ is acyclic. We call sets $A'$ with the latter property *feedback arc sets*.

Let $G = (V,A)$ be an arbitrary directed graph and $k$ an integer. We define $O = V$, $P = \{p_v : v \in V\}$, where $p_v = V \setminus \{v\} \succ v$, and $E = \{\langle b, a \succ \rangle : (a,b) \in A\}$. We note that for every assignment $w$ of weights to preferences in $P$, $s^{veto}_{P,w}(v) = \sum_{v' \in V, v' \neq v} w(p_{v'})$. We claim that $G$ contains a feedback arc set $A'$ such that $|A'| \leq k$ if and only if there exists a weight assignment $w$ to $P$ such that $(P,w)$ decides (under the veto rule) at least $|A| - k$ examples from $E$.

($\Rightarrow$) Let $A'$ be a feedback arc set for $G$ such that $|A'| \leq k$. Let $\langle v_1, \ldots, v_n \rangle$ be a topological ordering of $G' = (V, A \setminus A')$. Clearly, the strict order $v_n \succ \ldots \succ v_1$ is consistent with all examples in $E'\{\langle b, a \succ \rangle : (a,b) \in A \setminus A'\}$. By Lemma 5, there is a weight assignment $w$ such that $(P,w)$ decides all examples in $E'$. Since $|E'| \geq |A| - k$, the implication follows.

($\Leftarrow$) Let us assume that $w$ is a weight assignment such that the profile $(P,w)$ decides at least $|A| - k$ examples from $E$. Let $E'$ be the set of examples that are not decided. Clearly, $|E'| \leq k$. We define $A' = \{(a,b) : \langle b, a \succ \rangle \in E'\}$. We will show that $G' = (V, A \setminus A')$ is acyclic. Let $\varepsilon = \langle v_1, \ldots, v_n \rangle$ be any enumeration of elements in $V$ such that $s_{P,w}(v_i) < s_{P,w}(v_j)$ implies $i < j$. Let $(a,b) \in A \setminus A'$. It follows that the example $\langle b, a, \succ \rangle$ is decided

and so, $s_{P,w}(a) < s_{P,w}(b)$. Let $i$ and $j$ be the positions of $a$ and $b$ in the enumeration $\varepsilon$. It follows that $i < j$. Thus $\varepsilon$ is a topological ordering of $G'$ and so, $G'$ is acyclic. □

This theorem is mainly of technical interest. We will use it to prove the complexity of the optimization problem under a general effective positional scoring rule.

An effective positional scoring rule $F$ is called trivial if $f(n,m) = f(n,m')$ for every $1 \leq m, m' \leq n$. The optimization problem is in P under a trivial effective positional scoring rule since every candidate gets the same score no matter how to assign the weights to votes. For any non-trivial effective positional scoring rule, the optimization problem is NP complete.

**Theorem 40.** *The* WEIGHT-MATCH-MANY-$F$ *problem is NP-complete for every non-trivial effective positional scoring rule* $F$.

*Proof.* The problem is in NP because we can guess a weight assignment $w$ of size polynomial in the size of input (cf. Corollary 3), compute the scores of every candidate, and use these scores to verify whether there are at least $k$ examples are decided. As each of the latter two tasks can be accomplished in polynomial time, the membership in NP follows.

For the hardness part, we use a reduction from the problem in Theorem 39. In that problem we are given a set $O$ of $n$ candidates, a set $P = \{p_c : c \in O\}$ of votes, where $p_c = O \setminus \{c\} \succ c$, a set $E$ of strict and equivalence examples, and an integer $k$; the objective is to decide whether there is a weight assignment $w$ such that at least $k$ examples from $E$ are decided by $(P,w)$ under the veto rule.

We construct the sets $O'$ and $E'$ of candidates and examples by setting $O' = O$ and $E' = E$. Next, for each $c \in O$, we fix an enumeration $c_1, \ldots, c_{n-1}$ of elements in $O \setminus \{c\}$ and define votes $p_c^i$, $i = 1, \ldots, n-1$ by setting

$$p_c^i = c_i \succ \ldots \succ c_{n-1} \succ c_1 \succ \ldots \succ c_{i-1} \succ c.$$

Finally, we define $P_c = \{p_c^i : i = 1, \ldots, n-1\}$ and $P' = \bigcup_{c \in O} P_c$. We will show that there is a weight assignment $w$ on $P$ such that at least $k$ examples in $E$ are decided by $(P,w)$

under the veto rule if and only if there is a weight assignment $w'$ on $P'$ such that at least $k$ examples in $E'$ $(= E)$ are decided by $(P', w')$ under the rule $F$. $(\Rightarrow)$ Let $w$ be a weight assignment for $P$ such that at least $k$ examples in $E$ are decided by $(P, w)$ under the veto rule. Clearly, for every $x \in O$, $s_{P,w}(x) = W - w(p_x)$, where $W = \sum_{c \in O} w(p_c)$.

We now define a weight assignment $w'$ on $P'$. Namely, for every $p \in P_c$, we set $w'(p) = w(p_c)$. Clearly,

$$s^F_{P',w'}(x) = \sum_{c \in O} s^F_{P_c,w'}(x).$$

Since $x$ appears in the last position (position $n$) in every vote in $P_x$, and since each of these votes has the same weight $w(p_x)$,

$$s^F_{P_x,w'}(x) = \sum_{p \in P_x} \alpha_n w(p_x) = (n-1)\alpha_n w(p_x)$$

where we denote $f(n,m)$ by $\alpha_m$. For every $c \in O \setminus \{x\}$, $x$ appears exactly once in each position $i$, $1 \leq i \leq n-1$, in votes from $P_c$. Since each of these votes has the same weight $w(p_c)$,

$$s^F_{P_c,w'}(x) = (\alpha_1 + \cdots + \alpha_{n-1}) w(p_c)$$

Thus,

$$\begin{aligned} s^F_{P'w'}(x) &= \sum_{c \in O} s^F_{P_c,w'}(x) \\ &= (n-1)\alpha_n w(p_x) + A \sum_{c \neq x} w(p_c) \\ &= (n-1)\alpha_n w(p_x) + A(W - w(p_x)) \\ &= AW - (A - (n-1)\alpha_n) w(p_x), \end{aligned}$$

where we denote $\alpha_1 + \cdots + \alpha_{n-1}$ by $A$ (and we recall that we defined $W = \sum_{c \in O} w(p_c)$).

We now observe that since $F$ is non-trivial, $A - (n-1)\alpha_n > 0$. Consequently, for every $x, y \in C$,

$$s_{P,w}(x) > s_{P,w}(y) \quad \text{if and only if} \quad s^F_{P_c,w'}(x) > s^F_{P_c,w'}(y).$$

Since $(P,w)$ decides at least $k$ examples in $E$ (under the veto rule), $(P',w')$ decides at least $k$ examples in $E'$ $(= E)$ under the rule $F$.

($\Leftarrow$) Let $w'$ be a weight assignment on $P'$ such that at least $k$ examples in $E'$ $(= E)$ are decided by $P'$ under $F$. Let $E'' \subseteq E$ be the set of examples that are decided by $(P',w')$ under $F$. It follows that the preorder $\succ^F_{P',w'}$ is consistent with $E''$. By Lemma 5, there is a weight assignment $w$ on $P$ such that $(P,w)$ is consistent with $E''$. In other words, $(P,w)$ decides at least $k$ examples in $E$. □

### 6.4 Conclusions

In this work, we studied the problem of learning the importance of preferences when the preferences and a set of pairwise ordered options are given. Two settings are considered, preference aggregation with Pareto principle and voting with a positional scoring rule. We proved that for both of these settings, it can be decided in polynomial time that whether a rank/weight assignment exists such that all given examples are decided by the corresponding ranked/weighted profile. For the ranked profile, we also provided a polynomial algorithm to solve the consistency problem. It computes a rank assignment with minimal number of ranks if the profile is consistent with given examples. If the profile is not consistent with all examples, we considered an optimization problem which decides whether a rank/weight assignment exists such that the corresponding ranked/weighted profile can decide at least $k$ examples. For both of ranked and weighted profiles, we proved that the optimization problem is NP-complete. Therefore, a preference profile cannot be ranked/weighted easily to decide as many examples as possible.

In the future work, we will implement and evaluate our algorithm solving the RANK-CONSISTENCY problem for ranked profiles. We will also study heuristic algorithms for the optimization problem. In the current work, for voting profiles, we only considered the positional scoring voting rules and the results can be extended other voting rules. We can also study conditional learning problems, such as learning the importance of preferences

given that preference $a$ is more important than preference $b$. Moreover, the active learning problem is also interesting which studies how to recover the importance of preferences with minimal number of queries.

## Chapter 7 Conclusions and Future Work

In this dissertation, we studied the following problems for preferences in the ASO formalism: computing conditional optimal solutions, strategic behaviors for a multi-agent profile, and importance learning with preferences and pairwise ordered options provided.

In Chapter 4, we studied four kinds of computational problems related to reasoning with preferences in the ASO formalism: to find an optimal outcome, to find an optimal outcome different from a given interpretation, to find an optimal outcome which is similar to or dissimilar from a given interpretation, and to find a diverse set of optimal outcomes. We provided the computation complexity for deciding these optimization problems and discuss two methods to compute these optimal solutions. One method separates each optimization problem to a series of tasks which can be modeled as ASP programs, and solves these basic tasks using ASP solvers. The other method models each optimization problem as a single disjunctive logic program and uses disjunctive ASP solvers to solve it. We also designed three different data sets to experiment with these two methods on all problems when the preferences are ranked or unranked.

According to the experimental results, when all preferences are equally important, the method compiling the entire reasoning task into a disjunctive logic program works better. However, when the preferences have different importance, especially when they are ranked into many levels, the performance of the declarative method (modeling the problem as a single disjunctive logic program) drops sharply, since the size of the program increases dramatically when the importance levels need to be represented in the problem. The other method shows a good performance for the problems computing an optimal outcome and a different one, once an interpretation is given. For the other two problems, we proposed complementary approaches for this method with different regions of good performance. To conclude, the method modeling problems as disjunctive logic programs shows a potential

on simple instances (have no rank or a few ranks), and the other method performs very well on basic problems (finding an optimal solution or a different optimal solution).

To extend the current results, we will study possible improvements to our second method based on disjunctive logic program encodings. We also intend to study other methods for the optimization problems.

In Chapter 5, we studied the problem of misrepresenting preferences strategically by some of the agents in a group. We considered the situation when an agent secure that the group decision is more desirable to her by misrepresenting her preferences (manipulation) or coercing another agent to represent his preferences insincerely (simple bribery). We provided the conditions with which the manipulation and simple bribery are possible and how hard to decide the possibility when the preferences (or agents) are equally ranked and strictly ranked. The results show that it is impossible for any agent to strategically misrepresent preferences, or it is intractable to decide the existence of manipulation or simple bribery.

Our work leaves several interesting open problems. First, methods to lift preorders from sets to power sets can be defined axiomatically in terms of properties for the lifted preorders to satisfy. Are there general results characterizing the existence of manipulation (simple bribery) for lifted preorders specified only by axioms they satisfy? Second, we do not know the exact complexity of the problems $ESB^{cw}$, $EM^{ar}$ and $ESB^{ar}$ for the equally ranked preferences, nor for $EM^{lmin}$ and $ESB^{lmin}$ for the strictly ranked preferences (the superscript indicates the set comparison method used). Finally, in the setting of equally ranked preferences, most aggregation rules of practical significance properly extend the Pareto one. We conjecture that at least for some of these rules, one can derive results on existence of manipulation and simple bribery from our results concerning the Pareto rule.

In Chapter 6, we considered the problem learning the importance of preferences when the profile and a set of pairwise ordered options are given. For both preference aggregation with Pareto principle and voting with a positional scoring rule, we studied the problem

to decide whether a rank assignment for preferences or a weight assignment for votes exists such that all given outcomes are correctly ordered by the ranked profile or weighted profile. For a preference profile, we provided a polynomial algorithm to solve this problem and computes a rank assignment if one exists. For a voting profile, we proved the problem is NP-complete. If the profile is not consistent with all examples, we considered an optimization problem to decide whether a rank/weight assignment exists such that the ranked/weighted profile can satisfy at least $k$ examples. For both of the ranked and weighted profile, we proved the optimization problem is NP-complete.

In the future work, for voting profiles, we only considered the positional scoring voting rules and the results can be extended other voting rules. We can also study conditional learning problem, such as learning the importance of preferences given that preference $a$ is more important than preference $b$. Moreover, the active learning problem is also interesting which studies how to recover the importance of preferences with minimal number of queries.

## Bibliography


[1] Kenneth J. Arrow. *Social Choice and Individual Values*. Cowles Foundation Monographs Series. Yale University Press, 1963.

[2] Kenneth J. Arrow, Amartya Sen, and Kotaro Suzumura, editors. *Handbook of Social Choice and Welfare*. North-Holland, 2002.

[3] Salvador Barberà, Walter Bossert, and Prasanta K Pattanaik. *Ranking sets of objects*. Springer, 2004.

[4] Nathanaël Barrot, Laurent Gourvès, Jérôme Lang, Jérôme Monnot, and Bernard Ries. Possible winners in approval voting. In Patrice Perny, Marc Pirlot, and Alexis Tsoukiàs, editors, *Algorithmic Decision Theory*, volume 8176 of *Lecture Notes in Computer Science*, pages 57–70. Springer Berlin Heidelberg, 2013.

[5] John Bartholdi, Craig Tovey, and Michael Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[6] John J. Bartholdi, Craig A. Tovey, and Michael A. Trick. How hard is it to control an election? *Mathematical and Computer Modelling*, 16(8):27 – 40, 1992.

[7] Dorothea Baumeister, Magnus Roos, Jörg Rothe, Lena Schend, and Lirong Xia. The possible winner problem with uncertain weights. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pages 133–138, 2012.

[8] Salem Benferhat, Claudette Cayrol, Didier Dubois, Jerome Lang, and Henri Prade. Inconsistency management and prioritized syntax-based entailment. pages 640–645. Morgan Kaufmann, 1993.

[9] Nadja Betzler, Susanne Hemmann, and Rolf Niedermeier. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 53–58, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[10] Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera. Learning conditionally lexicographic preference relations. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 269–274, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[11] Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. Ucp-networks: A directed graphical representation of conditional utilities. *CoRR*, abs/1301.2259, 2013.

[12] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH*, 21:135–191, 2004.

[13] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Reasoning with conditional ceteris paribus preference statements. In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 71–80, 1999.

[14] Ronen I. Brafman and Yannis Dimopoulos. Extended semantics and optimization algorithms for cp-networks. *Computational Intelligence*, 20(2):218–245, 2004.

[15] Ronen I. Brafman and Carmel Domshlak. Introducing variable importance tradeoffs into cp-nets. *CoRR*, abs/1301.0558, 2013.

[16] Steven J. Brams and Peter C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare*, volume 1 of *Handbook of Social Choice and Welfare*, chapter 4, pages 173–236. Elsevier, 2002.

[17] F. Brandt. Group-strategyproof irresolute social choice functions. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 79–84. IJCAI/AAAI, 2011.

[18] F. Brandt and M. Brill. Necessary and sufficient conditions for the strategyproofness of irresolute social choice functions. In Krzysztof R. Apt, editor, *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge, TARK-2011*, pages 136–142. ACM, 2011.

[19] F. Brandt and C. Geist. Finding strategyproof social choice functions via SAT solving. In A. L. C. Bazzan, M. N. Huhns, A. Lomuscio, and P. Scerri, editors, *International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2014*, pages 1193–1200. IFAAMAS/ACM, 2014.

[20] Darius Braziunas and Craig Boutilier. Local utility elicitation in GAI models. *CoRR*, abs/1207.1361, 2012.

[21] Eric Brelsford, Henning Schnoor, Piotr Faliszewski, Edith Hemaspaandra, and Ilka Schnoor. Approximability of manipulating elections. In *In Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 44–49. AAAI Press, 2008.

[22] Gerhard Brewka. Logic programming with ordered disjunction. *CoRR*, cs.AI/0207042, 2002.

[23] Gerhard Brewka. Answer sets: From constraint programming towards qualitative optimization. In *Logic Programming and Nonmonotonic Reasoning, 7th International Conference, LPNMR 2004, Fort Lauderdale, FL, USA, January 6-8, 2004, Proceedings*, pages 34–46, 2004.

[24] Gerhard Brewka. Complex preferences for answer set optimization. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, pages 213–223, 2004.

[25] Gerhard Brewka, James P. Delgrande, Javier Romero, and Torsten Schaub. asprin: Customizing answer set preferences without a headache. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1467–1474, 2015.

[26] Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen. Implementing ordered disjunction using answer set solvers for normal programs. In *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*, pages 444–455, 2002.

[27] Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. Answer set optimization. In *IJCAI*, pages 867–872, 2003.

[28] Gerhard Brewka, Miroslaw Truszczynski, and Stefan Woltran. Representing preferences among sets. In *AAAI*, 2010.

[29] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing disjunctive datalog by constraints. *IEEE Trans. on Knowl. and Data Eng.*, 12(5):845–860, September 2000.

[30] Urszula Chajewska, Lise Getoor, Joseph Norman, and Yuval Shahar. Utility elicitation as a classification problem. *CoRR*, abs/1301.7367, 2013.

[31] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making Rational Decisions Using Adaptive Utility Elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 363–369. AAAI Press / The MIT Press, 2000.

[32] Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. Preference handling in combinatorial domains: From AI to social choice. *AI Magazine*, 29(4):37–46, 2008.

[33] Yann Chevaleyre, Frédéric Koriche, Jérôme Lang, Jérôme Mengin, and Bruno Zanuttini. Learning ordinal preferences on multiattribute domains: The case of cp-nets. In *Preference Learning.*, pages 273–296. 2010.

[34] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artif. Int. Res.*, 10(1):243–270, May 1999.

[35] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *J. ACM*, 54, 2007.

[36] Jessica Davies, George Katsirelos, Nina Narodytska, and Toby Walsh. Complexity of and algorithms for borda manipulation. *CoRR*, abs/1105.5667, 2011.

[37] James P. Delgrande, Torsten Schaub, and Hans Tompits. A framework for compiling preferences in logic programs. *CoRR*, cs.AI/0203005, 2002.

[38] Yannis Dimopoulos, Loizos Michael, and Fani Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1890–1895, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[39] József Dombi, Csanád Imreh, and Nándor Vincze. Learning lexicographic orders. *European Journal of Operational Research*, 183(2):748 – 756, 2007.

[40] Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. Preferences in AI: An overview. *Artif. Intell.*, 175(7-8):1037–1052, 2011.

[41] Didier Dubois, Jérôme Lang, and Henri Prade. A brief overview of possibilistic logic. In *ECSQARU*, pages 53–57, 1991.

[42] Thomas Eiter, Esra Erdem, Halit Erdogan, and Michael Fink. Finding similar or diverse solutions in answer set programming. In Patricia M. Hill and David Scott Warren, editors, *Proceedings of the 25th International Conference on Logic Programming, ICLP 2009*, volume 5649 of *LNCS*, pages 342–356. Springer, 2009.

[43] Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Computing preferred answer sets by meta-interpretation in answer set programming. *CoRR*, cs.LO/0201013, 2002.

[44] Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.

[45] Edith Elkind, Piotr Faliszewski, and Arkadii M. Slinko. Swap bribery. *CoRR*, abs/0905.3885, 2009.

[46] Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *In Proceedings of the 16th International Symposium on Algorithms and Computation*, pages 206–215. Springer-Verlag, 2005.

[47] Wolfgang Faber, Miroslaw Truszczynski, and Stefan Woltran. Strong equivalence of qualitative optimization problems. *CoRR*, abs/1112.0791, 2011.

[48] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. The complexity of bribery in elections. In *AAAI*, pages 641–646, 2006.

[49] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. How hard is bribery in elections? *J. Artif. Int. Res.*, 35(1):485–532, July 2009.

[50] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Using complexity to protect elections. *Commun. ACM*, 53(11):74–82, November 2010.

[51] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Multimode control attacks on elections. *J. Artif. Intell. Res. (JAIR)*, 40:305–351, 2011.

[52] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. *Algorithmic Aspects in Information and Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings*, chapter Copeland Voting Fully Resists Constructive Control, pages 165–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[53] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Llull and copeland voting computationally resist bribery and constructive control. *J. Artif. Intell. Res. (JAIR)*, 35:275–341, 2009.

[54] Piotr Faliszewski and Ariel D. Procaccia. Ai's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[55] P.C. Fishburn. Even-chance lotteries in social choice theory. *Theory and Decision*, 3:18–40, 1972.

[56] Peter C. Fishburn. Lexicographic Orders, Utilities and Decision Rules: A Survey. *Management Science*, 20(11):1442–1471, July 1974.

[57] Zack Fitzsimmons, Edith Hemaspaandra, and Lane A. Hemaspaandra. Control in the presence of manipulators: Cooperative and competitive cases. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 113–119. AAAI Press, 2013.

[58] Peter Flach and Edson Takashi Matsubara. A simple lexicographic ranker and probability estimator. In *Proceedings of the 18th European Conference on Machine Learning*, ECML '07, pages 575–582, Berlin, Heidelberg, 2007. Springer-Verlag.

[59] Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In *ECML*, pages 145–156, 2003.

[60] Johannes Fürnkranz and Eyke Hüllermeier, editors. *Preference Learning*. Springer, 2010.

[61] P. Gärdenfors. Manipulation of social choice functions. *Journal of Economic Theory*, 13(2):217–228, 1976.

[62] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. pages 1070–1080. MIT Press, 1988.

[63] Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):pp. 587–601, 1973.

[64] Judy Goldsmith and Ulrich Junker, editors. *Special Issue on Preferences*, volume 29(4) of *AI Magazine*. 2008.

[65] Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, pages 224–234, 2004.

[66] Joshua T. Guerin, Thomas E. Allen, and Judy Goldsmith. Learning cp-net preferences online from user queries. In *Late-Breaking Developments in the Field of Artificial Intelligence, Bellevue, Washington, USA, July 14-18, 2013*, 2013.

[67] Vu A. Ha and Peter Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. *CoRR*, abs/1302.1544, 2013.

[68] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172(16-17):1897–1916, November 2008.

[69] Souhila Kaci. *Working with Preferences: Less Is More*. Cognitive Technologies. Springer, 2011.

[70] J. Kelly. Strategy-proofness and social choice functions without single-valuedness. *Econometrica*, 45(2):439–446, 1977.

[71] Kathrin Konczak and Jerome Lang. Voting procedures with incomplete preferences. In *JCAI-05 Workshop on Advances in Preference Handling*, pages 124–129, 2005.

[72] Frédéric Koriche and Bruno Zanuttini. Learning conditional preference networks. *Artificial Intelligence*, 174(11):685 – 703, 2010.

[73] Frédéric Koriche and Bruno Zanuttini. Learning conditional preference networks with queries. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1930–1935, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[74] Jérôme Lang and Jérôme Mengin. The complexity of learning separable ceteris paribus preferences. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 848–853, 2009.

[75] Jérôme Lang and Jérôme Mengin. Learning preference relations over combinatorial domains. In *In NMR'08*, 2008.

[76] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, July 2006.

[77] Juntao Liu, Yi Xiong, Caihua Wu, Zhijun Yao, and Wenyu Liu. Learning conditional preference networks from inconsistent examples. *IEEE Trans. Knowl. Data Eng.*, 26(2):376–390, 2014.

[78] Xudong Liu and Miroslaw Truszczynski. Aggregating conditionally lexicographic preferences using answer set programming solvers. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT)*, volume 8176, pages 244–258. Springer, 2013.

[79] D. S. Johnson M. R. Garey. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[80] Victor W. Marek and Miroslaw Truszczynski. Stable models and an alternative logic programming paradigm. *CoRR*, cs.LO/9809032, 1998.

[81] Victor W. Marek and Miroslaw Truszczynski. Stable models and an alternative logic programming paradigm. In K.R. Apt, V.W. Marek, M. Truszczynski, and D.S. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer, Berlin, 1999.

[82] Reshef Meir, Ariel D. Procaccia, Jeffrey S. Rosenschein, and Aviv Zohar. Complexity of strategic behavior in multi-winner elections. *J. Artif. Int. Res.*, 33(1):149–178, September 2008.

[83] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.

[84] Christos H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007.

[85] Gadi Pinkas. Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, pages 525–530, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[86] Mark A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10, 1975.

[87] Michael Schmitt and Laura Martignon. On the complexity of learning lexicographic strategies. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:55–83, 2006.

[88] Patrik Simons, Ilkka Niemelá, and Timo Soininen. Extending and implementing the stable model semantics. *Artif. Intell.*, 138(1-2):181–234, June 2002.

[89] Tran Cao Son and Enrico Pontelli. Planning with preferences using logic programming. *CoRR*, abs/cs/0508132, 2005.

[90] A.D. Taylor. *Social Choice and the Mathematics of Manipulation*. Cambridge University Press, 2005.

[91] Toby Walsh. Uncertainty in preference elicitation and aggregation. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 3–8. AAAI Press, 2007.

[92] Lirong Xia and Vincent Conitzer. Determining possible and necessary winners given partial orders. *CoRR*, abs/1401.3876, 2014.

[93] Lirong Xia, Michael Zuckerman, Ariel D. Procaccia, Vincent Conitzer, and Jeffrey S. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 348–353, 2009.

[94] Fusun Yaman, Thomas J. Walsh, Michael L. Littman, and Marie desJardins. Learning lexicographic preference models. In *Preference Learning.*, pages 251–272. 2010.

[95] Fusun Yaman, Thomas J. Walsh, Michael L. Littman, and Marie desJardins. Democratic approximation of lexicographic preference models. *Artificial Intelligence*, 175:1290 – 1307, 2011. Representing, Processing, and Learning Preferences: Theoretical and Practical Challenges.

[96] Ying Zhu and Miroslaw Truszczynski. On optimal solutions of answer set optimization problems. In Pedro Cabalar and TranCao Son, editors, *Logic Programming and Nonmonotonic Reasoning*, volume 8148 of *Lecture Notes in Computer Science*, pages 556–568. Springer Berlin Heidelberg, 2013.

[97] Ying Zhu and Miroslaw Truszczynski. Manipulation and bribery when aggregating ranked preferences. In *Algorithmic Decision Theory - 4th International Conference, ADT 2015, Lexington, KY, USA, September 27-30, 2015, Proceedings*, pages 86–102, 2015.

[98] Michael Zuckerman, Ariel D. Procaccia, and Jeffrey S. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2):392 – 412, 2009.

**Vita**

**Name:** Ying Zhu
**Place of Birth:** Zhengzhou, Henan, China

## Education

- B.E., Software Engineering, East China Normal University, June 2010.

## Employment

- 2011 - current, Teaching Assistant, Department of Computer Science, University of Kentucky, Lexington, KY.
- 2015, Software Engineering Intern, Groupon, Palo Alto, CA.
- 2014, Software Engineering Intern, Amazon, Seattle, WA.
- 2010 - 2013, Research Assistant, Department of Computer Science, University of Kentucky, Lexington, KY.

## Honors and Awards

- Program committee for ADT-2017.
- Verizon Communications Graduate Fellowship, University of Kentucky, 2015.
- Reviewer for ECAI-2014, AAAI-2014, IJCAI-2013.
- Conference organizer for 27th International Conference on Logic Programming, Lexington, KY, 2011.
- Conference organizer for Thirty Years of Nonmonotonic Reasoning, Lexington, KY, USA, 2010.
- Gillis Graduate School Scholarship, University of Kentucky, 2010.
- First Grade Scholarship, East China Normal University, 2010

## Talks

- "Manipulation and Bribery when Aggregating Ranked Preferences", International Conference on Algorithmic Decision Theory, Lexington, Kentucky, September 2015.
- "Manipulation and Bribery in Preference Reasoning under Pareto Principle", 8th Multidisciplinary Workshop on Advances in Preference Handling (MPREF), Quebec, Canada, July 2014.

- "On Optimal Solutions of Answer Set Optimization Problems", Keeping Current Seminar, Department of Computer Science, University of Kentucky, August 2014.
- "On Optimal Solutions of Answer Set Optimization Problems", 12th International Conference on Logic Programming and Non-Monotonic Reasoning, Corunna, Spain, September 2013.
- "Tools for Preference Reasoning", AAAI-13 Eighteenth AAAI/SIGART Doctoral Consortium, Seattle, July 2013.
- "On the Semantics of Logic Programs with Preferences", AI Seminar, Department of Computer Science, University of Kentucky, February 2013.
- "Finding Similar or Diverse Solutions in Answer Set Programming and Answer Set Optimization", AI Seminar, Department of Computer Science, University of Kentucky, October 2012.
- "Tools for Answer Set Programming", Keeping Current Seminar, Department of Computer Science, University of Kentucky, March 2011.

## Publications

- Ying Zhu, Mirosław Truszczyński: Learning Importance of Preferences. ECAI. (under review)
- Ying Zhu, Mirosław Truszczyński: Manipulation and Bribery when Aggregating Ranked Preferences. International Conference on Algorithmic Decision Theory. (2015)
- Ying Zhu, Mirosław Truszczyński: Manipulation and Bribery in Preference Reasoning under Pareto Principle. AAAI-MPREF. (2014)
- Ying Zhu, Mirosław Truszczyński: On Optimal Solutions of Answer Set Optimization Problems. International Conference on Logic Programming and Non-monotonic Reasoning. (2013)
- Ying Zhu: Tools for Preference Reasoning. AAAI-SIGART Doctoral Consortium. (2013)