CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Reasoning with imprecise trade-offs in decision making under certainty and uncertainty
Authors	Razak, Abdul
Publication date	2014
Original Citation	Razak, A. 2014. Reasoning with imprecise trade-offs in decision making under certainty and uncertainty. PhD Thesis, University College Cork.
Type of publication	Doctoral thesis
Rights	© 2014, Abdul Razak. - http://creativecommons.org/licenses/by-nc-nd/3.0/
Download date	2026-11-01 21:30:52
Item downloaded from	https://hdl.handle.net/10468/1922



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Reasoning with Imprecise Trade-offs in Decision Making under Certainty and Uncertainty

Abdul Razak

MSc Mathematics with Computer Science

110132171



National University of Ireland, Cork

Faculty of Science

Department of Computer Science

Cork Constraint Computation Center (4C)

Thesis submitted for the degree of Doctor of Philosophy

August 21, 2014

Supervisors: Dr. Nic Wilson
Dr. Steve Prestwich
Head of Department/School: Professor Barry O'Sullivan

Research supported by IRCSET and IBM through the Enterprise Parternship Scheme

Contents

Acknowledgements vi
Abstract viii
List of Figures x
List of Tables xiii

1 Introduction **1**
1.1 Background 1
1.2 Problem Statement 2
1.3 Research Aim 4
1.4 Research Objectives 4
1.5 Contributions 4
1.6 Outline of the Thesis 6
1.7 Publications 7

2 Definitions and Concepts **9**
2.1 Introduction 9
2.2 Probability Theory 10
2.2.1 Events 10
2.2.2 Definition of Probability 11
2.2.3 Axioms 13
2.2.4 Conditional and Joint Probabilities 14
2.2.5 Chance Variables and Probability Distributions 15
2.2.6 Chain Rule for Variables 16
2.2.7 Bayes Rule 16
2.2.7.1 Rule of Total Probability and Marginalization . 16
2.2.8 Graphical Representation 18
2.3 Probabilistic Network 19
2.3.1 Bayesian Network 19
2.4 Decision and Utility Theory 21
2.4.1 Decision Making under Certainty 24
2.4.1.1 Multi-objective Constraint Optimization Problems 24
2.4.2 Decision Making under Uncertainty 26
2.4.2.1 Influence Diagrams 26
2.4.2.2 Influence Diagrams with Multiple Objectives . 30
2.4.3 Decision Trees 30

2.5 Summary 32

3 Related Work **34**
3.1 Introduction 34
3.2 Preferences 35
3.2.1 Preference Relations and Properties 35
3.2.1.1 Convex Cones and Preferences 39
3.2.2 Preference Representation 42
3.2.2.1 Cardinal Preference Representation 43
3.2.2.2 Relational Preference Representation 49
3.2.2.3 Representation using Convex Cones 50
3.3 Multi-objective Constraint Optimization 51
3.3.1 Graph Concepts 52
3.3.2 AND/OR Search Spaces and Trees 53
3.3.2.1 Weighted AND/OR Search Tree 55
3.3.3 Multi-objective AND/OR Branch-and-Bound 58
3.3.4 Variable Elimination Algorithm 59
3.3.4.1 Bucket and Mini Bucket Elimination 61
3.3.4.2 Branch-and-Bound and Mini-Bucket Heuristics 62
3.4 Decision Making under Uncertainty: Influence Diagrams 70
3.4.1 Transformation-based Evaluation 70
3.4.2 Variable Elimination Techniques 72
3.4.2.1 Bucket Elimination 73
3.4.3 Other Algorithms 78
3.4.4 Approximation Techniques 79
3.5 Summary and Conclusion 80

4 Multi-objective Constraint Optimization with Trade-offs **82**
4.1 Introduction 83
4.2 Multi-objective Constraint Optimization 84
4.3 Multi-objective AND/OR Branch-and-Bound 86
4.4 Handling Imprecise Trade-offs 88
4.4.1 Deducing Preferences from Additional Inputs 88
4.4.2 Preference Models 89
4.4.3 Implementing Matrix-based Dominance Test 105
4.4.4 Testing Consistency 109
4.5 Reducing the Upper Bound Sets 112
4.5.1 Pareto (no trade-offs) Case 113

4.5.2 Trade-offs Case 116
4.6 Experiments 121
4.7 Summary and Conclusion 131

5 Multi-objective Influence Diagrams with Trade-offs 133
5.1 Introduction 134
5.2 Multi-objective Influence Diagrams 135
5.2.1 The Graphical Model 135
5.2.2 Arithmetic Operations and Distributive Properties 137
5.2.3 Equivalent Sets of Utility Values 138
5.2.4 Variable Elimination 139
5.3 Approximating the Pareto Set 141
5.4 Handling Imprecise Trade-offs 146
5.5 Experiments 149
5.6 Summary and Conclusion 155

6 Conclusion and Future Directions 158
6.1 Summary and conclusion 158
6.2 Future Directions 160
6.2.1 Distributed Constraint Optimization Problem 160
6.2.2 Limited Memory Influence Diagrams 161

I, Abdul Razak, certify that this thesis is my own work and I have not obtained a degree in this university or elsewhere on the basis of the work submitted in this thesis.

Abdul Razak

To my parents - for endless love, unconditional support, encouragement and continuous prayers.

Acknowledgements

First of all, I thank God, the Almighty for giving me this opportunity and granting me the ability to complete this research successfully. This thesis appears in its current form due to the assistance and guidance of several people. I would therefore like to acknowledge the help and support of the following people.

I am luckier than most PhD students to have a supervisor like Dr. Nic Wilson. Right from the beginning of this journey, he offered me continuous guidance in research and professional activities. I would like to express my deep appreciation and gratitude for his patience, valuable comments, encouragement, advice and critics which ensured the successful completion of this project. Besides research, I would like to thank him for the friendly chats on my favourite sport (cricket) on number of occasions.

I am also grateful to my second supervisor Dr. Steve Prestwich for his valuable comments in improving this thesis. My special thanks go to Dr. Radu Marinescu who contributed to my work with his continuous collaboration and valuable suggestions.

I greatly appreciate the support of 4C laboratory administrative staff, Eleanor O'Riordan, Caitriona Walsh, Linda O'Sullivan for all the administrative assistance, and also the 4C IT department Peter MacHale and Joe Scanlon for handling all my IT needs.

I am grateful to all current and previous members of the 4C, with whom I had many productive and enjoyable discussions, especially - Dr. Walid Trabelsi, Dr. Syed Imran, Dr. Franclin Foping, Dr. John Feehan, Dr. Mustafa Al Bado, Dr. Mohamed Wahbi, Lisa Swenson, Oscar Manzano, Saim Ghafoor and Jerome Arokkiam. I am also very grateful to Dr. Zubair Kabir, University College Cork; Dr. Lacour Ayompe, University College Cork, for their scientific advice and knowledge and many insightful discussions and suggestions.

I wish to thanks my friends Mohammed Tanveer, Benzir Ahmed, Jubair Kabir, Shahidul Haque, Mohammed Mohan Miah, Atikur Rahman, Nasir Abdul Quadir, Sajjad, Junaid Amin, Rehan Ahmed, Aswand, Juan Pablo and Ibad Ahmed (the little master) for all the emotional support, camaraderie, entertainment, and caring they provided.

My deepest gratitude is to my family: to my father Gulam Dastagir, my mother

Beepasha Begum, my brother Mohammed Taher and my sisters Fahima, Shaheen, Mumtaaz and Tasleem without whose blessings and love I would not have reached so far in my life. I would also like to thank my nieces: Sameena, Safia, Sabiha and Tabassum; and nephews: Sameer, Khalid, Altaf, Arbaaz in particular, Sharib and Shehzaan for making me smile all the time.

Finally, I would like to acknowledge the support of the IRCSET and IBM for this research project through the IRCSET Enterprise Partnership Scheme. Also, I would like to show my gratitude Science Foundation Ireland for supporting me financially (under the grant no 08/PI/I1912) in the last two months of this research work.

Abstract

In many real world situations, we make decisions in the presence of multiple, often conflicting and non-commensurate objectives. The process of optimizing systematically and simultaneously over a set of objective functions is known as multi-objective optimization. In multi-objective optimization, we have a (possibly exponentially large) set of decisions and each decision has a set of alternatives. Each alternative depends on the state of the world, and is evaluated with respect to a number of criteria. In this thesis, we consider the decision making problems in two scenarios. In the first scenario, the current state of the world, under which the decisions are to be made, is known in advance. In the second scenario, the current state of the world is unknown at the time of making decisions.

For decision making under certainty, we consider the framework of multi-objective constraint optimization and focus on extending the algorithms to solve these models to the case where there are additional trade-offs. We focus especially on branch-and-bound algorithms that use a mini-buckets algorithm for generating the upper bound at each node of the search tree (in the context of maximizing values of objectives). Since the size of the guiding upper bound sets can become very large during the search, we introduce efficient methods for reducing these sets, yet still maintaining the upper bound property. We define a formalism for imprecise trade-offs, which allows the decision maker during the elicitation stage, to specify a preference for one multi-objective utility vector over another, and use such preferences to infer other preferences. The induced preference relation then is used to eliminate the dominated utility vectors during the computation.

For testing the dominance between multi-objective utility vectors, we present three different approaches. The first is based on a linear programming approach, the second is by use of distance-based algorithm (which uses a measure of the distance between a point and a convex cone); the third approach makes use of a matrix multiplication, which results in much faster dominance checks with respect to the preference relation induced by the trade-offs. Furthermore, we show that our trade-offs approach, which is based on a preference inference technique, can also be given an alternative semantics based on the well known Multi-Attribute Utility Theory. Our comprehensive experimental results on common multi-objective constraint optimization benchmarks demonstrate that the

proposed enhancements allow the algorithms to scale up to much larger problems than before.

For decision making problems under uncertainty, we describe multi-objective influence diagrams, based on a set of p objectives, where utility values are vectors in $\mathbb{R}^p$, and are typically only partially ordered. These can be solved by a variable elimination algorithm, leading to a set of maximal values of expected utility. If the Pareto ordering is used this set can often be prohibitively large. We consider approximate representations of the Pareto set based on ϵ-coverings, allowing much larger problems to be solved. In addition, we define a method for incorporating user trade-offs, which also greatly improves the efficiency.

List of Figures

2.1 Venn Diagram 12
2.2 Venn diagram for conditional probability 12
2.3 A DAG representation of $P(X|Y_1 \ldots, Y_k)$. 18
2.4 A simple Bayesian Network with three variables. 20
2.5 A simple influence diagram with two decisions. 29
2.6 A decision tree of the oil wildcatter problem. 32

3.1 A finitely generated convex cone. 40
3.2 A pointed cone (left) and a non-pointed cone (right). 40
3.3 Pointed cone (grey) and its dual cone (yellow). 42
3.4 A MOCOP instance with 2 objectives. 54
3.5 AND/OR search space. 56
3.6 Weighted AND/OR search tree. 57
3.7 A MOCOP instance with bucket structure. 60
3.8 Variable elimination execution and bucket-tree. 61
3.9 Utility associated with a partial solution tree. 64
3.10 Pruning the partial solution tree. 65
3.11 A MOCOP instance and a corresponding pseudo tree. 66
3.12 Static bucket elimination heuristic. 67
3.13 Static mini-bucket heuristics for $i = 3$. 67
3.14 Dynamic mini-bucket heuristics for $i = 3$. 70
3.15 An influence diagram representation of car buying problem. . . 73
3.16 Initial bucket structure of car problem. 74
3.17 Bucket evaluation for car example. 77

4.1 Convex cone of the preference information $\Theta = \{((2,5),(3,3)), ((5,2),(2,3))\}$. 102
4.2 Distance Algorithm with three iterations to conclude that the point is inside the cone. 104
4.3 Distance Algorithm with four iterations to conclude that point is outside the cone. 105
4.4 Comparing matrix based versus LP based dominance checks on influence diagrams with 3 and 5 objectives. CPU time in seconds as a function of problem size. Time limit 20 minutes. 108
4.5 Dual cone. 109

4.6 A MOCOP problem (given in Example 5 in Section 3.3.2, Chapter 3) with $\succcurlyeq_\Theta$-dominance. 110
4.7 Number of problem instances solved for random networks with 4 objectives. Using different Pareto least upper bound based methods. Time limit 3 minutes. 116
4.8 Number of problem instances solved for random networks with 4 objectives using different linear program based methods (left) and for random networks with 3 objectives using different cluster sizes (k) with method **RKG** (right). Time limit 3 minutes. . . . 119
4.9 A bi-attribute weighted graph G (left) and its vertex cover (right) 122
4.10 CPU time in seconds (left) and number of problem instances solved (right) for random networks with 5 objectives, combinatorial auctions with 3 objectives and vertex covering problems with 5 objectives, respectively. Using the Pareto ordering. Time limit 30 minutes. 126
4.11 CPU time in seconds (left) and number of problem instances solved (right) for random networks with 5 objectives, combinatorial auctions with 3 objectives and vertex covering problems with 5 objectives, respectively. Trade-offs generated with parameters $(K = 6, T = 3)$ for random networks, $(K = 2, T = 1)$ for combinatorial auctions and $(K = 5, T = 2)$ for vertex covering. Time limit 20 minutes. 127
4.12 CPU time in seconds for vertex covering problems with 3 objectives and 110 variables (left); and 5 objectives with 110 variables (right) as a function of the upper bound set size. Using the Pareto ordering. Time limit 20 minutes. 128
4.13 CPU time in seconds (left) and number of problem instances solved (right) for random networks with 4 objectives and 80 variables as a function of the upper bound set size. Time limit 20 minutes. 129
4.14 CPU time in seconds (left) and number of problem instances solved (right) as a function of the number of pairwise trade-offs (K) for vertex covering problems with $n = 160$, 5 objectives and using $T = 1$. The mini-bucket i-bound is 10. Time limit 20 minutes. 130

4.15 CPU time in seconds (left) and number of problem instances solved (right) as function of the mini-bucket i-bound for vertex covering problems with $n = 160$, 5 objectives and $(K = 5, T = 2)$ trade-offs. Time limit 20 minutes. 130

5.1 A bi-objective influence diagram. 136
5.2 Examples of ϵ-covering. 142
5.3 Distribution (mean and standard deviation) of the ϵ-covering size as a function of the ϵ value and plots showing the percentage of instances solved out of 100. Time limit 20 minutes. 152
5.4 Distribution of the maximal sets as a function of the number of pairwise trade-offs K and fixed 3-way trade-offs (T). Mean and standard deviation are shown as well as the percentage of instances solved. Time limit 20 minutes. 155
5.5 Distribution of the maximal sets as a function of the number of pairwise trade-offs K and fixed 3-way trade-offs (T). Mean and standard deviation are shown as well as the percentage of instances solved. Time limit 20 minutes. 156

List of Tables

5.1 Results with algorithms ELIM-MOID and ELIM-MOID$_\epsilon$ on random influence diagrams. Time limit 20 minutes. 152
5.2 Results comparing algorithms ELIM-MOID and ELIM-MOID-TOFs on random influence diagrams with random trade-offs. Time limit 20 minutes. 154
5.3 Impact of the quality of the random trade-offs on bi-objective influence diagrams. Time limit 20 minutes. 154
5.4 Results comparing algorithms ELIM-MOID-TOF (MATRIX) and ELIM-MOID-TOF$_\epsilon$ (MATRIX) with varying ϵ values on random influence diagrams with random tradeoffs. Time limit 20 minutes. 157

Chapter 1

Introduction

1.1 Background

In many real world situations, we make decisions in the presence of multiple, often conflicting and non-commensurate objectives. For instance, when we book a hotel for a holiday, we would ideally like a less expensive hotel with a good star rating. Clearly, the objectives relating to *price* and *rating* in this case conflict with each other. The process of optimizing systematically and simultaneously over a set of objective functions is known as *multi-objective optimization* or *vector optimization*.

In a multi-objective optimization problem, we have a (possibly exponentially large) set of decisions and each decision is associated with a set of *actions* (also known as *alternatives* or *options*). Each action is evaluated on a number of criteria. That means, an outcome of an action is a vector of objective values, that depends on the state of the world, also commonly referred to as a *state of nature*. If the world state is known prior to making decisions, then each chosen action leads to a specific outcome. For instance, in the hotel booking example, an outcome $(€300, 3)$ could be associated with a particular action, representing a hotel which costs three hundred Euros for a selected week and has rated three stars.

In general, the multi-objective decision problems under certainty can be represented by using the multi-objective constraint optimization framework. A multi-objective constraint optimization problem consists of a set of decision variables, where each decision variable takes a finite values subject to a set of

constraints expressed by the decision maker; and an objective function, which is the sum of all the utility (or cost) functions in the problem domain, where each utility (or cost) function is defined on some subset of decision variables. A complete assignment to the decision variables is referred to as a *solution* and has an associated multi-objective utility value. The comparison between solutions then reduces to the comparison between outcomes. The task of solving a multi-objective constraint optimization problem is to determine the solutions that are not dominated by any other existing solution, known as the *optimal* or *best* solutions.

If the world state is unknown or hidden, then *probability theory*, in general, is used to quantify the uncertainty over world state. Unlike the case of certainty, under uncertainty each action maps to a set of outcomes, where each outcome occurs with a known probability. Influence diagrams [Howard and Matheson, 1981] are the standard modelling tools for representing and solving sequential decision making problems under uncertainty. An influence diagrams is a directed graphical model with different types of nodes representing the uncertain quantities, decision variables and utility information of the decision maker. Solving an influence diagram amounts to finding the optimal policy for each decision, i.e., the aim is to find the decision alternatives with the highest (or maximal) expected utility. Each optimal policy then maximizes the expected utility, resulting in an optimal strategy for the influence diagram. However, if there are several objectives involved in the decision making process, then the associated utility will be a vector of objective values, and there will no longer necessarily be a unique maximal expected utility value, but a set of them.

1.2 Problem Statement

Several algorithms have been proposed for solving multi-objective constraint optimization problems including search-based (e.g., depth-first Branch-and-Bound search) and inference-based (e.g., Variable Elimination, Tree Clustering) algorithms. For solving multi-objective influence diagrams several exact algorithms have been proposed over the past decade, which mainly adopt the classical variable elimination techniques. However, in both deterministic decision making and the decision making under uncertainty problem solving, the key issue is ordering the multi-objective utility vectors.

The two most common approaches are using the Pareto ordering (or product

ordering), or a weighted coefficients model. In general, the Pareto ordering (denoted with $>$) leads to a weak pre-order on multi-objective utility vectors and hence on the decisions. For instance, in a bi-objective decision making problem, with the Pareto ordering, which is defined by $\vec{u} > \vec{v} \iff u_1 > v_1$ and $u_2 > v_2$, two outcomes $(5, 3)$ and $(4, 5)$ are incomparable. In particular, if there are several objectives and many available options, the Pareto ordering on the decisions may lead to extremely large number of optimal outcomes, which is unhelpful for the decision maker.

With the weighted coefficients model, the decision maker has to assign different weights for the objectives. For example, a weight 0.7 for the first objective and a weight 0.3 for the second objective, evaluates to a utility value of $0.7 \times 5 + 0.3 \times 3 = 4.4$ for $(5, 3)$, and a utility value of $0.7 \times 4 + 0.3 \times 5 = 4.3$ corresponds to $(4, 5)$, thus $(5, 3)$ is preferred to $(4, 5)$. In a weighted coefficients model, the weights determine the trade-offs between different objectives, i.e., how much the decision maker is willing to lose one objective in order to gain one unit of another objective.

However, with the weighted coefficients model, determining the appropriate weights for the objectives is a difficult task because eliciting weights from the decision maker is time-consuming; the reason could be that the decision maker might not have clear idea about the weights that fit exactly with their preferences; or there can be a group of experts involved in the decision making process, and each one of them has different ideas about the relative importance of the objectives. It can be undesirable to force the decision maker to define precise trade-offs between the objectives since they may not have clear idea about them, and it may lead to somewhat arbitrary decisions.

Search-based algorithms for constraint optimization problems, such as depth-first branch-and-bound generate an upper bound, which is a set in the multi-objective context, during the search for optimal solutions. The other issue when considering these algorithms is that the guiding upper bound sets can become large and therefore can have a dramatic impact on the performance of the algorithms.

1.3 Research Aim

The aim of this research is: to help the decision maker in expressing his preferences (or trade-offs) in a multi-objective context; to develop the dominance relation between the multi-objective utility vectors with respect to the input preferences; and to integrate this dominance relation (with respect to input preferences) in the algorithms for solving multi-objective decision making problems under certainty and uncertainty.

1.4 Research Objectives

Specific objectives of this research are:

- To reason about the partial preferences of the decision maker.
- To construct a systematic procedure for testing the consistency of the input preferences because, in many cases, the decision maker may be inconsistent while expressing his preferences in this form.
- To develop mathematical model(s) based on the consistent input preferences to test the dominance relation between multi-objective vectors.
- To integrate the preference based dominance relation in the multi-objective constraint optimization algorithms and the algorithms that solve the multi-objective influence diagrams to eliminate the dominated utility vectors during the computation.

1.5 Contributions

The following are the list of contributions in this thesis:

- We define a simple formalism for representing imprecise trade-offs of the decision maker, which allows the decision maker, during the elicitation stage, to specify a preference for one multi-objective utility vector over another.

  For instance, in the hotel booking example, it is quite easy for a decision maker to specify that outcome $(€300, 3)$ is better than outcome $(€200, 2)$,

which implies a constraint on the possible weights vector, but without specifying the weights vector precisely.

- We show that the consistent set of input preferences induce a preference relation, which can be used to eliminate the dominated multi-objective utility vectors during the computation of multi-objective optimization problems.
- We show that our approach for preference handling can be given as an alternate semantics based on the well known Multi-Attribute Utility Theory (MAUT).
- We develop a computational method based on the linear programming approach for checking the resulting dominance condition, which can be determined by using a linear programming solver (e.g., lpsolve [Berkelaar et al., 2006]) and we show that the (incomplete) algorithm of [Zheng and Chew, 2009] can be used as an alternative approach for testing the dominance between multi-objective utility vectors.
- Since the multi-objective optimization algorithms need to make many dominance checks with respect to the preference relation induced by the imprecise trade-offs, we construct a matrix that represents the preferences of the decision maker. We then compile the dominance check by use of this matrix and show that it can achieve an order of magnitude speed up over the linear programming approach.
- When using the branch-and-bound algorithms, we propose efficient methods for reducing the guiding upper bound sets, yet still ensuring the upper bound property for both the Pareto and trade-offs case.
- We demonstrate empirically on a variety of multi-objective constraint optimization benchmarks that our improved algorithms outperform the current state-of-the-art solvers by a significant margin and therefore they can scale up to much larger problems than before.
- The Pareto ordering on multi-objective utility is a rather weak one; the effect of this is that the set of maximal values of expected utility can often become huge. We make use of the notion of ϵ-covering, which approximates the Pareto set, for the case of multi-objective influence diagrams, and we experimentally demonstrate that the ϵ-covering has the major effect on the size of the maximal values of expected utility and hence on the

computational efficiency and feasibility for larger problems.

1.6 Outline of the Thesis

In this section, we present the brief outline of this thesis.

Chapter 2: Definitions and Concepts

In this chapter, we give background material on probability theory, probabilistic networks and discuss in detail Bayesian networks. For decision making with certainty, we introduce the concepts of multi-objective constraint optimization. For decision making under uncertainty, we present the frameworks of influence diagrams, which is a generalization of Bayesian network augmented with decision variables and utility functions, and we describe decision trees.

Chapter 3: Related Work

In this chapter, we define a preference relation and discuss different preference relations and closely look into their properties. We also describe the notion of *convex cones* which we explore in Chapters 4 and 5 for testing the dominance relation between multi-objective utility vectors. We discuss *cardinal representation* and *relational representation* of preferences and present related work on these models. On the other hand, we study the multi-objective AND/OR branch-and-bound and variable elimination algorithms for solving multi-objective constraint optimization problems, and discuss the work related to these algorithms. We also discuss different algorithms for solving multi-objective influence diagrams, which include transformation based, variable elimination and arc reversal and node removal algorithms. Finally, we discuss the method for approximating the Pareto set, which will be presented in detail in Chapter 5.

Chapter 4: Multi-objective Constraint Optimization with Trade-offs

In this chapter, we present our approach for preference handling, which allows a decision maker to express his preferences in the form of comparison between multi-objective utility vectors. For testing the dominance relation between multi-objective utility vectors, we describe three different approaches, the first is based on convex cones (or linear programming); the second is based

on matrix multiplication, where we construct the matrix with the help of input preferences; and the third is based on the distance algorithm that computes distance between a point and a convex cone. We extend the multi-objective AND/OR branch-and-bound and variable elimination algorithms for the case where there are additional imprecise trade-offs. Since the upper bound sets generated during branch-and-bound search are sets of multi-objective utility vectors, these sets can become extremely large, and we introduce some simple and effective techniques for both no trade-offs and trade-offs cases to handle the cardinality of these sets. We present experimental results illustrating the effects of all our proposed methods on the standard benchmarks.

Chapter 5: Multi-objective Influence Diagrams with Trade-offs

In this chapter, we extend the variable elimination algorithm to solve multi-objective influence diagrams for the case of additional trade-offs. We implement three approaches defined in the previous chapter for testing the dominance relation between multi-objective utility values. We describe an approach for approximating the expected utility sets based on the notion of ϵ-covering. We present the experimental results describing the impact of imprecise trade-offs and approximation techniques on randomly generated multi-objective influence diagrams.

Chapter 6: Conclusion and Future Directions

In this chapter, we summarize the achievements of the thesis and briefly discuss distributed constraint optimization framework and limited memory influence diagrams as a possible future directions of this research.

1.7 Publications

The original work presented in this thesis has been done in collaboration with Dr. Nic Wilson and Dr. Radu Marinescu. Parts of this thesis have been published in the proceedings of international conferences, which have been subject to peer review. We list them below.

- Multi-objective Influence Diagrams. Radu Marinescu, Abdul Razak and Nic Wilson.

In Proceedings of the $28th$ Conference on Uncertainty in Artificial Intelligence (UAI 2012), 2012.

- Multi-objective Constraint Optimization with Trade-offs. Radu Marinescu, Abdul Razak and Nic Wilson.

  In Proceedings of the $19th$ International Conference on Principles and Practice of Constraint Programming (CP 2013), 2013.

Chapter 2

Definitions and Concepts

2.1 Introduction

This chapter gives the basic definitions, concepts and background needed to study decision making problems under certainty and under uncertainty, which are the two main areas of focus of this dissertation. In particular, we study the concepts of Multi-objective Constraint Optimization, which is a very well-known mathematical model for representing many real-world decision making problems without uncertainty. For reasoning problems under uncertainty we present the very famous and powerful framework of Bayesian Networks, and discuss in detail the influence diagram which is a graph based formalism and an extension of a Bayesian network, for representing and solving decision making problems under uncertainty.

The chapter is organized as follows: Section 2.2 gives the basic concepts of probability theory which are the foundation for Bayesian networks and influence diagrams, Section 2.3 defines probabilistic networks and explains Bayesian network model. Section 2.4 introduces decision theory and utility theory, the notion of multi-objective constraint optimization for deterministic decision making is presented in Section 4.2, whereas, decision making under uncertainty and framework of influence diagram is presented in Sections 2.4.2 and 2.4.2.1, respectively.

2.2 Probability Theory

Probabilistic networks, such as Bayesian network is a qualitative and quantitative probabilistic knowledge representations. The qualitative knowledge is modelled into a directed acyclic graph (DAG), which represents the dependence and independence relation between variables. A DAG is derived using basic axioms of the probability theory.

In this section we present some basic concepts and axioms of probability theory such as joint and conditional probability, which play a key role in reasoning and decision making under uncertainty.

2.2.1 Events

Probability theory is a study of *random experiments*, whose outcomes or results are unpredictable. Examples of such experiments are tossing a coin and rolling a dice. Even though we know the outcomes of a random experiment in advance, we cannot predict exactly which of them is going to happen during the experiment. The process of performing a random experiment is called a *trial* and the set of all possible outcomes of a random experiment is known as *sample space* and it is denoted with the letter $\boldsymbol{s}$. For instance, in an experiment of rolling a die the sample space is the set of all its six faces, i.e., $\boldsymbol{s} = \{1, 2, \ldots, 6\}$. In this case, the sample space $\boldsymbol{s}$ is called discrete.

Each possible collection of outcomes of a random experiment is called an *event*. Events are subsets of the sample space of the corresponding random experiment. If A is an event then $A \subseteq \boldsymbol{s}$. For example, suppose that a box contains 50 tickets which have a unique number between 1 and 50 written on them. We mix the tickets in the box thoroughly and draw a ticket randomly, and note down the number on the ticket that has been drawn. The sample space of this experiment will naturally be $\boldsymbol{s} = \{1, 2, \ldots, 50\}$. Now, let us define an event A as 'we draw a ticket that has prime number written on it', then we have $A = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47\}$, which is a subset of $\boldsymbol{s}$.

In a random experiment, all possible events are collectively called *exhaustive events* and their union is the entire sample space. If a random experiment is conducted then at least one of them must occur. For instance, when we roll a die then at least one of its six faces will show-up, thus all its six faces are

exhaustive.

Two events A and B are said to be *disjoint* or *mutually exclusive*, if the happening of one of them prevents the other, i.e., if A happens then B cannot and vice-versa. In the above example of drawing a ticket, if we define A to be 'we draw an odd numbered ticket', B to be 'we draw a prime numbered ticket' and C to be 'we draw an even numbered ticket' then A and C are mutually exclusive events because they cannot happen at the same time. The events B and C are not mutually exclusive because if we draw a ticket with number 2 written on it, then it implies the happening of both of these events. Similarly, the events A and B are not mutually exclusive because the tickets with numbers $3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47$ on them imply the occurrence of A and B in the same trial.

2.2.2 Definition of Probability

Probability is a widely used mathematical concept in the field of artificial intelligence, particularly, it plays a major role in decision making under uncertainty domain which is one of the main areas we focus in this dissertation. So, let us define informally what exactly is meant by probability.

Consider $\boldsymbol{s}$ be the sample space of a random experiment, let A be an event such that $A \subseteq \boldsymbol{s}$. Then the probability of occurrence of A is denoted by $P(A)$ and it is defined as:

$$P(A) = \frac{n(A)}{n(\boldsymbol{s})}$$

where, $n(A)$ and $n(\boldsymbol{s})$ are the number of elements in A and $\boldsymbol{s}$, respectively. Clearly, for every $A \subseteq \boldsymbol{s}$ the probability function P assigns a real value between 0 and 1. If an element is selected randomly from $\boldsymbol{s}$ then $P(A)$ denotes the chance that the element is selected from A. Geometrically, $P(A)$ is the percentage of area occupied by the event A in the sample space $\boldsymbol{s}$.

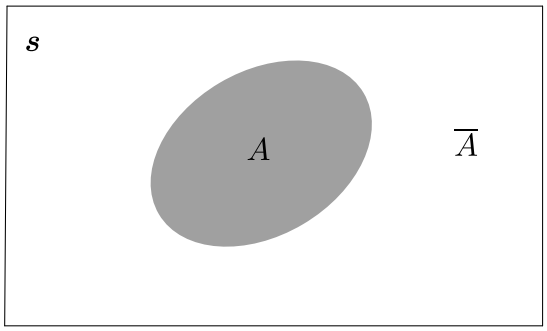


Figure 2.1: Venn Diagram

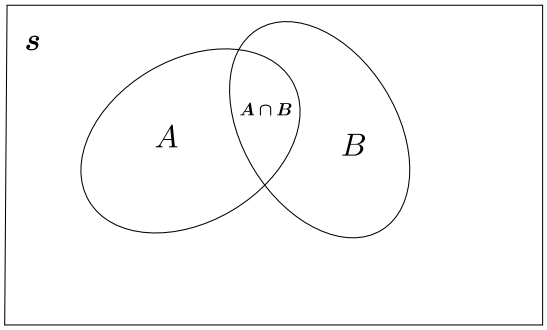


Figure 2.2: Venn diagram for conditional probability

If $\overline{A}$ is the complement of A, then its probability is defined as:

$$\begin{aligned} P(\overline{A}) &= \frac{n(\overline{A})}{n(\boldsymbol{s})} \\ &= \frac{n(\boldsymbol{s}) - n(A)}{n(\boldsymbol{s})} \\ &= 1 - \frac{n(A)}{n(\boldsymbol{s})} \\ &= 1 - P(A) \end{aligned}$$

In figure 2.1, the shaded region represents the area occupied by the event A and the unshaded region is the area of the complement event $\overline{A}$ in $\boldsymbol{s}$.

In an experiment, if A and B are two events as shown in figure 2.2, then the probability of either of them happening is defined as:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

where, $A \cap B$ is the region common to both A and B. For example, in an

experiment of rolling a dice, we want to know the probability of getting either an even number or prime number. Let us assume A to be 'getting even number' then the set of favourable outcomes will be $\{2, 4, 6\}$, and B to be 'getting prime number' then we have the favourable outcomes $\{2, 3\}$. Notice that, the outcome $\{2\}$ is common to both the events, i.e., 2 is favourable for the event $A \cap B$. Then the probability of happening of either A or B is:

$$P(A \cup B) = \frac{3}{6} + \frac{2}{6} - \frac{1}{6} = \frac{2}{3}$$

2.2.3 Axioms

The probability function P, must satisfy the following axioms. These axioms of probability are the base for the Bayes theory and thus foundation for Bayesian network (Section 2.3.1), Influence diagram (Section 2.4.2.1) and Decision tree (Section 2.4.3).

If ***s*** is a discrete sample space of a random experiment then we have:

Axiom 1 For any event A, $0 \leq P(A) < 1$. In particular, $P(A) = 0$ if and only if A is impossible and $P(A) = 1$ if and only if A is certain.

Axiom 1 describes that the probability of any event is a positive real number between 0 and 1, and it cannot be greater than 1. Also, if an event is impossible, meaning that it cannot occur, then its probability is 0. For example, in an experiment of rolling a die, getting a number greater than 6 is an impossible event. Whereas, the event which occurs for sure, always has the probability 1. In the above example, getting a number below 7 is a certain event.

Axiom 2 Probability of the sample space is 1, i.e., $P(\boldsymbol{s}) = 1$.

Axiom 2 simply says that the sum of the probabilities of all possible outcomes in a random experiment is 1.

Axiom 3 If A and B are mutually exclusive events then the probability of either of them happen is:

$$P(A \cup B) = P(A) + P(B)$$

In general, if $A_1, \ldots, A_t$ are t mutually exclusive events, then

$$P(\bigcup_{i=1}^{t} A_i) = \sum_{i=1}^{t} P(A_i).$$

Axiom 3 implies, if no pair of events occur together then probability of one of them occurs is equal to the sum of their individual probabilities.

2.2.4 Conditional and Joint Probabilities

Let A and B be two events. Then the probability of A given evidence that B has already occurred, is denoted by $P(A|B)$, and it is defined as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \text{ provided } P(B) \neq 0.$$

Similarly, the conditional probability of B given evidence that A is already occurred, is defined as

$$P(B|A) = \frac{P(A \cap B)}{P(A)}, \text{ provided } P(A) \neq 0.$$

where, $A \cap B$ denotes the area common to both A and B. For example, in an experiment of rolling a die, we would like to find, the probability of getting number 1 given that an odd number has showed up. Let us define the event A to be 'getting 1 on the dice' and suppose that the evidence B is 'odd number is noticed'. Then we have:

$$\boldsymbol{s} = \{1, 2, \ldots, 6\}, A = \{1\} \text{ and evidence } B = \{1, 3, 5\}$$

Then the probability of occurrence of event A is:

$$P(A|B) = \frac{1}{3}$$

The real value $P(A)$ is called the *prior probability* of A. Whereas, the real number $P(A|B)$ is called the *posterior probability* of A, which is updated probability of event A based on the observation that event B has already occurred.

We say that event A is *independent* of B, if $P(A|B) = P(A)$. Similarly, B is

independent of A, if $P(B|A) = P(B)$. If A and B are independent of each other then we can write, $P(A \cap B) = P(A) \cdot P(B)$.

Using the above definition, if A and B are two events of an experiment, then their joint probability is defined as follows:

$$P(A \cap B) = P(B|A)P(A) \text{ or } P(A \cap B) = P(A|B)P(B). \tag{2.1}$$

In the following sections, $P(A, B)$ is sometimes written as $P(A \cap B)$.

2.2.5 Chance Variables and Probability Distributions

Chance variable (or *Random variable*) is a function $X : \boldsymbol{s} \to \mathbb{R}$, where $\boldsymbol{s}$ is a sample space and $\mathbb{R}$ is set of real numbers. In other words, a chance variable is a function that assigns real values to the outcomes of a random experiment. The sample space $\boldsymbol{s}$ is called the *domain* of X and it is denoted as $dom(X) = \boldsymbol{s}$. Every event in $\boldsymbol{s}$ is called state, level, value, choice, option, etc. of X.

Consider an experiment of tossing two coins then we have $\boldsymbol{s} = \{TT, TH, HT, HH\}$, and if we define the chance variable X to be 'number of heads' then we have the following distribution.

$\boldsymbol{s}$	TT	TH	HT	HH
$X(\boldsymbol{s})$	0	1	1	2

We say that the random number X takes values $0, 1, 2$. In general, if X takes finite values $x_1, \ldots, x_m$ and we define $P(X = x_i) = p_i$, $i = 1, \ldots, m$. Then we say P is a probability function if the following conditions hold:

(i) $p_i \geq 0$, for all i and

(ii) $\sum_i p_i = 1$

In the above example we can define the probability function as $P(X = 0) = \frac{1}{4}, P(X = 1) = \frac{1}{2}$ and $P(X = 2) = \frac{1}{4}$.

A *Discrete Chance Variable* is one that takes a finite or countably infinite number of values. For instance, the chance variable in the above example is discrete.

A *Continuous Chance Variable* is one that takes an uncountable number of values. For example, if we define X to be a real number between 0 and 1 then there exists infinitely many possible values for X, thus it is continuous.

In this dissertation, we only consider the cases where chance variables are discrete.

2.2.6 Chain Rule for Variables

If X and Y are two chance variables and P is a probability distribution then the chain rule says that:

$$P(X,Y) = P(X|Y)P(Y) \text{ or } P(X,Y) = P(Y|X)P(X) \tag{2.2}$$

In general, if $X_1, \ldots, X_k$ be the k random variables, then

$$\begin{aligned} P(X_1, \ldots, X_k) &= P(X_k|X_1, \ldots, X_{k-1})P(X_1, \ldots, X_{k-1}) \\ &= P(X_1)P(X_2|X_1)\ldots P(X_k|X_1, \ldots, X_{k-1}) \\ &= \prod_{i=1}^{k} P(X_i|X_1, \ldots, X_{i-1}) \end{aligned} \tag{2.3}$$

2.2.7 Bayes Rule

If X and Y are two chance variables then it follows immediately from the chain rule given in equation (2.2) that:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{2.4}$$

which is known as *Bayes Rule*.

2.2.7.1 Rule of Total Probability and Marginalization

Let X and Y be two chance variables with domains $dom(X) = \{x_1, \ldots, x_m\}$ and $dom(Y) = \{y_1, \ldots, y_n\}$, respectively and $P(X,Y)$ is the joint probability distribution of X and Y. Suppose that $dom(X)$ and $dom(Y)$ are sets of exhaustive and mutually exclusive states of X and Y then the total probability rule is given by:

$$P(X = x_i) = \sum_{j=1}^{n} P(x_i, y_j), \text{ for all } i = 1, \ldots, m \tag{2.5}$$

Using (2.5) we can calculate $P(X)$ from $P(X,Y)$, which is given by:

$$\begin{aligned} P(X) &= \left(\sum_{j=1}^{n} P(x_1, y_j), \ldots, \sum_{j=1}^{n} P(x_m, y_j) \right) \\ &= \sum_{j=1}^{n} P(X, y_j) \\ &= \sum_{Y} P(X, Y) \end{aligned} \tag{2.6}$$

In equation (2.6) we can see that Y is *eliminated* from $P(X,Y)$ by summing over all domain values of Y, this elimination procedure is also called *marginalizing out* variable Y. This *mariginalization* operation plays an important role in probabilistic networks such as Bayesian networks and influence diagrams.

Example 1 » Marginalization

Let X and Y be two chance variables with $dom(X) = \{x_1, x_2, x_3\}$ and $dom(Y) = \{y_1, y_2, y_3\}$. Suppose that $P(X = x_1) = \frac{2}{10}, P(X = x_2) = \frac{3}{10}$ and $P(X = x_3) = \frac{5}{10}$. The probability of Y conditioned on X is given by the following table:

$P(Y|X) =$

	$X = x_1$	$X = x_2$	$X = x_3$
$Y = y_1$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{2}{9}$
$Y = y_2$	$\frac{3}{9}$	$\frac{2}{9}$	$\frac{3}{9}$
$Y = y_3$	$\frac{5}{9}$	$\frac{2}{9}$	$\frac{4}{9}$

Suppose that we want to compute $P(Y)$. First, we compute $P(X,Y)$, i.e., we compute the joint probability distribution of X and Y then using equation (2.6) we can eliminate X from $P(X,Y)$ to obtain $P(Y)$.

Using equation (2.2), we have:

$$\begin{aligned} P(X = x_1, Y = y_1) &= P(Y = y_1 | X = x_1) P(X = x_1) \\ &= \frac{2}{10} \cdot \frac{1}{9} \\ &= \frac{1}{45} \end{aligned}$$

Applying similar steps to calculate probability of the remaining combinations of states of X and Y then we obtain the complete joint probability distribution

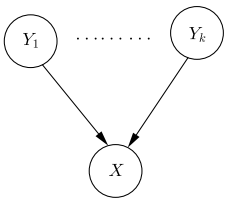


Figure 2.3: A DAG representation of $P(X|Y_1 \ldots, Y_k)$.

$P(X, Y)$, which is given in the following table:

$$P(X,Y) = \begin{array}{c|ccc} & X = x_1 & X = x_2 & X = x_3 \\ \hline Y = y_1 & \frac{1}{45} & \frac{1}{15} & \frac{1}{9} \\ Y = y_2 & \frac{1}{15} & \frac{1}{15} & \frac{1}{6} \\ Y = y_3 & \frac{1}{9} & \frac{1}{6} & \frac{2}{9} \end{array}$$

Now, through the marginalization we get:

$$\begin{aligned} P(Y) &= P(X = x_1, Y) + P(X = x_2, Y) + P(X = x_3, Y) \\ &= \begin{pmatrix} \frac{1}{45} \\ \frac{1}{15} \\ \frac{1}{9} \end{pmatrix} + \begin{pmatrix} \frac{1}{15} \\ \frac{1}{15} \\ \frac{1}{6} \end{pmatrix} + \begin{pmatrix} \frac{1}{9} \\ \frac{1}{6} \\ \frac{2}{9} \end{pmatrix} = \begin{pmatrix} \frac{1}{5} \\ \frac{3}{10} \\ \frac{1}{2} \end{pmatrix} \end{aligned}$$

That is, the probability of Y is: $P(Y = y_1) = \frac{1}{5}, P(Y = y_2) = \frac{3}{10}$ and $P(Y = y_3) = \frac{1}{2}$.

2.2.8 Graphical Representation

If X and $Y_1, \ldots, Y_k$ are the sets of variables then the conditional probability of the form $P(X|Y_1, \ldots, Y_k)$ can be represented using a directed acyclic graph (DAG) as shown in the figure 2.3

In figure 2.3, the nodes (circles) represent the chance variables and the directed arcs between the nodes denotes the probability dependence relation. The chance node X is called *child* and the nodes $Y_1, \ldots, Y_k$ are called *parents* of X.

2.3 Probabilistic Network

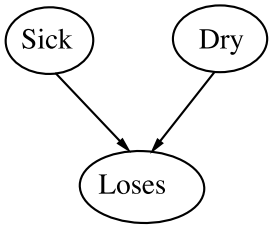
The performance of the intelligent systems [Rudas and Fodor, 2008] in real world requires the ability to make decisions under uncertainty based on the available evidence. Over the past decade, probabilistic networks [Kjærulff and Madsen, 2005] became a primary method for reasoning and acting under uncertainty [Binder et al., 1997]. Probabilistic network is a graphical language for representing the interactions between a set of variables. The nodes in the network denote the variables and the directed links (also called arcs or directed edges) between the nodes indicate the interactions (direct dependences) between the variables associated with the nodes. Any pair of non-adjacent nodes in the network represents conditional independence between the associated variables under particular circumstances, this can be easily captured from the network. Thus probabilistic network is a powerful intuitive language for encoding such dependence and independence relations, and therefore provides a perfect language for communicating and analysing the dependence and independence statements between variables of a problem-domain.

In the following section we discuss a very popular probabilistic network known as the Bayesian network.

2.3.1 Bayesian Network

The graph of the Bayesian network [Pearl, 1988] is a DAG consisting of nodes for chance variables (circles) and directed arcs between the nodes specify the probabilistic independence and dependence relations between the corresponding chance variables. Bayesian Network was first introduced into the field of artificial intelligence by [Pearl, 1982]. It is defined as a pair $\langle \mathbf{X}, \mathbf{P} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_k\}$ is the set of *chance variables* and $\mathbf{P}$ is the set of *conditional probability distributions*, containing one distribution, $P_i = P(X_i|pa(X_i))$, for each chance variable $X_i \in \mathbf{X}$, where $pa(X_i)$ is a set of parents of X_i in the graph.

The DAG of the Bayesian network encodes a joint probability distribution over $\mathbf{X}$, the conditional independence and dependence assumptions, and the properties of chain rule allow multiplicative factorization of the joint probability distribution over $\mathbf{X}$:



Sick contents	no	yes
P(S)	0.9	0.1

Dry contents	no	yes
P(D)	0.9	0.1

Probability of Lose (L)		P(L | D, S)	
D	S	no	yes
no	no	0.98	0.02
no	yes	0.1	0.9
yes	no	0.1	0.9
yes	yes	0.02	0.98

Figure 2.4: A simple Bayesian Network with three variables.

$$P(\mathbf{X}) = P(X_1, \ldots, X_k) = \prod_{i=1}^{k} P(X_i|pa(X_i)) \tag{2.7}$$

Each conditional probability $P(X_i|pa(X_i))$ can be represented as a table, where each entry in the table will specify the probability value of the variable X_i for the given joint assignment of the variables in $pa(X_i)$.

Solving a Bayesian network $\langle \mathbf{X}, \mathbf{P} \rangle$, with a set of random variables $\mathbf{X}$ having the conditional probabilities in $\mathbf{P}$ amounts to compute all posterior probabilities for a given evidence in the form of observations on the subset of variables Y in the model, i.e., computing $P(X|Y)$ for all $X \in \mathbf{X}$. If $Y = \emptyset$, i.e., no observations are made, then the task is to compute all prior probabilities, i.e., $P(X)$ for all $X \in \mathbf{X}$.

Example 2 » Bayesian Network

Figure 2.4 is a simple Bayesian Network taken from [Madsen, 1996], where it is observed that an apple tree has started losing its leaves. The owner of the tree wants to know, whether the tree is dry because of the drought or it is the sickness that causes it to lose its leaves. The qualitative knowledge of this situation is represented using a DAG, which consists of three nodes representing the chance variables Sick, Dry and Loses. All the variables have two states 'no' and 'yes'. The chance variable Sick tells us whether or not the tree is sick, whereas, the variable Dry tells us if the tree is dry or not and the variable Loses gives whether or not the tree is losing its leaves. The quantitative part of the Bayesian network is given by the probability distributions $P(D)$, $P(S)$ and $P(L|D,S)$ of the variables Dry, Sick and Loses respectively.

The variables Dry and Sick are independent variables since they do not have parents. Their probability distributions $P(D)$ *and* $P(S)$ *are marginal distributions, whereas, the prior distribution of the variable Loses is dependent on the states of the variables Dry and Sick. We can obtain its marginal distribution by marginalization of the variables Dry and Sick through the combination of the three probability distributions* $P(D)$*,* $P(S)$ *and* $P(L|D,S)$*, i.e., we finally obtain* $P(L = \textit{no}) = 0.82$ *and* $P(L = \textit{yes}) = 0.18$*.*

Since it is observed that the tree is losing its leaves, i.e., the variable Loses is in the state yes. Applying a similar procedure to compute the posterior distributions of Dry and Sick given Loses is in the state yes, we obtain, Dry: $P(D = \textit{no} \mid L = \textit{yes}) = 0.53$ *and* $P(D = \textit{yes} \mid L = \textit{yes}) = 0.47$ *and Sick:* $P(S = \textit{no} \mid L = \textit{yes}) = 0.51$ *and* $P(S = \textit{yes} \mid L = \textit{yes}) = 0.49$*. Hence, based on the given evidence, sickness is the most likely cause for the tree to lose its leaves.*

2.4 Decision and Utility Theory

A *decision* is a choice made by an individual, known as *decision maker*, among a set of available *alternatives* (or *outcomes*). In order to illustrate the distinction between decision and outcomes, think of an individual willing to invest his savings, debating between buying a property or buying some stocks on the market. In this situation, buying a property and buying some stocks on the market corresponds to outcomes, and a decision is to choose one of these outcomes. Howard [Howard, 2007] describes that people usually evaluate their decisions based on the outcomes, for instance, when someone's investment turned out to be a monetary loss then he usually says that he made a bad decision.

Decision theory is a theory of decisions which lies in the intersection of economics, psychology, statistics, game theory, operations research, artificial intelligence and many other [Hansson, 1994, Braziunas, 2011]. It is primarily based on the axioms of probability and utility [Howard, 2007], where probability theory provides a framework to represent the uncertain belief of the current state under which the decisions are to be taken, and utility theory presents a set of principles for consistency among beliefs, preferences and decisions [Henrion et al., 1991].

Boutilier et. al [Boutilier et al., 1997] describes that the specification of a decision making problem requires the following four components: (i) an estimate

of the current state or conditions under which the decisions are to be made; (ii) a set of decisions (or actions) that can be taken; (iii) a model of the system dynamics which describes the potential outcomes of any decision; and (iv) a set of preferences to qualify the relative goodness of particular outcomes. Although, extracting these four components can be difficult and time consuming, human decision analysts made enormous efforts to help elicit such information from the decision maker [Howard, 1984]. Examples of such works can be found in [Roy, 1989, Bouyssou, 1989, Pearl, 1996, Dyer et al., 1992, Kahneman and Tversky, 1979] and many others.

However, once a decision making problem has been identified completely then it can be classified according to various parameters (e.g., individual vs. group decision making), but the majority of the problems can be categorized into one of the following two main categories [Domshlak, 2002]:

(a) Decision making under certainty

(b) Decision making under uncertainty

A decision making problem falls into category (a) if the current state (or the condition) is known prior to making decisions, i.e., each action leads invariably to a specific outcome. For problems in (b) the current state is unknown at the time of making decisions, i.e., each action leads to one of a set of possible outcomes and each outcome occurs with known probability.

Multi-objective outcomes

In practice, often our decisions involve multiple objectives. Also, for each objective we associate an *attribute* (or *attributes*) which will indicate the degree to which the goal of the objective is met [Keeney, 1993]. These attributes may be measured in different measurement units. For instance, a shipping company wants to minimize the total duration of its routes in order to improve customer service and minimize the number of trucks used to reduce operating costs. In this situation, we may associate the objectives, “minimize the total duration of routes" and “minimize the trucks used” by attributes, “duration between cities" and “number of trucks in use”, respectively. Clearly, the attributes measurements are different. For simplicity, throughout this dissertation we assume that every objective is associated with only one attribute.

In a decision making problem assume that we have a set of p conflicting objectives $\{1, \ldots, p\}$ then the outcomes associated with each decision endowed with

p-dimensional structure. Under certainty, every decision maps to a point in p-dimensional space; whereas under uncertainty, a decision maps to a distribution over the points in that space [Braziunas, 2011].

Preference and Utility Modelling

The utility theory [Von Neumann and Morgenstern, 1945] is a foundation for both decision and game theories. Modelling preferences in the form of utility functions became a primary concern of decision theory. Roughly, utility function gives a unique (up to isomorphic linear transformations) description of the preferences [Domshlak, 2002]. Modelling preferences in a decision problem with several conflicting objectives is a very difficult task. To deal with this situation, decision theorists have developed multi-attribute (or multi-objective) utility theory (MAUT) as a way to model preferences [Keeney and Raiffa, 1993].

The most important stage of preference modelling is to identify the objectives or attributes of decision making problems. Once the objectives are identified then a utility function is assigned to each attribute. Each utility function then assesses the quantity of interest into a value scale whose increments are always equal in significance to the decision maker [Henrion et al., 1991]. The common approach in MAUT is, find a function that combines all attribute values into a single numerical value to compare outcomes. This function must be assessed with respect to the preferences of the decision maker. The utility model driven in this situation can be simple additive or more complex depending on how the attributes are judged, i.e., independently or there exists some dependencies among them. This and several other preference model forms are discussed in Chapter 3.

Utility functions also represent behaviour of an individual under uncertainty, for example, a person prefers a certain cash prize of $10 to a 60-percent chance of $20 and 40-percent chance of $0 [Keeney and Raiffa, 1993, Henrion et al., 1991]. Such preferences under uncertainty usually represented using simple lottery or gamble, which is a probability distribution over outcomes. For instance, if outcomes $o_1, \ldots, o_n$ are realized with probabilities $p_1, \ldots, p_n$, respectively, then it is represented as

$$l = \langle p_1, o_1; \ldots; p_n, o_n \rangle$$

Normally, outcomes with zero probability will not be included in the lottery notation. A variety of techniques have been developed to assist the decision mak-

ers in assessing their attitude towards risk by eliciting their relative preferences among lotteries, which are formed by varying probabilities and outcomes, such works can be found in [Keeney, 1993, Von Winterfeldt et al., 1986, Howard, 1967].

2.4.1 Decision Making under Certainty

We say that we are making decisions under *Certainty*, if each action is known to lead invariably to a particular outcome.

In the following section we discuss a very well-known mathematical model, multi-objective constraint optimization for representing and solving many decision making problems under certainty.

2.4.1.1 Multi-objective Constraint Optimization Problems

Constraint satisfaction and optimization are two important areas in the field of Artificial Intelligence [Gavanelli, 2002]. A *Constraint Satisfaction Problem* (CSP) is a framework for representing and solving many real-life decision making problems without uncertainty. A CSP consists of a set of decision variables which take finite values subject to a set of constraints or preferences expressed by the decision maker. The aim of solving a CSP is to find the assignments to the variables satisfying all the imposed constraints. A *Constraint Optimization Problem* (COP) is a CSP with an additional objective function that must be optimized. The goal of solving any COP is to determine the solutions that satisfy all the constraints and are not worse than any already existing solution. In a single-objective COP, it is easy to determine whether one solution is better than the other. As a result, we obtain a single *optimal* or *best* solution in the end. However, many real world applications, often involve two or more objective functions, which may conflict each other. Such problems are called *Multi-objective Constraint Optimization Problems* (MOCOPs). For instance, the shipping company problem discussed above is a MOCOP, because increasing the number of trucks reduces the duration of the routes, but it will increase the operation costs. Thus the objective functions are clearly conflicting.

Mathematically, a CSP is defined as a triple $\langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of (decision) variables with finite domains $\mathbf{D} = \{D_1, \ldots, D_n\}$ and $\mathbf{C}$ is the set of constraints or preferences. An assignment $\mathcal{A}$ is a function that maps some

of the variables to values in the corresponding domain, i.e., $\mathcal{A} : \{X_i\} \rightarrow D_i$. A complete value assignment $x = (X_1 = x_1, \ldots, X_n = x_n)$ to the variables is called a *solution*. A solution that satisfies all constraints is called a *feasible solution*.

Suppose that there are p (≥ 1) conflicting objectives and without loss of generality we assume maximization for all objectives, then a MOCOP is defined as $\langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ [Marinescu, 2011], where $\mathbf{F} = \{f_1, \ldots, f_r\}$ is the set of utility functions. A utility function $f_i \in \mathbf{F}$ is defined as $f_i : Y_i \rightarrow \mathbb{R}^p$, where $\mathbb{R}$ is set of real numbers and $Y_i \subseteq \mathbf{X}$ is called the *scope* of f_i. That is, for each configuration of variables in its scope utility functions return a vector in $\mathbb{R}^p$, which is called *utility vector*. The objective is to maximize the sum of all utility functions, i.e., maximize $\mathcal{F}(\mathbf{X}) = \sum_{i=1}^{r} f_i(Y_i)$ is the objective function. The solution to the problem is $x = (X_1 = x_1, \ldots, X_n = x_n)$, a complete value assignment to the decision variables that satisfies all constraints.

We say a solution x *weakly dominates* another solution x' if and only if each attribute values in the associated utility vector of x is better than or equal to the corresponding attribute values in associated utility vector of x'. In other words, if $\vec{u} = (u_1, \ldots, u_p)$ and $\vec{v} = (v_1, \ldots, v_p)$ are the utility vectors associated with the solutions x and x', then x weakly dominates x' if and only if $\vec{u} \geq \vec{v}$, i.e., $u_i \geq v_i$, for all $i = 1, \ldots, p$. This dominance relation is also known as *weak Pareto dominance relation* [Pareto, 1964], which is denoted by $\vec{u} \geq \vec{v}$, where $\geq$ is weak Pareto dominance relation. In particular, if $\vec{u} > \vec{v}$, i.e., $\vec{u} \geq \vec{v}$, and $\vec{u} \neq \vec{v}$ then we say that the solution x *dominates* or *Pareto dominates* the solution x'. The relation $>$ is called Pareto dominance relation.

Clearly, it is not always possible to determine between any pair of solutions if one is better than the other by using weak Pareto dominance relation (or Pareto dominance relation). Thus, the solution space of MOCOP is partially ordered with respect to $\geq$. By applying Pareto dominance relation, instead of a single optimal solution we get a set of solutions, known as *Pareto optimal solutions* or *non-dominated solutions*, with different trade-offs between the objectives. Although, there will be multiple Pareto optimal solutions, in practice, most probably the following steps will be taken for implementation:

(i) Generate the Pareto optimal solutions of the MOCOP.

(ii) Locate only those Pareto optimal solutions which satisfy preferences of the decision maker.

Since the Pareto optimal solutions are mathematically equal, the decision maker

requires to express his preferences to choose the *best* solutions of his interest. The main idea behind this is that the decision maker can be able to make better choices after knowing a variety of *good* solutions [Gavanelli, 2002]. However, this approach has several drawbacks that are discussed in detail in Chapters 3, 4 and 5.

2.4.2 Decision Making under Uncertainty

Uncertainty is the primary concern in making decisions and it is represented by probabilities. Suppose, in a decision making problem under uncertainty D is a decision variable with options $d_1, \ldots, d_m$, i.e., $dom(D) = \{d_1, \ldots, d_m\}$, H is a hypothesis with states $h_1, \ldots, h_n$, ϵ is a set of observations in the form of evidence and u is utility function. In order to evaluate the actions maximizing *expected utility* (EU) is probably the most common decision criteria adopted in literature. If $u(d_i, h_j)$ is the utility of an outcome (d_i, h_j) then the expected utility of performing an action d_i is defined as

$$EU(d_i) = \sum_j P(h_j \mid \epsilon) u(d_i, h_j),$$

where P is the probability distribution over H given the evidence ϵ. The utility function u represents the preferences of the decision maker on a numerical scale. Making optimal decision under uncertainty is to choose an option with highest expected utility, this is known as *maximum expected utility principle*. Choosing an action, which maximizes the expected utility is to find a decision option $\mathbf{d}$ such that

$$\mathbf{d} = \arg\max_{d_i \in dom(D)} EU(d_i)$$

In rest of this section we discuss two graph based formalisms *Influence Diagram* and *Decision Tree* for representing and solving sequential decision making problems under uncertainty.

2.4.2.1 Influence Diagrams

The success of Bayesian networks in representing probability information using a graphical model suggested extending it to represent both preferences and utility information of the decision maker in order to reason about expected utility

[Domshlak, 2002]. The idea of this extension leads to a graph based formalism known as *influence diagram*, which adds “utility nodes" or “value nodes" and “decision nodes" to Bayesian network, and represents the dependence of utility on conditions imposed in the model.

Influence diagrams were first introduced by [Howard and Matheson, 1984a], and it is defined as a quadruple $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of chance variables representing uncertainty, $\mathbf{D} = \{D_1, \ldots, D_m\}$ is a set of decision variables that are under control of the decision maker and specify the possible decisions are to be taken over the time, **P** is the set of conditional probability distributions one for each chance variable $X_i \in \mathbf{X}$, defined by $P_i = P(X_i|pa(X_i))$, where $P_i \in \mathbf{P}$ and $pa(X_i) \subseteq \mathbf{X} \cup \mathbf{D} \setminus \{X_i\}$, and $\mathbf{U} = \{U_1, \ldots, U_r\}$ is the set of *utility* (or *reward*) *functions* representing the preferences of the decision maker and are defined on the subsets of variables $Q = \{Q_1, \ldots, Q_r\}$, $Q_i \subseteq \mathbf{X} \cup \mathbf{D}$, called *scopes*.

Each decision $D_k \in \mathbf{D}$ in the model is dependent on some set of variables $pa(D_k) \subseteq \mathbf{X} \cup \mathbf{D} \setminus \{D_k\}$, called its parents whose states will be known at the time of making decision D_k and will have direct impact on it. It is assumed that a decision node and its parents are parents to all subsequent decisions in the model, this property is called *no-forgetting*. Like in Bayesian networks, the chance variables are divided into two categories, the ones observed during the evaluation of influence diagrams are called *observable* and *unobservable* are the ones that are never observed during the whole process [Pearl, 1988].

The graph of an influence diagram is a directed acyclic graph that contains nodes representing chance variables (circles), decision variables (rectangles) and utility functions (diamonds). An arc emerging from a node Y into a node representing random variable X_i denotes a probabilistic dependence relation of X_i on Y, while an arc from a variable X into a node representing a decision variable D_j denotes that the state of X is known when the decision D_j is to be made. An arc from a node Z into a node representing a utility function U_k denotes functional dependence of U_k on Z.

In an influence diagram, the decision variables are totally ordered, this is known as *regularity*. The regularity implies that there should be only one sequence, lets say $D_1, \ldots, D_k$, in which the decisions are to be made. The graph of the influence diagram contains a directed path from one decision variable to the next one in the decision sequence to impose the regularity. As a result of the total order on the decision variables, the chance variables are partitioned into

disjoint sets, denoted by $\mathbf{I}_o, \mathbf{I}_1, \ldots, \mathbf{I}_m$. The partition induces a strict partial order $\prec$ over $\mathbf{X} \cup \mathbf{D}$, given by

$$\mathbf{I}_0 \prec D_1 \prec \mathbf{I}_1 \prec \cdots \prec D_m \prec \mathbf{I}_m \text{ [Jensen et al., 1994]},$$

where $\mathbf{I}_o$ is the set of chance variables observed before first decision, $\mathbf{I}_i$ is the set of chance variables observed after making decision D_i and before making decision D_{i+1}, and $\mathbf{I}_m$ is the set of chance variables left unobserved or observed after the last decision D_m has been made. All these properties and steps are necessary in construction of the influence diagram otherwise the computed expected utilities will not be correct.

A *policy* (or *strategy*) for an influence diagram is defined as an ordered sequence $\Delta = (\delta_1, \ldots, \delta_m)$, where each δ_k, $k = 1, \ldots, m$ is called the *decision rule* for the decision variable $D_k \in \mathbf{D}$ and is a mapping $\delta_k : \Omega_{pa(D_k)} \rightarrow \Omega_{D_k}$, where $\Omega_{pa(D_k)}$ is cartesian product of the domains of the variables in $pa(D_k) \subseteq \mathbf{X} \cup \mathbf{D}$. A decision rule for the decision D_k determines the optimal action that the decision maker can take for all possible observations made prior to making decision D_k. We can see that a policy Δ assigns a value for each decision variable D_i which is dependent on its parents set $pa(D_i)$. For a policy Δ, the expected utility is given by the expression

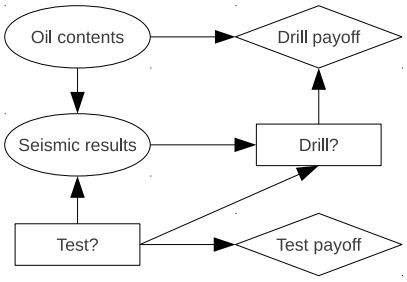
$$EU_\Delta = \sum_{\mathbf{X}} [(\prod_{i=1}^{n} P_i \times \sum_{j=1}^{r} U_j)]_\Delta$$

Solving an influence diagram amounts to finding an *optimal policy* that maximizes the expected utility, that is, need to find $\arg\max_\Delta EU_\Delta$. Therefore, the aim of the decision maker is to solve the following expression, known as *sum-max-sum rule*, which gives the maximum expected utility

$$\sum_{\mathbf{I}_0} \max_{D_1} \cdots \sum_{\mathbf{I}_{m-1}} \max_{D_m} \sum_{\mathbf{I}_m} \left(\prod_{i=1}^{n} P_i \times \sum_{j=1}^{r} U_j \right) \text{ [Jensen et al., 1994]}. \tag{2.8}$$

Example 3 » Standard Influence Diagram

Figure 2.5 shows the influence diagram of the oil wildcatter problem [Raiffa, 1993]. An oil wildcatter must decide either to drill *or* not to drill *for oil at a specific site. Before drilling, a* seismic test *could help determine the geological structure of the site. The test results can show a* closed *reflection pattern (indication of significant oil), an* open *pattern (indication of some oil), or a* diffuse



Oil contents	P(O)		
	dry	wet	soak
	0.5	0.3	0.2

Test payoff	$U_1(T)$
Test?	
yes	-10
no	0

Seismic results		P(S | O,T)			
Oil cnt.	Test?	closed	open	diffuse	notest
dry	yes	0.1	0.3	0.6	0
dry	no	0	0	0	1
wet	yes	0.3	0.4	0.3	0
wet	no	0	0	0	1
soak	yes	0.5	0.4	0.1	0
soak	no	0	0	0	1

Drill payoff		$U_2(O,D)$
Oil cnt.	Drill?	
dry	yes	-70
dry	no	0
wet	yes	50
wet	no	0
soak	yes	200
soak	no	0

Figure 2.5: A simple influence diagram with two decisions.

pattern (almost no hope of oil). The special value 'notest' *is used if no seismic test is performed. There are two decision variables,* T *(Test) and* D *(Drill), and two chance variables* S *(Seismic results) and* O *(Oil contents). The probabilistic knowledge consists of the conditional probability tables* $P(O)$ *and* $P(S|O,T)$*, while the utility function is the sum of* $U_1(T)$ *and* $U_2(O,D)$*.*

The following is the strict partial order $\prec$ *imposed by the total order of the decision variables:*

$$\{\} \prec \textit{Test} \prec \textit{Seismic} \prec \textit{Drill} \prec \textit{Oil}$$

Here we obtained the total ordering of the variables but in general this will not be the case. The empty set at the beginning of the ordering denotes that no observations are made before the decision on whether or not to Test. The decision maker will have Seismic test result after making the decision to Test the soil and before deciding whether or not to Drill the ground. Finally, Oil is observed after the decision Drilling has been made. The optimal policy is to perform the test and to drill only if the test results show an open or a closed pattern. The expected utility of this policy is 22.5.

2.4.2.2 Influence Diagrams with Multiple Objectives

The representation of influence diagram with only one objective is both unrealistic and inadequate for modelling most sequential decision making problems. Incorporating multiple objectives within influence diagrams has always been accomplished through a multi-attribute utility function as part of MAUT [Diehl and Haimes, 2004]. MAUT has provided a variety of techniques to include multi-objective utility functions in standard single objective influence diagrams.

Extending a single objective influence diagram to a multi-objective one, it only requires replacement of single attribute utility nodes (or utility functions) with new *multi-attribute utility nodes (or multi-attribute functions)*. In other words, in multi-objective influence diagrams, we allow more general notions of utility functions, which allow utility values to be vectors, known as utility vectors, in $\mathbb{R}^p$, where p is the number of objectives. Each element in the utility vector is a different measurable objective, for example, one based on monetary gain, and one based on risk to health. On the other hand, the graphical representation of multi-objective influence diagrams is identical to that of standard influence diagrams.

Formally, a multi-objective influence diagram with $p(\geq 1)$ objectives is defined as a quadruple $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of chance variables representing uncertainty, $\mathbf{D} = \{D_1, \ldots, D_m\}$ is a set of decision variables, $\mathbf{P}$ is the set of conditional probability distributions, and $\mathbf{U} = \{U_1, \ldots, U_r\}$ is the set of utility functions, which represents the preference of the decision maker over the p-objectives. Each utility function is defined as $U_j : \Omega_{Q_j} \rightarrow \mathbb{R}^p$, where Q_j is the scope of U_j. Since the utility vectors are now elements in $\mathbb{R}^p$, the Pareto (or Product) ordering (see Section 4.2), in general, is used to compare them. This indicates that we will no longer necessarily have a unique maximal values of expected utility, but a set of them. Different algorithms to solve multi-objective influence diagrams are discussed in Chapter 3 (Section 3.4) and we present our approach to solve these models in Chapter 5 (Section 5.2).

2.4.3 Decision Trees

The notion of decision trees was first presented by the pioneering work of [Von Neumann and Morgenstern, 1945] for game theory, and introduced by [Raiffa, 1993]. Decision tree is a graph based formalism for sequential deci-

sion making under uncertainty. A decision tree is defined as a triple $\langle \mathbf{X}, \mathbf{D}, \mathbf{P} \rangle$, where **X** is a set of chance variables representing the uncertain environment of the problem domain and **D** is a set of decision variables specifying the possible actions that the decision maker can take at specific point of time, though the decision variables are controlled by the decision maker but he does not know exactly the outcomes of the actions because the outcomes not only depends on his choices but also on some subset of unpredictable variables in **X**, and **P** is the set of *conditional probability distributions*, containing one distribution, $P_i = P(X_i|pa(X_i))$, for each chance variable $X_i \in \mathbf{X}$.

The graph of the decision tree consists of two types of nodes, decision nodes drawn as squares which represent the decision variables and chance nodes designed by circles for representing chance variables. The tree is constructed using top-down approach, i.e., starting from the root of the tree and going down towards its branches. It is assumed that, at each decision node in the tree, there are only a finite number of alternatives available for the decision maker. The decision maker then selects a course of action at each decision node, which is shown as branch emerging to the right side of the decision node in the tree. The chance node in the tree indicates that an event is expected at that point. The terminal branches of the tree determine the utility (or pay-off) associated with the series of actions and events that occur along each path.

The decision trees are usually solved based on the average-out and fold-back strategy, this implies, starting from the terminal node in each path, the task of averaging-out process is to compute the expected utility value. The task of fold-back process is to choose the decision alternative having the highest expected utility value. To solve a decision tree model, the decision maker starts with the leaf node in each path and move towards left until he reaches the root node by applying the process of averaging-out at chance nodes and fold-back at decision nodes.

Example 4 » Decision Tree

Figure 2.6 shows the decision tree of the problem described in Example 3. The decision variables test (T) and drill (D) are represented as square boxes and chance variables Seismic results (S) and oil contents (O) as circles. For simplicity, we assume that the chance variable Seismic results has only three outcomes diffuse, open and closed patterns, respectively. In figure, the paths emerging from the chance variable represent its possible outcomes and are assigned a probability value, whereas, the paths emerging from the decision nodes represent that the

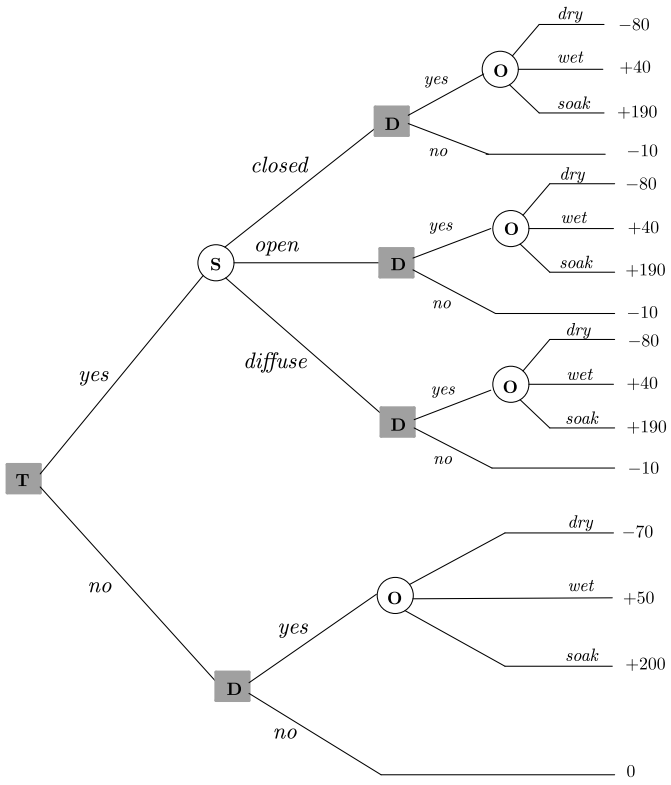


Figure 2.6: A decision tree of the oil wildcatter problem.

possible actions that can be taken. The terminal branches of the tree specifies the associated utility value of the actions and the events that occur along each path.

Despite the fact that the complex problem is better understood through its decision tree representation, decision tree has several disadvantages. Firstly, a small change in the input data cause large changes in the tree; in some cases such as changing variables and removing the duplicate information can lead to redrawing the tree. Secondly, its complexity; it is easy to compare the decision tree with any other decision making model but constructing it, specially with many branches is complex and time consuming.

2.5 Summary

This chapter presented the basic concepts of probability theory and introduced probabilistic networks for reasoning and decision making under uncertainty. A probabilistic network is an effective representation of dependence and independence relations among sets of random variables. The framework of Bayesian network was discussed to represent and process probabilistic knowledge. Since

they are the effective representation of probabilistic knowledge this makes them powerful tools for reasoning under uncertainty.

We have presented the concepts of the decision and utility theories and discussed decision making problems with multiple objectives under certainty and uncertainty. The framework of multi-objective constraint optimization was discussed for decision making without uncertainties. Whereas, influence diagrams and decision trees were introduced for representing and solving decision making under certainty. In particular, we discussed that influence diagrams are the extension of Bayesian networks augmented with decision variables, information precedence relations and preference relations. For the case of decisions with multiple objectives under uncertainty, we discussed how a standard influence diagram with single objectives can be extended to incorporate multiple objectives.

Chapter 3

Related Work

3.1 Introduction

This chapter gives the background and literature related to preference relations and their representation in multi-objective optimization settings. Different techniques for representing preferences in the literature are mainly classified as *cardinal representations* and *relational representations*, these two models will be discussed in detail and the related work on these models will be presented.

Since there is a huge literature related to solving multi-objective constraint optimization models, we only discuss the background and related work on the multi-objective AND/OR branch-and-bound algorithm (search-based algorithm), and variable elimination algorithm (inference-based algorithm) which is based on the framework of AND/OR search spaces [Dechter and Mateescu, 2004], which are the two main algorithms focused on solving multi-objective constraint optimization models in this dissertation.

Regarding solving influence diagram models, we discuss different algorithms including transformation-based method (which transform influence diagram models to other graphical models and apply the solution techniques related to the newly transformed model), variable elimination methods (such as the bucket elimination algorithm) and algorithms related to some other methods such as arc reversal and node removal.

Finally, we discuss methods approximating the Pareto ordering, here we only focus on ϵ-covering techniques presented in the literature.

The outline of the chapter is as follows: Introduction and background on pref-

erences, properties of preference relations and preference representations are presented in Section 4.4.1. Graph concepts, AND/OR search spaces and trees, and algorithms for solving multi-objective constraint optimization models are discussed in Section 3.3. Different methods and formalisms for solving influence diagram model, approximation techniques for approximating the Pareto optimal set, and methods for incorporating the preferences during influence diagram evaluation are discussed in Section 3.4. Concluding remarks are given in Section 3.5.

3.2 Preferences

Though the notion of preferences sounds simple, working with them can be a very difficult task. There are a number of reasons for this, the most obvious one is the cognitive difficulty of specifying preferences [Brafman and Domshlak, 2009]. However, this is often not an issue when there is only a single objective (or attribute) that the decision maker considers to be important. For instance, in a flight booking scenario, in which one wants to book a flight well in advance from Cork to Hyderabad, and all the user cares about is the price of the ticket. In this situation, finding the most suitable outcome among the set of all available outcomes might be an easy task as we only search for the cheapest flight. However, our preferences are typically quite complicated [Brafman and Domshlak, 2009], for instance, if the decision maker also cares about flight length (including changeover time during the journey), airline, flight class (e.g., business or economy) and so on. Once decisions involve multiple objectives, ordering even two outcomes can become cognitively difficult; the simple reason for this is, one needs to consider the decision maker's trade-offs and the interdependence of the objectives. As the number of objectives increases, additional computation and representational issues come into play.

3.2.1 Preference Relations and Properties

The preference relation is a crucial concept in modelling an individual's preferences in decision theory. Usually, a preference relation is a binary relation, which compares or eliminates the unwanted (or also called dominated) elements (with respect to the relation) from a set. In this section we give some introductory material on preference relations and their properties.

A *binary relation* on a set X is a set of ordered pairs (x, y) with $x, y \in X$ and a *universal binary relation* on X is a set $\{(x, y) : x, y \in X\}$. If $\mathcal{R}$ is any binary relation on X then it is a subset of universal binary relation. If any $(x, y) \in \mathcal{R}$ then it is denoted with $x\mathcal{R}y$ and $(x, y) \notin \mathcal{R}$ is denoted as $x \not\mathcal{R} y$.

If $\mathcal{R}$ is a binary relation on X then for any $x, y \in X$ exactly one of the following holds:

(i) $x\mathcal{R}y$ and $y\mathcal{R}x$

(ii) $x\mathcal{R}y$ and $y \not\mathcal{R}x$

(iii) $x \not\mathcal{R}y$ and $y\mathcal{R}x$

(iv) $x \not\mathcal{R}y$ and $y \not\mathcal{R}x$

When we deal with binary relations they usually are assumed to have certain properties. In the following definition we give some of the common properties.

Definition 1 » Properties of a binary relation

A binary relation $\mathcal{R}$ on X is

(i) reflexive *if $x\mathcal{R}x$ for all $x \in X$,*

(ii) irreflexive *if $x \not\mathcal{R}x$ for all $x \in X$,*

(iii) transitive *if $x\mathcal{R}y$ and $y\mathcal{R}z$ then $x\mathcal{R}z$ for all $x, y, z \in X$,*

(iv) negatively transitive *if $x \not\mathcal{R}y$ and $y \not\mathcal{R}z$ then $x \not\mathcal{R}z$ for all $x, y, z \in X$,*

(v) symmetric *if $x\mathcal{R}y$ then $y\mathcal{R}x$ for all $x, y \in X$,*

(vi) asymmetric *if $x\mathcal{R}y$ then $y \not\mathcal{R}x$ for all $x, y \in X$,*

(vii) antisymmetric *if $x\mathcal{R}y$ and $y\mathcal{R}x$ then $x = y$ for all $x, y \in X$,*

(viii) complete *if $x\mathcal{R}y$ or $y\mathcal{R}x$ (or possibly both) for all $x, y \in X$.*

A binary relation defined on a set, having or assumed to have certain properties, can define an ordering (or some systematic ranking) on the elements of that set. We distinguish between binary relations based on the properties they hold with special names. We now present few of them that we deal within this dissertation.

Definition 2 » Order relations

A binary relation $\mathcal{R}$ on X is

(i) a preorder, if it is reflexive and transitive,

(ii) a partial order, if it is preorder and antisymmetric,

(iii) a total preorder, if it is complete and transitive,

(iv) a total order, if it is a total preorder and antisymmetric,

(v) an equivalence relation, if it is reflexive, transitive and symmetric.

For instance, the relation $\geq$ on $\mathcal{A} = \{1, 2, 3, 4, 5\}$ is a total pre-order and a total order but it is not an equivalence relation because it is reflexive (i.e., $a \geq a$ for all $a \in \mathcal{A}$) and transitive (i.e., $a \geq b$ and $b \geq c$ then $a \geq c$ for all $a, b, c \in \mathcal{A}$) but not symmetric (eg., $3 \geq 2$ but $2 \ngeq 3$).

The relation $=$ on $\mathcal{A}$ is reflexive, transitive and symmetric, therefore it is an equivalence relation.

The equivalence relation is important, particularly when we deal preferences with multiple objectives (i.e., preferences that are expressed on more than one attribute). The main reason is that, it represents the notion of equally preferred outcomes. For instance, if $\mathcal{R}$ is an equivalence relation on a set X then $x\mathcal{R}y$ implies that x is equally preferred or equivalent to y with respect to $\mathcal{R}$. An equivalence relation on X partitions X into a class of non-empty disjoint subsets, called *equivalence classes*, such that any two elements of X are in the same class then they are equivalent. Formally, an equivalence class of X is defined as follows:

Definition 3 » Equivalence class

If X is a set and $\mathcal{R}$ is an equivalence relation on X then $\mathcal{R}[x] = \{y \in X : y\mathcal{R}x\}$ is called the equivalence class generated by x.

For any $x, y \in X$, it can be easily seen that for any equivalence relation $\mathcal{R}$ on X, $\mathcal{R}[x] = \mathcal{R}[y]$ if and only if $x\mathcal{R}y$ [Fishburn, 1970]. This means, any two equivalence classes are either identical or disjoint.

Indifference is another important relation that usually arises when one has to express his preferences on a set of multi-objective outcomes, which is defined as:

Definition 4 » Indifference relation

If $\mathcal{R}$ is a binary relation on a set X then the associated indifference or incompara-

bility relation $\sim$ on X is defined as

$$x \sim y \iff x \not\mathcal{R} y \text{ and } y \not\mathcal{R} y$$

For any x and y in X, the relation $x \sim y$ represents that an individual feels there is no real difference between x and y or uncertain about his preferences between x and y. For instance, in above flight booking example, an individual who prefers (€950, 3) (i.e., prefers a flight with ticket price €950 and three stopovers) over (€925, 4) might feel that (€950, 3) $\sim$ (€1050, 2).

Definition 5 » Additive independence

A binary relation $\mathcal{R}$ on $X \subseteq \mathbb{R}$ is

— *additively independent $\iff$ for all $x, y \in X$, $x\mathcal{R}y \Rightarrow (x + z)\mathcal{R}(y + z)$, for all $z \in X$.*

If additive independence holds, then we say that the relation $\mathcal{R}$ respects the operation addition $(+)$.

Definition 6 » Scale-invariance

A binary relation $\mathcal{R}$ on $X \subseteq \mathbb{R}$ holds

— *scale invariance $\iff$ for all $x, y \in X$, $x\mathcal{R}y \Rightarrow (\alpha.x)\mathcal{R}(\alpha.y)$, for any $\alpha \in \mathbb{R}$, $\alpha > 0$.*

Similarly, if scale-invariance holds, then we say that the relation $\mathcal{R}$ respects the operation multiplication $(\cdot)$.

We now discuss an important binary relation, namely *strict order*, which is defined as:

Definition 7 » Strict order

A binary relation $\mathcal{R}$ defined on X is a

— *strict order $\iff$ $\mathcal{R}$ on X is non-empty, transitive and irreflexive.*

For instance, again consider $\mathcal{A} = \{1, 2, 3, 4, 5\}$, then the relation $>$ (and $<$) on A is a strict order.

We denote a strict order that is defined on some set X with $\succ$. For any $x, y \in X$, $x \succ y$ represents that x is strictly preferred to y

Asymmetry and the antisymmetric property can be viewed as criteria of strict preference consistency, in the sense that if one strictly prefers x to y then he should not simultaneously prefer y to x. Transitivity is often assumed to be a reasonable criterion for an individual's preferences. The reason is, if x is preferred to y and y is preferred to z then it seems obvious that x should be preferred to z.

3.2.1.1 Convex Cones and Preferences

Convex cones (Chapter 2, page 10, [Boyd and Vandenberghe, 2004]) allow effective modelling of preferences. They represent the structure of preferences. In this section we discuss the characteristics and features of cones that are useful in our work.

Let $p > 0$ be an integer and $\vec{x}, \vec{y} \in \mathbb{R}^p$, where $\vec{x} = (x_1, \ldots, x_p)$ and $\vec{y} = (y_1, \ldots, y_p)$.

Definition 8 » Cones

A non-empty set $\mathcal{C} \subseteq \mathbb{R}^p$ is called a

(i) cone, if $\vec{x} \in \mathcal{C}$ implies that $\alpha\vec{x} \in \mathcal{C}$ for all $\alpha \in \mathbb{R}$, $\alpha \geq 0$.

(ii) convex cone, if (a) $\mathcal{C}$ is a cone ; (b) any $\vec{x}, \vec{y} \in \mathcal{C}$ implies that $\vec{x} + \vec{y} \in \mathcal{C}$.

(ii) positive convex cone, if (a) $\vec{0} \in \mathcal{C}$; (b) $\mathcal{C}$ is a convex cone (c) if any $\vec{x} \in \mathbb{R}^p$, $\vec{x} \geq \vec{0}$ then $\vec{x} \in \mathcal{C}$.

We can see that a cone is closed under positive scalar multiplication, whereas a convex cone is closed under positive scalar multiplication and closed under addition. The positive convex cone is closed under positive scalar multiplication, closed under addition and consists of all the elements of $\mathbb{R}^p$ whose components (or coordinates) are non-negative.

Definition 9 » Finitely generated cone

A convex cone $\mathcal{C} \subseteq \mathbb{R}^p$ is said to be generated or spanned by a finite set of vectors $\mathcal{G} = \{\vec{g_1}, \ldots, \vec{g_n}\}$, where $\vec{g_i} \in \mathbb{R}^p$ for all $i \in \{1, \ldots, n\}$ called generators, if it satisfies

$$\mathcal{C} = \{\vec{x} \in \mathbb{R}^p : \exists\ (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^n_+ \cup \{\vec{0}\} \text{ such that } \vec{x} = \sum_{i=1}^{n} \lambda_i \vec{g_i}\}$$

Where $\vec{0} = (0, \ldots, 0)$ is the zero vector in $\mathbb{R}^n$.

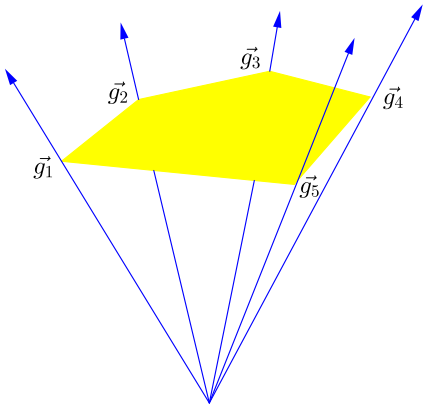


Figure 3.1: A finitely generated convex cone.

Figure 3.1 shows a finitely generated convex cone with five generators $\vec{g_1}, \vec{g_2}, \vec{g_3}, \vec{g_4}$ and $\vec{g_5}$.

We now introduce two special cones.

Definition 10 » Closed and pointed cone

A convex cone $\mathcal{C} \subseteq \mathbb{R}^p$ is called

(i) closed, if $\mathcal{C}$ coincides with its topological closure, where topological closure of $\mathcal{C}$ is the union of $\mathcal{C}$ and its boundary.

(ii) pointed, if (a) $\mathcal{C}$ is a closed; (b) any $\vec{x} \in \mathcal{C}$, $\vec{x} \neq \vec{0}$, implies that $-\vec{x} \notin \mathcal{C}$.

A pointed convex cone plays a key role in preferences, it gives the notion of consistency of the preferences.

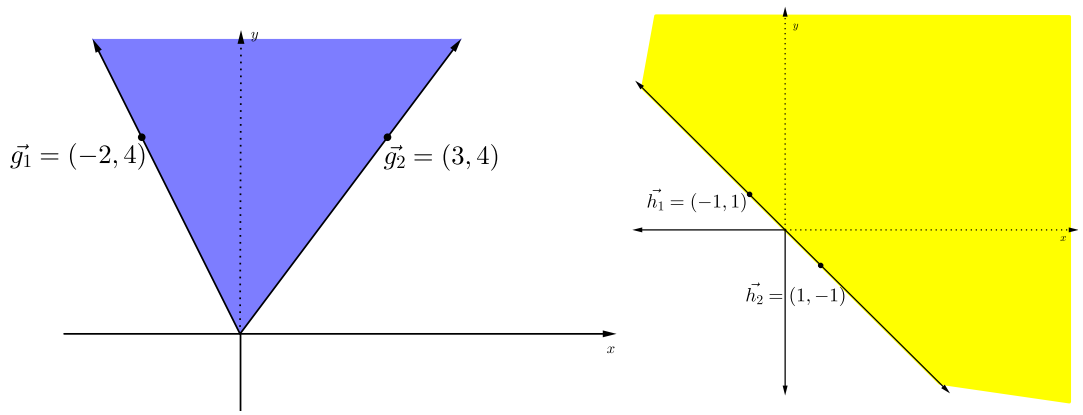


Figure 3.2: A pointed cone (left) and a non-pointed cone (right).

In figure 3.2, a convex cone generated by $\vec{g_1} = (-2, 4)$ and $\vec{g_2} = (3, 4)$ is pointed. Whereas a convex cone which is generated by $\vec{h_1} = (-1, 1)$ and $\vec{h_2} = (1, -1)$, is not pointed because it contains both $(-1, 1)$ and its additive inverse $(1, -1)$.

A pointed cone induces a partial order on $\mathbb{R}^p$, we next look into this.

Definition 11 » Pointed cone based partial orders

If $\mathcal{C}_{pc} \subseteq \mathbb{R}^p$ is a pointed convex cone, then the relation $\succcurlyeq_{pc}$ is a partial order on $\mathbb{R}^p$ (Chapter 2, page 104,[Dattorro, 2005]), defined by

$$\vec{x} \succcurlyeq_{pc} \vec{y} \iff \vec{x} - \vec{y} \in \mathcal{C}_{pc}.$$

Neither $\vec{x}$ or $\vec{y}$ is necessarily a member of $\mathcal{C}_{pc}$ for the above relations to hold. If $\vec{x} \succcurlyeq_{pc} \vec{y}$ then we may say that $\vec{x}$ *weakly dominates* $\vec{y}$ with respect to $\mathcal{C}_{pc}$. We say that two points $\vec{x}$ and $\vec{y}$ are *comparable* with respect to $\mathcal{C}_{pc}$ when $\vec{x} \succcurlyeq_{pc} \vec{y}$ or $\vec{y} \succcurlyeq_{pc} \vec{x}$.

We can easily see that the partial order $\succcurlyeq_{pc}$ satisfies the Scale-invariance and additive independence properties.

Proposition 1 » Scale-invariance and additivity of $\succcurlyeq_{pc}$

If $\mathcal{C}_{pc} \subseteq \mathbb{R}^p$ is a pointed convex cone and $\succcurlyeq_{pc}$ is a partial order induced by $\mathcal{C}_{pc}$ then

(i) for all $\vec{x}, \vec{y} \in \mathbb{R}^p$, if $\vec{x} \succcurlyeq_{pc} \vec{y}$ and $\lambda \in \mathbb{R}^+$, $\lambda > 0$ then $\lambda\vec{x} \succcurlyeq_{pc} \lambda\vec{y}$ (Scale-invariance).

(ii) for all $\vec{x}, \vec{y} \in \mathbb{R}^p$, if $\vec{x} \succcurlyeq_{pc} \vec{y}$ then $\vec{x} + \vec{z} \succcurlyeq_{pc} \vec{y} + \vec{z}$ for all $\vec{z} \in \mathbb{R}^p$ (Additive independence).

Proof. Let $\mathcal{C}_{pc} \subseteq \mathbb{R}^p$ be a pointed convex.

(i): Consider $\vec{x} \succcurlyeq_{pc} \vec{y}$; then we have $\vec{x} - \vec{y} \in \mathcal{C}_{pc}$. Since $\mathcal{C}_{pc}$ is closed under positive scalar multiplication, then for any real $\lambda > 0$, we have $\lambda(\vec{x} - \vec{y}) \in \mathcal{C}_{pc}$, i.e., $\lambda\vec{x} - \lambda\vec{y} \in \mathcal{C}_{pc}$, implying that $\lambda\vec{x} \succeq_{pc} \lambda\vec{y}$.

(ii): Suppose that $\vec{x} \succcurlyeq_{pc} \vec{y}$ then $\vec{x} - \vec{y} \in \mathcal{C}_{pc}$. We have $(\vec{x}+\vec{z})-(\vec{y}+\vec{z}) = \vec{x}-\vec{y} \in \mathcal{C}_{pc}$, which implies $(\vec{x} + \vec{z}) \succcurlyeq_{pc} (\vec{y} + \vec{z})$ for all $\vec{z} \in \mathbb{R}^p$.

□

We now define the *dual cone* of a cone, which is another important definition on which our one of the main contributions, i.e., representing preferences in

the matrix form in Chapter 4 rely.

Definition 12 » Dual cone

A subset $\mathcal{C}^* \subseteq \mathbb{R}^p$ is called the dual cone of a cone $\mathcal{C} \subseteq \mathbb{R}^p$, if it is given by

$$\mathcal{C}^* = \{\vec{y} \in \mathbb{R}^p : \vec{y} \cdot \vec{x} \geq 0 \text{ for all } \vec{x} \in \mathcal{C}\}$$

In the above definition, $\vec{y} \cdot \vec{x}$ is called the *dot product* between $\vec{y}$ and $\vec{x}$, which is defined by $\vec{y} \cdot \vec{x} = y_1 x_1 + \ldots + y_p x_p = \sum_{i=1}^{p} y_i x_i$. We can see from the definition that the dual cone $\mathcal{C}^*$ consists all the elements which make an acute angle (i.e., angle less than or equal to $90°$) with every element of the cone $\mathcal{C}$.

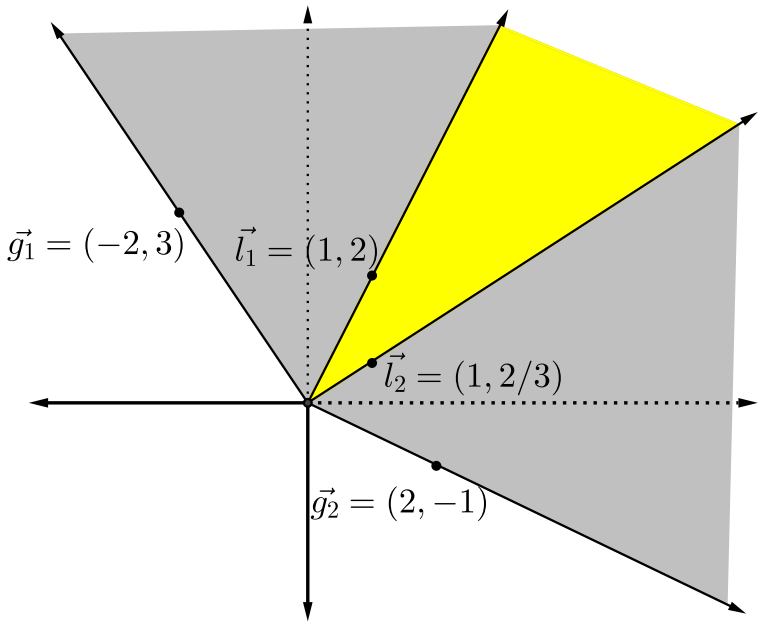


Figure 3.3: Pointed cone (grey) and its dual cone (yellow).

Figure 3.3 shows a convex cone (grey in colour) generated by $\vec{g_1} = (-2, 3)$ and $\vec{g_2} = (2, -1)$ and its dual cone (yellow in colour) generated by $\vec{l_1} = (1, 2)$ and $\vec{l_2} = (1, 2/3)$.

3.2.2 Preference Representation

Representation of preferences of an individual or a group, in a multi-objective setting, is a very difficult task in general and it is a major goal of decision analysis [Fishburn, 1970, Raiffa and Keeney, 1976, French, 1986, Keeney, 1993, Howard and Matheson, 2005]. One of the simplest techniques is, to

present the decision maker with a set of outcomes and ask him to compare pairwise. However, this method is not always feasible; clearly one can not apply this technique when we have a very large (possibly exponential) number of outcomes.

Many different formalisms have been proposed and studied to represent preferences. These formalisms can be classified into two different categories, namely *cardinal preference representation* and *relational preference representation* [Walsh, 2007]. In a cardinal preference representation, a numerical valuation is given to each outcome, whereas in a relational preference representation, the outcomes are ordered by means of a binary preference relation. In this dissertation, we focus on a relational representation of preferences and also give its relationship with the cardinal representation.

In the following sections we discuss different preference representation techniques. We denote the set of outcomes, on which a preference relation is defined with U. The notation $\vec{u} \succcurlyeq \vec{v}$ represents the outcome $\vec{u}$ is *weakly preferred to* $\vec{v}$; that is, outcome $\vec{u}$ is deemed to be at least as good as outcome $\vec{v}$ and $\vec{u} \succ \vec{v}$ means that $\vec{u}$ is *strictly preferred to* $\vec{v}$, i.e., $\vec{u} \succcurlyeq \vec{v}$ and $\vec{v} \not\succcurlyeq \vec{u}$.

3.2.2.1 Cardinal Preference Representation

This section presents different cardinal preference representations, which include utility-based representation and relative importance of criteria such as lexicographic models and weighted coefficient models.

Utility-based Preference Representation

In this form of preference representation the decision maker asked to assign a real number, also called the *utility*, to each outcome such that the preference relation $\succcurlyeq$ is defined as: $\vec{u} \succcurlyeq \vec{v}$ if and only if the utility associated with $\vec{u}$ is greater than or equal to the utility associated with $\vec{v}$. More formally, the decision maker requires to define a real-valued function, called *utility function* or *value function*, $f : U \to \mathbb{R}$ that represents his ability to prefer a particular outcome. The *fundamental theorem of utility* [Fishburn, 1970, Von Neumann and Morgenstern, 1945] states that the numbers $f(\vec{u})$ and $f(\vec{v})$ are assigned to the elements $\vec{u}$ and $\vec{v}$ in U in such a way that

$$\vec{u} \succcurlyeq \vec{v} \text{ if and only if } f(\vec{u}) \geq f(\vec{v})$$

The advantage of representing preferences in this way is that it is always possible to answer very common questions such as "which of the two outcomes is better?" or "what is the optimal outcome?". However, the main disadvantage with this form of representation is that, it is time consuming and tedious when one has to deal with the large number of outcomes with multiple objectives (or attributes) [Domshlak et al., 2011]. Also, with this representation the decision maker may find two outcomes incomparable. It can be possible that the decision maker expresses the utility function in a convenient and compact form and if such form exists then outcome-by-outcome quantification can be eliminated [Brafman and Domshlak, 2009]. If U is finite or countably large then the existence of such utility function f is guaranteed by a certain number of axioms [Von Neumann and Morgenstern, 1945].

A variety of methods have been emerged to construct the decision maker's utility function (for instance [Raiffa and Keeney, 1976, Saaty, 1988, Keeney, 1982]). Most of these methods are categorized based on constructing the utility function either explicitly or implicitly. In the first category of methods, a utility function is directly constructed from the decision maker's revealed preferences [Von Winterfeldt et al., 1986]. A good example of this method is given by Keeney and Raiffa in their classic book [Raiffa and Keeney, 1976]. In the second category of methods, the utility function is constructed on axiomatic basis of multi-attribute utility theory, examples of such methods include the UTA (*Utilités Additives*) method proposed by Jacquet-Largreze and Sisko [Jacquet-Lagreze and Siskos, 1982, Siskos et al., 2005] which requires the utility function associated with each objective to be piece-wise linear (i.e., utility function composed of some linear segments defined over an equal number of intervals), the AHP (Analytic Hierarchy Process) method developed by Saaty [Saaty, 2005] which assumes the underlying utility function is linear, and the MACBETH (Measuring Attractiveness by a Categorical based Evaluation Technique) method developed by Bana e Costa and Vansnick [Bana et al., 2005] which constructs the underlying utility function in an interactive procedure based on preference difference measurement. One feature of the AHP method is that, it informs the decision maker about his inconsistency while expressing his preferences, but it does not necessarily remove the inconsistencies.

Several interactive approaches have been developed to elicit the underlying utility function of which the exact form is not known to the decision maker. These methods are primarily based on the assumption that the utility function is either linear (defined by $f(\vec{u}) = p\vec{u} + q$, where $\vec{u} \in U$ and $p, q \in \mathbb{R}$) [Zionts

and Wallenius, 1980, Koksalan, 1984], or *quasiconcave* ($f(\lambda\vec{u} + (1-\lambda)\vec{v}) \leq \min(f(\vec{u}), f(\vec{v}))$, where $\lambda \in [0,1]$ and $\vec{u}, \vec{v} \in U$) [Korhonen et al., 1984, Murat Köksalan and Taner, 1992, Koksalan et al., 1984, Malakooti, 1989] or general *monotone* ($\vec{u} \geq \vec{v} \Rightarrow f(\vec{u}) \geq f(\vec{v})$, for $\vec{u}, \vec{v} \in U$) [Köksalan and Sagala, 1995]. In addition, Koksalan and Sagala [Murat Köksalan and Sagala, 1995] developed an approach to test whether the expressed preferences are consistent with the underlying utility function which is either linear, or quasi-concave, or general monotone. Despite the fact that these methods are simple and clear, they require the decision maker to be consistent with the underlying utility function while expressing his preferences.

Additive Independence

Given the fact that a utility function elicitation over a large number of outcomes is a cognitively difficult task [Domshlak et al., 2011, Braziunas and Boutilier, 2006], we need to make additional efforts concerning decision maker preferences so that the utility function can be compactly representable. One such attempt is structural assumption of preferences, known as *additive independence* [Raiffa and Keeney, 1976, Fishburn, 1970] which is commonly used in practice and applied to partition a set of attributes into a single attributes or arbitrary sets of attributes. When additive independence hold then it leads to a simple and compact additive utility representation.

A set of objectives is said to be preferentially independent of its complement, if the preference order over the outcomes with varying its objective values does not change when the objective values of its complement are fixed to any value. A set of objectives is mutually preferentially independent, if every subset of it, is preferentially independent of its complement set.

If mutual preferential independence holds, the utility function f, representing preferences of the decision maker can be broken down into single-objective *sub-utility functions*, namely $f_1, \ldots, f_p$ such that f can be written as:

$$f(\vec{u}) = \sum_{j=1}^{p} f_j(u_j) \tag{3.1}$$

where u_j is the value of j^{th} objective in $\vec{u}$.

Each sub-utility function f_j, often written as a product of *local value function* g_j and *scaling constants*, or *weights*, w_j, i.e., $f_j = w_j g_j(u_j)$, the utility function f

can be represented as:

$$f(\vec{u}) = \sum_{j=1}^{p} w_j g_j(u_j) \tag{3.2}$$

The additive representations, given in (3.1) and (3.2) are equivalent. In weighted representation, it is usually assumed that each scaling constant is non-negative and their sum is equal to unity, i.e., $\sum_{j=1}^{p} w_j = 1$, where $w_j \geq 0$. We discuss this model in detail in Section 3.2.2.1.1.

Generalized Additive Independence

Additive independence relations require very strong assumptions regarding the structure of the preferences. *Generalized additive independence* (GAI) models have gained popularity in the recent years because of their additional flexibility [Bacchus and Grove, 1995, Boutilier et al., 2001, Boutilier et al., 2003, Gonzales and Perny, 2004, Boutilier et al., 2005, Braziunas and Boutilier, 2012, Braziunas and Boutilier, 2007]. A GAI model is a generalization of an additive model and the conditions under which the GAI model defines an accurate representation of decision maker's utility function is given by Fishburn [Fishburn, 1967, Fishburn, 1970]. In GAI models, independence holds among certain subsets (known as factors) of objectives (or attributes), rather than individual objectives (or attributes). For instance, in a flight booking scenario, if an individual's preferences over flight length and flight class (e.g., economy or business) depends on price of the ticket then the GAI model groups such preferences into two GAI factors, namely F_1 and F_2, where $F_1 = \{\text{length}, \text{ price}\}$ and $F_2 = \{\text{class}, \text{ price}\}$. With these two factors, a GAI utility function representing such conditional preference can be defined as:

$$f_{GAI}(\vec{u}) = f_1(\text{length, price}) + f_2(\text{class, price})$$

where $\vec{u} = (\text{price, length, class})$. In contrast, the simple additive utility function defined as

$$f_A(\vec{u}) = f'_1(\text{price}) + f'_2(\text{length}) + f'_3(\text{class})$$

cannot adequately represent such conditional preferences of the decision maker.

3.2.2.1.1 Relative Importance of Criteria (or Objectives) In many multi-objective optimization problems, all objectives are not supposed to be equally important. In other words, the decision maker may consider only a subset of objectives which are more important among all other objectives. For instance, in a flight booking scenario, an individual may consider the price (P) of the ticket is relatively more important than the number of stopovers (S).

Generally, relative importance of objectives is modelled by using the very well-known preference models *lexicographic ordering* and *weighting coefficients*. In the following section we discuss them in detail.

Lexicographic Models

The concept of *lexicographic preference models* (or orderings) [Fishburn, 1974, Freuder et al., 2010] with multiple objectives implies that if one outcome has better objective values than another outcome on the most important objectives (or criteria) then it is considered to be better overall, regardless on how poor values it has on the rest of the objectives [Brafman et al., 2006]. For instance, again in a flight booking scenario, suppose that the decision maker is also concerned about the length (L) of the journey, and expresses his preferences as, P is more important than S and L, and S is more important than L. Suppose that an outcome $\vec{u}$ has better value for P and the worse values for less important objectives S and L in comparison with another outcome $\vec{v}$, then lexicographic preference model implies that $\vec{u}$ is still preferred over $\vec{v}$.

A *lexicographic order* is total, i.e., any arbitrary pair of outcomes are comparable under the lexicographic order. The notion of lexicographic ordering is often used in multi-objective optimization for representing qualitative preferences. The main disadvantage of using a lexicographic approach is that not all objective values might be considered while comparing the outcomes. However, [Yaman et al., 2011] argues that lexicographic preference models are simple and can be well-understood by the individuals.

Weighted Coefficient Models

This is the most common approach used in multi-objective optimization for modelling the decision maker preferences. In weighted coefficient models with p objectives $\{1, \ldots, p\}$, the weights $w_i, i = 1, \ldots p$ are assigned to the objectives. These weights are non-negative numbers which represent the relative importance of the objectives. The weights are independent from the measurement

units of the objectives scales [Brans and Mareschal, 2005]. Usually, the higher the weight, more important the objective is.

The dominance relation between vectors of $\mathbb{R}^p$ with respect to weighted coefficients model is defined as:

Definition 13 » Weighted dominance relation

For a given weights vector $\vec{w} = (w_1, \ldots, w_p)$, where w_i for all $i \in \{1, \ldots, p\}$ is the weight of the objective i, then a vector $\vec{u} = (u_1, \ldots, u_p)$ dominates another vector $\vec{v} = (v_1, \ldots, v_p)$, written as $\vec{u} \succcurlyeq_{\vec{w}} \vec{v}$ if

$$\vec{u} \succcurlyeq_{\vec{w}} \vec{v} \iff \sum_{i=1}^{p} w_i u_i \geq \sum_{i=1}^{p} w_i v_i$$

Assigning weights to the objectives is not straightforward because it involves priorities and perceptions of the decision maker. However, many researchers have developed systematic approaches for selecting weights. For instance, [Eckenrode, 1965] provides a method for choosing weights for each of the six objectives involved in designing a specific air defence and a general air defence systems, and selecting a personnel subsystem manager for a development program. In general, the approaches for selecting weights are categorized into two main classes known as *rating* and *ranking methods*. With rating methods, the decision maker assigns independent weights of relative importance to each objective. With ranking methods [Yoon and Hwang, 1995], the objective functions are ordered by importance in such a way that least important objective receives a weight of one, and the objectives that are more important receive integer weights which will then be incremented consistently.

The *paired comparison methods* rate objective functions by pairwise comparing them. For instance, [Saaty, 1977, Saaty, 2003, Saaty and Hu, 1998] provide an eigenvalues method for determining the weights, this involves $p(p-1)/2$, pairwise comparisons between objectives, where p is the number of objectives. The pairwise comparison of objectives is represented by using a matrix, which is known as the *judgement* or *comparison* matrix. The eigenvalues of the matrix are then used as weights for the objectives. While [Gennert and Yuille, 1988] suggests a method which disregards the preferences of the decision maker in assigning the weights to the objectives. This yields choosing the weights which maximize the final minimum value of the weighted sum function.

3.2.2.2 Relational Preference Representation

In relational preference representation, outcomes are ordered by means of a binary preference relation. Expressing preferences in relational form is perhaps much easier for the decision maker than the cardinal form [Walsh, 2007]. For instance, in a flight booking scenario, the decision maker may choose a cheapest flight among all the available options without assigning the weights. In addition, expressing the conditional preferences using the relational formalism is straightforward. For example, if there is somewhat expensive direct flight from origin to the destination, the decision maker concerns about change-overs during the journey so prefers "direct flight" to the "cheapest flight". Such conditional preferences are difficult to express using cardinal formalisms [Walsh, 2007].

Domshlak et al. [Domshlak et al., 2011] argues that for a decision maker providing binary preference relations is much easier to specify than value functions, because the qualitative comparison of pair of multi-objective outcomes (indicating one outcome is better than another one) needs much lesser burden than the quantitative assessment of single multi-objective outcomes. For instance, the simple qualitative statements (of preferences) of the form "I like the flight which costs €1000 with 2 stopovers more than the flight which costs €900 with 4 stopovers" or "I prefer a car like this, but white in color" indicate an ordering relation between outcomes [Brafman and Domshlak, 2009]. Such preference statements provide much more information because they implicitly encode many comparisons between multi-objective outcomes.

Although learning a binary preference relation that compare outcomes in pairwise manner is much simpler (mainly because suggesting that one alternative is better than the other one can be used directly instead of translating it into constraints on a value function), the prediction step may become more difficult [Domshlak et al., 2011]. The main reason for this is that the binary preference relation learned from the qualitative input preference statements usually does not define an ordering of the outcomes in a unique way. Therefore, it is then required to map a preference relation to a maximally consistent ranking. One such efficient techniques is simple voting, which is also known as Borda count procedure used in social chose theory, often gives good approximations [Hüllermeier and Fürnkranz, 2010]. Another approach for learning ranking functions is based on the model assumptions, that is, assumptions about the structure of the preference relations. Such approach is less generic than the previous one,

as it only depends on the assumptions made on the preference structure.

3.2.2.3 Representation using Convex Cones

Convex cones allow more effective modelling of preferences from the trade-offs perspective [Hunt et al., 2010]. In this thesis we use convex cones to model preferences of the decision maker, revealed in the form of pairwise comparison between utility vectors. In this section we look at the work related to ours and also present some literature on modelling preferences using convex cones.

The use of convex cones to model a decision maker's preferences in multi-objective decision making became popular with the seminal work of Yu [Yu, 1974]. To locate the set of undominated solutions for a given $Y \subset \mathbb{R}^p$, Yu presented two different mathematical methods, based on the concepts of *polar cones* (a polar cone W_* of a $W \subset \mathbb{R}^p$ is defined as, $W_* = \{\vec{v} \in \mathbb{R}^p \mid \vec{v} \cdot \vec{u} \leq 0 \text{ for all } \vec{u} \in S\}$) and the cone convexity of Y. This work is mainly based on the study of 'cone extreme points', where extreme points of a convex cone are the ones which are not dominated by any other element in it.

Berman and Naumov [Berman et al., 1993] uses interval trade-offs and construct a matrix of a cone, which represents the decision maker's preferences. Noghin [Noghin, 1997] makes use of objective weights as relative importance of criteria and construct the convex hull of the Pareto cone, this idea extended further in [Noghin and Tolstykh, 2000, Noghin, 2001] to construct an estimate of the undominated solution set. Wiecek [Wiecek, 2007] extended the works of Berman and Naumov, and Noghin to model relative importance of objectives using convex cones, where two cone-based models were presented to model the preferences. In the first model, one objective is designated as less important while all the others are more important. In the second model, a group (more than one) of objectives are classified as less important while all the others are considered more important. These models were represented by use of matrices. In addition, the relation between these models were also studied.

Dehnokhalaji et al. [Dehnokhalaji et al., 2011] developed a framework to model preferences, which revealed in the form of pairwise comparison between utility vectors, with multiple convex cones. Each cone represents a subset of preference information. This work is based on the assumption that the decision maker's value function is unknown, but it has a quasi-concave form. Engau [Engau, 2008] assumes scale-invariance and additive properties to the decision

maker's preference relation, in addition they assume ideal-symmetric property, which is given as, if $Y \subset \mathbb{R}^p$ and $\vec{y} \in Y$, any equal length vectors $\vec{r}, \vec{t} \in \mathbb{R}^p$ with $\vec{y} \cdot \vec{r} = \vec{y} \cdot \vec{t}$ then $\vec{y} - \vec{r}$ is preferred over $\vec{y}$ if and only if $\vec{y} - \vec{t}$ is preferred over $\vec{y}$. The ideal-symmetric convex cones are then used to model the decision maker's preferences.

3.3 Multi-objective Constraint Optimization

In constraint satisfaction problems (CSPs) the aim is to find the assignments for the variables that satisfy all the constraints [Dechter, 2003]. Since multi-objective constraint optimization problems (MOCOPs) consist of a set objective functions, the primary goal of solving them is to search for the set of Pareto optimal solutions, known as *Pareto frontier* or *efficient frontier*, providing the decision maker with a set of outcomes to choose from [Lin, 2010]. Most complete algorithms for solving MOCOPs typically fall within one of the following two categories [Marinescu, 2009]:

(i) Inference-based algorithms

(ii) Search-based algorithms

Inference-based algorithms (e.g., variable elimination) are good at making use of the structural information encoded in the problem. In addition, these algorithms perform a sequence of transformations that reduce the problem size, while preserving the solution space of the problem. The time and space complexity of these algorithms is exponential in a topological parameter called *tree-width* [Mateescu and Dechter, 2012]. The main draw back of these algorithms is that, due to their high space requirements these are impractical particularly for problems with larger tree-widths [Marinescu, 2008, Marinescu, 2009].

Search-based algorithms (e.g., depth-first Branch and Bound) transform MOCOPs into a set of sub-problems by selecting a variable and considering the assignment of each of its domain values. The sub-problems are then solved in sequence applying recursively the same transformation rule [Marinescu, 2009]. These algorithms are not sensitive to the problem structure, consequently, they do not capture the in-dependencies represented by the structural information encoded in the problem. For this reason, search-based algorithms may not be as effective as inference-based algorithms in using this information. These have a time complexity which is exponential in the number of variables, but can

operate in linear memory [Marinescu, 2008].

The introduction of AND/OR search space [Dechter and Mateescu, 2007a] for graphical models has changed the situation for the past few years. In particular, the framework of AND/OR search space is sensitive to the in-dependencies in the model, which often results in exponential reduction of complexities. In addition, the search is guided by a *pseudo tree* [Freuder and Quinn, 1985, Bayardo and Miranker, 1995] that captures the in-dependencies in the underlying graphical model. As a result, the search space is exponential in the depth of the pseudo tree, rather than in the number of variables.

The following section defines the basic graph concepts and AND/OR search space for MOCOPs.

3.3.1 Graph Concepts

A *directed graph* $\mathcal{G} = \{\mathbf{V}, \mathbf{E}\}$, where $\mathbf{V} = \{X_1, \ldots, X_n\}$ is a set of *vertices* or *nodes*, and $\mathbf{E} = \{(X_i, X_j) \mid X_i, X_j \in V\}$ is a set of directed *edges* or *arcs*. The edge (X_i, X_j) implies that X_i points to X_j, where X_i is called *parent* and X_j is named its *child*. For each node X_i, the set of parents of X_i in $\mathcal{G}$ is denoted with $pa(X_i)$, while the set of child nodes of X_i is denoted with $ch(X_i)$. The *degree* of a vertex is defined as the number of incident arcs to it. The *family* of node X_i is denoted with $fam(X_i)$ and it is defined as $fam(X_i) = \{X_i\} \cup pa(X_i)$. If $\mathcal{G}$ has no directed cycles then we say that $\mathcal{G}$ is *acyclic* graph. An *undirected* graph $\mathcal{G}'$ is similar to a directed graph $\mathcal{G}$, but the only difference is that, arcs are undirected, i.e., $(X_i, X_j) \in \mathbf{E} \iff (X_j, X_i) \in \mathbf{E}$.

An *ordered graph* $\mathcal{G}_o = (\mathcal{G}, o)$ is an undirected graph, where $o = (X_1, \ldots, X_n)$ is an *ordering* of the nodes. The *width of the node* X_i is defined as the number of neighbours of X_i that precede it in the ordering. The *width of the ordering* o is the *maximum width* over all nodes. The *induced width* of $\mathcal{G}_o$ is denoted as $w^*(o)$ and it is the width of the *induced ordered graph* which is obtained as follows: nodes are processed from last to first; when node X_i is processed then all its neighbours are connected. The induced width of a graph $\mathcal{G}$ is denoted by w^* and it is the *minimal induced width* over all its orderings.

A *hyper-graph* is a pair $\mathcal{G}_h = (\mathbf{X}, \mathbf{S})$, where $\mathbf{X}$ is a set of variables and $\mathbf{S} = \{S_1, \ldots, S_t\}$ is a set of subsets of $\mathbf{X}$. Elements of $\mathbf{S}$ are called *hyper-edges* of $\mathcal{G}_h$. Let $T = (\mathbf{V}, \mathbf{E})$ be a tree, where $\mathbf{V}$ is a set of nodes, also called *clusters* and $\mathbf{E}$

is the set of edges. Then T is called a *tree decomposition* of $\mathcal{G}_h$ together with a labelling function $\mathcal{L}$, which associates with each node $v \in \mathbf{V}$ a set $\mathcal{L}(v) \subseteq \mathbf{X}$, if the following properties hold:

(1) For each $S_i \in \mathbf{S}$ $\exists v \in \mathbf{V}$ such that $S_i \subseteq \mathcal{L}(v)$;

(2) For each $X_i \in \mathbf{X}$, the set $\{v \in \mathbf{V} \mid X_i \in \mathcal{L}(v)\}$ induces a connected sub-tree of T. This is also known as *running intersection property*.

The *tree width* of a tree decomposition of a hyper-graph is defined as the size of the largest cluster minus 1, i.e., width $= \max_v |\mathcal{L}(v) - 1|$. Whereas, the *tree-width* of a hyper-graph is defined as the minimum width among all its tree decompositions.

The set of maximal cliques or clusters in the induced graph provide a tree-decomposition of the graph [Mateescu and Dechter, 2005]. The tree-width is the maximal number of variables in a cluster of an optimal cluster-tree decomposition of the graph [Arnborg, 1985]. It is well known that the induced width of any graph is identical to its tree-width [Dechter and Pearl, 1989].

3.3.2 AND/OR Search Spaces and Trees

The AND/OR search space for solving MOCOPs has recently been introduced in [Marinescu, 2009]; its important characteristic includes exploiting the independencies between the variables during the search and captures the underlying graphical model. Let $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ be a MOCOP instance, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of (decision) variables with finite domains $\mathbf{D} = \{D_1, \ldots, D_n\}$, $\mathbf{C}$ is the set of constraints and $\mathbf{F} = \{f_1, \ldots, f_r\}$ is the set of utility functions. Each utility function f_i is defined as $f_i : Y_i \rightarrow \mathbb{R}^p$, where $\mathbb{R}$ is the set of real numbers and $Y_i \subseteq \mathbf{X}$, known as its scope. The objective function is $\mathcal{F}(\mathbf{X}) = \sum_{i=1}^{r} f_i(Y_i)$. Then we define its *primal graph* and *pseudo tree* as follows:

Definition 14 » primal graph

A primal graph of $\mathcal{M}$ is an undirected graph $G = (V, E)$ whose vertices, V, are the variables in $\mathcal{M}$ and an edge in E connects any two variables that appear in the scope of the same utility function.

Definition 15 » pseudo tree

A pseudo tree *of a primal graph $G = (V, E)$ is a directed rooted tree $\mathcal{T} = (V, E')$,*

such that every arc of G not included in E' is a back-arc in $\mathcal{T}$, namely it connects a node in $\mathcal{T}$ to an ancestor in $\mathcal{T}$.

Example 5 » An example of MOCOP

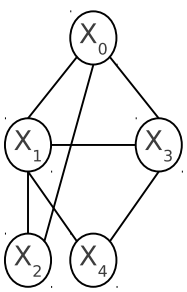
Figure 3.4 shows a MOCOP instance with 5 bi-valued decision variables $\{X_0, X_1, X_2, X_3, X_4\}$ and 3 ternary utility functions $f_1(X_0, X_1, X_2)$, $f_2(X_0, X_1, X_3)$, and $f_3(X_1, X_3, X_4)$ defined in Figure 3.4(a). The objective is to maximize the sum of all utility functions, i.e., objective is to maximize $\mathcal{F}(X_0, X_1, X_2, X_3) = f_1(X_0, X_1, X_2) + f_2(X_0, X_1, X_3) + f_3(X_1, X_3, X_4)$. Figures 3.4(b) and 3.4(c) represents its corresponding primal graph and pseudo tree, respectively.

X_0	X_1	X_2	f_1
0	0	0	(5,3)
0	0	1	(2,8)
0	1	0	(3,6)
0	1	1	(2,9)
1	0	0	(2,1)
1	0	1	(3,0)
1	1	0	(1,6)
1	1	1	(2,4)

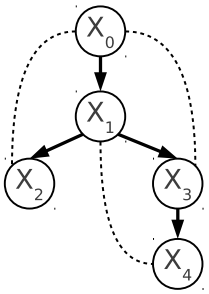
X_0	X_1	X_3	f_2
0	0	0	(1,5)
0	0	1	(4,2)
0	1	0	(0,6)
0	1	1	(2,4)
1	0	0	(6,1)
1	0	1	(5,5)
1	1	0	(6,4)
1	1	1	(5,2)

X_1	X_3	X_4	f_3
0	0	0	(1,4)
0	0	1	(3,2)
0	1	0	(5,1)
0	1	1	(4,3)
1	0	0	(1,9)
1	0	1	(3,2)
1	1	0	(5,6)
1	1	1	(4,8)

(a) utility functions



(b) primal graph



(c) pseudo tree

Figure 3.4: A MOCOP instance with 2 objectives.

An AND/OR state space representation of a MOCOP problem can be defined by a quadruple $\langle S, O, S_g, s_o \rangle$, where:

(1) S is a set of *states* which can be either OR or AND states;

(2) O is a set of *operators* which can be either OR or AND operators;

(3) $S_g \subseteq S$ is a set of goal states;

(4) $s_o \in S$ is a start node.

The OR states represent alternate ways for solving the problem, whereas, the AND states represent the problem decomposition into sub-problems which need to be solved. An OR node transforms an OR state into another state, and an AND state transforms an AND state into a set of states.

The AND/OR state space representation of the problem includes an explicit AND/OR *search tree* $S_{\mathcal{T}}(\mathcal{M})$, where each node represents a state (or a variable) and child nodes are obtained by applicable AND or OR operators. The search

graph includes start node s_o, known as its *root node*. The nodes having no children are known as *terminal nodes*, and are labelled as SOLVED or UNSOLVED.

The structure of an AND/OR search tree $S_{\mathcal{T}}(\mathcal{M})$ is given by the underlying pseudo tree $\mathcal{T}$. It has alternating levels of OR and AND nodes. The OR nodes correspond to the variables in $\mathcal{M}$ and are labelled X_i, and AND nodes are corresponding to the value assignments to the variables and are labelled $\langle X_i, x_i \rangle$ or $\langle x_i \rangle$. The root of $S_{\mathcal{T}}(\mathcal{M})$ is an OR node labelled with the root of $\mathcal{T}$. The children of an OR node X_i are AND nodes labelled with the assignments $\langle X_i, x_i \rangle$ which are consistent with the assignments along the path from the root. The children of an AND node $\langle X_i, x_i \rangle$ are OR nodes labelled with the children of variable X_i in $\mathcal{T}$.

The *solution tree* of the search tree $S_{\mathcal{T}}(\mathcal{M})$ is a AND/OR sub-tree T which:

(1) contains the root node s_o of $S_{\mathcal{T}}(\mathcal{M})$;

(2) if $n \in T$ is an OR node then it contains exactly one of its child nodes in $S_{\mathcal{T}}(\mathcal{M})$, and if $n \in T$ is an AND node then it contains all its children in T;

(3) all its terminal nodes are labelled SOLVED.

An AND/OR search tree $S_{\mathcal{T}}(\mathcal{M})$ can have a utility associated with each arc, and the utility of a solution sub-tree is a function (e.g., additive utility) of the arcs included in the solution sub-tree. The goal is to search for a solution sub-tree with optimal (maximum) utility.

Example 6 » AND/OR search tree for MOCOP

Figure 3.5 shows the AND/OR search tree corresponding to the MOCOP instance given in Example 5. From the figure we can see that the AND/OR search tree has alternating levels of OR and AND nodes. A solution tree of the problem is highlighted (in orange colour). Once the variables X_0 and X_1 are instantiated, the search space below the AND node labelled $\langle X_1, 0 \rangle$ decomposes the problem into two independent sub-problems, the first one is rooted at node labelled X_2 and the second is rooted at node labelled X_3.

3.3.2.1 Weighted AND/OR Search Tree

The arcs in $S_{\mathcal{T}}(\mathcal{M})$ from OR nodes X_i's to AND nodes $\langle X_i, x_i \rangle$'s are assigned *weights* (which are vectors in $\mathbb{R}^p$) derived from the multi-objective utility functions in **F**. Once all the weights are assigned, $S_{\mathcal{T}}(\mathcal{M})$ is then called a weighted

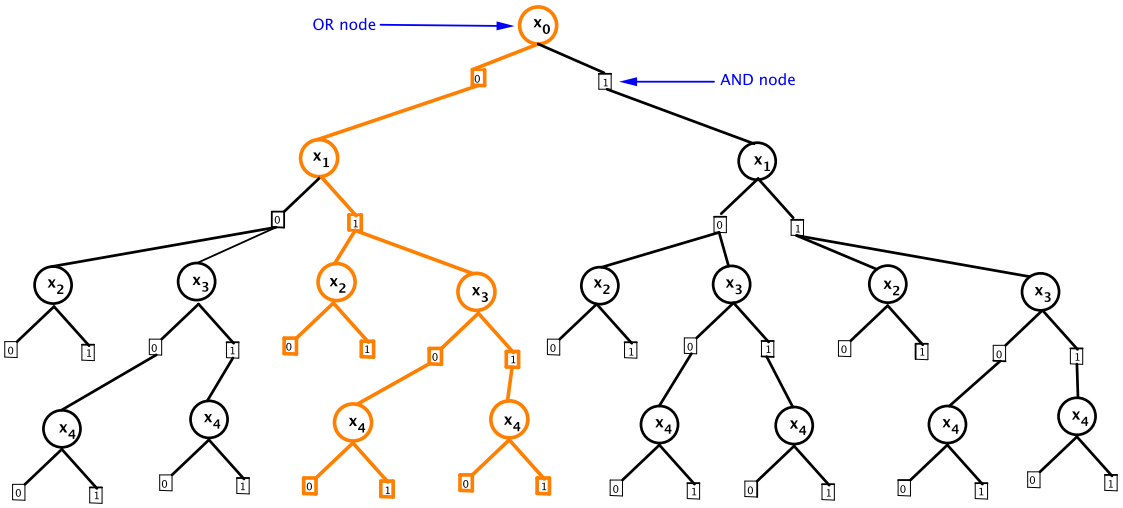


Figure 3.5: AND/OR search space.

AND/OR search tree.

Since we deal with sets of multi-objective utility values, in the following we give some definitions on utility sets that will be used in rest of the thesis.

Definition 16 » Sum of two utility sets

If $U, V \subset \mathbb{R}^p$ are two sets of utility values then we define their sum as:

$$U + V = \{\vec{u} + \vec{v} \mid \vec{u} \in U, \vec{v} \in U\}.$$

The dominance relation between sets of utility vectors is defined as:

Definition 17 » Dominance relation for sets of utility values

If U and V are two subsets of $\mathbb{R}^p$ then we say that U dominates *V with respect to $\succcurlyeq$ if and only if $\forall v \in V \; \exists u \in U$ such that $u \succcurlyeq v$. If U* dominates *V, then we denote it as $U \succcurlyeq V$.*

Definition 18 » Maximal or Pareto Set

Let $\succcurlyeq$ be a partial order and suppose that $U \subset \mathbb{R}^p$ is the set of utility vectors, then the maximal set *of U with respect to $\succcurlyeq$ is the set of undominated elements of U, i.e., the* maximal set *of U is defined by $\max_{\succcurlyeq}(U) = \{\vec{u} \in U \mid \nexists \vec{v} \in U, \vec{v} \succ \vec{u}\}$. If $\succcurlyeq$ is the weak Pareto order $\geq$ then we call $max_{\succcurlyeq}(U)$ as the Pareto set or Pareto frontier.*

Definition 19 » Arc weight

The weight $w(n, n')$ of the arc from the OR node n labelled X_i to the AND node

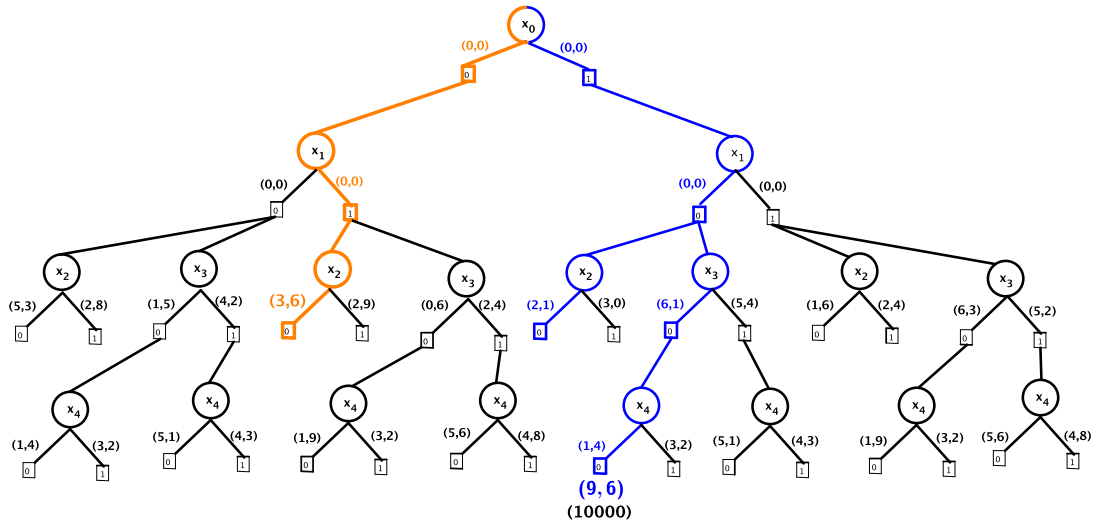


Figure 3.6: Weighted AND/OR search tree.

n' labelled $\langle X_i, x_i \rangle$ is a utility vector defined as the sum of all the multi-objective utility functions whose scope includes X_i and is fully assigned along the path from the root to $\langle X_i, x_i \rangle$, evaluated at the values along the path.

Each node n in the weighted search tree is associated with a *value* $v(n)$ (is a subset of $\mathbb{R}^p$) which is computed recursively as shown in [Marinescu, 2009]. The value $v(n)$ represents the optimal solution associated with the conditioned sub-problem below n.

Definition 20 » Node value

The value $v(n)$ of a node n in a weighted AND/OR search tree of a MOCOP instance $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F} \rangle$ with p objectives is defined recursively as follows:

(1) $v(n) = \{\vec{0}\}$, *if* $n = \langle X_i, x_i \rangle$ *is a terminal AND node;*

(2) $v(n) = \sum_{n' \in succ(n)} v(n')$, *if* $n = \langle X_i, x_i \rangle$ *is an internal AND node, where* $succ(n)$ *are the children of* n*;*

(3) $v(n) = \max_{\preccurlyeq} \{w(n, n') + v(n') \mid n' \in succ(n)\}$, *if* $n = X_i$ *is an OR node.*

From the above definition, the value $v(n)$ of a node in an AND/OR search tree $S_{\mathcal{T}}$ is the set of utility vectors representing the optimal solutions of the sub-problem rooted at n conditioned on the variable assignment along the path from the root to n [Marinescu, 2009]. In particular, if n is the root node then $v(n)$ is the efficient frontier of the initial problem.

Example 7 » Weighted AND/OR search tree

Figure 3.6 shows the weighted AND/OR search tree of the MOCOP instance given

in Example 5, relative to the pseudo tree given in Figure 3.4(c). The utility vectors displayed on the OR-to-AND arcs are the weights corresponding to the input utility functions. For instance, the weight of the arc from OR node X_2 *to its AND child* $\langle X_2, 1\rangle$ *is* $(3, 6)$ *given along the path* $(X_0, \langle X_0, 0\rangle, X_1, \langle X_1, 1\rangle, X_2, \langle X_2, 0\rangle)$*. An optimal solution tree corresponding to the assignment* $(X_0 = 1, X_1 = 0, X_2 = 1, X_3 = 0, X_4 = 0)$ *with utility vector* $(9, 6)$*, which is obtained by summing the utility values associated on the arcs from the root node* X_0 *all the way to terminal AND node labelled* $\langle X_4, 0\rangle$*, i.e.,* $(0, 0) + (0, 0) + (2, 1) + (6, 1) + (1, 4) = (9, 6)$*, is highlighted (blue colour).*

The value of the OR node X_2 *along the path* $(X_0, \langle X_0, 1\rangle, X_1, \langle X_1, 1\rangle, X_2)$ *is computed as follows:*

$$\begin{aligned} v(X_2) &= \max_{\geq}\{w(X_2, \langle W_2, 0\rangle) + v(\langle W_2, 0\rangle), w(X_2, \langle W_2, 1\rangle + v(\langle W_2, 1\rangle)\} \\ &= \max_{\geq}\{(1, 6) + (0, 0), (2, 4) + (0, 0)\} \\ &= \max_{\geq}\{(1, 6), (2, 4)\} \\ &= \{(1, 6), (2, 4)\} \end{aligned}$$

where, $\geq$ *is the weak Pareto ordering. Notice that* $v(\langle W_2, 0\rangle) = v(\langle W_2, 1\rangle) = (0, 0)$ *because the nodes* $\langle W_2, 0\rangle$ *and* $\langle W_2, 1\rangle$ *are the terminal AND nodes for the independent sub-problem rooted at the node labelled* X_2*.*

3.3.3 Multi-objective AND/OR Branch-and-Bound

Rollon and Larrosa [Rollon and Larrosa, 2007, Rollón and Larrosa, 2006a] showed that multi-objective problems can be solved using a depth-first Branch-and-Bound schema, which traverses the traditional OR search tree in depth-first manner and results in the set of optimal solutions of the initial problem. During the search, the algorithm Multi-Objective Branch-and-Bound (MOBB) records the best solutions found so far, which is the underestimate of the optimal solutions of the initial problem [Marinescu, 2009, Rollon and Larrosa, 2007]. At each visited node during the search, MOBB computes an upper bound (which is basically a set of utility vectors in $\mathbb{R}^p$) of the sub-problem below the current node using a heuristic evaluation function (see Definition 22 in Section 3.3.4.2). The algorithm then backtracks if the current best solutions found so far dominates the upper bound set, because traversing below the current node cannot lead to any new optimal solution.

Exact Multi-objective AND/OR Branch-and-Bound (MOAOBB) is one of the most efficient algorithms to solve MOCOPs (recently introduced by [Marinescu, 2009]). MOAOBB is an extension of the AND/OR Branch-and-Bound (AOBB) algorithm introduced by [Marinescu and Dechter, 2005] for the mono objective case. MOAOBB applies the general principles of AND/OR search and extends the MOBB into a Branch-and-Band algorithm guided by an AND/OR instead of traditional OR search tree. We start the description of the algorithm with the notion of *partial solution tree* which represents the sets of full solution trees.

Definition 21 » Partial solution tree

A partial solution tree T' of an AND/OR search tree $S_{\mathcal{T}}(\mathcal{M})$ is a sub-tree which:

(1) contains the root node s_o of $S_{\mathcal{T}}(\mathcal{M})$;

(2) if n is an OR node in T' then it contains one of its AND child nodes in $S_{\mathcal{T}}(\mathcal{M})$, and if n is an AND node it contains all its OR children in $S_{\mathcal{T}}(\mathcal{M})$.

A node of T' is a tip node if it has no children in T'. A tip node is either a terminal node (if it has no children in $S_{\mathcal{T}}(\mathcal{M})$), or a non-terminal node (if it has children in $S_{\mathcal{T}}(\mathcal{M})$).

In above definition, we can see that a partial solution tree whose tip nodes are terminal is a solution tree. A partial solution tree may be extended in several ways to a full solution tree; the set of these extensions is denoted with $extension(T')$.

3.3.4 Variable Elimination Algorithm

Variable elimination algorithms [Bertele and Brioschi, 1972, Dechter, 1999, Shafer and Shenoy, 1990] are the facets of inference-based methods. Let $G = \langle X, D, F \rangle$ be a graphical model with an ordering $d = (X_1, \ldots, X_n)$ of its variables. The ordering d represents the *variable elimination* order, from last to first, for variable elimination [Mateescu and Dechter, 2012]. The utility functions in F are placed into the buckets of the variables that appear latest in their scope. For instance, all functions that contain variable X_i and do not contain any X_j for $j > i$ are placed in the bucket of X_i.

Let $\{B(X_1), \ldots, B(X_n)\}$ be a set of buckets, one for each variable given in the order $d = (X_1, \ldots, X_n)$. We say that bucket $B(X_i)$ is connected to bucket $B(X_j)$ if the function generated in bucket $B(X_i)$ is placed in $B(X_j)$ [Marinescu and

Dechter, 2009a]. The variables of bucket $B(X_i)$, are the variables in the union of the scopes of its new and old functions.

Once the functions are partitioned into the buckets, then buckets are processed from the last variable X_n to the first X_1 by eliminating the buckets variable. The variables are eliminated by combining all functions and removing the variable by a marginalization operation and place the resulting function, also known as the *message*, in the bucket of its latest variable in d. A variable elimination algorithm also generates a *bucket-tree* [Kask et al., 2005] (where the bucket-tree is a tree whose nodes are buckets) by linking each bucket X_i to the one where the newly generated functions in bucket X_i are placed, which is called the parent of bucket X_i in the bucket-tree.

Example 8 » Variable elimination

Figure 3.7(a) shows the primal graph of a MOCOP instance which has the utility functions $f_1(X_0, X_1)$, $f_2(X_0, X_2)$, $f_3(X_0, X_1, X_4)$ and $f_4(X_1, X_2, X_3)$, a pseudo tree that drives its weighted AND/OR search tree is shown in figure 3.7(b), and the initial partition of the utility functions into the buckets along the ordering $d = (X_0, X_1, X_4, X_3, X_2)$ is shown in figure 3.7(c).

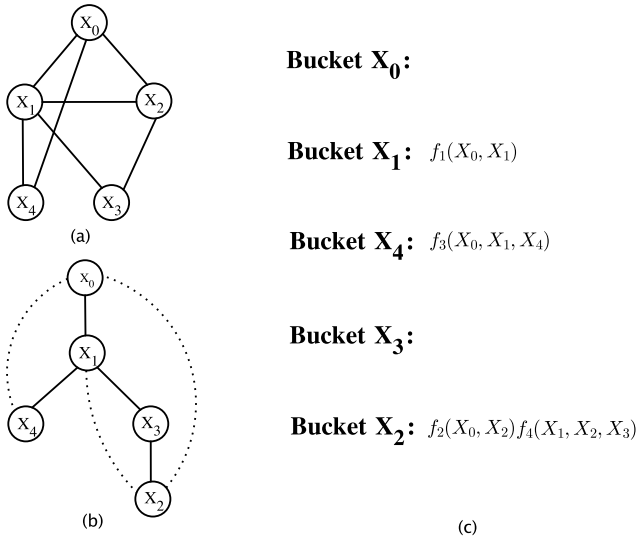


Figure 3.7: A MOCOP instance with bucket structure.

In this case the bucket of X_4 contains the function $f_3(X_0, X_1, X_4)$, the bucket of X_2 contains the functions $f_2(X_0, X_2)$ and $f_4(X_1, X_2, X_3)$ and the bucket X_1 has the function $f_1(X_0, X_1)$. We can notice that the buckets of X_0 and X_3 are initially empty.

Figure 3.8(a) shows the execution of the variable elimination and figure

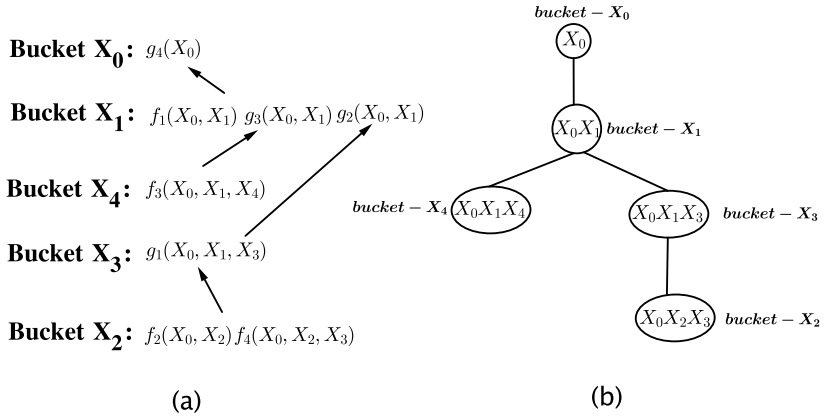


Figure 3.8: Variable elimination execution and bucket-tree.

3.8(b) represents its bucket-tree structure. The variable elimination algorithm first processes the bucket X_2, it combines the bucket's functions $f_2(X_0, X_2)$ and $f_4(X_1, X_2, X_3)$, the resulting function will be a utility function, lets say $f(X_0, X_1, X_2, X_3)$, which involves the variables X_0, X_1, X_2, X_3. Eliminate the variable X_2 from $f(X_0, X_1, X_2, X_3)$ using the marginalisation operation and, as shown in figure 3.8(a), the newly generated function, namely $g_1(X_0, X_1, X_3)$, is then placed into the bucket of next latest variable in d, i.e., X_3. Since the bucket of X_3 only has the function $g_1(X_0, X_1, X_3)$ then we apply the marginalisation operation directly to eliminate X_3. In this case the newly constructed function, namely $g_2(X_0, X_1)$, is placed into the bucket of the next latest variable X_4 in d. Similarly, the marginalisation operation is applied on the only function $f_3(X_0, X_1, X_4)$ residing in the bucket of X_4, and the newly constructed function $g_3(X_0, X_1)$ is placed in the bucket of X_1. Now, the bucket of X_1 has two new functions $g_2(X_0, X_1)$ and $g_3(X_0, X_1)$, and an already existing function $f_1(X_0, X_1)$. These functions are then combined to eliminate the variable X_1 by using the marginalisation. Finally, the newly generated function $g_4(X_0)$ is placed in the first variable X_0 in the ordering.

The order in which the variables are eliminated is important because it determines the complexity of the algorithm. The elimination ordering d we compute using greedy *Min-fill* heuristics [Dechter, 2003, Rollon and Larrosa, 2011].

3.3.4.1 Bucket and Mini Bucket Elimination

Given an optimization problem and its variable ordering d, the *Bucket Elimination* (BE) algorithm partitions utility functions into buckets, where each bucket is associated with a variable. Utility functions are placed into buckets according to the variables in their argument that appear latest in the variable ordering.

The algorithm has two phases, namely *top-down* and *bottom-up*. During the top-down phase, the algorithm processes each bucket from last to first by a variable elimination procedure (see Section 3.3.4). The variable elimination procedure computes a new function by combining all the functions in the bucket and eliminates the bucket's variable. The newly generated function is then placed into lower bucket. During the bottom-up phase, the algorithm assigns a value to each variable in the ordering and constructs a solution to the problem. The time and space complexity of the algorithm is exponential in w^*, where w^* is the induced width of the corresponding primal graph along the ordering d of the variables [Dechter, 1999].

The *mini-bucket elimination* (MBE) algorithm is designed to avoid time and space complexity of BE by partitioning the large buckets into smaller buckets, called *mini-buckets* [Dechter and Rish, 2003]. The sizes of these mini-buckets are controlled by an input parameter, called the i-bound. For simplicity sometimes we denote the i-bound with i. For given i the algorithm divides the large buckets into mini-buckets containing at most i number of distinct variables. Assuming maximization of the objectives, the algorithm processes each mini-bucket separately. Recently, [Rollon, 2008] extended the MBE algorithm for mono-objective to the multi-objective case, which is known as *multi-objective mini-bucket elimination* (MO-MBE). The important feature of MO-MBE is, for any given MOCOP instance along with the optimal solution it also outputs the collection of augmented buckets, which are the foundation for the generated heuristics. The time and space complexity of the algorithm is exponential in i, where $i < n$ [Dechter and Rish, 1998].

For given a MOCOP instance, if we set the i value to be very large, in particular, if $i > w^*$ then the algorithm MBE coincides with the BE algorithm. The BE algorithm can be viewed as a message passing algorithm, where messages pass from leaves to the root along the bucket-tree.

3.3.4.2 Branch-and-Bound and Mini-Bucket Heuristics

As mentioned previously, the MOAOBB performs the depth-first traversal of the search tree defined by the problem. The internal nodes during the search represent the partial assignments to the variables and leaf nodes denote the complete ones. For maximization problems, the algorithm maintains a global lower bound (global upper bound for minimization problems) on the cost of optimal

solutions, which corresponds to the best full assignment of the variables found thus far. In addition, at each node, the MOAOBB computes a heuristic estimate of the best solution extending the current partial assignment. MOAOBB then prunes the respective sub-tree if the heuristic estimate is worse than the global upper bound.

On the other hand, the effectiveness of the Branch-and-Bound algorithm depends on the quality of the heuristic function [Marinescu and Dechter, 2009b]. [Marinescu, 2009] extended the notion of heuristic evaluation function from the mono objective to the multi-objective case. Like in the mono objective case, with each node a heuristic evaluation function $h(n)$ is associated such that $h(n)$ is the underestimate of the efficient frontier $v(n)$ of the conditioned sub-problem below the node n.

Definition 22 » Heuristic evaluation function

Given a node n and a partial solution tree T'_n rooted at n in the AND/OR search tree $S_{\mathcal{T}}$. Then the heuristic evaluation function $f(T'_n)$, is defined recursively as follows:

(1) If T'_n has only node n then $f(T'_n) = h(n)$;

(2) If n is OR node having the AND child m in T'_n, then $f(T'_n) = w(n,m) + f(T'_m)$;

(3) If n is an AND node having OR children $m_1, \ldots, m_k$ in T'_n, then $f(T'_n) = \sum_{i=1}^{k} f(T'_{m_i})$.

From the above definition, $f(T'_n)$ is an upper bound that underestimates the efficient frontier (optimal solutions) of the sub-problem represented by T'_n. During the search, at the root node s_o, the algorithm maintains the best solutions found so far, denoted with $v(s_o)$. If every element of $f(T'_n)$ is dominated by at least one element of $v(s_o)$ (i.e., for all $\vec{v} \in f(T'_n)$ $\exists \vec{u} \in v(s_o)$ such that $\vec{u} > \vec{v}$, implying that $v(s_o) > f(T'_n)$, where $>$ is Pareto ordering) then it prunes the sub-problem below the current node n because no optimal solution exists under the node n.

In the following example we describe heuristic evaluation function of a partial solution tree.

Example 9 » Heuristic evaluation function of a partial solution tree

Consider the MOCOP instance presented in Example 5. Figure 3.9 shows the partially explored AND/OR search tree relative to the pseudo tree given in Figure 3.4(c). The current partial solution tree T' is highlighted (orange colour).

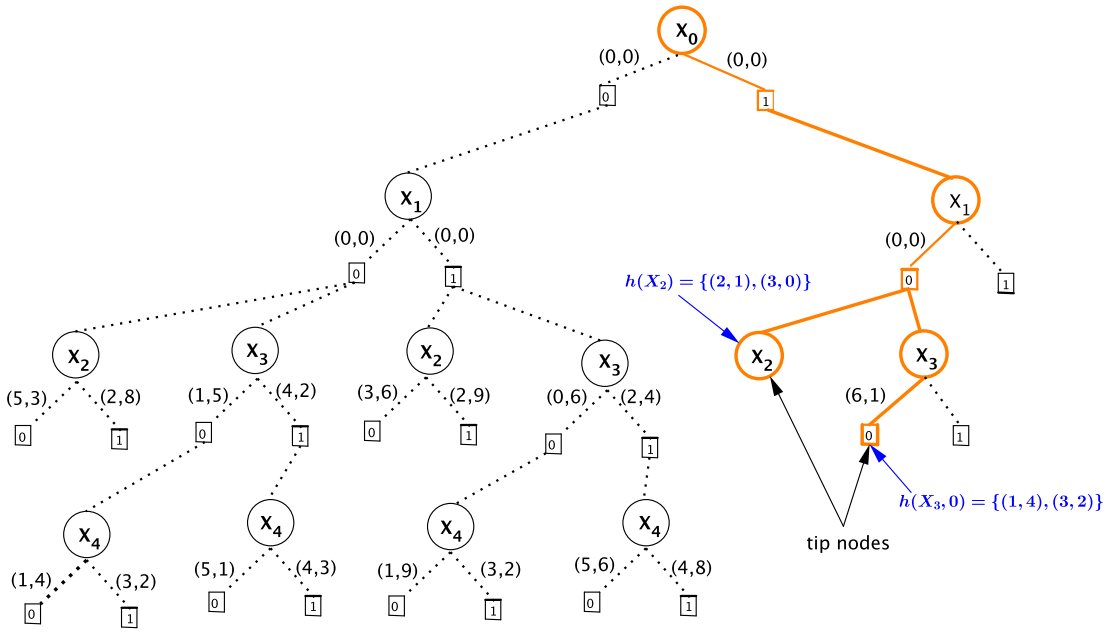


Figure 3.9: Utility associated with a partial solution tree.

The nodes of the partial solution tree are: $X_0, \langle X_0, 1\rangle, X_1, \langle X_1, 0\rangle, X_2, X_3$ *and* $\langle X_3, 0\rangle$. *The node* X_2 *is a terminal tip node and its corresponding heuristic estimate is* $h(X_2) = \{(2,1),(3,0)\}$. *Assuming that the search is currently at the tip node* $\langle X_3, 0\rangle$ *of* T' *and its corresponding heuristic estimate is* $h(\langle X_3, 0\rangle) = \{(1,4),(3,2)\}$. *By using Definitions 20 and 22, the heuristic evaluation function of* T' *is computed recursively as follows:*

$$
\begin{aligned}
f(T') =& w(X_0,1) + f(T'_{\langle X_0,1\rangle}) \\
=& w(X_0,1) + f(T'_{X_1}) \\
=& w(X_0,1) + w(X_1,0) + f(T'_{\langle X_1,0\rangle}) \\
=& w(X_0,1) + w(X_1,0) + f(T'_{X_2}) + f(T'_{X_3}) \\
=& w(X_0,1) + w(X_1,0) + h(X_2) + w(X_3,0) + f(T'_{\langle X_3,0\rangle}) \\
=& w(X_0,1) + w(X_1,0) + h(X_2) + w(X_3,0) + h(\langle X_3,0\rangle) \\
=& (0,0) + (0,0) + \underbrace{\{(2,1),(3,0)\}}_{h(X_2)} + \underbrace{(6,1)}_{w(X_3,0)} + \underbrace{\{(1,4),(3,2)\}}_{h(\langle X_3,0\rangle)} \\
=& \underbrace{\{(2,1),(3,0)\}}_{h(X_2)} + \underbrace{\{(7,5),(9,3)\}}_{w(X_3,0)+h(\langle X_3,0\rangle)} \\
=& \{(2,1)+(7,5),(2,1)+(9,3),(3,0)+(7,5),(3,0)+(9,3)\} \\
=& \{(9,6),(11,4),(10,5),(12,3)\}
\end{aligned}
$$

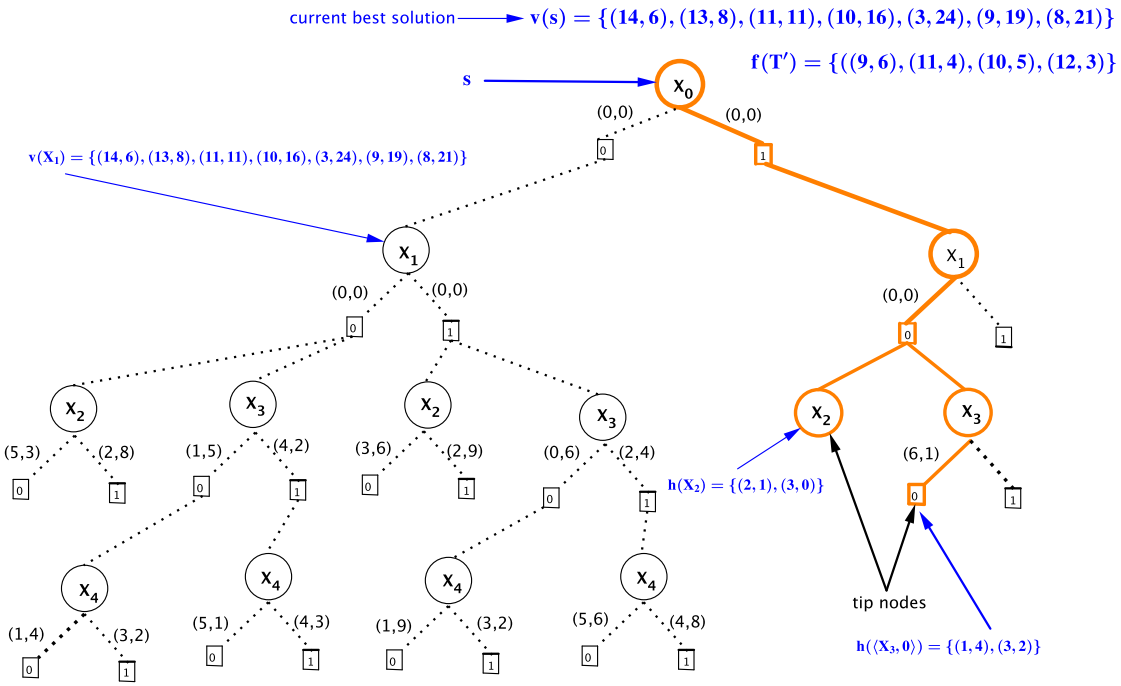


Figure 3.10: Pruning the partial solution tree.

In the following example, we explain the pruning mechanism of MOAOBB.

Example 10 » Pruning mechanism

Continuing Example 9, in Figure 3.10, the left sub-tree rooted at node X_1 is fully explored, producing the current best solution found so far, which is equal to $v(s) = \{(14,6), (13,8), (11,11), (10,16), (3,24), (9,19), (8,21)\}$. Suppose that the search is currently at AND node $\langle X_3, 0\rangle$ of the current partial solution tree T' (highlighted in Figure 3.10). Extending the sub-tree T' rooted at current tip node $\langle X_3, 0\rangle$, we obtained $f(T') = \{(9,6), (11,4), (10,5), (12,3)\}$. Clearly, the current best solution set, v(s), Pareto dominates $f(T')$, which implies that extending the sub-tree rooted at current tip node $\langle X_1, 0\rangle$ cannot yield a better solution and search can be pruned.

Static and Dynamic Mini-Bucket Heuristics

We now discuss two general schemes for generating heuristic estimates based on mini-bucket approximation. These schemes are controlled by the mini-bucket i-bound, which allows a controllable trade-off between preprocessing and computational overhead [Kask and Dechter, 2001].

Static Mini-Bucket Heuristics: For a given MOCOP instance the intermediate functions generated by the mini-bucket approximation can be used to com-

pute a heuristic function that overestimates the minimal extension of the current assignment in an AND/OR search [Marinescu and Dechter, 2005, Kask and Dechter, 2001]. For instance, consider $\{B(X_1), \ldots, B(X_n)\}$ the ordered set of augmented buckets generated by MO-MBE along a depth-first search traversal of the corresponding pseudo tree $\mathcal{T}$. The static mini-bucket heuristic function $h(n)$, where n is the node in the AND/OR search tree relative to $\mathcal{T}$ is computed as follows:

(1) If n is an AND node labelled $\langle X_j, x_j \rangle$ then $h(n)$ is equal to the sum of all intermediate functions generated by MO-MBE in buckets corresponding to the descendants of X_j in $\mathcal{T}$ and reside in bucket $B(X_j)$ or the buckets corresponding to the ancestors of X_j in $\mathcal{T}$;

(2) If n is an OR node labelled X_j then $h(n)$ is defined as the undominated closure (i.e., the set of elements that are not dominated by any other elements) of $\{w(n,m) + h(m) \mid m \in succ(n)\}$.

In the following example [Marinescu and Dechter, 2006, Marinescu, 2008] we describe the static bucket and mini-bucket heuristics.

Example 11 » Static bucket and mini-bucket heuristics

Consider a MOCOP instance with variables $\{X_0, X_1, X_2, X_3, X_4, X_5, X_6\}$, Figures 3.11(a) and 3.11(b) show its primal graph and corresponding pseudo tree. The variable ordering of the problem is given by $d = (X_0, X_1, X_2, X_3, X_4, X_5, X_6)$. The corresponding bucket and mini-bucket configurations are displayed in Figures 3.12 and 3.13, respectively. In Figures 3.12 and 3.13 we also show the execution of the bucket and mini-bucket eliminations along the bucket tree corresponding to the elimination ordering d. The functions (labelled h) on the arcs denote the messages sent from a bucket node to its parent in the tree.

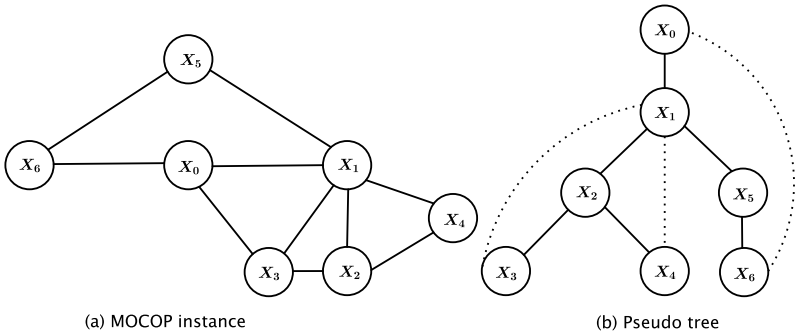


Figure 3.11: A MOCOP instance and a corresponding pseudo tree.

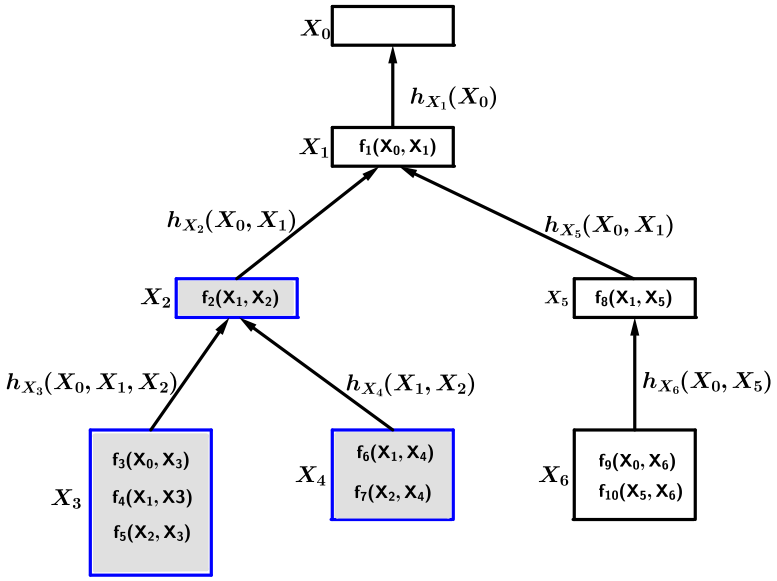


Figure 3.12: Static bucket elimination heuristic.

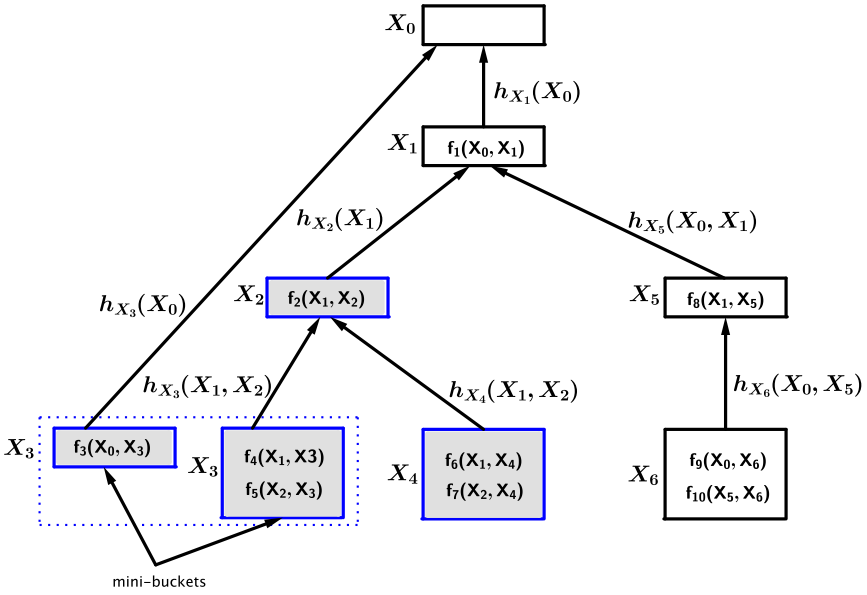


Figure 3.13: Static mini-bucket heuristics for $i = 3$.

Suppose that the variables X_0 and X_1 have been instantiated during the search. Let us assume that $h^*(x_0, x_1, x_2)$ is the maximal utility of the sub-problem rooted at node X_2 in the pseudo tree, conditioned on $(X_0 = x_0, X_1 = x_1, X_2 = x_2)$. In the AND/OR search tree, this is represented by the sub-problem rooted at the AND node labelled $\langle X_2, x_2 \rangle$, ending the path $\{X_0, \langle X_0, x_0 \rangle, X_1, \langle X_1, x_1 \rangle, X_2, \langle X_2, x_2 \rangle\}$.

Thus, we obtain:

$$\begin{aligned} h^*(x_0, x_1, x_2) &= \max_{X_3, X_4}(f_7(x_2, x_4) + f_6(x_1, x_4) + f_3(x_0, x_3) + f_5(x_2, x_3) + f_4(x_1, x_3) \\ &= \max_{X_3}(f_3(x_0, x_3) + f_5(x_2, x_3) + f_4(x_1, x_3)) + \max_{X_4}(f_7(x_2, x_4) + f_6(x_1, x_4)) \\ &= h_{X_3}(x_0, x_1, x_2) + h_{X_4}(x_1, x_2) \end{aligned}$$

where,

$$\begin{aligned} h_{X_3}(x_0, x_1, x_2) &= \max_{X_3}(f_3(x_0, x_3) + f_5(x_2, x_3) + f_4(x_1, x_3)) \\ h_{X_4}(x_1, x_2) &= \max_{X_4}(f_7(x_2, x_4) + f_6(x_1, x_4)) \end{aligned}$$

As shown in figure 3.12, the heuristic functions $h_{X_3}(x_0, x_1, x_2)$ and $h_{X_4}(x_1, x_2)$ are generated by the bucket elimination algorithm using the maximization operation (since we assume maximization of objectives) over the variables X_3 and X_4. On the other hand, computing these functions, particularly, the function $h_{X_3}(x_0, x_1, x_2)$ may be too hard to compute as it involves a computation with four variables. Alternatively, we can replace bucket elimination by a partition based mini-bucket approximation, which yields an upper bound approximation, $h(x_0, x_1, x_2)$, defined as follows:

$$\begin{aligned} h^*(x_0, x_1, x_2) &= \max_{X_3}(f_3(x_0, x_3) + f_5(x_2, x_3) + f_4(x_1, x_3)) + h_{X_4}(x_1, x_2) \\ &\leq \max_{X_3}(f_3(x_0, x_3)) + \max_{X_3}(f_5(x_2, x_3) + f_4(x_1, x_3)) + h_{X_4}(x_1, x_2) \\ &= h_{X_3}(x_0) + h_{X_3}(x_1, x_2) + h_{X_4}(x_1, x_2) \\ &= h(x_0, x_1, x_2) \end{aligned}$$

where,

$$\begin{aligned} h_{X_3}(x_0) &= \max_{X_3}(f_3(x_0, x_3)) \\ h_{X_3}(x_1, x_2) &= \max_{X_3}(f_5(x_2, x_3) + f_4(x_1, x_3)) \end{aligned}$$

As shown in Figure 3.13, the functions $h_{X_3}(x_0)$ and $h_{X_3}(x_1, x_2)$ are computed by the mini-bucket algorithm with $i = 3$. Thus, during search we can compute the function $h(x_0, x_1, x_2)$ from the pre-compiled mini-buckets, which yields an upper bound on the respective sub-problem.

For OR nodes, such as X_2, ending the path $\{X_0, \langle X_0, x_0\rangle, X_1, \langle X_1, x_1\rangle, X_2\}$, $h(X_2)$

can be obtained by maximizing over the values of the sum between the weights $w(C, \langle C, c\rangle)$ *and the heuristic evaluation function* $h(\langle X_2, x_2\rangle)$ *below the AND child* $\langle X_2, x_2\rangle$ *of* C*, i.e.,* $h(X_2) = \max_{\langle X_2, x_2\rangle}(w(X_2, \langle X_2, x_2\rangle) + h(\langle X_2, x_2\rangle))$.

Dynamic Mini-Bucket Heuristics: Instead of pre-compiling the mini-bucket heuristic, it is also possible to generate it dynamically during the search [Rollon and Larrosa, 2006, Marinescu and Dechter, 2005]. For instance, given the order set of augmented buckets $\{B(X_1), \ldots, B(X_n)\}$ along a depth-first search traversal of $\mathcal{T}$. If $asgn(\pi_n)$ is the current partial assignment path to the node n then the dynamic mini-bucket heuristic function is computed as follows:

(1) If n is an AND node labelled $\langle X_j, x_j\rangle$, then $h(n)$ is the sum of all intermediate functions that reside in bucket $B(X_j)$ and were generated by MO-MBE in buckets corresponding to the descendants of X_j in $\mathcal{T}$, conditioned on $asgn(\pi_n)$;

(2) If n is an OR node labelled X_j then $h(n)$ is defined as the undominated closure of $\{w(n, m) + h(m) \mid m \in succ(n)\}$.

For a specified i-bound, dynamic mini-bucket heuristic in comparison with the static mini-bucket heuristic implies a much higher computational overhead. However, the bounds generated dynamically may be much more accurate because some of the variables are already assigned and therefore require less partitioning and smaller number of functions [Marinescu, 2009, Marinescu and Dechter, 2009a].

Example 12 » Dynamic mini-bucket heuristics

Figure 3.14 shows the bucket tree structure corresponding to a MOCOP instance given in Example 11. The dynamic mini-bucket heuristic estimate $h(x_0, x_1, x_2)$ *of the AND node labelled* $\langle X_2, x_2\rangle$ *of variable* X_2 *ending the path* $\{X_0, \langle X_0, x_0\rangle, X_1, \langle X_1, x_1\rangle, X_2, \langle X_2, x_2\rangle\}$ *is computed by MO-MBE with* $i = 3$ *on the sub-problem represented by the buckets* X_3 *and* X_4*, conditioned on the partial assignment* $(X_0 = x_0, X_1 = x_1, X_2 = x_2)$*. The algorithm MO-MBE produces buckets* X_3 *and* X_4 *by eliminating the corresponding variables. As shown in Figure 3.14, it generates two new functions, namely,* $h_{X_3}(X_2)$ *and* $h_{X_4}(X_2)$*. These functions are constants since variables* X_0*,* X_1 *and* X_2 *are already assigned in the scope of the input functions:* $f_3(x_0, X_3), f_4(x_1, X_3), f_5(x_2, X_3), f_6(x_1, X_4)$ *and* $f_7(x_2, X_4)$*, respectively. Thus,* $h(x_0, x_1, x_2) = h_{X_3}(x_2) + h_{X_4}(x_2)$ *and it is equal to the exact* $h^*(x_0, x_1, x_2)$ *in this case.*

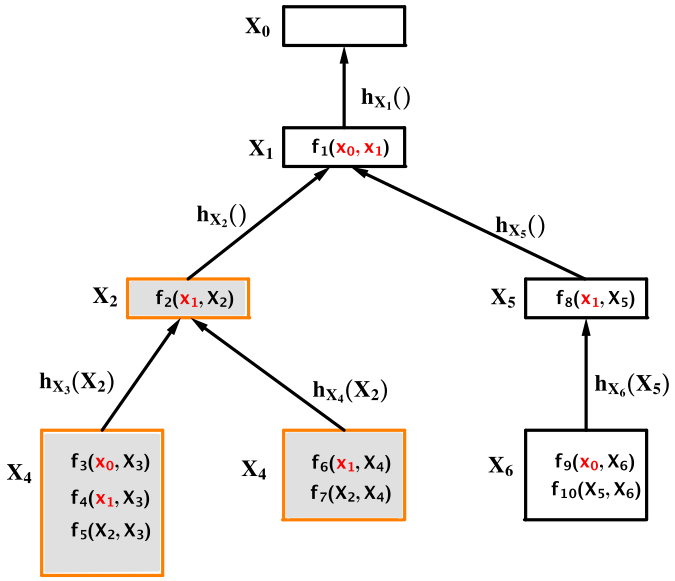


Figure 3.14: Dynamic mini-bucket heuristics for $i = 3$.

3.4 Decision Making under Uncertainty: Influence Diagrams

Influence diagrams are widely used by AI researches for knowledge representation and decision making under uncertainty. In this section we present the development of influence diagrams and discuss various methods that have been used for solving them. These methods solve influence diagrams either by converting them to other graphical models such as Bayesian networks and decision trees, or by evaluating them directly.

3.4.1 Transformation-based Evaluation

Reduction to Decision Tree:

Influence diagrams can be transformed into a decision tree and then solved using an average-out and fold-back strategy [Howard and Matheson, 1984b]. The transformation involves two phases. In the first phase, a regular influence diagram is first transformed into a *decision tree network* [Howard and Matheson, 1984b, Shachter, 1986]. Whereas in second phase, a decision tree is constructed from the decision tree network. The major drawback with this method is that the size of the decision tree is exponential in the number of variables in the influence diagram. Therefore, it requires an enormous amount of space to store

the tree [Pearl, 1988]. Similar kinds of algorithms can be found in [Qi, 1994, Qi and Poole, 1993, Qi et al., 1994], which transforms influence diagram into a decision tree in such a way that an optimal solution graph of the decision graph corresponds to an optimal policy of the influence diagram.

Qi and Poole [Qi and Poole, 1995] made an attempt to convert an influence diagram into a much smaller decision tree than Howard and Matheson's. This approach is based on asymmetry in the decision problem to avoid unnecessary computation and makes use of heuristic search techniques and domain dependent knowledge.

Reduction to Bayesian Network:

The approach of transforming influence diagrams to Bayesian Networks was first given by Cooper [Cooper, 1988]. Since the influence diagram is similar to a Bayesian network, the transformation only requires to ensure that all nodes in the graphical model are assigned proper probability distributions to allow one to perform inference. That means we only need to convert all decision and utility functions (or utility nodes) to chance nodes.

Cooper's method applies only in the case when there is just one utility node. Consider an influence diagram that has the set of decision nodes $\{D_1, \ldots, D_m\}$ having the finite domains $\{dom(D_1), \ldots, dom(D_m)\}$, then these nodes convert to chance nodes using the following distribution:

$$\forall d_i \in dom(D_i), P(d_i \mid pa(D_i)) = \frac{1}{|dom(D_i)|}$$

where $pa(D_i)$ is the parents of D_i in the graph of influence diagram and $|dom(D_i)|$ is the number of elements in $dom(D_i)$. Suppose that it has only one utility function denoted by U. Without loss of generality we can assume that U is non-negative. The utility node U is then transformed to a binary chance node with the following conditional probability distribution:

$$P(U = 1 \mid pa(U)) = \frac{U(pa(U))}{M_U}$$
$$P(U = 0 \mid pa(U)) = 1 - P(U = 1 \mid pa(U))$$

where $pa(U)$ is the set of parents (or scope) of U and $M_U = \max_{pa(U)} U(pa(U))$. This transformation is known as *Cooper's transformation* [Crowley, 2004].

Once the transformation process is finished decision variables are enumerated

according to the regularity constraint (which says that there is a defined order of decisions). If $D_1, \ldots, D_m$ is the order in which decisions are to be taken then it is shown that an optimal decision rule for decision D_k can be computed by

$$\delta_k^* = \arg\max_{D_k} P(D_k, pa(D_k) \mid U = 1).$$

The conditional probability of D_k is then set to $P_{\delta_k^*}(D_k \mid pa(D_k))$. Similarly, the optimal decision rule for decision variables $D_{k-1}, \ldots, D_1$ is computed using the above expression.

Shachter and Peot [Shachter and Peot, 1992] introduced an improved algorithm that takes advantage of the independence structure of the Bayesian network and the regularity constraint. This algorithm breaks the influence diagram into Bayesian inference sub-problems, which implies that the computations can be done locally using Bayesian network technique for belief propagation [Pearl, 1986]. In addition, using this approach we can recursively optimize each decision rule starting from the last until we reach the first.

Zhang [Zhang, 1998] introduced an algorithm that greatly reduces the number of nodes being considered in each stage of the evaluation. This algorithm is primarily based on the previous works of Zhang and Poole given in [Zhang and Poole, 1992], which deals with the stepwise decomposable influence diagrams, and [Zhang and Poole, 1994], which is based on Bayesian inference techniques. This method is primarily based on partitioning the set of all nodes in an influence diagram into disjoint sets with respect to the last decision D_m (which is called the *tail decision node*). Using these partitions they ignore the variables that are irrelevant to decision D_m and apply the Bayesian inference technique to compute the optimal decision rule for D_m. Xiang and Ye [Xiang and Ye, 2001] proposed an algorithm that transforms influence diagrams into a Bayesian network and then use junction tree-based [Jensen and Jensen, 1994] inference algorithms. This algorithm claims to be efficient and simpler than Zhang's algorithm but there is no experimental backup given for the claims.

3.4.2 Variable Elimination Techniques

Our approach to multi-objective influence diagram computation is based on the axiomatic framework of [Wilson and Marinescu, 2012], which allows partially ordered utility values. This general framework is applicable for influence

diagrams with multi-objective utilities, or interval-valued utilities, as well as quality decision theory based on order-of-magnitude probabilities and utilities.

3.4.2.1 Bucket Elimination

We adopt the variable elimination approach to solve (multi-objective) influence diagram, making use of Dechter's bucket elimination framework [Dechter, 2000a, Wilson and Marinescu, 2012]. To describe the bucket elimination algorithm for influence diagrams, consider a car buying problem [Pearl, 1988] described as follows:

Example 13 » Used cars

A buyer wants to buy one of the two used cars which can be carried out for various tests for various costs. Depending on the test results buyer can decide which car to buy. The quality of the car purchased determines the payoff he gets. The objective of the buyer is to maximize the expected monetary value.

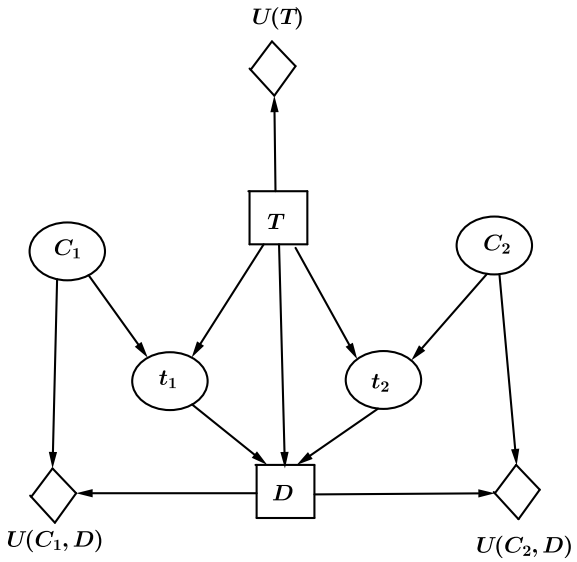


Figure 3.15: An influence diagram representation of car buying problem.

Figure 3.15 is the influence diagram representation of the situation. The decision variable T represents the choice of test to be carried out and it has three possible options, namely t_0 (no test), t_1 (test car 1) and t_2 (test car 2). Each decision option has three possible consequences, namely, pass, fail *and* null. *The* null *value represents that the test is carried out to car 1 and the results are given for car 2, and vice versa. The decision variable D represents the choice of which car to buy, its possible options are d_1 and d_2, representing the decision to buy car 1 and the*

decision to buy car 2, respectively. The two chance variables C_1 and C_2 represent the quality of car 1 and the quality of car 2, and each of them can either be good quality, q_1, or bad quality, q_2. The utility of testing is defined by the function $U(T)$, the utility in buying car 1 is defined by $U(C_1, D =$ buy car 1$)$ and the utility for buying car 2 is given by the function $U(C_2, D =$ buy car 2$)$. Note that the utility $U(C_1, D =$ buy car 2$) = 0$ and $U(C_2, D =$ buy car 1$) = 0$. The total utility of the problem is given by the sum of all utility functions, i.e., the total utility is $U(T) + U(C_1, D) + U(C_2, D)$.

The objective of the buyer is to determine the actions for the decision variables T and D which maximizes the following expected utility:

$$EU = \max_{T,D} \sum_{t_2,t_1,C_2,C_1} P(t_2 \mid C_2, T)P(C_2)P(t_1 \mid C_1, T)P(C_1). \\ [U(T) + U(C_1, D) + U(C_2, D)] \qquad (3.3)$$

Like the MOCOP case, the bucket elimination algorithm for influence diagrams partitions the probability and utility components into ordered buckets. Each of these functions are placed into a bucket of the variable that appear latest in their scopes. For instance, in the above car example, the initial bucket structure for the ordering $d = (T, t_2, t_1, D, C_2, C_1)$ is given in the following figure.

Bucket C_1: $\mathbf{P(C_1), P(t_1|C_1, T), U(C_1, D)}$

Bucket C_2: $\mathbf{P(C_2), P(t_2|C_2, T), U(C_2, D)}$

Bucket D:

Bucket t_1:

Bucket t_2:

Bucket T: $\mathbf{U(T)}$

Figure 3.16: Initial bucket structure of car problem.

After initial partitioning of the utility (denoted by ψ) and probability (denoted by ϕ) functions, buckets are processed from the last variable to the first in the ordering d. When processing a chance bucket a new probability component and a new utility component are computed, which are then placed into a closest lower bucket of a variable in their scopes. The chance variable is then

eliminated from the ordering. The decision buckets are processed in a similar manner and eliminate the bucket's variable by maximization ($\max$) operation.

Since the elimination ordering is $d = (T, t_2, t_1, D, C_2, C_1)$ the expected utility expression 3.3 can be reorganized as follows:

$$EU = \max_T \sum_{t_2} \sum_{t_1} \max_D \sum_{C_2} P(t_2 \mid C_2, T)P(C_2).\sum_{C_1} P(t_1 \mid C_1, T)P(C_1).$$
$$[U(T) + U(C_1, D) + U(C_2, D)]$$

When processing Bucket C_1:

$$EU = \max_T \sum_{t_2} \sum_{t_1} \max_D \sum_{C_2} P(t_2 \mid C_2, T)P(C_2).\sum_{C_1} \underbrace{P(t_1 \mid C_1, T)P(C_1)}_{\text{Bucket } C_1}.$$
$$[U(T) + \underbrace{U(C_1, D)}_{\text{Bucket } C_1} + U(C_2, D)]$$

$$EU = \max_T \sum_{t_2} \sum_{t_1} \max_D \sum_{C_2} P(t_2 \mid C_2, T)P(C_2).\underbrace{\phi(t_1, T)}_{\text{Bucket } C_1}.[U(T) + \underbrace{\psi(t_1, T, D)}_{\text{Bucket } C_1} + U(C_2, D)] \tag{3.4}$$

$$\text{where } \phi(t_1, T) = \sum_{C_1} P(t_1 \mid C_1, T)P(C_1)$$

$$\text{and } \psi(t_1, T, D) = \frac{1}{\phi(t_1, T)}.\sum_{C_1} P(t_1 \mid C_1, T)P(C_1)U(C_1, D).$$

We can see in expression (3.4) that eliminating chance variable C_1 (by summation) created a new probability ($\phi(t, T)$) and a utility ($\psi(t_1, T, D)$) components. Now, we proceed to eliminate the next variable, C_2.

When processing Bucket C_2: Reorganising the expression (3.4), we get

$$EU = \max_T \sum_{t_2} \sum_{t_1} \phi(t_1, T) \max_D \sum_{C_2} \underbrace{P(t_2 \mid C_2, T)P(C_2)}_{\text{Bucket } C_2}.[U(T) + \psi(t_1, T, D) + \underbrace{U(C_2, D)}_{\text{Bucket } C_2}]$$

$$EU = \max_T \sum_{t_2} \sum_{t_1} \phi(t_1, T) \max_D \underbrace{\phi(t_2, T)}_{\text{Bucket } C_2}.[U(T) + \psi(t_1, T, D) + \underbrace{\psi(t_2, T, D)}_{\text{Bucket } C_2}] \tag{3.5}$$

$$\text{where } \phi(t_2, T) = \sum_{C_2} P(t_2 \mid C_2, T) P(C_2)$$

$$\text{and } \psi(t_2, T, D) = \frac{1}{\phi(t_2, T)} . \sum_{C_2} P(t_2 \mid C_2, T) P(C_2) U(C_2, D).$$

In expression (3.5), processing chance bucket C_2 created a new probability ($\phi(t_2, T)$) component and a new utility ($\psi(t_2, T, D)$) component. We now move onto the next variable, D, in the ordering.

When processing Bucket D: Rearranging (3.5), we get

$$EU = \max_T \sum_{t_2} \phi(t_2, T) \sum_{t_1} \phi(t_1, T)[U(T) + \max_D \underbrace{\{\psi(t_1, T, D) + \psi(t_2, T, D)\}}_{\text{Bucket } D}]$$

maximizing over D, we obtain

$$EU = \max_T \sum_{t_2} \phi(t_2, T) \sum_{t_1} \phi(t_1, T)[U(T) + \psi(t_1, t_2, T)] \tag{3.6}$$

$$\text{where } \phi(t_1, t_2, T) = \max_D \{\psi(t_1, T, D) + \psi(t_2, T, D)\}.$$

For the decision bucket D, since no probability component involves D we simply maximize over the relevant sum of utility components, generating the new utility component, namely, $\psi(t_1, t_2, T)$. Now, we move onto the next variable t_1 in the ordering.

When processing Bucket t_1: Since everything is properly ordered in expression (3.6), we apply the summation operation over the relevant probability and utility components to remove t_1 in (3.6)

$$EU = \max_T \sum_{t_2} \phi(t_2, T) \sum_{t_1} \underbrace{\phi(t_1, T)}_{\text{Bucket } t_1}[U(T) + \underbrace{\psi(t_1, t_2, T)}_{\text{Bucket } t_1}]$$

$$EU = \max_T \sum_{t_2} \phi(t_2, T) \phi(T)[U(T) + \psi(t_2, T)] \tag{3.7}$$

$$\text{where } \phi(T) = \sum_{t_1} \phi(t_1, T) \text{ and } \psi(t_2, T) = \frac{1}{\phi(T)} \sum_{t_1} \phi(t_1, T) . \psi(t_1, t_2, T)$$

The elimination of chance variable t_1 created a new probability component and a new utility component, namely, $\phi(T)$ and $\psi(t_2, T)$. The next elimination variable in the ordering is t_2.

When processing Bucket t_2: Rearranging the probability and utility components in (3.7) to eliminate t_2, we get

$$EU = \max_T \phi(T) \sum_{t_2} \underbrace{\phi(t_2, T)}_{\text{Bucket } t_2} [U(T) + \underbrace{\psi(t_2, T)}_{\text{Bucket } t_2}]$$

$$EU = \max_T \phi(T)\phi'(T).[U(T) + \psi(T)] \tag{3.8}$$

$$\text{where } \phi'(T) = \sum_{t_2} \phi(t_2, T) \text{ and } \psi(T) = \frac{1}{\phi'(T)} \sum_{t_2} \phi(t_2, T).\psi(t_2, T)$$

The new probability and utility components after eliminating the chance variable t_2 are $\phi'(T)$ and $\psi(T)$. Now we move to the final variable T in the elimination ordering.

When processing Bucket T: Finally, we choose the options for T that maximizes:

$$EU = \max_T \phi(T)\phi'(T).[U(T) + \psi(T)]$$

	probability components	*utility components*
Bucket C_1:	$P(C_1), P(t_1 \mid C_1, T)$	$U(C_1, D)$
Bucket C_2:	$P(C_2), P(t_2 \mid C_2, T)$	$U(C_2, D)$
Bucket D:		$\psi(t_1, T, D), \psi(t_2, T, D)$
Bucket t_1:	$\phi(t_1, T)$	$\psi(t_1, t_2, T)$
Bucket t_2:	$\phi(t_2, T)$	$\psi(t_2, T)$
Bucket T:	$\phi(T), \phi'(T)$	$U(T), \psi(T)$

Figure 3.17: Bucket evaluation for car example.

Figure 3.17 shows the recorded functions in the buckets after processing in

reverse order of d.

It was shown in [Dechter, 2000a] that the dependencies created by the multiple utility components can be avoided, whenever the utility functions are defined on chance variables only. This shows that computing expected utility is not necessarily that much harder than the task of belief-updating over the underlying probabilistic sub-networks.

In principle, the bucket elimination for influence diagrams is similar to the variable elimination algorithms proposed by Shachter [Shachter, 1986, Shachter, 1988, Shachter, 1990], Tatman and Shachter [Tatman and Shachter, 1990], Shachter and Peot [Shachter and Peot, 1992], Shenoy [Shenoy, 1992] and Zhang [Zhang, 1998]. In particular it is very much similar to the join-tree clustering algorithm of F. Jensen, V. Jensen and Dittmer [Jensen et al., 1994] for evaluating influence diagrams.

Maua et al. [Maua and de Campos, 2011] proposed a variable elimination algorithm for solving Limited Memory Influence Diagrams (LIMIDs), which are generalization of influence diagrams allowing decision making with limited information. LIMIDs have similar graphical structure that of standard influence diagrams but relax the two fundamental assumptions, no-forgetting assumption and the assumption of a total order on the decisions. Although they consider standard totally ordered utilities, their algorithm also manipulates partially ordered sets of different undominated policies.

3.4.3 Other Algorithms

Previously, influence diagrams were evaluated using arc reversal and node removal method, in which each node (or variable) is removed through some *value-preserving reduction*, which is a sequence of transformations to the diagram that maintain feasibility and has the ability not to modify the maximal expected utility value. Nodes are removed one after the other until only one utility node remains, holding the utility of the optimal policy. Removal of chance variable involves averaging the utility that depends on the chance node using the conditional probability (involving that chance variable). In contrast, the removal of decision variable involves combining the utility that depends on the decision node and then maximizing the resulting utility over the decision node. Such works can be found in [Olmsted, 1983, Tatman and Shachter, 1990, Shachter, 1986, Ezawa, 1986].

A branch-and-bound approach for evaluating standard influence diagrams is suggested by Pearl [Pearl, 1988], as an improvement over the classical method of unfolding influence diagram into a decision tree. This approach was first implemented by Yuan et al. [Yuan et al., 2012], to overcome the difficulty of wasting computation in solving decision scenarios that have zero probability or that are not reachable from any initial state by following an optimal decision policy. They take the approach of [Nilsson and Höhle, 2001, Yuan and Hansen, 2009] for computing upper bounds on maximum utility and prune branches of the search tree that cannot be part of an optimal policy, and also prune branches that have zero probability. Marinescu [Marinescu, 2010] proposed AND/OR branch-and-bound algorithm which traverses AND/OR search spaces associated with an influence diagram.

Zhou et al. [Zhou et al., 2009] consider influence diagrams with interval-valued utility which is similar to bi-objective utility. Their approach is based on Cooper's transformation for solving influence diagrams based on Bayesian network algorithms [Cooper, 1988], which has a restriction on the influence diagrams to have a unique utility node. Kikuti and Cozman [Kikuti and Cozman, 2007], Kikuti et al. [Kikuti et al., 2011] and DeCampos [de Campos and Ji, 2008] allow interval probability but precise single-objective utility, and Lopez and Rodriguez [López-Díaz and Rodríguez-Muñiz, 2007] consider generalized influence diagrams based on fuzzy random variables.

Diehl and Haimes [Diehl and Haimes, 2004] describe a computational technique for influence diagrams with multiple objectives, with the restriction of just a single value node (utility function). The solution method is based on influence diagram transformations [Shachter, 1986]. Pareto dominance is used to prune sub-optimal utility vectors during the computation (with trade-offs being taken into account at the end of the computation).

3.4.4 Approximation Techniques

Our other approach for multi-objective influence diagrams is based on approximation technique that relies on a log transformation of the solution space. Papadimitriou and Yannakakis [Papadimitriou and Yannakakis, 2000] proposed the use of the logarithmic grid in $(1+\epsilon)$, where $\epsilon > 0$ is a very small real number, to generate an ϵ-*covering* of the Pareto set, which is based on ϵ-*dominance* ($\geq_\epsilon$). For any $\vec{u}, \vec{v} \in \mathbb{R}^p_+$ we say that $\vec{u}$ ϵ-dominates $\vec{v}$ if and only if $(1+\epsilon).\vec{u} \geq \vec{v}$,

whereas the ϵ-covering of any arbitrary finite set $\mathcal{U} \subset \mathbb{R}^p_+$ is the set $\mathcal{U}_\epsilon$, which has the property that $\forall\ \vec{v} \in \mathcal{U}\ \exists \vec{u} \in \mathcal{U}_\epsilon$ such that $\vec{u} \geq_\epsilon \vec{v}$.

The set $\mathcal{U}_\epsilon$ is not unique; however, it was shown in [Papadimitriou, 1991] that it is possible to compute an ϵ-covering of a finite set $\mathcal{U} \in \mathbb{R}^p_+$ by mapping each vector $\vec{u} \in \mathcal{U}$ onto a hyper grid using a log transformation $\varphi : \mathbb{R}^p_+ \to \mathbb{Z}^p_+$, defined by $\varphi(\vec{u}) = (\varphi(u_1), \ldots, \varphi(u_p))$, where $\forall i, \varphi(u_i) = \lceil \log u_i / \log(1+\epsilon) \rceil$, then for any $\vec{u}, \vec{v} \in \mathbb{R}^p_+$, we have the following (see Section 5.3 of Chapter 5 for more):

$$\varphi(\vec{u}) \geq \varphi(\vec{v}) \Rightarrow \vec{u} \geq_\epsilon \vec{v}.$$

However, it is not straightforward to replace the Pareto dominance with ϵ-dominance because ϵ-dominance is not transitive. For instance, in a bi-objective decision making problem, if we have $\vec{u} = (10, 50)$, $\vec{v} = (9, 54)$ and $\vec{w} = (7, 58)$ then for $\epsilon = 0.1$ we have $\vec{u} \geq_\epsilon \vec{v}$ and $\vec{v} \geq_\epsilon \vec{w}$ but $\vec{u} \not\geq_\epsilon \vec{w}$. To overcome this issue, we adopt a finer ϵ-dominance relation, called (ϵ, λ)-dominance ($\geq^\lambda_\epsilon$), where $\lambda \in (0, 1)$, developed by Dubus et al. [Dubus et al., 2009] for a generalized additive decomposable (GAI) network, which represents a (multi-objective) generalized decomposable additive utility functions [Gonzales and Perny, 2004]. For any $\vec{u}, \vec{v} \in \mathbb{R}^p_+$ we say that $\vec{u}$ (ϵ, λ)-dominates $\vec{v}$ if and only if $(1+\epsilon)^\lambda \cdot \vec{u} \geq \vec{v}$. The (ϵ, λ)-covering of any finite $\mathcal{U} \subset \mathbb{R}^p_+$, $\mathcal{U}_{(\epsilon,\lambda)}$, is obtained by eliminating all (ϵ, λ)-dominated elements from $\mathcal{U}$, i.e., $\mathcal{U}_{(\epsilon,\lambda)} = \{\vec{v} \in \mathcal{U} |\ \nexists \vec{u} \in \mathcal{U}$ such that $\vec{u} \geq^\lambda_\epsilon \vec{v}\}$.

It is now possible to compute an (ϵ, λ)-covering of a finite set $\mathcal{U} \subset \mathbb{R}^p_+$ with $\log$ grid mapping $\varphi_\lambda : \mathbb{R}^p_+ \to \mathbb{Z}^p_+$, defined by $\varphi_\lambda(\vec{u}) = (\varphi_\lambda(u_1), \ldots, \varphi_\lambda(u_p))$ where $\forall i$, $\varphi_\lambda(u_i) = \lceil \log u_i / \log(1+\epsilon)^\lambda \rceil$, For any $\vec{u}, \vec{v} \in \mathbb{R}^p_+$, we have the following:

$$\varphi_\lambda(\vec{u}) \geq \varphi_\lambda(\vec{v}) \Rightarrow \vec{u} \geq^\lambda_\epsilon \vec{v}.$$

3.5 Summary and Conclusion

In this chapter, we presented some introductory material for preferences and discussed various properties that a preference relation can take, which form the background to our preference handling techniques in this thesis. We also discussed various preference representation methods such as utility-based representation and lexicographic models. In particular, we described the weighted coefficients and relational representation models for preferences, which are closely related to the work on preferences in this dissertation. Despite the fact

that, weighted coefficients model is much easier when compare to the relational representation of preferences, the main drawback is that defining the weights to the objectives is much burden on the decision maker. In addition, representing the true preferences of the decision maker with weighted coefficients model is a very difficult task. Nevertheless, we show in Chapter 4, our formalism for preferences is an alternative approach to the weighted coefficients model.

For solving multi-objective constraint optimization models, we discussed in detail both inference-based and search-based algorithms. In addition, we provided introductory material to the graph-based concepts to help understanding the AND/OR search spaces for graphical models and some other graph-based formalisms. We explained AND/OR branch-and-bound and a variable elimination algorithm (based on bucket structure) for solving the model. These are the two main algorithms we focus in this dissertation in solving multi-objective constraint optimization models in Chapter 4.

For influence diagrams, various algorithms are presented to solve the model. In particular, we highlighted (in Section 3.4) related works in solving influence diagrams. Variable elimination techniques are discussed in detail and we see in Chapter 5 that it can be used effectively to solve multi-objective influence diagrams in both situations, first, when using ϵ-covering techniques for approximating the Pareto sets, and second, when there are some additional input preferences of the decision maker.

Chapter 4

Multi-objective Constraint Optimization with Trade-offs

This chapter discusses particular enhancements in multi-objective constraint optimization algorithms. The main focus is incorporating the decision maker's preference information into the multi-objective constraint optimization algorithms. In particular, we focus on branch-and-bound algorithms which use a mini-buckets algorithm for generating the upper bound (a set with vector of objective values) at each node. We consider the problems with maximization instead of minimization of the objectives and hence use the terminology of utility and utility function instead of cost and cost function.

The upper bound sets that guide the branch-and-bound algorithm can often be very large. We introduce simple and efficient methods to control the cardinality of the generating upper bound sets. The reduced upper bound sets still maintain the upper bound property.

The chapter is organized as follows: Section 4.2 gives the basic definition of multi-objective constraint optimization problem, Section 4.3 discusses the AND/OR Branch-and-Bound algorithm for multi-objective constraint optimization problems. The main focus of this chapter, handling the decision maker's imprecise trade-offs is given in Section 4.4. Methods for deducing preferences (imprecise trade-offs) are given in Section 4.4.1 and the implementation of the dominance test between the multi-objective utility vectors with respect to the deduced preferences is discussed in Section 4.4.3. Methods for controlling the upper bound set size for both Pareto (no trade-offs) and trade-offs cases are discussed in Section 4.5. Section 4.6 gives the experimental results on various

common multi-objective constraint optimization benchmarks. Finally, concluding remarks are given in Section 4.7.

4.1 Introduction

The goal of solving a Multi-Objective Constraint Optimization Problem (MO-COP) is to minimize or maximize the objective function subject to a set of constraints involving multiple, often conflicting and some times non-commensurate objectives. Examples of such problems arise in job-shop scheduling [Xia and Wu, 2005], machine maintenance [Al-Najjar and Alsyouf, 2003], product or process design [Ashby, 2000, Cavin et al., 2004] and many other.

We assume that there are $p(> 1)$ number of objectives in a multi-objective constraint optimization problem, where each complete assignment of the decision variables is a utility vector in $\mathbb{R}^p$. The solutions can be compared based on the Pareto or point-wise ordering. However, the Pareto ordering is rather weak, leading to the set of Pareto-undominated solutions becoming potentially very large. On the other hand, it is very difficult task for the decision maker to specify precise trade-offs between the objectives. As mentioned in earlier chapters, there are number of reasons for this, the most obvious one is the cognitive difficulty of specifying preferences [Brafman and Domshlak, 2009].

In this chapter, we discuss our approach for preference handling that allows the decision maker to express his preferences in the form of comparison between the utility vectors. The utility vectors presented to the decision maker for comparison can be chosen, e.g., using a brief elicitation technique. These preferences will then be used to strengthen the preference relation over $\mathbb{R}^p$. Our experimental results show that even a small number of such preferences can greatly reduce the size of the undominated set.

We present three different approaches for testing the dominance between utility vectors of $\mathbb{R}^p$ with respect to the input preferences of the decision maker. The first approach is based on linear programming, the second is based on the distance algorithm that computes distance between point and a convex cone, and the third is based on matrix multiplication, where we construct the matrix using the input preferences. The experimental results show that the matrix based approach speeds up almost an order of magnitude over the linear programming approach.

In this chapter, we focus on the extension of MOCO algorithms for the case where there are additional trade-offs, in particular we extend the multi-objective branch-and-bound and variable elimination algorithms. The branch-and-bound approach which we consider is based on [Marinescu, 2009], that performs the depth-first traversal of an AND/OR search tree and use a mini-buckets algorithm for generating an upper bound which is basically a set of utility vectors at each node. Since the generating upper bound sets are subsets of $\mathbb{R}^p$ they can become extremely large during the search. We propose very simple and effective methods to handle the cardinality of the upper bound sets for both Pareto (no trade-offs) and trade-offs case. These methods iteratively replace a collection of elements with their upper bound until the upper bound set reduces to the desired size. For the case where there are no trade-offs the upper bound of a collection of elements is generated using the Pareto ordering, whereas, for trade-offs case the upper bound is generated using the input preferences. The new reduced upper bound set still maintains the key upper bound property (defined in Section 4.5).

4.2 Multi-objective Constraint Optimization

A Multi-objective Constraint Optimization Problem (MOCOP) is a classical formulation of an operations research model which is based on maximization or minimization of some objective function subject to some constraints. We define a MOCOP problem with p objectives as a triple $\langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, where:

- $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of variables.
- $\mathbf{D} = \{D_1, \ldots, D_n\}$ is a set of finite domains of variables in $\mathbf{X}$.
- $\mathbf{F} = \{f_1, \ldots, f_r\}$ is a set of utility functions.

Each utility function $f_i \in \mathbf{F}$ is defined as $f_i : Y_i \rightarrow \mathbb{R}^p$, where $Y_i \subseteq \mathbf{X}$ is called *scope* of the function f_i. That means, for each configuration of variables in its scope, a utility function in $\mathbf{F}$ associates a vector in $\mathbb{R}^p$, which is called *utility vector*. A utility vector $\vec{u} = (u_1, \ldots, u_p)$ is a vector of p components, where $u_i \in \mathbb{R}$ represents the utility associated with the i-th objective. The objective function is sum of all utility function in the problem domain, i.e., $\mathcal{F}(\mathbf{X}) = \sum_{i=1}^{r} f_i(Y_i)$. Any complete value assignment, $x = (X_1 = x_1, \ldots, X_n = x_n)$ to the variables is called a *solution*.

Solutions are compared based on the associated utility vectors. The most common approach for comparing the solutions is the Pareto ordering, which is defined as: a solution x *weakly-Pareto dominates* another solution x' if and only if the utility vector associated with x is at least as good as the utility vector associated with x' for all $i \in \{1, \ldots, p\}$, and we say that x *Pareto dominates* x' if and only if the utility vector associated with x has strictly better value in at least one $j \in \{1, \ldots, p\}$ and better values in the remaining objectives than the associated utility vector of x'. The weak Pareto dominance and the Pareto dominance relations are denoted with $\geq$ and $>$, respectively. Formally, we define:

Definition 23 » weak Pareto order

If $\vec{u} = (u_1, \ldots, u_p)$ and $\vec{v} = (v_1, \ldots, v_p)$ are the utility vectors associated with any two solutions, then the relation $\geq$ defined by

$$\vec{u} \geq \vec{v} \text{ if and only if } u_i \geq v_i \text{ for all } i \in \{1, \ldots, p\}$$

is called the weak Pareto order.

Definition 24 » Pareto order

If $\vec{u} = (u_1, \ldots, u_p)$ and $\vec{v} = (v_1, \ldots, v_p)$ are the utility vectors associated with any two solutions, then the relation $>$ defined by

$$\vec{u} > \vec{v} \text{ if and only if } \vec{u} \geq \vec{v} \text{ and } \vec{u} \neq \vec{v}$$

is called the Pareto order.

With weak Pareto order (or Pareto order) it is not always possible to determine the most preferred (or the best) solution between a pair of solutions. For instance, in a bi-objective case, if we have $\vec{u} = (3, 2)$ and $\vec{v} = (2, 3)$, then neither of them dominates the other, i.e., $\vec{u} \not\succcurlyeq \vec{v}$ and $\vec{v} \not\succcurlyeq \vec{u}$. Consequently, the solution space of a MOCOP is partially ordered with respect to the weak Pareto order (or Pareto order).

In this thesis, we focus on the partial orders $\succcurlyeq$ satisfying the following monotonicity properties:

Independence: if $u \succcurlyeq v$ then $u + w \succcurlyeq v + w$ for any $u, v, w \in \mathbb{R}^p$.

Scale-Invariance: if $\vec{u} \succcurlyeq \vec{v}$ and $q \in \mathbb{R}$, $q \geq 0$ then $q \times \vec{u} \succcurlyeq q \times \vec{v}$ for any $u, v \in \mathbb{R}^p$ and $q \in \mathbb{R}$, $q \geq 0$.

An example of a partial order satisfying these monotonicity properties is the weak Pareto order.

Given any $u, v \in \mathbb{R}^p$, and $\succcurlyeq$ is any relation then $u \succcurlyeq v$ denotes that u *weakly dominates* v with respect to $\succcurlyeq$. As usual the asymmetric part of $\succcurlyeq$ denoted by $\succ$, which represents the *strict dominance* or simply *dominance* relation, defined as $u \succ v$ if and only if $u \succcurlyeq v$ and $v \not\succcurlyeq u$.

Solving a MOCOP problem consists of finding the set of undominated solutions that generate the maximal utility values (see Definition 18 in Chapter 3), i.e., finding the assignments of the decision variables such that the corresponding utility is a member of the set $max_{\succcurlyeq}\{\mathcal{F}(X) \in \mathbb{R}^p \mid X \in \mathbf{X}\}$.

4.3 Multi-objective AND/OR Branch-and-Bound

In this section, we present the Multi-objective AND/OR Branch-and-Bound (MOAOBB) to solve MOCOPs, which is introduced by [Marinescu, 2009]. MOAOBB is an extension of the AND/OR Branch-and-Bound (AOBB) algorithm presented by [Marinescu and Dechter, 2005] for the mono objective case. MOAOBB applies the general principles of AND/OR search and extends the Multi-objective Branch-and-Bound (MOBB) into a Branch-and-Bound algorithm guided by an AND/OR instead of traditional OR search tree. During the search, MOAOBB records the best solutions found so far. At each visited node during the search, MOAOBB computes an upper bound (which is basically a set of utility vectors in $\mathbb{R}^p$) of the sub-problem below the current node using a heuristic evaluation function $h(n)$ (see definition 22, Chapter 3). The heuristic function $h(n)$ is used in our experiments is the multi-objective mini-bucket heuristic (see Section 3.3.4.2, Chapter 3). The algorithm backtracks if the current best solutions found so far dominates the upper bound set, because traversing below the current node cannot lead to any new optimal solution.

MOAOBB is described by Algorithm 1. It performs a depth-first traversal of the weighted AND/OR search tree (see Section 3.3.2.1, Chapter 3), while maintaining at the root node s of the search tree the set $v(s)$ of the best solutions found so far. It expands alternating levels of OR and AND nodes (lines 3–10). A perimeter of the search is maintained by a stack called OPEN, whereas the nodes already expanded are contained by the list called CLOSED. During the search, an expanded node n having an empty set of successors propagates the

Algorithm 1: MOAOBB

Data: $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, heuristic h.
Result: Maximal set of $\mathcal{M}$, $\max_{\succcurlyeq}\{\mathcal{F}(\mathrm{X}) \mid \mathrm{X} \in \mathcal{X}\}$.

```
1  create an OR node s labeled by the root of T ;
2  OPEN ← {s}; CLOSED ← ∅;
3  while OPEN ≠ ∅ do
4      move top node n from OPEN to CLOSED;
5      expand n by creating its successors succ(n);
6      foreach n' ∈ succ(n) do
7          evaluate h(n') and add n' on top of OPEN;
8          let T' be the current partial solution tree with tip n';
9          if v(s) ≽ f(T') then
10             remove n' from OPEN and succ(n);
11     while ∃n ∈ CLOSED s.t. succ(n) = ∅ do
12         remove n from CLOSED and let n_p be n's parent;
13         if n_p is AND then v(n_p) ← v(n_p) + v(n);
14         else v(n_p) ← max_≽{v(n_p) ∪ {w(n_p, n) + v(n)}};
15         remove n from succ(n_p);
16 return v(s)
```

value $v(n)$ to its parent n_p in the search tree which, in turn, updates the value $v(n_p)$ (lines 11–15).

The algorithm computes node values recursively in bottom-up manner. That is starting from the leaf nodes, it removes the OR nodes by maximization and AND nodes by summation, where the maximization operation yields undominated closure with respect to the relation $\succcurlyeq$. The algorithm also prunes any partial solution tree T' with tip node n' if the corresponding heuristic evaluation function $f(T')$ is dominated by the current best solutions found so far $v(s)$ (i.e., for every $\vec{u} \in f(T')$ $\exists \vec{v} \in v(s)$ such that $\vec{u} \succcurlyeq \vec{v}$) maintained by the root node s on the maximal set $\max_{\succcurlyeq}\{\mathcal{F}(X) \in \mathbb{R}^p \mid X \in \mathcal{X}\}$ (lines 9–10). Finally, the search terminates when the root s is evaluated, resulting $v(s)$ as the set of utility vectors corresponding to the optimal solutions.

The time complexity of MOAOBB is given by $O(n \cdot k^{d_p})$, where d_p is the depth of the pseudo tree, k is the maximum domain size and n is the number of variables. Since the maximal set $\max_{\succcurlyeq}\{\mathcal{F}(X) \in \mathbb{R}^p \mid X \in \mathcal{X}\}$ is a subset of $\mathbb{R}^p$, it is not an easy task to predict its size, as a consequence MOAOBB may use prohibitively large amounts of memory to store the maximal sets.

4.4 Handling Imprecise Trade-offs

In this section we present our approach for handling the decision maker preferences.

4.4.1 Deducing Preferences from Additional Inputs

We assume that we have learned some preferences of the decision maker, i.e., we are given a set Θ of pairs of the form $(\vec{u}, \vec{v})$ meaning that the decision maker prefers $\vec{u}$ to $\vec{v}$, where the utility vectors $\vec{u}$ and $\vec{v}$ can be chosen using some elicitation technique. We will use this information to deduce further preferences in two different ways, the first based on Partial order inference [Marinescu et al., 2012], the second based on a well-known Multi-Attribute Utility Theory (MAUT) model.

Partial Order-based Inference: Suppose that we are given a set of input preferences Θ. Based on this preference information we infer a preference relation $\succcurlyeq_\Theta$ over $\mathbb{R}^p$. As mentioned earlier, we are interested in the partial orders which satisfies the Independence and Scale-Invariance properties. We assume that the decision maker has some unknown partial order $\succcurlyeq$ that represents Θ over $\mathbb{R}^p$ and further $\succcurlyeq$ satisfies the following properties:

— Extends weak Pareto: $\vec{u} \geq \vec{v} \Longrightarrow \vec{u} \succcurlyeq \vec{v}$ for all $\vec{u}, \vec{v} \in \mathbb{R}^p$.

— Scale Invariance: $\vec{u} \succcurlyeq \vec{v} \Longrightarrow q\vec{u} \succcurlyeq q\vec{v}$ for all $\vec{u}, \vec{v} \in \mathbb{R}^p$ and $q \geq 0, q \in \mathbb{R}$.

— Independence: $\vec{u} \succcurlyeq \vec{v} \Longrightarrow \vec{u} + \vec{w} \succcurlyeq \vec{v} + \vec{w}$ for all $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^p$

Since the decision maker's preference relation $\succcurlyeq$ includes Θ then it naturally gives rise to the following definitions.

Definition 25 » Consistency

The set Θ is said to be consistent if there exists some partial order $\succcurlyeq$ on $\mathbb{R}^p$ that extends Θ, extends Pareto, and satisfies Scale-Invariance and Independence.

Definition 26 » Induced preference relation

If Θ is consistent then we define an induced preference relation $\succcurlyeq_\Theta$ on $\mathbb{R}^p$ by $\vec{u} \succcurlyeq_\Theta \vec{v} \iff \vec{u} \succcurlyeq \vec{v}$ for all partial orders $\succcurlyeq$ extending Θ, extending Pareto, and satisfy Scale-Invariance and Independence.

Definition 25 gives the consistency of the input preferences. It is very crucial for us to test the consistency because it is quite common situation that revealed preferences might have inconsistencies [Wang et al., 2009, Tversky, 1996]. Definition 26 defines the induced preference relation over the set of input preferences. The relation $\succcurlyeq_\Theta$ should be the smallest among all partial orders which include the preferences of a decision maker and satisfying the assumed monotonicity properties.

MAUT-based Inference: Our other approach for preference inference is based on the well-known Multi-attribute Utility Theory. Assume that the decision maker uses a weighted sum of the objectives to compare the utility vectors (Chapter 5, Page 168 of [Figueira et al., 2005]), i.e., there exists non-negative real scalars $q_1, \ldots, q_p$, such that, for given any two utility vectors $\vec{u}, \vec{v} \in \mathbb{R}^p$ the decision maker prefers $\vec{u}$ to $\vec{v}$ ($\vec{u} \succcurlyeq \vec{v}$) if and only if $\sum_{i=1}^{p} q_i u_i \geq \sum_{i=1}^{p} q_i v_i$. We call the preference relation $\succcurlyeq$ a MAUT-based total pre-order. If we know the values for weight vectors $q_1, \ldots, q_p$ in advance then the problem reduces to the single objective. Since we are given a set of input preferences Θ this allows us to define the induced preference relation $\succcurlyeq^2_\Theta$, defined as follows:

Definition 27 » MAUT-based preference relation

$\vec{u} \succcurlyeq^2_\Theta \vec{v}$ if and only if $\vec{u} \succcurlyeq \vec{v}$ holds for all MAUT-based pre-orders extending Θ.

Thus $\vec{u}$ is preferred to $\vec{v}$ then it holds for all MAUT-based orderings satisfying preferences, and vice versa.

4.4.2 Preference Models

In this section we present different preference models for representing the decision maker's input preferences.

The consistent set of preferences Θ induces a relation $\succcurlyeq_\Theta$ which specifies the input preferences. For given any $\vec{u}, \vec{v} \in \mathbb{R}^p$, Definition 26 implies that $(\vec{u}, \vec{v})$ ($\vec{u}$ is preferred to $\vec{v}$) *can be deduced from* Θ if $\vec{u} \succcurlyeq \vec{v}$ holds for all partial orders $\succcurlyeq$ that extend Θ, extend Pareto, and satisfy Scale-Invariance and Independence properties. In this case we denote $\vec{u} \succcurlyeq_\Theta \vec{v}$.

The following well-known result (see e.g., [Casas-Garriga and Balcázar, 2004], Example 1.8, page 7, [Harju, 2006]) shows that the intersection of a set of partial orders is also a partial order. By definition, we can see that the induced

preference relation $\succcurlyeq_\Theta$ is also the intersection of all partial orders extending Θ. If Θ is consistent then the following result implies that $\succcurlyeq_\Theta$ is a partial order.

Lemma 1 » Intersection of arbitrary family of partial orders

The intersection of a set of partial orders is a partial order.

Proof. Let Q be a set of partial orders and $\succcurlyeq$ be the intersection of all partial orders in Q, i.e., $\succcurlyeq = \bigcap Q$. We need to show that $\succcurlyeq$ is a partial order.

(i) For all partial orders $\succeq$ in Q, by reflexivity we have $\vec{u} \succeq \vec{v}\ \forall \vec{u}$, which implies that $\vec{u} \succcurlyeq \vec{v}$ holds $\forall \vec{u}$. Thus, $\succcurlyeq$ is reflexive.

(ii) For all $\vec{u}, \vec{v}, \vec{w}$ suppose that $\vec{u} \succcurlyeq \vec{v}$ and $\vec{v} \succcurlyeq \vec{w}$.

$\vec{u} \succcurlyeq \vec{v} \implies \vec{u} \succeq \vec{v}$ holds for all partial orders $\succeq$ in Q. Similarly, $\vec{v} \succcurlyeq \vec{w} \implies \vec{v} \succeq \vec{w}$ holds for all partial orders $\succeq$ in Q. By transitivity of partial orders $\succeq$, it follows immediately that $\vec{u} \succeq \vec{w}$. The definition of $\succcurlyeq$ then implies that $\vec{u} \succcurlyeq \vec{w}$. Hence, $\succcurlyeq$ is transitive.

(iii) For all $\vec{u}, \vec{v}$ suppose that $\vec{u} \succcurlyeq \vec{v}$ and $\vec{v} \succcurlyeq \vec{u}$.

$\vec{u} \succcurlyeq \vec{v}$ and $\vec{v} \succcurlyeq$ implies that $\vec{u} \succeq \vec{v}$ and $\vec{v} \succeq \vec{u}$ holds for all partial orders $\succeq$ in Q. The antisymmetric property of $\succeq$ implies that $\vec{u} = \vec{v}$. Thus, $\succcurlyeq$ is antisymmetric.

□

For a consistent set of input preferences Θ, if $Q(\Theta)$ is the set of all partial orders extending Θ and Pareto, satisfying Scale-Invariance and Independence properties then by Definition 26, we can write the induced preference relation $\succcurlyeq_\Theta$ as $\succcurlyeq_\Theta = \bigcap Q(\Theta)$. The following result gives the important characterization of $\succcurlyeq_\Theta$, which says, for a given set of consistent input preferences Θ the relation $\succcurlyeq_\Theta$ is a partial order, satisfies Scale-Invariance and Independence. We need these properties for the MOCOP algorithms, such as MOAOBB, to be correct (up to equivalence).

Proposition 2 » Consistent input preferences induce a partial order

If Θ is consistent then $\succcurlyeq_\Theta$ is a partial order extending Θ and Pareto, and satisfying Scale-Invariance and Independence.

Proof. Let Θ be the consistent set of input preferences and $Q(\Theta)$ be the set of all partial orders extending Θ and Pareto, and satisfying the Scale-Invariance and Independence properties.

(i) Since $\succcurlyeq_\Theta = \bigcap Q(\Theta)$, Lemma 1 implies that $\succcurlyeq_\Theta$ is a partial order.

(ii) Suppose that $(\vec{u}, \vec{v}) \in \Theta$. Then for all partial orders $\succcurlyeq$ in $Q(\Theta)$ we have that $\vec{u} \succcurlyeq \vec{v}$, which implies $\vec{u} \succcurlyeq_\Theta \vec{v}$. Therefore, $\succcurlyeq_\Theta$ extends Θ.

(iii) Let $\vec{u} \geq \vec{v}$ for all $\vec{u}, \vec{v}$, where $\geq$ is the Pareto ordering. Since all partial orders $\succcurlyeq$ in $Q(\Theta)$ extends the Pareto ordering, i.e., $\vec{u} \geq \vec{v} \implies \vec{u} \succcurlyeq \vec{v}$ holds for all Partial orders $\succcurlyeq$ in $Q(\Theta)$, which implies that $\vec{u} \succcurlyeq_\Theta \vec{v}$ also holds. Thus, $\succcurlyeq_\Theta$ extends the Pareto ordering.

(iv) Let $\vec{u} \succcurlyeq_\Theta \vec{v}$ for all $\vec{u}, \vec{v}$, and let $\lambda \geq 0$ be a real number.

$\vec{u} \succcurlyeq_\Theta \vec{v} \implies \vec{u} \succcurlyeq \vec{v}$ holds for all partial orders $\succcurlyeq$ in $Q(\Theta)$. Since all partial orders in $Q(\Theta)$ satisfy the Scale-Invariance then it holds $\lambda\vec{u} \succcurlyeq \lambda\vec{v}$ for all partial orders $\succcurlyeq$ in $Q(\Theta)$, which implies $\lambda\vec{u} \succcurlyeq_\Theta \lambda\vec{v}$. Therefore, $\succcurlyeq_\Theta$ satisfies Scale-Invariance.

(v) Let $\vec{u} \succcurlyeq_\Theta \vec{v}$ for all $\vec{u}, \vec{v}$.

$\vec{u} \succcurlyeq_\Theta \vec{v} \implies \vec{u} \succcurlyeq \vec{v}$ for all partial orders $\succcurlyeq$ in $Q(\Theta)$. Then for all $\vec{w}$ the Independence property implies that $\vec{u} + \vec{w} \succcurlyeq \vec{v} + \vec{w}$ for all partial orders $\succcurlyeq$ in $Q(\Theta)$, implying that $\vec{u} + \vec{w} \succcurlyeq_\Theta \vec{v} + \vec{w}$. Thus, $\succcurlyeq_\Theta$ satisfies Independence. This completes the proof.

□

Since the relation $\succcurlyeq_\Theta$ satisfies the Independence property then with the following result we show that every element of the set $\{\vec{u} - \vec{v} \mid (\vec{u}, \vec{v}) \in \Theta\}$ is preferred to $\vec{0}$.

Lemma 2 » Difference vector always preferred to zero vector

Let $\succcurlyeq$ be a pre-order on $\mathbb{R}^p$ that satisfies Independence. Then, for any vectors $\vec{u}$ and $\vec{v}$, we have $\vec{u} \succcurlyeq \vec{v} \iff \vec{u} - \vec{v} \succcurlyeq \vec{0}$.

Proof. Suppose that $\vec{u} \succcurlyeq \vec{v}$ for $\vec{u}, \vec{v} \in \mathbb{R}^p$. The Independence property of $\succcurlyeq$ implies that $\vec{u} + (-\vec{v}) \succcurlyeq \vec{v} + (-\vec{v})$, i.e., $\vec{u} - \vec{v} \succcurlyeq \vec{0}$.

Conversely, suppose that $\vec{u} - \vec{v} \succcurlyeq \vec{0}$ for $\vec{u}, \vec{v} \in \mathbb{R}^p$. Again using the Independence property we have $\vec{u} - \vec{v} + \vec{v} \succcurlyeq \vec{0} + \vec{v}$, i.e., $\vec{u} \succcurlyeq \vec{v}$.

□

The following result gives the generalized Independence property.

Lemma 3 » Generalized Independence

Suppose that a partial order $\succcurlyeq$ satisfies Independence, and that for all $i = 1, \ldots, k$, $\vec{u_i} \succcurlyeq \vec{v_i}$ then $\sum_{i=1}^{k} \vec{u_i} \succcurlyeq \sum_{i=1}^{k} \vec{v_i}$.

Proof. We prove this by induction on k. If $k = 1$ then the statement is true, i.e., $\vec{u_1} \succcurlyeq \vec{v_1}$. Assume that $\sum_{i=1}^{k-1} \vec{u_i} \succcurlyeq \sum_{i=1}^{k-1} \vec{v_i}$. Independence and commutative properties implies that $\sum_{i=1}^{k-1} \vec{u_i} + \vec{u_k} \succcurlyeq \vec{u_k} + \sum_{i=1}^{k-1} \vec{v_i}$, i.e.,

$$\sum_{i=1}^{k} \vec{u_i} \succcurlyeq \vec{u_k} + \sum_{i=1}^{k-1} \vec{v_i}. \tag{4.1}$$

Considering $\vec{u_k} \succcurlyeq \vec{v_k}$, we can repeat the Independence property with $\sum_{i=1}^{k-1} \vec{v_i}$ then

we obtain $\vec{u_k} + \sum_{i=1}^{k-1} \vec{v_i} \succcurlyeq \vec{v_k} + \sum_{i=1}^{k-1} \vec{v_i}$, i.e.,

$$\vec{u_k} + \sum_{i=1}^{k-1} \vec{v_i} \succcurlyeq \sum_{i=1}^{k} \vec{v_i}. \tag{4.2}$$

Relations (4.1), (4.2) and the transitivity of $\succcurlyeq$ implies that $\sum_{i=1}^{k} \vec{u_i} \succcurlyeq \sum_{i=1}^{k} \vec{v_i}$. This completes the proof by induction on k.

□

The relation $\succcurlyeq'_\Theta$ based on pre-orders, and its relationship with $\succcurlyeq_\Theta$

If Θ be the set of input preferences and $\succcurlyeq'_\Theta$ be the induced relation defined as follows: for given $\vec{u}, \vec{v} \in \mathbb{R}^p$ we define, $\vec{u} \succcurlyeq'_\Theta \vec{v}$ if and only if $\vec{u} \succcurlyeq \vec{v}$ holds for all pre-orders $\succcurlyeq$ that extend Θ, extend Pareto, and satisfy Scale-Invariance and Independence.

Using the above definition we can write the relation $\succcurlyeq'_\Theta$ as $\bigcap R(\Theta)$, where $R(\Theta)$ is the set of all pre-orders that extend Θ, extend Pareto, and satisfy Scale-Invariance and Independence. The relation $\succcurlyeq_\Theta$ is defined as $\bigcap Q(\Theta)$, where $Q(\Theta)$ is the set of all partial orders that extend Θ, extend Pareto, and satisfy Scale-Invariance and Independence.

The following result shows that the intersection of any arbitrary family of pre-orders that contains a partial order is always a partial order.

Lemma 4 » Intersection of arbitrary family of pre-orders including a partial order

---

The intersection of a set of pre-orders that includes a partial order is a partial order.

Proof. Let R be the set of pre-orders. Let $\succcurlyeq$ be a pre-order and $\succeq$ be a partial order in R. We need to show that $\succcurlyeq$ is a partial order, i.e., we need to show $\succcurlyeq$ is antisymmetric.

Let $\vec{u} \succcurlyeq \vec{v}$ and $\vec{v} \succcurlyeq \vec{u}$.

$$\vec{u} \succcurlyeq \vec{v} \implies \vec{u} \succeq \vec{v}$$
$$\vec{v} \succcurlyeq \vec{u} \implies \vec{v} \succeq \vec{u}$$
$$\vec{u} \succeq \vec{v} \text{ and } \vec{v} \succeq \vec{u} \implies \vec{u} = \vec{v}.$$

Therefore, $\vec{u} \succcurlyeq \vec{v}$ and $\vec{v} \succcurlyeq \vec{u}$ implies that $\vec{u} = \vec{v}$. This completes the proof.

□

We make use of the above result in the following result, which gives the relation between $\succcurlyeq'_\Theta$ and $\succcurlyeq_\Theta$.

Lemma 5 » Equivalence of $\succcurlyeq_\Theta$ and $\succcurlyeq'_\Theta$

The following holds for an arbitrary binary relation defined for the input preferences set Θ on $\mathbb{R}^p$.

(i) *$\succcurlyeq_\Theta$ and $\succcurlyeq'_\Theta$ are both pre-orders that extend Θ, extend Pareto, and satisfy Scale-Invariance and Independence.*

(ii) *Θ is consistent $\iff$ $\succcurlyeq_\Theta$ is a partial order.*

(iii) *$\succcurlyeq'_\Theta \subseteq \succcurlyeq_\Theta$.*

(iv) *If Θ is consistent then $\succcurlyeq'_\Theta = \succcurlyeq_\Theta$.*

Proof. Let Θ be the set of input preferences on $\mathbb{R}^p$.

(i): Proof follows from Proposition 2.

(ii): Suppose that Θ is a consistent set, which implies $Q(\Theta)$ is non-empty. Since $\succcurlyeq_\Theta$ is the intersection of all partial orders then $\succcurlyeq_\Theta$ itself a partial order.

Conversely, suppose that $\succcurlyeq_\Theta$ is a partial order extending Θ and Pareto, and satisfying Scale-Invariance and Independence then the set $Q(\Theta)$ is non-empty, i.e., there exists a partial order that extends Θ and Pareto, and satisfying Scale-Invariance and Independence properties then Definition 25 implies that Θ is consistent.

(iii): Since a partial order is pre-order, then we have $Q(\Theta) \subseteq R(\Theta)$ which implies $\succcurlyeq'_\Theta \subseteq \succcurlyeq_\Theta$.

(iv): Suppose that Θ is consistent. Then $Q(\Theta)$ is non-empty. From (iii), we have $Q(\Theta) \subseteq R(\Theta)$ which implies $R(\Theta)$ includes a partial order. Lemma 4 implies that the intersection of a set of pre-orders that includes a partial order is a partial order, that means $\succcurlyeq'_\Theta = \bigcap R(\Theta)$ is a partial order. Using (i), we have $\succcurlyeq'_\Theta$ is a partial order that extends Θ, extends Pareto, and satisfies Scale-Invariance and Independence. Therefore, $\succcurlyeq'_\Theta$ is a member of the set $Q(\Theta)$, which then implies $\succcurlyeq_\Theta \subseteq \succcurlyeq'_\Theta$. It follows using (iii) that $\succcurlyeq'_\Theta = \succcurlyeq_\Theta$.

□

The cone-based ordering $\succcurlyeq^\Theta$

In this section, we present our preference model, which is based on the positive convex cones (which is a convex cone, contains $\vec{0}$ and contains all vectors $\vec{u}$ such that $\vec{u} \geq \vec{0}$).

Let $W = \{\vec{w_1}, \ldots, \vec{w_k}\}$ with $W \subseteq \mathbb{R}^p$. We define the set $\mathbf{C}(W)$ to be the set of all vectors $\vec{u}$ of $\mathbb{R}^p$ such that there exists $k \geq 0$ and non-negative real scalars $q_1, \ldots, q_k$ such that $\vec{u} \geq \sum_{i=1}^{k} q_i \vec{w_i}$, where $\geq$ is weak Pareto order. Note that an empty summation on the right-hand-side taken to be equal to $\vec{0}$, the zero vector $(0, \ldots, 0)$ in $\mathbb{R}^p$. In other words, the set $\mathbf{C}(W)$ consists all the vectors that weakly-Pareto dominate some (finite) linear combination of elements of W, i.e., $\mathbf{C}(W) = \{\vec{u} \in \mathbb{R}^p \mid \exists q_1, \ldots q_k \in \mathbb{R}^+ \cup \{0\}$ such that $\vec{u} \geq \sum_{i=1}^{k} q_i \vec{w_i}\}$.

The following well-known result (see e.g., Chapter 19, pages 170-178, [Rockafellar, 1997]) gives some properties of the set $\mathbf{C}(W)$.

Lemma 6 » $\mathbf{C}(W)$ is a positive convex cone

Let W be any finite subset of $\mathbb{R}^p$.

(i) If $\vec{u} \geq \vec{0}$ then $\vec{u} \in \boldsymbol{C}(W)$.

(ii) If $\vec{u}, \vec{v} \in \boldsymbol{C}(W)$ then $\vec{u} + \vec{v} \in \boldsymbol{C}(W)$.

(iii) $\boldsymbol{C}(W) \supseteq W$.

(iv) If $\vec{u} \in \boldsymbol{C}(W)$ and q is a non-negative real number then $q\vec{u} \in \boldsymbol{C}(W)$.

Proof. Let W be any finite subset of $\mathbb{R}^p$.

(i): Assume that $\vec{u} \geq \vec{0}$. Choose $q_1 = 0, \ldots, q_k = 0$ then we have $\sum_{i=1}^{k} q_i\vec{w}_i = \vec{0}$, i.e., $\vec{u} \geq \sum_{i=1}^{k} q_i\vec{w}_i$, which implies $\vec{u} \in \mathbf{C}(W)$.

(ii): Suppose $\vec{u}, \vec{v} \in \mathbf{C}(W)$, then there exists non-negative real scalars $q_1, \ldots, q_k$ and $q'_1, \ldots, q'_k$, and vectors $\vec{w}_1, \ldots, \vec{w}_k \in W$ such that $\vec{u} \geq \sum_{i=1}^{k} q_i\vec{w}_i$ and $\vec{v} \geq \sum_{i=1}^{k} q'_i\vec{w}_i$. Then $\vec{u} + \vec{v} \geq \sum_{i=1}^{k} (q_i + q'_i)\vec{w}_i$, implying that $\vec{u} + \vec{v} \in \mathbf{C}(W)$.

(iii): Consider $\vec{w}_1 \in W$ and choose $q_1 = 1, q_2 = 0, \ldots, q_k = 0$. Then we can write $\vec{w}_1 = \sum_{i=1}^{k} q_i\vec{w}_i$, which implies $\vec{w}_1 \in \mathbf{C}(W)$. Similarly $\vec{w}_2, \ldots, \vec{w}_k \in \mathbf{C}(W)$. Hence $\mathbf{C}(W) \supseteq W$.

(iv): Let $\vec{u} \in \mathbf{C}(W)$, then there exists non-negative real scalars $q_1, \ldots, q_k$ and vectors $\vec{w}_1, \ldots, \vec{w}_k \in W$ such that $\vec{u} \geq \sum_{i=1}^{k} q_i\vec{w}_i$. Consider real $q \geq 0$. Then $q\vec{u} \geq \sum_{i=1}^{k} (qq_i)\vec{w}_i$, implying that $q\vec{u} \in \mathbf{C}(W)$.

□

In the above result, the properties (ii) (closure under addition) and (iv) (closure under positive scalar multiplication) imply that $\mathbf{C}(W)$ is a convex cone. In addition, we name the property (i) as *positivity*. For this reason we call $\mathbf{C}(W)$ a *positive convex cone*.

For $W \subseteq \mathbb{R}^p$, we define W^* to be the set of vectors $\vec{u} \in \mathbb{R}^p$ such that $\vec{u} \cdot \vec{v} \geq 0$ for all $\vec{v} \in W$, where $\vec{u} \cdot \vec{v}$ is the dot product between $\vec{u}$ and $\vec{v}$ given by $\vec{u} \cdot \vec{v} = \sum_{i=1}^{p} u_i v_i$,

i.e., $W^* = \{\vec{u} \in \mathbb{R}^p \mid \vec{u} \cdot \vec{v} \geq 0 \; \forall \vec{v} \in W\}$. The set W^* is called dual of W.

The following standard result from [Güler, 2010] (Page 151, Section 6.3, Chapter 6) states that a finitely generated convex cone is always closed and the dual of the dual of closed convex cone is the convex cone itself.

Proposition 3 » Dual of the dual is primal

If $W \subseteq \mathbb{R}^p$ finite set then $\boldsymbol{C}(W)$ is topologically closed and $W^{**} = \boldsymbol{C}(W)$, where $W^{**} = (W^*)^*$.

Suppose that $\Theta = \{(\vec{u_i}, \vec{v_i}) \mid \vec{u_i}, \vec{v_i} \in \mathbb{R}^p, i = 1, \ldots, k\}$ is the set of k input preferences. Then we define the set $W_\Theta = \{\vec{u_i} - \vec{v_i} \mid (\vec{u_i}, \vec{v_i}) \in \Theta, i = 1, \ldots, k\}$. Using the results in Lemma 6, the set $\mathbf{C}(W_\Theta)$ defined as the set of all elements in $\mathbb{R}^p$ that weakly-Pareto dominates some positive linear combination of W_Θ, i.e.,

$$\mathbf{C}(W_\Theta) = \{\vec{u} \in \mathbb{R}^p \mid \exists q_1, \ldots, q_k \in \mathbb{R}^+ \cup \{0\} \text{ such that } \vec{u} \geq \sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i})\}$$

is a *positive convex cone* (by Lemma 6). We now define the relation $\succcurlyeq^\Theta$ based on the positive convex cones as:

Definition 28 » Positive convex cone-based relation

If Θ be the finite set of input preferences then we define the relation $\succcurlyeq^\Theta$ by $\vec{u} \succcurlyeq^\Theta \vec{v}$ if and only if $\vec{u} - \vec{v} \in \boldsymbol{C}(W_\Theta)$.

The following result gives some important properties of the relation $\succcurlyeq^\Theta$.

Proposition 4 » Properties of $\succcurlyeq^\Theta$

$\succcurlyeq^\Theta$ is a pre-order that extends Θ, extends Pareto, and satisfies Scale-Invariance and Independence.

Proof. Let $\mathbf{C}(W_\Theta)$ be the positive convex cone generated by W_Θ. Then by Definition 28, we have $\vec{u} \succcurlyeq^\Theta \vec{v}$ if and only if $\vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$.

(i) Let $\vec{u} \in \mathbb{R}^p$. By Lemma 6(i) we have $\vec{0} \in \mathbf{C}(W_\Theta)$ and $\vec{0} = \vec{u} - \vec{u} \Longrightarrow \vec{u} \succcurlyeq^\Theta \vec{u}$, showing that $\succcurlyeq^\Theta$ is reflexive.

(ii) Suppose that $\vec{u} \succcurlyeq^\Theta \vec{v}$ and $\vec{v} \succcurlyeq^\Theta \vec{w}$ then $\vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$ and $\vec{v} - \vec{w} \in \mathbf{C}(W_\Theta)$. Lemma 6(ii) implies that $\vec{u} - \vec{w} = (\vec{u} - \vec{v}) + (\vec{v} - \vec{w}) \in \mathbf{C}(W_\Theta)$, i.e., $\vec{u} \succcurlyeq^\Theta \vec{w}$, showing that $\succcurlyeq^\Theta$ is transitive. Hence $\succcurlyeq^\Theta$ is a pre-order.

(iii) To show that $\succcurlyeq^\Theta$ extends Θ we need to show that $\vec{u} \succcurlyeq^\Theta \vec{v}$ holds for any pair $(\vec{u}, \vec{v}) \in \Theta$. Let $(\vec{u}, \vec{v}) \in \Theta$ then to prove $\vec{u} \succcurlyeq^\Theta \vec{v}$ we need to show $\vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$. By definition $\vec{u} - \vec{v} \in W_\Theta$ and by Lemma 6(iii) the proof immediately follows.

(iv) Consider any $\vec{u}, \vec{v} \in \mathbb{R}^p$ and assume that $\vec{u} \geq \vec{v}$. To prove that $\succcurlyeq^\Theta$ extends Pareto we need to show $\vec{u} \succcurlyeq^\Theta \vec{v}$, i.e., we need to show that $\vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$. Now, $\vec{u} \geq \vec{v} \Longrightarrow \vec{u} - \vec{v} \geq \vec{0}$, it follows immediately using Lemma 6(i) that $\vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$. Thus $\succcurlyeq^\Theta$ extends the Pareto.

(v) Suppose that $\vec{u} \succcurlyeq^\Theta \vec{v}$ then we have $\vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$, and let $\vec{w}$ be any arbitrary vector in $\mathbb{R}^p$. $(\vec{u}+\vec{w})-(\vec{v}+\vec{w}) = \vec{u}-\vec{v} \in \mathbf{C}(W_\Theta)$, i.e., $\vec{u}+\vec{w} \succcurlyeq^\Theta \vec{v}+\vec{w}$, showing that $\succcurlyeq^\Theta$ satisfies Independence.

(vi) Let $q \geq 0$ be a real number and assume that $\vec{u} \succcurlyeq^\Theta \vec{v}$. Then we have $\vec{u}-\vec{v} \in \mathbf{C}(W_\Theta)$. By Lemma 6(iv), it follows that $q\vec{u}-q\vec{v} = q(\vec{u}-\vec{v}) \in \mathbf{C}(W_\Theta)$, i.e., $q\vec{u} \succcurlyeq^\Theta q\vec{v}$, showing that $\succcurlyeq^\Theta$ satisfies Scale-Invariance.

□

In the following section we look at the relationship between the relations $\succcurlyeq^\Theta$ and $\succcurlyeq'_\Theta$.

The relationship between $\succcurlyeq^\Theta$ and $\succcurlyeq'_\Theta$

We show now that the cone-based partial order $\succcurlyeq^\Theta$ is equivalent to the induced preference relation $\succcurlyeq'_\Theta$. In particular, we present a theorem which basically gives a way to check the dominance between the utility vectors in $\mathbb{R}^p$ with respect to the decision maker's revealed preferences Θ.

The following result shows that the relation $\succcurlyeq^\Theta$ is a subset of the relation $\succcurlyeq'_\Theta$.

Proposition 5 » $\succcurlyeq^{\Theta}$ is a subset of $\succcurlyeq'_{\Theta}$

Let Θ be the finite set of input preferences and $\vec{u}, \vec{v} \in \mathbb{R}^p$ be any two arbitrary vectors. If $\vec{u} \succcurlyeq^{\Theta} \vec{v}$ then $\vec{u} \succcurlyeq'_{\Theta} \vec{v}$.

Proof. Let $\Theta = \{(\vec{u_i}, \vec{v_i}) \mid \vec{u_i}, \vec{v_i} \in \mathbb{R}^p, i = 1, \ldots, k\}$ be the set of input preferences.

Suppose that $\vec{u} \succcurlyeq^{\Theta} \vec{v}$. To prove $\vec{u} \succcurlyeq'_{\Theta} \vec{v}$, it is sufficient to show that $\vec{u} \succcurlyeq \vec{v}$ holds for any pre-order that extends Θ, extends Pareto, and satisfy Scale-Invariance and Independence.

If $\vec{u} \succcurlyeq^{\Theta} \vec{v}$ then by definition $\vec{u} - \vec{v} \in \mathbf{C}(W_{\Theta})$ then there exists positive real scalars $q_1, \ldots, q_k$ and given $\vec{u_i} - \vec{v_i}, i = 1, \ldots, k$ such that $\vec{u} - \vec{v} \geq \sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i})$. Let us suppose that $\succcurlyeq$ is a pre-order that extends Θ, extends Pareto, and satisfies Scale-Invariance and Independence.

Since $\succcurlyeq$ extends Θ, then we have for all $i = 1, \ldots, k$, $\vec{u_i} \succcurlyeq \vec{v_i}$. Scale-Invariance implies that $q_i\vec{u_i} \succcurlyeq q_i\vec{v_i}$. Independence property and Lemma 3 imply that $\sum_{i=1}^{k} q_i\vec{u_i} \succcurlyeq \sum_{i=1}^{k} q_i\vec{v_i}$. Lemma 2 implies that $\sum_{i=1}^{k} q_i\vec{u_i} - \sum_{i=1}^{k} q_i\vec{v_i} \succcurlyeq \vec{0}$, i.e., $\sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i}) \succcurlyeq \vec{0}$. Since we have $\vec{u} - \vec{v} \geq \sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i})$, then transitivity implies that $\vec{u} - \vec{v} \succcurlyeq \vec{0}$. Lemma 2 implies that $\vec{u} \succcurlyeq \vec{v}$, as required.

□

Proposition 6 » The relations $\succcurlyeq^{\Theta}$ and $\succcurlyeq'_{\Theta}$ are identical

---

For any finite Θ, we have $\succcurlyeq^{\Theta} = \succcurlyeq'_{\Theta}$.

Proof. By Proposition 4, $\succcurlyeq^{\Theta}$ is a pre-order that extends Θ, extends Pareto, and satisfies Scale-Invariance and Independence. The definition of $\succcurlyeq'_{\Theta}$ then entails that $\succcurlyeq'_{\Theta} \subseteq \succcurlyeq^{\Theta}$.

Proposition 5 implies that $\succcurlyeq^{\Theta} \subseteq \succcurlyeq'_{\Theta}$. Thus $\succcurlyeq^{\Theta} = \succcurlyeq'_{\Theta}$.

□

In the following section we discuss the relationship between the relations $\succcurlyeq_\Theta$ and $\succcurlyeq^2_\Theta$.

The relationship between $\succcurlyeq_\Theta$ and $\succcurlyeq^2_\Theta$

The following result shows that the induced preference relation $\succcurlyeq_\Theta$ is equivalent to the MAUT-based total pre-orders extending Θ, $\succcurlyeq^2_\Theta$, which is given in Definition 27.

Proposition 7 » The relations $\succcurlyeq_\Theta$ and $\succcurlyeq^2_\Theta$ are equal

---

$\vec{u} \succcurlyeq^2_\Theta \vec{v} \iff \vec{u} - \vec{v} \in \boldsymbol{C}(W_\Theta)$. Therefore, if Θ is consistent then the relations $\succcurlyeq_\Theta$ and $\succcurlyeq^2_\Theta$ are equal.

Proof. Let $\vec{u} = (u_1, \ldots, u_p)$ and $\vec{v} = (v_1, \ldots, v_p)$ be two p-dimensional utility vectors. Any MAUT-based order $\succcurlyeq$ on $\mathbb{R}^p$ has an associated weights vector $\vec{w} = (w_1, \ldots, w_p)$, so we write $\succcurlyeq$ as $\succcurlyeq_{\vec{w}}$. We then have $\vec{u} \succcurlyeq_{\vec{w}} \vec{v} \iff (\vec{u} - \vec{v}) \cdot \vec{w} \geq \vec{0}$. We say that $\succcurlyeq_{\vec{w}}$ extends Θ if and only if, for all $\vec{u} \in W_\Theta, \vec{w} \cdot \vec{u} \geq \vec{0}$, i.e., iff $\vec{w} \in (W_\Theta)^*$. Thus,

$$\begin{aligned}
\vec{u} \succcurlyeq^2_\Theta \vec{v} &\iff (\vec{u} - \vec{v}) \cdot \vec{w} \geq 0, \text{ for all } \vec{w} \in (W_\Theta)^* \\
&\iff \vec{u} - \vec{v} \in (W_\Theta)^{**} \\
&\iff \vec{u} - \vec{v} \in \mathbf{C}(W_\Theta) \text{ [by Proposition 3, } (W_\Theta)^{**} = \mathbf{C}(W_\Theta)] \\
&\iff \vec{u} \succcurlyeq_\Theta \vec{v}.
\end{aligned}$$

□

The following theorem gives a way to test the dominance between utility vectors of $\mathbb{R}^p$ with respect to the partial order $\succcurlyeq_\Theta$.

Theorem 1 » Testing the dominance between utility vectors

Let Θ be a consistent set of pairs of vectors in $\mathbb{R}^p$. Then $\vec{u} \succcurlyeq_\Theta \vec{v}$ if and only if $\vec{u} - \vec{v} \in \boldsymbol{C}(W_\Theta)$.

Proof. Suppose that Θ is consistent. Lemma 5(iv) implies that $\succcurlyeq'_\Theta = \succcurlyeq_\Theta$. Proposition 6 gives that $\succcurlyeq^\Theta = \succcurlyeq_\Theta$. This completes the proof.

□

Theorem 1 states that, for given any set of input preferences $\Theta = \{(\vec{u_i}, \vec{v_i}) \mid \vec{u_i}, \vec{v_i} \in \mathbb{R}^p, i = 1, \ldots, k\}$ checking if $\vec{u}$ dominates $\vec{v}$ with respect to the induced preference relation $\succcurlyeq_\Theta$, it is equivalent to check if $\vec{u} - \vec{v}$ is in the positive convex cone $\boldsymbol{C}(W_\Theta)$, which is equivalent of checking if there exists a set of non-negative real scalars $q_1, \ldots, q_k$ such that $\vec{u} - \vec{v} \geq \sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i})$.

In the following example we explain the procedure for testing the dominance between utility vectors with respect to the induced preference relation $\succcurlyeq_\Theta$, which is given by Theorem 1.

Example 14 » Cone-based dominance

Suppose $\Theta = \{((2,5),(3,3)),((5,2),(2,3))\}$ is the set of consistent input preferences. Then we obtain $W_\Theta = \{(-1,2),(3,-1)\}$. Suppose that we are given the utility vector, $\vec{u} = (7,2)$ and $\vec{v} = (2,3.5)$. Then using Theorem 1, $\vec{u} \succcurlyeq_\Theta \vec{v}$ if and only if $\vec{u} - \vec{v}$ is in the convex cone $\boldsymbol{C}(W_\Theta)$ generated by W_Θ. In other words, $\vec{u} \succcurlyeq_\Theta \vec{v}$ if and only if $\vec{u} - \vec{v}$ weakly-Pareto dominates some positive linear combination of elements of W_Θ, i.e., $\vec{u} \succcurlyeq_\Theta \vec{v}$ if and only if $\exists q_1, q_2 \geq 0$ such that $\vec{u} - \vec{v} \geq q_1(-1,2) + q_2(3,-1)$. Finally, $\vec{u} \succcurlyeq_\Theta \vec{v}$ if and only if there exists a solution for the linear inequalities: $5 \geq -q_1 + 3q_2$ and $-1.5 \geq 2q_1 - q_2$. The system in this case has a solution, e.g., $q_1 = 0.1$ and $q_2 = 1.7$, thus implying that $\vec{u} \succcurlyeq_\Theta \vec{v}$.

In Figure 4.1, the shaded area represents the positive convex cone $\boldsymbol{C}(W_\Theta)$. Since the relation $\succcurlyeq_\Theta$ extends Pareto, $\boldsymbol{C}(W_\Theta)$ always include the positive quadrant of the xy-plane. We can see that the point $\vec{u} - \vec{v}$ is in $\boldsymbol{C}(W_\Theta)$, thus $\vec{u} \succcurlyeq_\Theta \vec{v}$.

Using Theorem 1 and looking at the above example, to perform the dominance test between utility vectors with respect to $\succeq_\Theta$, i.e., to check if $\vec{u} \succcurlyeq_\Theta \vec{v}$ we need

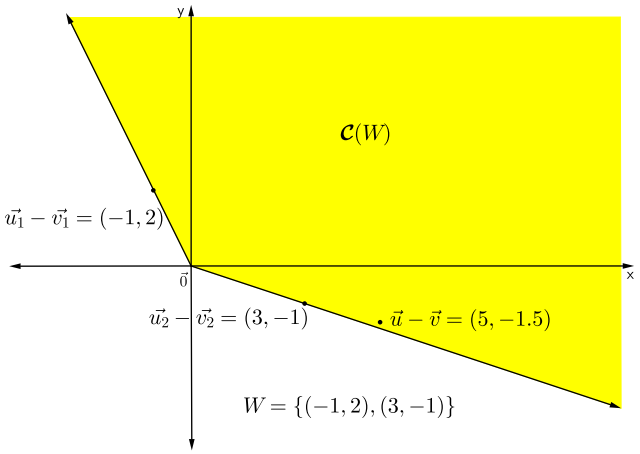


Figure 4.1: Convex cone of the preference information $\Theta = \{((2, 5), (3, 3)), ((5, 2), (2, 3))\}$.

to check the consistency of the system of linear inequalities, which can be done using a linear programming solver (e.g., lpsolve). Alternatively, we can use the (incomplete) algorithm of [Zheng and Chew, 2009], which is known as the *Distance Algorithm* (DA). The DA is introduced for calculating the distance between a vector and a convex cone in p-dimensional space. In the following section we discuss the use of DA for testing the dominance between utility vectors in $\mathbb{R}^p$.

Testing Dominance using Distance Algorithm

We implemented DA to calculate the distance between vector $\vec{u} - \vec{v}$ and the convex cone, $\mathbf{C}(W_\Theta)$, generated by $W_\Theta = \{\vec{u_i} - \vec{v_i} \mid (\vec{u_i}, \vec{v_i}) \in \Theta, i = 1, \ldots, k\}$ plus p unit vectors in $\mathbb{R}^p$. The algorithm depends on a very small positive real number ε, called the termination tolerance. If the distance between the point and the convex cone is zero or less than ε the algorithm results that the point is in the cone, i.e., the vector $\vec{u}$ dominates $\vec{v}$ with respect to $\succcurlyeq_\Theta$. Otherwise, the point is outside and the vector $\vec{u}$ does not dominate $\vec{v}$ with respect to $\succcurlyeq_\Theta$. Moreover, the distance algorithm takes a finite number of iterations to conclude position of the point with respect to the convex cone.

To describe the algorithm, let $W \subset \mathbb{R}^p$ and $\mathbf{C}(W)$ be the convex cone generated by W. Let $\vec{u}$ be the point that we want to locate its position (i.e., inside or outside) with respect to $\mathbf{C}(W)$. Let $\vec{v}$ be the closest point on $\mathbf{C}(W)$ to $\vec{u}$. The following function gives the maximum dot product value, computed between an element $\vec{r} \in \mathbb{R}^p$ and the elements of W:

$$h_W(\vec{r}) = \max_{\vec{w} \in W} \vec{w} \cdot \vec{r}$$

where, $\vec{w} \cdot \vec{r}$ is the dot product between vectors $\vec{w}$ and $\vec{r}$. The vector $\vec{r}$ is usually defined as $\vec{r} = \vec{u} - \vec{v}$ and its length gives the distance between the vectors $\vec{u}$ and $\vec{v}$. The function $s_W : \mathbb{R}^p \to W$, finds the vector $s_W(\vec{r}) \in W$ such that $h_W(\vec{r}) = s_W(\vec{r}) \cdot \vec{r}$, i.e., the function s_W finds a vector in W that gives the maximum dot product with $\vec{r}$. The vector $s_W(\vec{r})$ is not necessarily unique, but considering any one of them will be fine.

Let $\mathbf{C}(W_k) \subseteq \mathbf{C}(W)$ be the convex cone generated by $W_k = \{\vec{w_1}, \ldots, \vec{w_k}\} \subseteq W$. Let $\vec{v_k}$ be the closest vector on $\mathbf{C}(W_k)$ to $\vec{u}$ and $\vec{r_k} = \vec{u} - \vec{v_k}$ be the difference vector. Let $\hat{W}_k$ be a minimal subset of W_k such that $\vec{v_k}$ is a strictly positive combination of $\hat{W}_k$. In order to locate the position of $\vec{u}$ with respect to $\mathbf{C}(W)$, the algorithm performs the following steps:

(i) Set $\vec{v_0} = \vec{0}, \vec{r_0} = \vec{u}, W_0 = \emptyset, \hat{W}_0 = \emptyset$, and $k = 0$.

(ii) If $h_W(r_k) < 0$ then $\vec{u}$ is outside the cone $\mathbf{C}(W)$ or if $h_W(r_k) > 0$ and $h_W(r_k) < \epsilon$ then $\vec{u}$ is inside the cone $\mathbf{C}(W)$ and algorithm stops.

(iii) Compute $s_W(\vec{r_k})$ and set $W_{k+1} = \hat{W}_k \cup \{s_W(\vec{r_k})\}$.

(iv) Compute $c_{W_k} = (W_{k+1}^T W_{k+1})^{-1} W_{k+1}^T \vec{u}$. Partition the set W_{k+1} into W_{k+1}^-, W_{k+1}^0 and W_{k+1}^+, which consist of the vectors in W_{k+1}, to which the corresponding components of c_{W_k} are strictly negative, zero, and strictly positive, respectively.

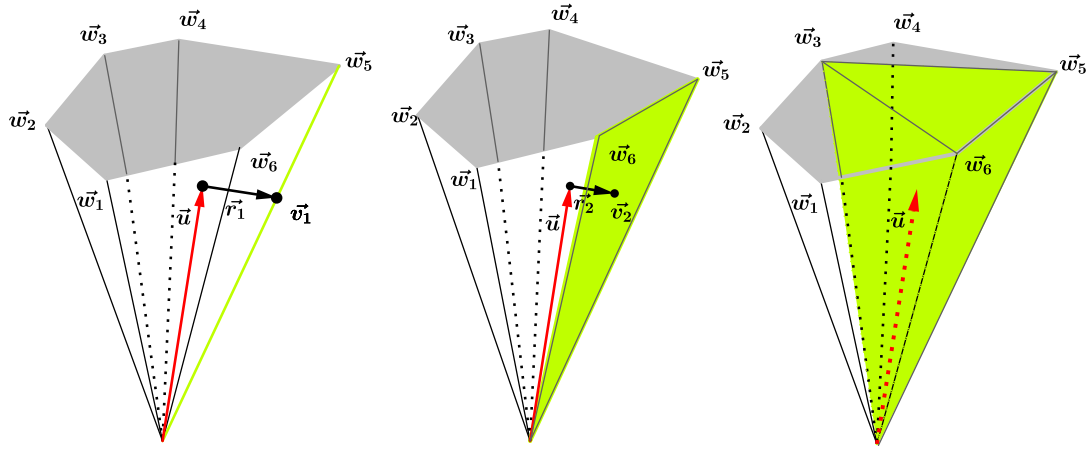
(v) If $W_{k+1}^- = \emptyset$ then $\vec{v_{k+1}} = W_{k+1} c_{W_k}$. Else if W_{k+1}^- is a singleton then remove W_{k+1}^- from W_{k+1} and go to (iv).

(vi) If W_{k+1}^- is not singleton then take W'_{k+1} to be a proper subset of W_{k+1} including $s_W(\vec{r_k})$ and excluding at least one point in W_{k+1}^-. If all subsets have been checked, then $\vec{v_{k+1}} = \vec{v_{W'_{k+1}}}$ and $\hat{W_{k+1}} = W'_{k+1}$.

(vii) $\vec{r_{k+1}} = \vec{u} - \vec{v_{k+1}}$, and $k = k + 1$. Return to (ii).

Example 15 » Distance Algorithm

To describe the DA, consider $W = \{\vec{w_1}, \vec{w_2}, \vec{w_3}, \vec{w_4}, \vec{w_5}, \vec{w_6}\} \subset \mathbb{R}^p$ and $C(W)$ be the pointed convex cone generated by W. Suppose that we are given a point $\vec{u} \in \mathbb{R}^p$ and let ϵ be the small positive number (e.g. 10^{-5}). Figure 4.2 shows the working procedure of the algorithm to locate the point $\vec{u}$ with respect to the cone $C(W)$. In first iteration (Figure 4.2(a)), algorithm finds that $\vec{w_5} \in W$ using the functions h_W and s_W, i.e., $h_W(\vec{u}) = s_W(\vec{u}) \cdot \vec{u}$, where $s_W(\vec{u}) = \vec{w_5}$. In this case we set $W_1 = \{w_5\}$. The coefficient vector c_{W_1} is calculated as $c_{W_1} = (W_1^T W_1)^{-1} W_1^T \vec{u}$ (step

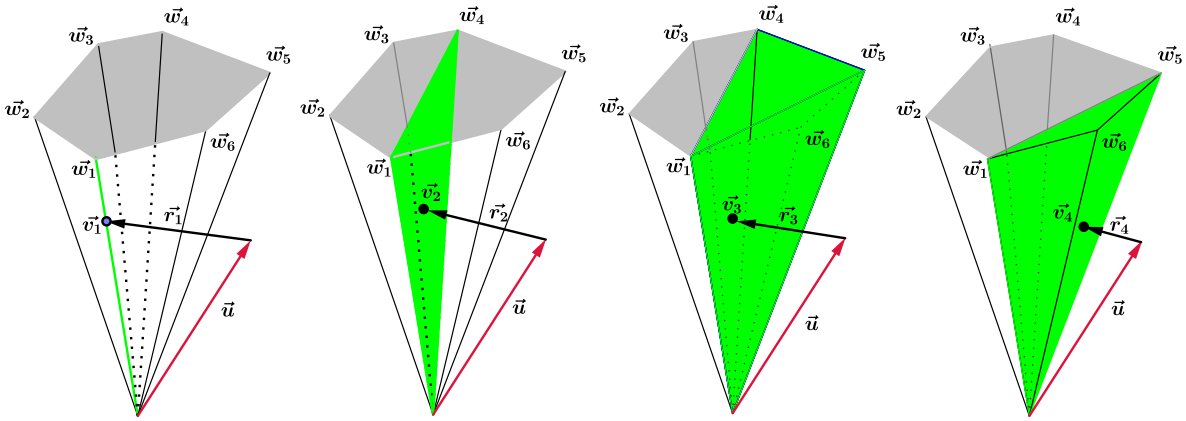


(a) First iteration (b) Second iteration (c) Third iteration

Figure 4.2: Distance Algorithm with three iterations to conclude that the point is inside the cone.

(iv)). The algorithm then computes the closest vector of $\vec{u}$*,* $\vec{v_1}$*, in first iteration, which is equal to* $\vec{v_1} = W_1 c_{W_1}$*. The distance of* $\vec{u}$ *with respect to* $C(W)$ *is the length of the residual vector,* $\vec{r_1} = \vec{u} - \vec{v_1}$*. In second iteration (Figure 4.2(b)), DA finds the vector* $\vec{w_6}$ *such that* $h_W(\vec{r_1}) = \vec{w_6} \cdot \vec{r_1}$*, i.e.,* $s_W(\vec{r_1}) = \vec{w_6}$*. We set* $W_2 = \hat{W}_1 \cup \{\vec{w_6}\} = \{\vec{w_5}, \vec{w_6}\}$*. The coefficient vector* c_{W_2} *is given by* $c_{W_2} = (W_2^T W_2)^{-1} W_2^T$*. The closest vector of* $\vec{u}$ *in the second iteration is constructed as* $\vec{v_2} = W_2 c_{W_2}$*. The updated shortest distance between* $\vec{u}$ *and* $C(W)$ *is given by the length of the new residual vector,* $\vec{r_2} = \vec{u} - \vec{v_2}$*. The algorithm moves to third iteration (Figure 4.2(c)), where it finds that* $h_W(\vec{r_2}) = \vec{w_3} \cdot \vec{r_2}$*, in this case* $s_W(\vec{r_2}) = \vec{w_3}$*. DA then sets* $W_3 = \hat{W}_2 \cup \{w_5, w_6\} = \{w_3, w_5, w_6\}$*. The coefficient vector in third iteration is computed as,* $c_{W_3} = (W_3^T W_3)^{-1} W_3^T \vec{u}$*. For simplicity assume that the elements of* c_{W_3} *are non-negative, otherwise, DA executes steps (iv) - (vi), to find the minimal set* $\hat{W}_3$ *from* W_3 *such that the closest vector* $\vec{v_3}$ *can be expresses as their positive linear combination. Since all components of* c_{W_3} *are non-negative, then we have* $\vec{v_3} = W_3 c_{W_3}$*. The residual vector is obtained as* $\vec{r_3} = \vec{u} - \vec{v_3}$*. In fourth iteration, algorithm finds that the value* $h_W(\vec{r_3})$ *is less than* ϵ*. Thus, DA discovers that the point* $\vec{u}$ *lies inside the cone* $C(W)$*, in particular, it concludes that* $\vec{u}$ *lies in the cone* $C(W_3)$ *generated by the elements of* W_3*.*

Figures in 4.3 display the case where the given point $\vec{u}$ *lies outside the cone* $C(W)$*. The algorithm takes four iterations (Figures 4.3(a), 4.3(b), 4.3(c) and 4.3(d)) to discover that the point* $\vec{u}$ *is outside* $C(W)$*. In fourth iteration, we have* $\hat{W}_4 = \{\vec{w_1}, \vec{w_5}, \vec{w_6}\}$*, the closest vector of* $\vec{u}$ *on* $C(W)$ *is* $\vec{v_4}$ *and the residual vector is* $\vec{r_4} =$



(a) First iteration (b) Second iteration (c) Third iteration (d) Fourth iteration

Figure 4.3: Distance Algorithm with four iterations to conclude that point is outside the cone.

$\vec{u} - \vec{v_4}$. *When processing next iteration, we obtain the negative value for* $h_W(r_4)$*, which indicates that the given point is outside the convex cone.*

For our experiments (in Section 5.5) on influence diagrams in the next chapter we set $\varepsilon = 10^{-7}$ and the experimental results indicate that the incompleteness of the algorithm does not appear to make a significant difference.

4.4.3 Implementing Matrix-based Dominance Test

Proposition 7 gives the equality of the two induced preference relations $\succcurlyeq_\Theta$ and $\succcurlyeq^2_\Theta$, whereas, Theorem 1 gives the condition for the dominance check with respect to $\succcurlyeq_\Theta$ i.e., for given any set of consistent preferences Θ in order to test whether utility vector $\vec{u}$ dominates the utility vector $\vec{v}$ we need to check if $\vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$, where W_Θ is given by $\{\vec{u_i} - \vec{v_i} : (\vec{u_i}, \vec{v_i}) \in \Theta\}$. As we have already seen that testing if $\vec{u} - \vec{v}$ is in $\mathbf{C}(W_\Theta)$ leads to solving a system of k linear inequalities which can be done using a linear program (LP) solver or by use of distance algorithm. Clearly, for multi-objective constraint optimization this can be an expensive task as we need to perform many dominance checks. In this section we present a procedure to compile this dominance check by use of a matrix which we construct using the input preferences Θ. Later we show that it greatly reduces the computational overhead.

Let $(\mathbf{C}(W_\Theta))^*$ be the dual cone of $\mathbf{C}(W_\Theta)$ which is defined as (also see Definition

12 in Section 3.2.1.1, Chapter 3):

$$(\mathbf{C}(W_\Theta))^* = \{\vec{u} \in \mathbb{R}^p \mid \sum_{i=1}^{p} \vec{u} \cdot \vec{v} \geq 0,\ \forall\ \vec{v} \in \mathbf{C}(W_\Theta)\}$$

Since $\mathbf{C}(W_\Theta)$ is finitely generated then $\mathbf{C}(W_\Theta)^*$ will also be a finitely generated set. Moreover, the set $(\mathbf{C}(W_\Theta))^*$ is characterized by its generators, i.e., the set of vectors in $\mathbb{R}^p$ which represent $(\mathbf{C}(W_\Theta))^*$. To find these set of generators we use the approach of [Tamura, 1976], where two different scenarios are presented for constructing the generators of the dual cone. In first case, the generators are constructed for a *pointed convex cone*, whereas the second case focuses on constructing generators for a *non-pointed convex cone*, where a pointed convex cone $\mathcal{C}$ has the property that, if $\vec{u} \in \mathcal{C}$ then $-\vec{u} \notin \mathcal{C}$. If $\mathcal{C}$ is not pointed then it is a non-pointed convex cone.

As shown in [Tamura, 1976], the procedure for constructing generators for pointed convex cones is much simpler than for non-pointed ones. We assume that the input preferences set Θ is consistent which then implies that $\mathbf{C}(W_\Theta)$ is pointed [Yu, 1974, Wiecek, 2007, Engau, 2008]. In addition, the result presented in Proposition 8 guarantees that Θ is consistent.

Once we find the set of generators of $(\mathbf{C}(W_\Theta))^*$ then we arrange them into the rows of a matrix $\mathbf{A}$ (the number of rows of $\mathbf{A}$ is then equal to the number of generators of $(\mathbf{C}(W_\Theta))^*$). The matrix $\mathbf{A}$ is in general rectangular form (i.e., the number of rows and columns are not necessarily equal in $\mathbf{A}$) because the number of generators of $(\mathbf{C}(W_\Theta))^*$ depends on the input preferences Θ [Dattorro, 2005]. The positive convex cone, $\mathbf{C}(W_\Theta)$, contains the set of vectors $\vec{u} \in \mathbb{R}^p$ such that $\mathbf{A}\vec{u} \geq \vec{0}$ (the utility vector $\vec{u}$ here is arranged into a column matrix of size $p \times 1$). The following result gives a way to test the dominance between utility vectors with respect to $\mathbf{A}$.

Lemma 7 » Dominance with respect to A

Suppose matrix A is such that $\vec{w} \in \mathbf{C}(W_\Theta) \iff \mathbf{A}\vec{w} \geq \vec{0}$. Then for any $\vec{u}, \vec{v} \in \mathbb{R}^p$, $\vec{u} \succcurlyeq_\Theta \vec{v}$ if and only if $\mathbf{A}\vec{u} \geq \mathbf{A}\vec{v}$ where $\geq$ is the weak Pareto order.

Proof. $\vec{u} \succcurlyeq_\Theta \vec{v} \iff \vec{u} - \vec{v} \in \mathbf{C}(W_\Theta) \iff \mathbf{A}(\vec{u} - \vec{v}) \geq \vec{0} \iff \mathbf{A}\vec{u} \geq \mathbf{A}\vec{v}$.

□

The following result illustrates that the matrix **A** represents the induced preference relation $\succcurlyeq_\Theta$. In fact, if **A** represents $\succcurlyeq_\Theta$ then the dual cone of $\mathbf{C}(W_\Theta)$ is equal to the cone generated by the rows of **A**.

Lemma 8 » A represents $\succcurlyeq_\Theta$

---

*Matrix **A** represents $\succcurlyeq_\Theta$ if and only if the dual cone $(\mathbf{C}(W_\Theta))^*$ is equal to the cone generated by the rows of **A**, i.e., every row of **A** is in $(\mathbf{C}(W_\Theta))^*$ and every element of $(\mathbf{C}(W_\Theta))^*$ is a positive convex combination of rows of **A**.*

Proof. Let R be the set of all rows of matrix **A**. Let $\mathbf{C}(R)$ be the convex cone generated by R then we have $\vec{u} \in (\mathbf{C}(R))^*$ if and only if $\mathbf{A}\vec{u} \geq 0$. If **A** represents the induced preference relation $\succcurlyeq_\Theta$ then it easily follows from Theorem 1 and Lemma 7 that for any $\vec{u} \in \mathbb{R}^p$, $\vec{u} \in \mathbf{C}(W_\Theta) \iff \mathbf{A}\vec{u} \geq \vec{0} \iff \vec{u} \in (\mathbf{C}(R))^*$. We have every element of $\mathbf{C}(W_\Theta)$ is also a member of $(\mathbf{C}(R))^*$ and vice versa, thus if **A** represents $\succcurlyeq_\Theta$ then $\mathbf{C}(W_\Theta) = (\mathbf{C}(R))^*$.

Now, $\mathbf{C}(W_\Theta) = (\mathbf{C}(R))^*$ implies that $(\mathbf{C}(W_\Theta))^* = (\mathbf{C}(R))^{**} = \mathbf{C}(R)$ by Proposition 3. Conversely, if $(\mathbf{C}(W_\Theta))^* = \mathbf{C}(R)$ then $\mathbf{C}(W_\Theta) = (\mathbf{C}(W_\Theta))^{**} = (\mathbf{C}(R))^*$.

Thus **A** represents $\succcurlyeq_\Theta$ if and only if $(\mathbf{C}(W_\Theta))^* = (\mathbf{C}(R))^*$.

□

Figure 4.4 compares the experimental results between matrix based and the linear program (LP) based dominance checks performed on the randomly generated influence diagrams with 3 and 5 objectives respectively. For these problem instances the number of decision variables is fixed to 5. The experiments were conducted on randomly generated 100 problem instances. Each data point on the graphs represents an average over the number of instances solved by both methods. The results show that the former method clearly dominates the latter one. We can clearly see that the former method is almost one order of magnitude faster than the latter.

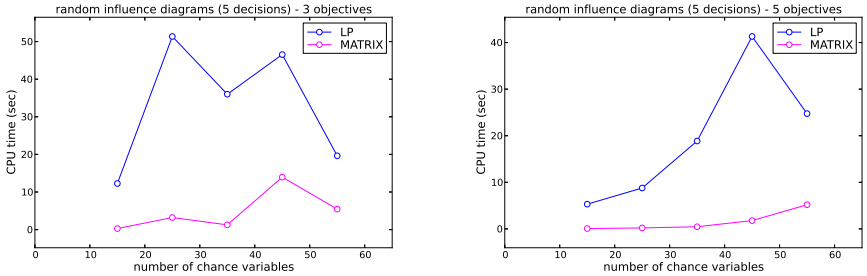


Figure 4.4: Comparing matrix based versus LP based dominance checks on influence diagrams with 3 and 5 objectives. CPU time in seconds as a function of problem size. Time limit 20 minutes.

In the following example we present the representation of preferences discussed above and also discuss preference representation in matrix form.

Example 16 » Matrix-based dominance test

Suppose that the decision maker reveals a unit of second objective is more valuable than a unit of first objective, i.e., $(0,1)$ *is preferred over* $(1,0)$*. With just this information we have* $\Theta = \{((0,1),(1,0))\}$*. Let* $\succcurlyeq_\Theta$ *be the induced preference relation. In addition,* $\succcurlyeq_\Theta$ *extends the Pareto ordering and satisfies the Scale-Invariance and Independence properties. The Independence property implies that* $(0,1)+(-1,0) \succcurlyeq_\Theta (0,1)+(0,-1)$*, i.e.,* $(-1,1) \succcurlyeq_\Theta (0,0)$*. We have* $W_\Theta = \{(-1,1)\}$ *and by Lemma 6,* $\boldsymbol{C}(W_\Theta)$ *is the (positive) convex cone generated by* W_Θ *and the unit vectors of* $\mathbb{R}^2$*. In Figure 4.5,* $\boldsymbol{C}(W_\Theta)$ *is represented by the shaded (grey and blue coloured) region which is generated by the vectors* $(-1,1)$ *and* $(1,0)$*.* $(\boldsymbol{C}(W_\Theta))^*$ *is the dual cone of* W_Θ *and using the approach of [Tamura, 1976], we obtain its generators* $(0,1)$ *and* $(1,1)$*. In Figure 4.5, the dual cone* $(\boldsymbol{C}(W_\Theta))^*$ *is represented by the convex region (only blue coloured surface) generated by its generators. We arrange these generators into the rows of a matrix* $\boldsymbol{A}$*, thus*

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

Using Lemma 7, the induced preference relation $\succcurlyeq_\Theta$ *is given by* $\vec{u} \succcurlyeq_\Theta \vec{v}$ *if and only if* $\boldsymbol{A}(\vec{u}-\vec{v}) \geq \vec{0}$*, where* $\vec{0}$ *is the zero vector and* $\vec{u}-\vec{v}$ *is arranged into a column vector.*

For instance, let $\vec{u} = (3,37)$ *and* $\vec{v} = (12,37)$ *be two utility vectors, then* $\vec{u}-\vec{v} = (-9,14)$*. To test whether* $\vec{u}$ $\succcurlyeq_\Theta$*-dominates* $\vec{v}$*, consider the product of the matrix* $\boldsymbol{A}$

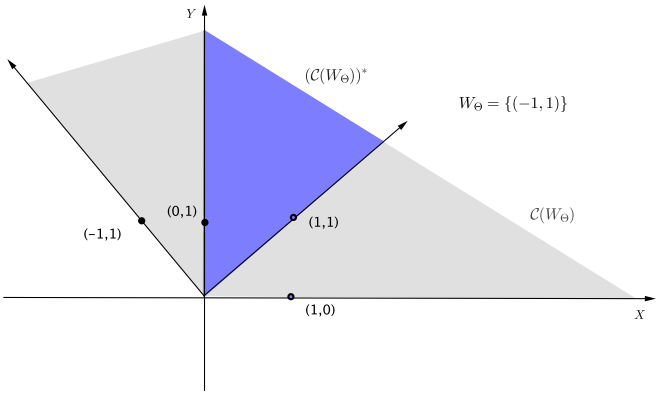


Figure 4.5: Dual cone.

and $\vec{u} - \vec{v}$*, i.e.,*

$$\begin{aligned} \boldsymbol{A} \cdot (\vec{u} - \vec{v}) &= \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} -9 \\ 14 \end{pmatrix} \\ &= \begin{pmatrix} 14 \\ 5 \end{pmatrix} \\ &\geq \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \vec{0}. \end{aligned}$$

Thus, $\vec{u} \succcurlyeq_\Theta \vec{v}$*.*

With Pareto ordering, the MOCOP problem given in Example 5 in Section 3.3.2 (Chapter 3), has the set $\{(14,6),(13,8),(3,24),\ (10,16),(9,19),$ $(8,21),(11,14),(12,12)\}$ *of Pareto optimal (or undominated) solutions corresponding to the value assignments* (00010)*,* (00011)*,* (01100)*,* (01010)*,* (01110)*,* (01111)*,* (11010)*, and* (11110) *to the variables* $\{X_0, X_1, X_2, X_3, X_4\}$ *of the problem, where* (00010) *represents the value assignment* $X_0 = 0, X_1 = 0, X_2 = 0, X_3 = 1$ *and* $X_4 = 0$*. With the additional preference information* $\Theta = \{((0,1),(1,0))\}$*, we obtain that* $(3,24)$ *and* $(8,21)$ *are the only* $\succcurlyeq_\Theta$*-undominated solutions. This illustrates that even a single trade-off can greatly reduce the number of undominated solutions. In Figure 4.6, we show both* $\succcurlyeq_\Theta$*-dominated (red colour) and* $\succcurlyeq_\Theta$*-undominated (blue colour) solutions.*

4.4.4 Testing Consistency

We need to check whether the decision maker is consistent with his preferences because he may state preference values which can conflict with his own

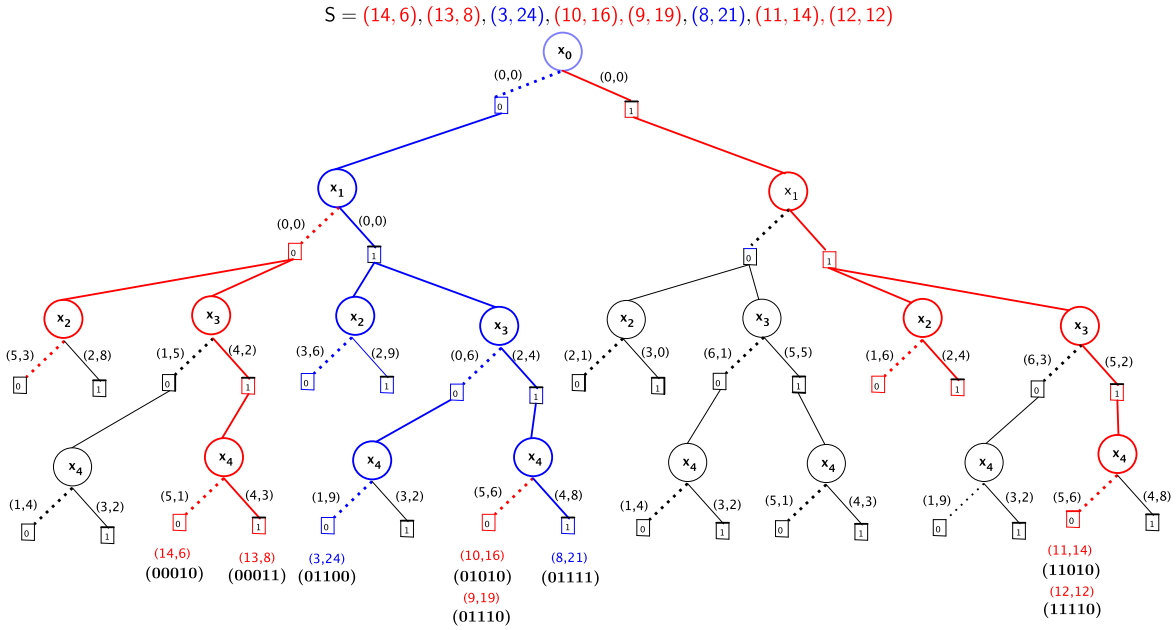


Figure 4.6: A MOCOP problem (given in Example 5 in Section 3.3.2, Chapter 3) with $\succcurlyeq_\Theta$-dominance.

preferences stated earlier. For instance, in a hotel booking scenario, one's preferences for the selected hotel can conflict with his limited budget. In many of the multi-objective problem settings it is a common challenge that the revealed preferences should maintain their consistency.

Suppose that Θ is the set of decision maker's input preferences. The following result gives a sufficient condition for Θ to be consistent.

Proposition 8 » Sufficient condition for consistency

Set of input preferences $\Theta = \{(\vec{u_i}, \vec{v_i}) \mid \vec{u_i}, \vec{v_i} \in \mathbb{R}^p, i = 1, \ldots, k\}$ *is consistent if there exist real scalars* w_j $(j = 1, \ldots, p)$ *such that the linear inequalities*

$$\textit{for all vectors } i = 1, \ldots, k; \sum_{j=1}^{p} w_j(u_{i(j)} - v_{i(j)}) > 0 \qquad (4.3)$$

hold (where, $u_{i(j)}$ is the j^{th} component of $\vec{u_i}$).

Proof. Given $\Theta = \{(\vec{u_i}, \vec{v_i}) \mid \vec{u_i}, \vec{v_i} \in \mathbb{R}^p, i = 1, \ldots, k\}$ is a set of input preferences.

Let $W_\Theta = \{\vec{u_i} - \vec{v_i} \ : \ \vec{u_i}, \vec{v_i} \in \mathbb{R}^p, i = 1, \ldots, k\}$ and $\mathbf{C}(W_\Theta)$ be the set of all positive linear combination of the elements of W_Θ, i.e.,

$$\mathbf{C}(W_\Theta) = \{\vec{u} \in \mathbb{R}^p \mid \exists q_1, \ldots, q_k \in \mathbb{R}^+ \cup \{0\} \text{ such that } \vec{u} \geq \sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i})\}$$

By Lemma 6, we have that $\mathbf{C}(W_\Theta)$ is a positive convex cone containing W_Θ. By Definition 28, the ordering $\succcurlyeq^\Theta$ is defined on $\mathbb{R}^p$ by: $\vec{u} \succcurlyeq^\Theta \vec{v} \iff \vec{u} - \vec{v} \in \mathbf{C}(W_\Theta)$.

By Proposition 4 the relation $\succcurlyeq^\Theta$ is a pre-order extending Θ, extending Pareto and satisfying Scale-Invariance and Independence properties.

Suppose that vector $\vec{w}$ satisfies (4.3), i.e., $\vec{w} \cdot (\vec{u_i} - \vec{v_i}) > 0$ for all $i = 1, \ldots, k$.

The definition of $\mathbf{C}(W_\Theta)$ implies that for all non-zero elements $\vec{z} \in \mathbf{C}(W_\Theta)$, $\vec{w} \cdot \vec{z} > 0$. Since $\vec{z}$ is a positive linear combination of the vectors $(\vec{u_i} - \vec{v_i})$, for all $i = 1, \ldots, k$. This implies that if $\vec{z}$ and $-\vec{z}$ are both in $\mathbf{C}(W_\Theta)$ then $\vec{z} = \vec{0}$: since otherwise $\vec{w} \cdot \vec{z} > 0$ and $\vec{w} \cdot (-\vec{z}) > 0$, which is a contradiction since $\vec{w} \cdot (-\vec{z}) = -(\vec{w} \cdot \vec{z}) < 0$.

This implies that $\succcurlyeq^\Theta$ is a partial order: since if $\vec{u} \succcurlyeq^\Theta \vec{v}$ and $\vec{v} \succcurlyeq^\Theta \vec{u}$ then $\vec{u} - \vec{v}$ and $\vec{v} - \vec{u}$ are in $\mathbf{C}(W_\Theta)$, which implies that $\vec{u} - \vec{v} = \vec{0}$, thus $\vec{u} = \vec{v}$.

This implies that Θ is consistent.

□

To check the satisfiability of the linear inequalities in (4.3), we can make use of a linear programming solver (e.g., lpsolve). Since we can use a linear programming solver, we can replace zero on the right-hand side of system of linear inequalities in equation (4.3) with a small positive number δ and also add constraints relating to the positivity condition for each w_j ($j = 1, \ldots, p$), i.e., we replace the system of linear inequalities in equation (4.3) with the following system.

$$\text{for all } i = 1, \ldots, k; \sum_{j=1}^{p} w_j(u_{i(j)} - v_{i(j)}) \geq \delta \text{ and}$$
$$\text{for all } j = 1, \ldots, p; w_j \geq \epsilon, (\text{ for some } \epsilon > 0)$$

Since δ doesn't effect the satisfiability of the system (4.3), we can use, e.g., $\delta = 1$. Also, we can choose, e.g., $\epsilon = 10^{-6}$ and the objective function we can set as, minimization of $\sum_{j=1}^{p} w_i$. In summary, to check whether the set of

input preferences Θ is consistent, we check the feasibility of the following linear program by using a linear programming solver:

$$\text{Minimize } \sum_{j=1}^{p} w_j \tag{4.4}$$

subject to:

$$\text{for all } i = 1, \ldots, k; \sum_{j=1}^{p} w_j(u_{i(j)} - v_{i(j)}) \geq \delta$$

$$\text{for all } j = 1, \ldots, p; w_j \geq \epsilon$$

We make use of above result to validate the user preferences and to generate the random trade-offs for experimental evaluation.

If the input preferences Θ is not consistent then the induced preference relation $\succcurlyeq^{\Theta}$ is not a partial order, consequently, it implies that everything else is preferred to everything, i.e., $\vec{u} \succcurlyeq^{\Theta} \vec{v}$ and $\vec{v} \succcurlyeq^{\Theta} \vec{u}$ for any $\vec{u}, \vec{v} \in \mathbb{R}^p$.

4.5 Reducing the Upper Bound Sets

One of the important features of the MOAOBB (see Algorithm 1 in Section 4.3) is that it generates an upper bound at each node during the search for an optimal solution. If n is the current node then for a single objective problem the upper bound (generated by the mini-bucket heuristic estimates) is usually a single element whereas for the case with p objectives since the utility vectors are typically only partially ordered, it will be a subset of utility vectors in $\mathbb{R}^p$. The key property of the upper bound set is, any assignment below the current node is weakly dominated by some element of it. Formally, an upper bound in a multi-objective setting is defined as:

Definition 29 » Upper bound set

Let V be any finite subset of $\mathbb{R}^p$ then a set UB(V) is an upper bound set of V if $\forall \vec{v} \in V$, $\exists \vec{u} \in$ UB(V) such that $\vec{u}$ weakly dominates $\vec{v}$, i.e., $\vec{u} \succcurlyeq \vec{v}$.

During the computation, the sizes of the upper bound sets can become very large. Here we discuss the methods to restrict the cardinality of the upper bound sets without damaging their key upper bound property.

Let $B(\geq 1)$ be any positive integer. Suppose that one wants to restrict the upper

bound sets to have cardinality at most B at each node during the search. Let $\mathcal{U}$ be an upper bound set at some node. In this section, we develop methods that iteratively choose a subset of k elements $\{\vec{v_1}, \ldots, \vec{v_k}\} \subseteq \mathcal{U}$ and generate their upper bound denoted by $\vec{u}$. The selected subset of elements in $\mathcal{U}$ is then replaced with $\vec{u}$. We also remove the other elements in $\mathcal{U}$ that are dominated by newly added $\vec{u}$. We repeat this process until $\mathcal{U}$ contains at most B elements. The reduced $\mathcal{U}$ still satisfies the upper bound property. For the last iteration we choose $k = |\mathcal{U}| - B + 1$ to avoid removing too many elements so that the cardinality of the final upper bound set is closer to B. For the special case when $B = 1$ (i.e., choosing $k = |\mathcal{U}|$) we replace all the elements of $\mathcal{U}$ with an upper bound in one iteration.

In the following sections we will discuss in detail two cases, firstly no trade-offs (Pareto), and secondly when the decision maker provides some additional trade-offs. In each of these cases we make use of the Pareto least upper bound (PLUB) of the subset of elements $\{\vec{v_1}, \ldots, \vec{v_k}\}$, given by $\vec{v} = \max_{j=1}^{k} \vec{v_j}$, where the operation $\max$ is applied point-wise.

4.5.1 Pareto (no trade-offs) Case

When there are no additional trade-offs, we developed the following five different methods for reducing the upper bound sets during the search. In each of these methods we replace only two elements in every iteration, i.e., we set $k = 2$ for reducing the upper bound sets. In some of these methods we calculate the *Manhattan distance* and the *dot product* between two vectors $\vec{w_1}, \vec{w_2} \in \mathbb{R}^p$, where Manhattan distance is defined as, $MD(\vec{w_1}, \vec{w_2}) = \sum_{i=1}^{p} |w_{1_i} - w_{2_i}|$, and dot product is defined as, $DP(\vec{w_1}, \vec{w_2}) = \sum_{i=1}^{p} w_{1_i} w_{2_i}$, where w_{1_i} and w_{2_i} are the i^{th} components of utility vectors $\vec{w_1}$ and $\vec{w_2}$. In all these five methods we assume a finite subset $\mathcal{U}$ of $\mathbb{R}^p$ is an upper bound set that needs to be reduced.

Method 1 » Random-Manhattan (RM)

(i) Choose randomly $\vec{w_i} \in \mathcal{U}$ and locate an element $\vec{w_j} \in \mathcal{U}$ that minimizes the Manhattan distance from $\vec{w_i}$.

(ii) Remove $\vec{w_i}$ and $\vec{w_j}$ from $\mathcal{U}$ and add their Pareto least upper bound $\vec{v}$ to $\mathcal{U}$.

(iii) Remove the elements which are Pareto dominated by $\vec{v}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise go to (i).

In step (i), method **RM** randomly selects a utility vector $\vec{w}_i$ from $\mathcal{U}$ and finds a utility vector $\vec{w}_j$ in $\mathcal{U}$, which is close to $\vec{w}_i$ with respect to Manhattan distance. If there is more than one $\vec{w}_j$ then the method selects one of them randomly. In step (ii), **RM** replaces both $\vec{w}_i$ and $\vec{w}_j$ with a Pareto least upper bound $\vec{v}$ computed as $\vec{v} = \max(\vec{w}_i, \vec{w}_j)$, where $\max$ is applied point-wise. In step(iii), **RM** removes the elements from $\mathcal{U}$ that are dominated by $\vec{v}$. This process repeats until cardinality of the upper bound set $\mathcal{U}$ reduces to less than or equal to B.

Method 2 » Minimum-Manhattan (MM)

(i) Find $\vec{w}_i, \vec{w}_j \in \mathcal{U}$ that minimize the Manhattan distance, i.e., $MD(\vec{w}_i, \vec{w}_j) \leq MD(\vec{w}_t, \vec{w}_s)$ for all $\vec{w}_t, \vec{w}_s \in \mathcal{U}$, $s \neq t$.

(ii) Replace $\vec{w}_i$ and $\vec{w}_j$ from $\mathcal{U}$ and add their Pareto least upper bound $\vec{v}$.

(iii) Remove the elements which are Pareto dominated by $\vec{v}$ from $\mathcal{U}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise go to (i).

Method **MM** is similar to method **RM**, the only differences being in step (i), where method MM calculates Manhattan distance between every two elements of the upper bound set $\mathcal{U}$ and selects a pair of vectors $\vec{w}_i$ and $\vec{w}_j$ that have minimum Manhattan distance between them. In other words, method **MM** selects a pair of vectors that are close to each other with respect to Manhattan distance.

Method 3 » Dot Product (DP)

(i) Choose $\vec{w}_i, \vec{w}_j \in \mathbb{R}^p$ which maximize the dot product value, i.e., $DP(\vec{w}_i, \vec{w}_j) \geq DP(\vec{w}_t, \vec{w}_s)$ for all $\vec{w}_t, \vec{w}_s \in \mathcal{U}$, $s \neq t$.

(ii) Remove $\vec{w}_i$ and $\vec{w}_j$ from $\mathcal{U}$ and add their Pareto least upper bound $\vec{v}$.

(iii) Remove the elements which are Pareto dominated by $\vec{v}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise go to (i).

Method **DP** computes dot product values between every two elements of the upper bound set $\mathcal{U}$ and selects a pair that has maximum dot product value. The reason for choosing maximum dot product value is that it gives the vectors close to each other. If there are two or more pair of vectors having the maximum dot product value then the method selects randomly one of them.

Method 4 » Normalized Dot Product (NDP)

(i) Find $\vec{w_i}, \vec{w_j} \in \mathbb{R}^p$ which maximize the normalized dot product (NDP) value, where NDP between $\vec{w_i}$ and $\vec{w_j}$ is defined as:

$$NDP(\vec{w_i}, \vec{w_j}) = \frac{\sum_{k=1}^{p} w_{i_k} \cdot w_{j_k}}{\sqrt{\sum_{k=1}^{p} w_{i_k}^2} \sqrt{\sum_{k=1}^{p} w_{j_k}^2}}$$

(ii) Remove $\vec{w_i}$ and $\vec{w_j}$ from $\mathcal{U}$ and add their Pareto least upper bound $\vec{v}$.

(iii) Remove the elements which are Pareto dominated by $\vec{v}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise go to (i).

Method **NDP** is similar to method **DP**, the only difference is, it normalizes the vectors before calculating the dot product values.

Method 5 » Random (RND)

(i) Choose randomly $\vec{w_i}, \vec{w_j} \in \mathbb{R}^p$.

(ii) Remove $\vec{w_i}$ and $\vec{w_j}$ from $\mathcal{U}$ and add their Pareto least upper bound $\vec{v}$.

(iii) Remove the elements which are dominated by $\vec{v}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise go to (i).

In each iteration, method **RND** chooses the vectors $\vec{w_i}$ and $\vec{w_j}$ randomly from the upper bound set $\mathcal{U}$ and replace them with their Pareto least upper bound $\vec{v}$. Method **RND** is a simplest one, but because of the random selection, it might select a pair of vectors that are far from each other, whereas, all other methods try to find a pair of vectors close to each other.

Figure 4.7 shows the number of instances solved by MOAOBB using each of these methods out of 50 randomly generated MOCOP graphs. The upper bound set size B was set to 5, so that all the methods restrict the cardinality of the upper bound sets to at most 5 at each node during the search for optimal solution. It can be seen that there is no huge difference in the results; however **RM** is overall the best. Hereafter when $B > 1$ we only use **RM** to restrict the cardinality of the upper bound sets and we simply call this method as "PLUB". For the special case, when we set $B = 1$ to restrict the upper bound sets to singleton

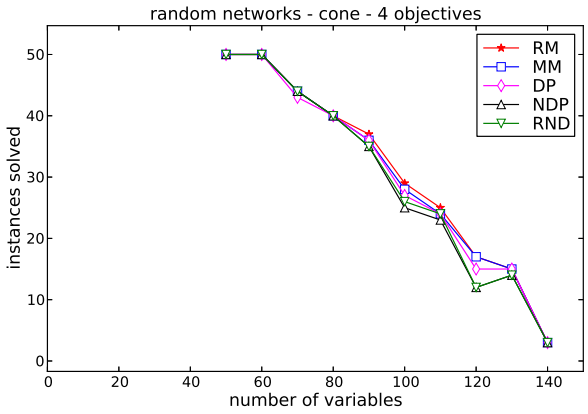


Figure 4.7: Number of problem instances solved for random networks with 4 objectives. Using different Pareto least upper bound based methods. Time limit 3 minutes.

sets, instead of removing only two elements in every iteration, we replace all the elements with their Pareto least upper bound in a single iteration.

The following example illustrates how PLUB will be applied in solving MOCOPs.

Example 17 » Methods for reducing the upper bound set (Pareto case)

Continuing Example 16, suppose that we are given an upper bound set $\mathcal{U} = \{(21,3),(3,15),(24,1)\}$ *and we want to reduce its size to 2, i.e., we set* $B = 2$. *Let* $\vec{w}_1 = (21,3), \vec{w}_2 = (3,15)$ *and* $\vec{w}_3 = (24,1)$. *Assume that PLUB randomly selects* $\vec{w}_1$ *and then obtains Manhattan distances* $MD(\vec{w}_1,\vec{w}_2) = 30$ *and* $MD(\vec{w}_1,\vec{w}_3) = 5$ *from utility vectors* $\vec{w}_2$ *and* $\vec{w}_3$*, respectively. Since the utility vector* $\vec{w}_3$ *minimizes the Manhattan distance from* $\vec{w}_1$*, PLUB replaces* $\vec{w}_1$ *and* $\vec{w}_3$ *with their Pareto least upper bound* $\vec{v} = (24,3)$ *from* $\mathcal{U}$. *Since* $(24,3) \not\succcurlyeq (3,15)$ *then the reduced upper bound set will then be* $\{(24,3),(3,15)\}$ *which still maintains the upper bound property.*

If we set $B = 1$*, i.e., reducing* $\mathcal{U}$ *to a singleton set, then PLUB replaces* $\vec{w}_1$*,* $\vec{w}_2$ *and* $\vec{w}_3$ *with their Pareto least upper bound* $\vec{v} = (24,15)$ *from* $\mathcal{U}$ *in a single iteration. The new upper bound set will then be* $\{(24,15)\}$.

4.5.2 Trade-offs Case

As described above, we assume that we are given a consistent set of input preferences Θ from which we derive a preference relation $\succcurlyeq_\Theta$. As discussed in Section 4.4.3, let us suppose matrix $\mathbf{A}$ represents $\succcurlyeq_\Theta$. Assume that we are producing an upper bound for a subset of utility vectors in $\mathcal{U}$ with respect to Θ.

We will make use of the following result to generate an upper bound $\vec{u}$ of the vectors $\vec{v_1}, \ldots, \vec{v_k}$ with respect to Θ.

Proposition 9 » Upper bound with respect to Θ

Let $\vec{v}_1, \ldots, \vec{v}_k$ be vectors in $\mathbb{R}^p$, and let $\vec{v} = \max_{j=1}^{k} \vec{v}_j$ be their Pareto least upper bound, and let $\vec{w} = \max_{j=1}^{k} \boldsymbol{A}\vec{v}_j$ (with $\max$ being applied point-wise in both cases). Then,

(i) $\vec{v} \succcurlyeq_\Theta \vec{v}_1, \ldots, \vec{v}_k$, i.e., the Pareto least upper bound is an upper bound with respect to $\succcurlyeq_\Theta$;

(ii) for $\vec{u} \in \mathbb{R}^p$, $\vec{u} \succcurlyeq_\Theta \vec{v}_1, \ldots, \vec{v}_k \iff \boldsymbol{A}\vec{u} \geq \vec{w}$; and

(iii) $\boldsymbol{A}\vec{v} \geq \vec{w}$.

Proof. (i): Since $\succcurlyeq_\Theta$ extends Pareto, also we have $\vec{v} \geq \vec{v_j}$, for all $j \in \{1, \ldots, k\}$ it follows immediately that $\vec{v} \succcurlyeq_\Theta \vec{v_j}$, for all $j \in \{1, \ldots, k\}$.

(ii): Let $\vec{u} \in \mathbb{R}^p$ then $\vec{u} \succcurlyeq_\Theta \vec{v_j}$ for all $j \in \{1, \ldots, k\} \iff$ (using Lemma 7) $\mathbf{A}\vec{u} \geq \mathbf{A}\vec{v_j}$ for all $j \in \{1, \ldots, k\} \iff \mathbf{A}\vec{u} \geq \max_{j=1}^{k} \mathbf{A}\vec{v_j} \iff \mathbf{A}\vec{u} \geq \vec{w}$.

(iii): From (i) we have $\vec{v} \succcurlyeq_\Theta \vec{v_j}$ for all $j \in \{1, \ldots, k\}$. It follows immediately from (ii) that $\mathbf{A}\vec{v} \geq \vec{w}$.

□

If $\succcurlyeq_\Theta$ is much stronger than the Pareto ordering $\geq$, then we can obtain a much tighter upper bound, which can lead to much stronger pruning. Solving a system of linear inequalities $\mathbf{A}\vec{u} \geq w$, given in Proposition 9(ii), may lead to an upper bound $\vec{u}$ with respect to $\succcurlyeq_\Theta$ such that $\vec{u} \geq \vec{v}$, or $\vec{u}$ and $\vec{v}$ may not be comparable. Thus, we may obtain an upper bound with respect to $\succcurlyeq_\Theta$ which is even worse than the Pareto least upper bound. For this reason, we add an additional constraint $\vec{u} \leq \vec{v}$, i.e., $\vec{u} \leq \max_{j=1}^{k} \vec{v_j}$, which guarantees that the generated upper bound with respect to the induced preference relation $\succcurlyeq_\Theta$ is tighter than the Pareto least upper bound. Thus the additional constraint on the upper bound with respect to $\succcurlyeq_\Theta$ may lead to stronger pruning.

To obtain an upper bound $\vec{u}$ with respect to $\succcurlyeq_\Theta$ we solve the following linear

program:

$$\text{Minimize } \sum_i u_i \text{ (where } u_i \text{ is the } i^{th} \text{ component of } \vec{u}) \tag{4.5}$$

subject to:

$$\mathbf{A}\vec{u} \geq \vec{w}$$
$$\vec{u} \leq \max_{j=1}^{k} v_j$$

The above linear program is feasible because Proposition 9(iii) shows that the system of linear inequalities are satisfiable, since $\vec{v}$ is a solution.

The following are the methods we tried with this approach, in all these methods k is the cluster size:

Method 1 » k-nearest neighbours (KNN)

(i) Choose an element $\vec{w}$ in $\mathcal{U}$ and find its $k-1$ nearest neighbours in $\mathcal{U}$ with respect to Manhattan distance.

(ii) Remove $\vec{w}$ and its $k-1$ nearest neighbours form $\mathcal{U}$ and add their upper bound $\vec{u}$ generated with respect to $\succcurlyeq_\Theta$ by using (4.5).

(iii) Remove the elements which are dominated by $\vec{u}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise go to (i).

In step(i), method **KNN** selects a utility vector $\vec{w}$ randomly from the upper bound set $\mathcal{U}$ and computes the Manhattan distance between $\vec{w}$ and the remaining elements of $\mathcal{U}$. It then selects $k-1$ utility vectors that have the smallest Manhattan distances from $\vec{w}$, this clearly gives the $k-1$ nearest neighbours of $\vec{w}$ with respect to Manhattan distance. In step (ii), the method replaces $\vec{w}$ and its $k-1$ nearest neighbours with the $\succcurlyeq_\Theta$-based upper bound, $\vec{u}$, computed by using (4.5). In step (iii), the method removes the elements from $\mathcal{U}$ that are dominated by the newly added upper bound $\vec{u}$. The method repeats these steps until the upper bound set $\mathcal{U}$ contains the at most B elements.

Method 2 » Random k-group (RKG)

(i) Choose randomly k elements $\vec{v_1}, \ldots, \vec{v_k}$ from $\mathcal{U}$.

(ii) Replace $\vec{v_1}, \ldots, \vec{v_k}$ with their upper bound $\vec{u}$ generated with respect to $\succcurlyeq_\Theta$ by using (4.5).

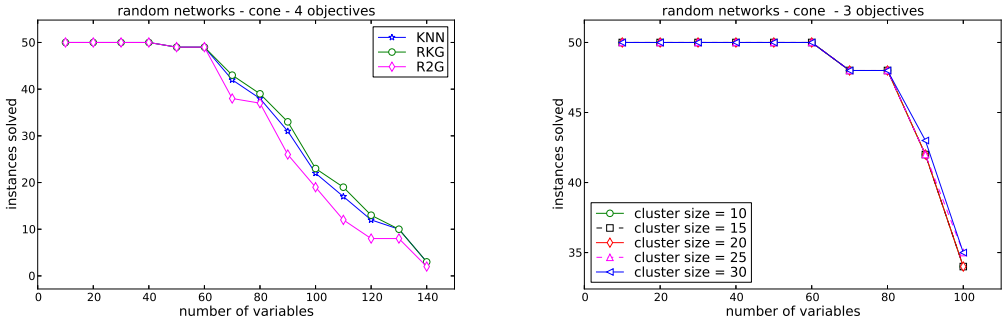


Figure 4.8: Number of problem instances solved for random networks with 4 objectives using different linear program based methods (left) and for random networks with 3 objectives using different cluster sizes (k) with method **RKG** (right). Time limit 3 minutes.

(iii) Remove the elements which are dominated by $\vec{u}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise go to (i).

Method **RKG** randomly selects k (cluster size) different elements from $\mathcal{U}$ (step (i)) and replaces them with the $\succcurlyeq_\Theta$ based upper bound, $\vec{u}$, computed using (4.5) (step (ii)). It also removes the elements in $\mathcal{U}$ dominated by $\vec{u}$. This process repeats until the upper bound set $\mathcal{U}$ contains at most B elements.

Method 3 » Random 2-group (R2G)

Choosing cluster size $k = 2$ in the above method we have the following.

(i) Set $k = 2$ and choose randomly two utility vectors $\vec{v_1}, \vec{v_2} \in \mathcal{U}$.

(ii) Replace $\vec{v_1}$ and $\vec{v_2}$ with their upper bound $\vec{u}$ generated with respect to $\succcurlyeq_\Theta$.

(iii) Remove the elements which are dominated by $\vec{u}$.

(iv) If $|\mathcal{U}| \leq B$ then stop. Otherwise, go to (i).

The above method **R2G** is obtained by setting the cluster size $k = 2$ in method **RKG**.

In Figure 4.8, the left-hand-side results show the number of problem instances solved for randomly generated MOCOP graphs with 4 objectives using all the three methods discussed above. Clearly, the results are similar, but for these instances **RKG** is the best. The right-hand-side results in Figure 4.8 for random graphs with 3 objectives show the number of problem instances solved by **RKG** with cluster sizes $k = 10, 15, 20, 25$ and 30. These results indicate that changing the cluster size does not really effect the number of instances solved much. In

the following section for the experiments with trade-offs we use **RKG** which we simply call "LP" and set the cluster size $k = 30$. For the special case, when $k = |\mathcal{U}|$, we replace all the elements of $\mathcal{U}$ with their upper bound generated with respect to $\succcurlyeq_\Theta$ in a single iteration.

The following example illustrates the procedure for generating upper bound with respect to induced preferences and how we apply LP method in solving MOCOPs.

Example 18 » Method for reducing upper bound set (Trade-offs case)

Continuing Example 17 in Section 4.5.1, suppose that we are given an upper bound set $\mathcal{U} = \{(21, 3), (3, 15), (24, 1)\}$*, where* $\vec{w}_1 = (21, 3), \vec{w}_2 = (3, 15)$ *and* $\vec{w}_3 = (24, 1)$*; and B = 2. For simplicity assume that the cluster size* $k = 2$*. Suppose LP randomly selects* $\vec{w}_1$ *and* $\vec{w}_2$*. As in Example 16, suppose we are given* $\Theta = \{(0, 1), (1, 0)\}$ *leading to the matrix*

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Let $\vec{u} = (u_1, u_2)$ *be* $\succcurlyeq_\Theta$*-based upper bound, we have* $\vec{v} = (21, 15)$ *is the Pareto least upper bound of* $\vec{w}_1$ *and* $\vec{w}_2$*. As described in Proposition 9, we get* $\vec{w} = \max_{j=1}^{2} \boldsymbol{A}\vec{w}_j = (15, 24)$*. To generate* $\vec{u}$ *we formulate and solve the following linear program:*

Minimize

$$u_1 + u_2$$

subject to the constraints:

$$\begin{aligned} u_2 &\geq 15 \\ u_1 + u_2 &\geq 24 \\ u_1 &\leq 21 \\ u_2 &\leq 15 \end{aligned}$$

Solving the above linear program (which in this case has a unique optimal solution) we get $\vec{u} = (9, 15)$*, which is clearly a much tighter upper bound than* $(21, 15)$*. We then replace* $\vec{w}_1$ *and* $\vec{w}_2$ *with* $\vec{u}$*. The reduced new upper bound set will then be* $\{(9, 15), (24, 1)\}$.

4.6 Experiments

In this Section, we present the experimental evaluation of our new approach to solve multi-objective constraint optimization problems. We focus mainly on three classes of standard MOCOP benchmarks namely *random networks*, *combinatorial auctions* and *vertex coverings*. The algorithms for generating the problem instances and the proposed enhancements in MOCOP algorithms were implemented in C++ (32 bits). The experiments were run on a 2.6GHz quad-core processor with 4 GB of RAM.

The random generators for generating the problem instances are as follows:

Random Networks

These graphs are basically characterized by two parameters, the number of variables (or vertices) n and the number of utility functions c. As described in [Marinescu, 2009, Rollón and Larrosa, 2006b] the instances are generated by randomly selecting the edges between the variables (or vertices), with values $n \in [10, 160]$ and $c = 1.6n$. The number of objectives are $2, 3, 4$ and 5, respectively and the objective values are uniformly distributed between 0 and 10. The induced width of these graphs range between 5 and 14, respectively.

Weighted Vertex Coverings

Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges between them, a vertex covering of G is a set of vertices $S \subseteq V$ such that each edge in E is incident to at least one vertex in S, i.e., $\forall (u, v) \in E$, either $u \in S$ or $v \in S$. Assume that each vertex $u \in V$ has an associated multi-attribute utility $U(u) = (u_1, \ldots, u_p)$ then the task is to find S such that for each edge $(u, v) \in E$, either $u \in S$ or $v \in S$, and $F(S) = \sum_{u \in S} U(u)$ is maximized.

Figure 4.9 shows a bi-attribute weighted graph G with six vertices $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and six edges $E = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5), (v_4, v_6)\}$. By definition, the vertex cover of graph G having the maximum sum of weights of its vertices is $S = \{v_1, v_3, v_4\}$ (orange coloured vertices), where the sum of the weights is given by $(3, 2) + (5, 6) + (6, 3) = (14, 11)$.

Following [Marinescu, 2009], we generated random vertex covers with $|V| = [10, 180]$ vertices, $|E| = 1.6V$ edges and having $2, 3, 4$ and 5 objectives. Problem

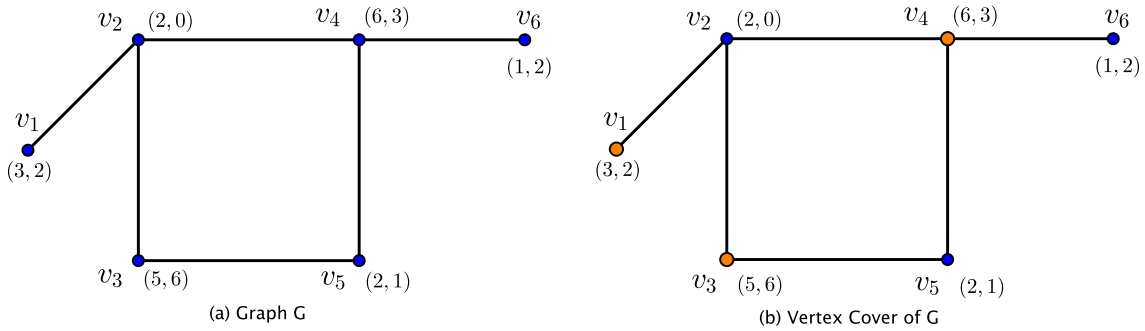


Figure 4.9: A bi-attribute weighted graph G (left) and its vertex cover (right)

instances were generated by randomly selecting E edges. The utility vector components were generated randomly between -10 between 0. The induced width of these graphs ranged between 9 and 25, respectively.

Combinatorial Auctions

In a combinatorial auction, an auctioneer wants to sell a set of distinguishable goods amongst a group of bidders, where the amount that the bidders are willing to pay is either unknown or unpredictably changeable over time [Leyton-Brown et al., 2000]. However, the auctioneer wants to allocate the goods in a way that maximizes his revenue (for instance, selling a pair of shoe gives more profit than the profit of selling a left shoe alone plus the profit of selling a right shoe alone). On the other hand, he wants to control the risk of not being paid after a bid has been accepted [Holland, 2005] and wants to improve the quality of services associated with the bids.

Let $\alpha = \{\alpha_1, \ldots, \alpha_l\}$ be the set of goods to be auctioned and $\beta = \{\beta_1, \ldots, \beta_m\}$ be the set of bids. For our experiments, we define each bid $\beta_j \in \beta$ as a quadruple $(s_{\beta_j}, r_{\beta_j}, p_{\beta_j}, q_{\beta_j})$, where

- s_{β_j} is the subset of goods that bid β_j is placed for, i.e., $s_{\beta_j} \subseteq \alpha$.
- r_{β_j} is the price of bid β_j.
- p_{β_j} is the probability of failing the payment upon acceptance of β_j.
- q_{β_j} is the quality of service measure associated with bid β_j.

Given that each good is allocated to at most one bid, the goal of the auctioneer is to determine the subset of winning bids that simultaneously maximize the profit, minimize the risk of not getting the full revenue and maximize the overall quality of the services represented by the selected bids.

For our experiments, we adopt the technique of generating auctions with 30 goods (in each bundle) and increasing number of bids from the *paths* distribution of the CATS (Combinatorial Auction Test Suite) given in [Leyton-Brown et al., 2000]. The probabilities of failing payments will be generated randomly between 0 and 0.3 and the quality of services associated with bids is uniformly at random between 1 and 10. The induced width of these graphs ranged between 6 and 61.

Consistent random trade-offs

We generate consistent random trade-offs, namely *two-way* and *three-way*, between the objectives defined as follows.

Two-way trade-offs: Let i, j be the different elements in $\{1, \ldots, p\}$ (where each element in the set is corresponding to an objective) and for $i = 1, \ldots, p$ let $\vec{e_i}$ be the unit vector in i^{th} direction in $\mathbb{R}^p$. We randomly choose real numbers $a, b, c \in [0.1, 1)$. Then we generate the vectors $a\vec{e_i} - b\vec{e_j}$ and $b\vec{e_j} - ac\vec{e_i}$, which we call two-way or binary trade-offs. These vectors correspond to the constraints $aw_i \geq bw_j$ and $bw_j \geq acw_i$, where we consider a vector of unknown weights $(w_1, \ldots, w_n)$ for the objectives. Moreover, the first vector represents that the decision maker is willing to gain a units of objective i at the cost of losing b units of objective j, and the second is vice versa. Note that the parameter c can be used to control the strength of the trade-offs. For instance, if $c = 1$ then the constraints on w_i and w_j imply $aw_i = bw_j$, So, the i and j objectives collapse into a single objective implying that we have precise rates of exchange between objectives i and j.

For example, if $p = 3$ and $i = 3$, $j = 1$ then we get the trade-off vectors $(-b, 0, a)$ and $(b, 0, -ac)$.

Three-way trade-offs: Let i, j and k be different elements in $\{1 \ldots, p\}$. We randomly choose real numbers a, b, c in $[0.1, 1)$ then we generate the trade-off vector $a\vec{e_i} + b\vec{e_j} - c\vec{e_k}$, which corresponds to the constraint $aw_i + bw_j \geq cw_k$; we call it as three-way trade-off between the objectives i, j and k. This represents the decision maker's willingness to gain a units of objective i and b units of objective j at the cost of losing c units of objective k.

For example, if $p = 3$ and $i = 2$, $j = 3$ and $k = 1$ then we get the three-way trade-off vector $(-c, a, b)$.

The algorithm RAND-TRADE-OFFS describes the whole process for generat-

Algorithm 2: RANDOM-TRADE-OFFS

```
1  if p = 2 then
2      C ← ∅;
3      r ← Rand(0, 1);
4      if r ≤ 0.5 then
5          generate trade-offs for the pair (1, 2);
6      else
7          generate trade-offs for the pair (2, 1);
8      place all trade-offs in C;
9  else if p > 2 then
10     repeat
11         C ← ∅;
           // generate binary trade-offs
12         for t = 1 to K do
13             pick random all different elements i, j in {1, ..., p};
14             generate binary trade-offs for (i, j);
           // generate 3−way trade-offs
15         for h = 1 to T do
16             pick random all different elements i, j, k in {1, ..., p};
17             generate 3−way "+ + −" trade-offs for (i, j, k);
18         place all trade-offs in C;
19     until Proposition 8 implies that C is consistent, i.e., the linear program (4.4)
       is feasible.;
20 return C;
```

ing consistent random trade-offs. The function $Rand(0, 1)$ generates a random real number between 0 and 1. For bi-objective case ($p = 2$), with chance 0.5 we generate trade-offs for the pair $(1, 2)$ of objectives. Otherwise, we generate trade-offs for the pair $(2, 1)$ of objectives (lines 1-8). For more than two objectives case, we consider two input parameters K and T (positive integers), and generate K binary (lines 12-14) and T 3-way random trade-offs (lines 15-17). In this case, we apply Proposition 8 to ensure that the generated random trade-offs are consistent (line 19).

For simplicity we use the following terminology in our experiments:

- **MOAOBB(i)** – is the multi-objective AND/OR branch and bound algorithm described in Section 4.3, where i is the i-bound size which will be passed as an argument to control the accuracy of the guiding heuristic. In [Marinescu, 2009] it was shown that the larger values of i typically give more accurate results but they are more expensive to compute.

- **B=b (PLUB)** – is the MOAOBB(i) that uses the Pareto least upper bound based method "PLUB" described above to reduce the upper bound sets described in Section 4.5.1 to at most $b(\geq 1)$ during the search at each node. This will be used for both Pareto and trade-offs cases.
- **B=b (LP)** – is the MOAOBB(i) that uses the $\succcurlyeq_\Theta$-based upper bound method "LP" described above to reduce the upper bound sets described in Section 4.5.2. This will be used for both Pareto and trade-offs cases. For all experiments we replace a cluster of 30 elements from the upper bound set in each iteration.
- **VE** – is the variable elimination algorithm given in [Marinescu et al., 2012] for solving multi-objective influence diagrams. We use the same algorithm to solve MOCOP instances. In contrast with a branch-and-bound algorithm which is linear in space, VE is time and space exponential in the underlying induced tree width of the problem instance.

All the algorithms we focus on here are restricted to a static variable ordering that is obtained using a depth-first traversal of the corresponding pseudo tree. The min-fill heuristic given in [Dechter and Mateescu, 2007b, Marinescu, 2009] is used to compute these pseudo trees. On the other hand, the AND/OR search algorithms use lexicographic order to order the sub-problems rooted at each node in the search tree.

Comparison with State of the Art Approaches:

Figure 4.10 shows the results obtained using the Pareto ordering for random networks with 5 objectives, combinatorial auctions with 3 objectives and vertex covering problems with 5 objectives, respectively. The data points on the graph represent the averages over 10 randomly generated instances. The mini-bucket i-bounds (see Section 3.3.4.1, Chapter 3) are 8, 10 and 12 for random networks, combinatorial auctions and vertex covering problems, respectively. The time allotted to solve each instance is 30 minutes. That means the instances which are not solved within 30 minutes will be labelled as unsolved.

Overall, for Pareto case, among all the algorithms, the algorithm using B = 1 (PLUB), the single element upper bound set, is the best and it outperforms significantly the state-of-the-art MOAOBB across all problem instances. The B = 2 (PLUB) is slightly slower than B = 1 (PLUB) because of the computational overhead issues (such as computing Manhattan distances). For the case of auctions,

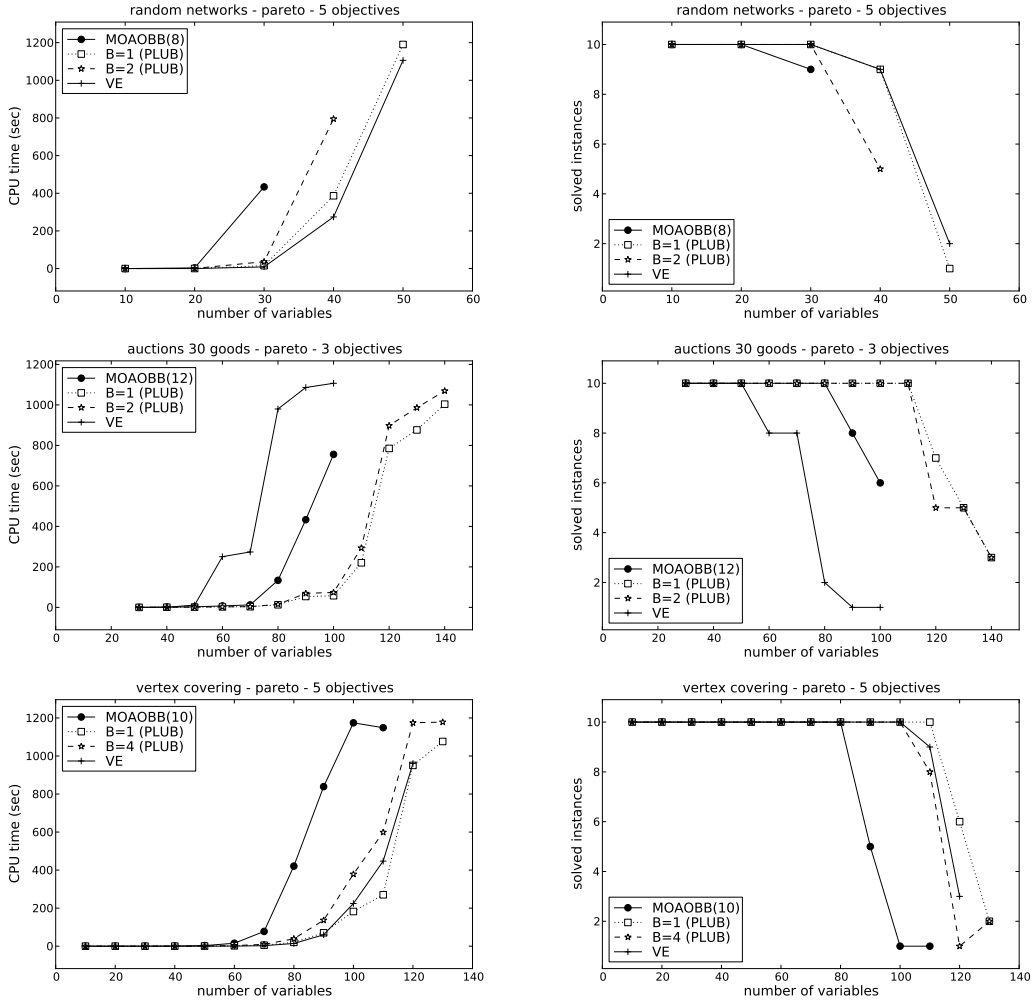


Figure 4.10: CPU time in seconds (left) and number of problem instances solved (right) for random networks with 5 objectives, combinatorial auctions with 3 objectives and vertex covering problems with 5 objectives, respectively. Using the Pareto ordering. Time limit 30 minutes.

algorithms with B = 1 (PLUB) and B = 2 (PLUB) are able to solve instances with 140 bids, whereas MOAOBB is hardly able to solve up to instances with 100 bids. Since VE is time and space exponential in the induced width of the problem instance, it is competitive only for small and medium size problems and runs out of memory on large size problems such as combinatorial auctions with more than 100 bids.

Figure 4.11 shows the results obtained for the same problem classes for trade-offs case. The number of two-way and three-way trade-offs used for each class is given by the parameters K and T. Here we see a different picture, the algorithms using B = 1 for both Pareto and trade-offs cases perform less well compared to the algorithms with B = 2 for auctions and vertex coverings and B = 5 for

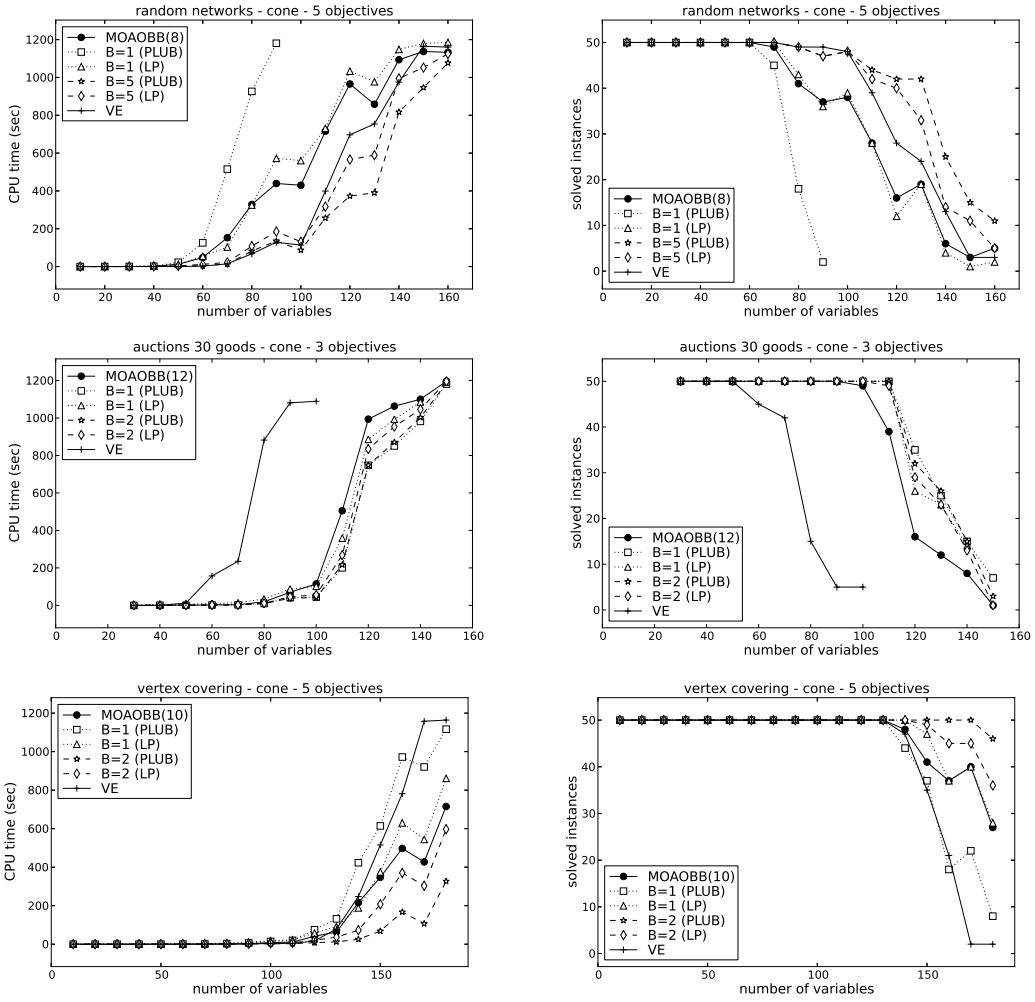


Figure 4.11: CPU time in seconds (left) and number of problem instances solved (right) for random networks with 5 objectives, combinatorial auctions with 3 objectives and vertex covering problems with 5 objectives, respectively. Trade-offs generated with parameters $(K = 6, T = 3)$ for random networks, $(K = 2, T = 1)$ for combinatorial auctions and $(K = 5, T = 2)$ for vertex covering. Time limit 20 minutes.

random networks. The results suggest that the upper bound sets generated by the algorithms for B = 1 (PLUB) case are very weak, and for B = 1 (LP) we have additional computational overhead issues such as often calling LP solver to solve the associated linear programs.

In summary, for the Pareto (i.e., no trade-offs) case, algorithms using singleton upper bound sets dominate the algorithms using relatively small cardinality of the upper bound sets. Most importantly, in many cases the singleton upper bounds significantly dominate the current state-of-the-art algorithm. For the trade-offs case, algorithms using relatively small cardinality of the upper bound

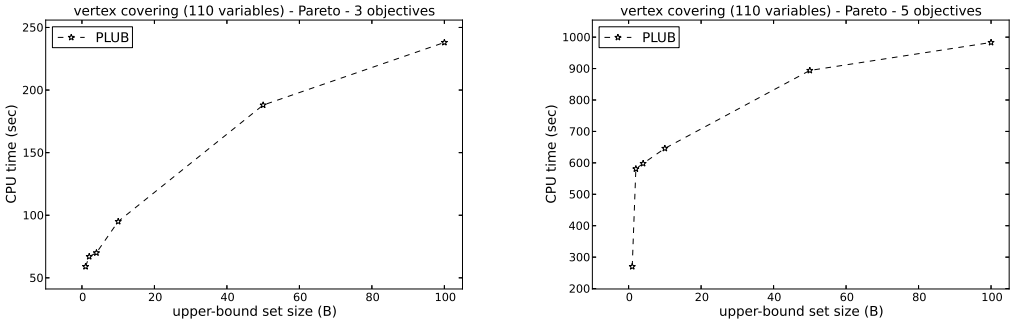


Figure 4.12: CPU time in seconds for vertex covering problems with 3 objectives and 110 variables (left); and 5 objectives with 110 variables (right) as a function of the upper bound set size. Using the Pareto ordering. Time limit 20 minutes.

sets are superior over the singleton upper bound sets and the state-of-the-art algorithm.

Impact of the Upper Bound Set Size

Pareto (no trade-offs) case:

Figure 4.12 shows the CPU times in seconds for varying the upper bound set size $B \in \{1, 2, 4, 10, 50, 100\}$ using vertex covering problems with 3 and 5 objectives. The results are averaged over 10 randomly generated instances. The mini bucket i-bound is 10. We can see that the singleton upper bound set performs best. However, the upper bound sets with relatively small cardinality (e.g., $B = 2$), with the slight additional computational overheads (such as calculating the Manhattan distances) perform quite well. For problems with 5 objectives, the MOAOBB algorithm involves the use of large upper bound sets with approximately 5000 elements, and is able to solve only one problem instance. In contrast, the algorithm with $B = 1$ solved all the instances within 5 minutes on average.

Trade-offs case:

Figure 4.13 displays the average CPU time in seconds (left) and number of problem instances solved (right) as a function of the upper bound set size (B) for random network problems with 4 objectives and 80 variables. The methods using Pareto least upper bounds (PLUB) and trade-offs based upper bounds (LP) have been applied for $B \in \{1, 2, 4, 16, 32, 64, 128\}$. For each problem size we generated 10 random instances and for each random instance we generated 10 random sets of consistent trade-offs. For the LP case we set the cluster size to 30, replacing a collection of 30 elements with their LP based upper bound.

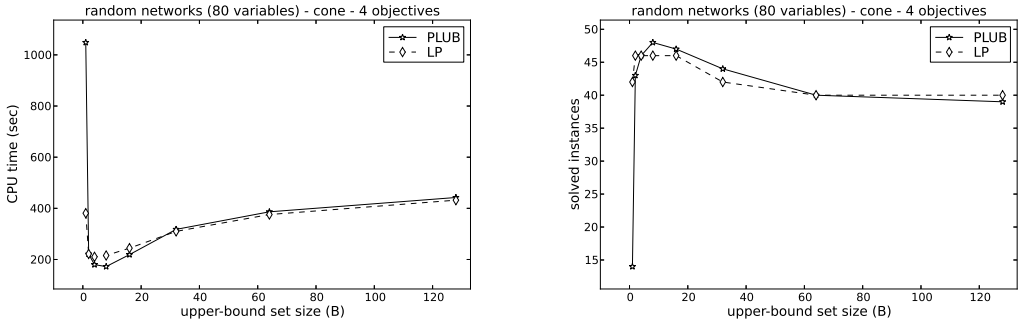


Figure 4.13: CPU time in seconds (left) and number of problem instances solved (right) for random networks with 4 objectives and 80 variables as a function of the upper bound set size. Time limit 20 minutes.

The number of two-way trade-offs (K) and number of three-way trade-offs (T) were set to 3 and 2 respectively. The mini-bucket i-bound was set to 8.

The graphs help us to choose the best value for B when using the Pareto and LP based upper bounds. For instance, the algorithm using B = 1 (PLUB) has the worst performance due to the loose upper bounds, whereas the algorithm using B = 1 (LP) is not the worst but due to the computation of the upper bounds (using LP solver) the average solving time is much higher than other cases. In summary, the best option is to use an upper bound set with small cardinality when using both PLUB and LP based bounding schemes.

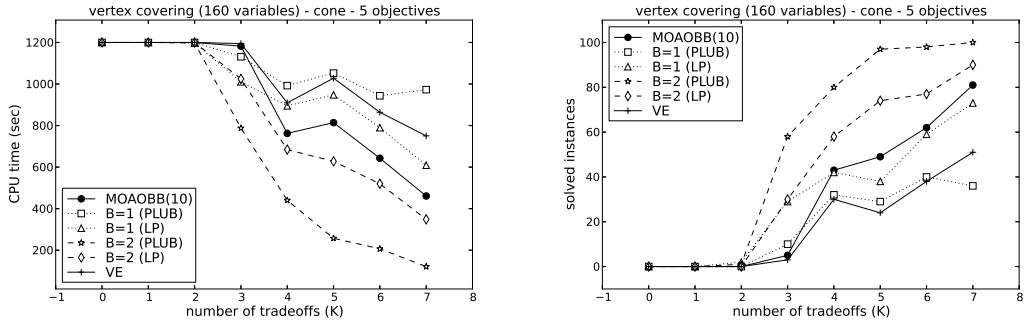


Figure 4.14: CPU time in seconds (left) and number of problem instances solved (right) as a function of the number of pairwise trade-offs (K) for vertex covering problems with $n = 160$, 5 objectives and using $T = 1$. The mini-bucket i-bound is 10. Time limit 20 minutes.

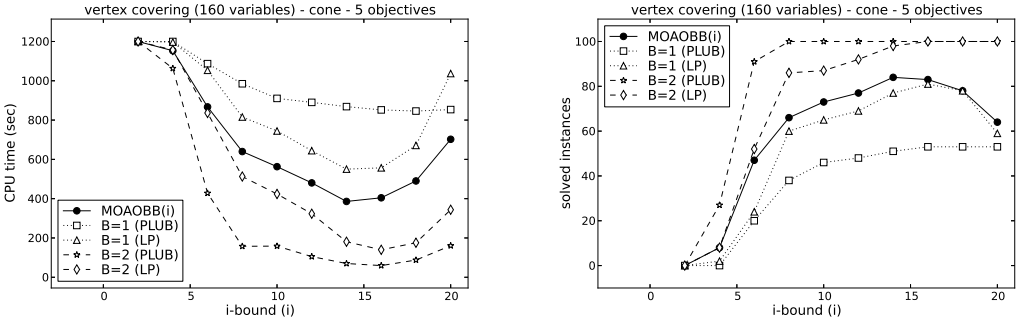


Figure 4.15: CPU time in seconds (left) and number of problem instances solved (right) as function of the mini-bucket i-bound for vertex covering problems with $n = 160$, 5 objectives and $(K = 5, T = 2)$ trade-offs. Time limit 20 minutes.

Impact of the Number of Trade-offs

Figure 4.14 displays the impact of the number of pair-wise trade-offs K for vertex covering problems with $n = 160$ variables and for fixed number of three-way trade-offs $T = 1$. As K increases we can see that the induced preference relation $\succcurlyeq_\Theta$ gets stronger and consequently more problems are being solved, with continuously reducing CPU time. The algorithms using B = 1 (PLUB) and B = 1 (LP) have flatter performances due to their use of a weaker upper bound set.

Impact of the Heuristic Information

Results in Figure 4.15 display the impact of increasing mini-bucket i-bound size for vertex covering problems. We can see that each algorithm using the mini-bucket heuristics produced a U-shaped curve because the mini-bucket heuristics get stronger for a certain range of i-bound values and consequently prune the

search space more effectively. For example, for $i \in [0, 12]$ each algorithm overall solving time decreased monotonically.

Using Proposition 9, and the additional constraints ensuring that the LP-based upper bound to be weakly dominated by the Pareto upper bound, yielded a much tighter upper bound that led to much stronger pruning of the search space across all benchmarks. But having to solve the associated linear programs to generate upper bounds the performance of LP-based algorithms bit slower compare to the Pareto-based algorithms.

4.7 Summary and Conclusion

In this chapter, we presented the enhancements in the multi-objective constraint optimization algorithms such as branch-and-bound and variable elimination. In particular, we defined a formalism for the decision maker's preference information and incorporating it in different variations of a branch-and-bound algorithm. In addition to this, we showed that our approach for preference inference can be given an alternative semantics based on Multi-Attribute Utility Theory, where it is assumed that the decision maker compares utility vectors by a weighted sum of the individual values.

We presented a method for testing of the dominance between multi-objective utility vectors using a linear program approach, and introduced a much faster approach for dominance testing which involved compilation to matrix multiplication. We experimentally showed that the matrix based approach substantially reduces the time for testing the dominance between utility vectors.

We focused on a branch-and-bound algorithm that uses a mini-buckets procedure to generate an upper bound set at each node during the search for optimal solutions. Since the utility vectors are compared on p objective values, the generating upper bound sets can be large in size. We introduced different techniques which are simple, and at the same time are very effective, to control the cardinality of the upper bound sets. These methods choose a collection of elements and replace them with their upper bound. We presented techniques for generating upper bounds for a set of elements for both Pareto (no trade-offs) and trade-offs cases. It is clear that the trade-offs based upper bound can be much tighter than the Pareto based one. Our experimental results with different multi-objective constraint optimization benchmarks showed that

using a singleton upper bound set for the Pareto case, and non-singleton upper bound set with quite small cardinality for the trade-offs case, appears to be best. The results clearly indicate that our proposed enhancements bring significant improvement over the current approaches.

Chapter 5

Multi-objective Influence Diagrams with Trade-offs

In this chapter we discuss sequential decision making problem under uncertainty represented by using influence diagram. We extend standard influence diagram to include p (> 1) objectives, in this case utility values are vectors in $\mathbb{R}^p$, and are typically only partially ordered. We present the variable elimination algorithm to solve these models, which results in a set of maximal values of expected utilities (or Pareto set). Since the Pareto ordering of the multi-objective utility vectors is not very discriminating, these sets can grow extremely large. We present an approximation technique which is based on the notion of ϵ-covering of the expected utility sets, this allows us to scale up to much larger problems than before. In addition, we also bring the formalism defined in the previous chapter for modelling the decision maker's imprecise trade-offs between the objectives to solve multi-objective influence diagram, and we experimentally demonstrate that this greatly improves the efficiency.

The chapter is set out as follows: Section 5.2 defines multi-objective influence diagram and explains how a variable elimination can be applied to solve it up to a form of equivalence; Section 5.3 discusses in detail ϵ-covering of the maximal values of expected utility set; and Section 5.4 describes the formalism for modelling imprecise trade-offs and presents a method to test the dominance relation between the multi-objective utility vectors with respect to the induced preference relation. Experimental results based on Pareto ordering, ϵ-covering of the Pareto set and the dominance relation induced by the decision maker trade-offs are presented in Section 5.5. Finally, Section 5.6 contains a summary

of the chapter and the conclusions.

5.1 Introduction

The influence diagram [Howard and Matheson, 1984a] is used to represent and analyse sequential decision making problems under uncertainty. It involves a directed acyclic graph consisting of nodes representing variables in the decision problem. The uncertainty is represented by chance variables (or random variables) whose outcomes are dependent on other variables known as its parents. Decision variables represent points in time where the decision maker has to make the decision based on some observation on other variables. Utility functions (also known as value functions) assign utilities for each configuration of its domain variables in the model. Similar to the Bayesian network, uncertainty is modelled by a collection of conditional probabilities, one for each chance variable.

As described in the previous chapter, in many situations, including our daily activities, involve decisions with multiple, often conflicting and non-commensurate, objectives such as selecting a hotel for dinner and buying furniture for our house. It may not always be possible to map different objectives into a single utility scale, since the decision maker may not be willing to do or he may not have clear idea about the precise trade-offs between the objectives [Keeney and Raiffa, 1993, Roy, 1996, Ehrgott, 1999]. For this reason, we consider multi-objective influence diagrams, which include utility values elements in $\mathbb{R}^p$, where $p\ (> 1)$ is the number of objectives associated with each decision. Since utility vectors are only partially ordered, for example using the Pareto ordering, we no longer have a unique maximal expected utility value but a set of them known as the maximal expected utility set (see Definition 18 of Chapter 4).

As we have already seen in Chapter 4, the Pareto ordering of the multi-objective utility vectors is rather weak, in the sense that it does not discriminate much between the utility vectors, and consequently the maximal expected utility set can often be large. We introduce a method based on ϵ-covering which approximates the Pareto set. We show experimentally that this greatly reduces the size of the undominated sets and solves much larger problems than the original one.

On the other hand, we make use of the formalism defined in Chapter 4 for

handling the decision maker's imprecise trade-offs, in the form of choosing one multi-objective utility vector over another (discussed in Section 5.4), and use these input preferences to infer other preferences. The induced preference relation is then used to eliminate the dominated utility vectors during the computation. Our experimental results indicate that even a small number of such input preferences can greatly reduce the undominated set of expected utility values.

5.2 Multi-objective Influence Diagrams

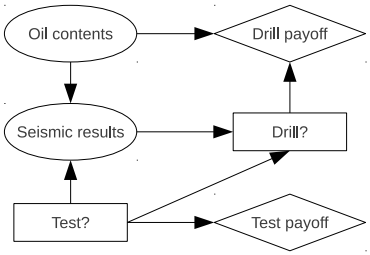
Multi-objective influence diagrams are an extension of standard influence diagrams that include multi-objective utility functions which are additive decomposable [Gonzales et al., 2011]. In the following sections we discuss in detail the graphical model and present a variable elimination algorithm to evaluate the model. For simplicity and without loss of generality, we assume that all objectives are to be maximized.

5.2.1 The Graphical Model

A *multi-objective influence diagram* (MOID) is similar to a standard influence diagram (ID), the only difference is, it allows a multi-objective utility function defined on p (> 1) objectives. As in the standard ID, the graphical model of MOID is a directed acyclic graph that contains *chance nodes* (circles), which represent discrete random variables denoted with $\mathbf{X} = \{X_1, \ldots, X_n\}$, *decision nodes* (rectangles) represent the decision variables $\mathbf{D} = \{D_1, \ldots, D_m\}$, and *utility nodes* (diamonds) for the local utility functions (also known as *value functions*) $\mathbf{U} = \{U_1, \ldots, U_r\}$.

The graphical model is built in such a way that the decision maker can easily determine exactly what information is available before making each decision. The directed arcs represent the information flow between the variables in the model. The directed arc into a node for a random variable represents the probabilistic dependence on the random variable. The directed arc into a node for the decision variable denotes the information available before making that decision. A link into a node for local utility function denotes functional dependence.

The model contains a conditional probability distribution $P(X_i \mid pa(X_i))$ for each discrete random variable $X_i \in \mathbf{X}$, where $pa(X_i)$ is the parent set of X_i.



Oil contents	P(O)		
	dry	wet	soak
	0,5	0,3	0,2

Test payoff	U_1(T)
Test?	
yes	{(-10,10)}
no	{(0,0)}

Seismic results		P(S | O,T)			
Oil cnt.	Test?	closed	open	diffuse	notest
dry	yes	0.1	0.3	0.6	0
dry	no	0	0	0	1
wet	yes	0.3	0.4	0.3	0
wet	no	0	0	0	1
soak	yes	0.5	0.4	0.1	0
soak	no	0	0	0	1

Drill payoff		U_2(O,D)
Oil cnt.	Drill?	
dry	yes	{(-70,18)}
dry	no	{(0,0)}
wet	yes	{(50,12)}
wet	no	{(0,0)}
soak	yes	{(200,8)}
soak	no	{(0,0)}

Figure 5.1: A bi-objective influence diagram.

We assign a utility function to each utility node $U_j \in \mathbf{U}$ which represents the preference of the decision maker over the p-objectives, where $U_j : \Omega_{Q_j} \to \mathbb{R}^p$, here Q_j is called the scope of the utility function U_j.

A *policy* for a MOID is defined as an ordered sequence $\Delta = (\delta_1, \ldots, \delta_k)$, where each δ_i is called a *decision rule* for the decision variable D_i and it is defined as a mapping between $\Omega_{pa(D_i)}$ and Ω_{D_i}. Since the utility vectors are the members of $\mathbb{R}^p$ then for each policy Δ we have the expected utility $EU_\Delta \subseteq \mathbb{R}^p$. Solving a MOID is nothing but finding the policies with undominated expected utility value. In other words, solving a MOID is equivalent to identifying the policies that generate the maximal values of expected utility, i.e., finding the policies that generate the set $\max_{\succcurlyeq}\{EU_\Delta \mid \text{ all policies } \Delta\}$. A policy with undominated expected utility is called *optimal*.

Example 19 » Multi-objective influence diagram

Figure 5.1 is the extension of the decision problem presented in Example 3 (Section 2.4.2.1, Chapter 2), where the utility functions have two attributes, namely testing/drilling payoff and environmental damage, respectively. The utility associated with testing for oil is $(-10, 10)$*, the utility of drilling when the hole is dry is* $(-70, 18)$*, the utility of drilling when the hole is wet is* $(50, 12)$ *and when the hole is soak the utility of drilling is* $(200, 8)$*. The goal of the decision maker is to find the policies that maximize the payoff and minimize the environmental*

damage as much as possible. If $\vec{u} = (u_1, u_2)$ *and* $\vec{v} = (v_1, v_2)$ *are utility vectors in* $\mathbb{R}^2$ *then the weak Pareto dominance relation for this problem is defined as* $\vec{u} \geq \vec{v} \Leftrightarrow u_1 \geq v_1$ *and* $u_2 \leq v_2$*. For instance,* $(10, 2)$ *dominates* $(8, 4)$ *because* $10 \geq 8$ *and* $2 \leq 4$*. Whereas* $(10, 2)$ *does not dominate* $(8, 1)$ *because* $2 \not\leq 1$*. The Pareto or maximal set of the problem is* $\{(22.5, 17.56)$*,* $(20, 14.2)$*,* $(11, 12.78)$*,* $(0, 0)\}$ *(i.e.,* $\max_{\geq}\{EU_\Delta \mid$ *policies* $\Delta\}$*), corresponding to the four optimal policies shown in the following table.*

	Δ_1	Δ_2	Δ_3	Δ_4
δ_T	*yes*	*no*	*yes*	*no*
δ_D	*yes (S = closed)* *yes (S = open)* *no (S = diffuse)*	*yes (S = notest)*	*yes (S = closed)* *no (S = open)* *no (S = diffuse)*	*no (S = notest)*
EU_{Δ_i}	$\{(22.5, 17.56)\}$	$\{(20, 14.2)\}$	$\{(11, 12.78)\}$	$\{(0, 0)\}$

5.2.2 Arithmetic Operations and Distributive Properties

Since the utility values are only partially ordered, the $\max$ (or maximization) operator does not necessarily lead to a single element. Thus, we need to deal with sets of utility values. In this section we present the arithmetic operations addition, multiplication and $\max$ operator for finite sets of utility values. We assume that $\succcurlyeq$ is a partial order on $\mathbb{R}^p$ that satisfies the properties Independence, Scale-Invariance and extends the Pareto.

Suppose that $\mathcal{U}, \mathcal{V}$ are two finite subsets of $\mathbb{R}^p$ and $q \geq 0$ be any non-negative real number. The addition$(+)$ and multiplication$(\times)$ operations defined as follows, $\mathcal{U} + \mathcal{V} = \{\vec{u} + \vec{v} \mid \vec{u} \in \mathcal{U}, \vec{v} \in \mathcal{V}\}$ and $q \times \mathcal{U} = \{q \times \vec{u} \mid \vec{u} \in \mathcal{U}\}$. The $\max$ operator is defined as $\max(\mathcal{U}) = \max_{\succcurlyeq}\{\vec{u} \in \mathcal{U} \mid \nexists \vec{v} \in \mathcal{U}, \vec{v} \succ \vec{u}\}$ and also between $\mathcal{U}$ and $\mathcal{V}$ the $\max$ operator is defined as $\max(\mathcal{U}, \mathcal{V}) = \max_{\succcurlyeq}(\mathcal{U} \cup \mathcal{V})$. The operators $+$, $\times$ and $\max$ satisfies commutative and associative properties.

The key properties that we need for the variable elimination procedure are distributive properties. It is clear that the left distributive property $q \times (\mathcal{U} + \mathcal{V}) = (q \times \mathcal{U}) + (q \times \mathcal{V})$ holds, whereas the right distributive property $(q_1 + q_2) \times \mathcal{U} = (q_1 \times \mathcal{U}) + (q_2 \times \mathcal{U})$, where $q_1, q_2 \geq 0$ are real numbers, does not always hold. For instance, if we have $q_1 = q_2 = 0.5$ and $\mathcal{U} = \{(1, 0), (0, 1)\}$ then we obtain $(q_1 + q_2) \times \mathcal{U} = \{(1, 0), (0, 1)\}$ whereas $q_1 \times \mathcal{U} + q_2 \times \mathcal{U} = \{(0.5, 0), (0, 0.5)\} + \{(0.5, 0), (0, 0.5)\} = \{(1, 0), (0.5, 0.5), (0, 1)\}$. Clearly, $(q_1 + q_2) \times \mathcal{U} \neq (q_1 \times \mathcal{U}) + (q_2 \times \mathcal{U})$. However, if the utility sets $\mathcal{U}$ and $\mathcal{V}$ are restricted to convex then the property holds. In the following section we discuss this in detail.

5.2.3 Equivalent Sets of Utility Values

Recall the definition of dominance relation for the sets of utility vectors, i.e., for given any $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$, we say that $\mathcal{U} \succcurlyeq \mathcal{V}$ if and only if for every $\vec{v} \in \mathcal{V}$ there exists $\vec{u} \in \mathcal{U}$ such that $\vec{u} \succcurlyeq \vec{v}$, i.e., every element of $\mathcal{V}$ is weakly dominated (with respect to $\succcurlyeq$) by some element of $\mathcal{U}$.

We define the relation $\approx$ between the sets of utility vectors as follows:

Definition 30 » Relation $\approx$

For any $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$, $\mathcal{U} \approx \mathcal{V}$ if and only if $\mathcal{U} \succcurlyeq \mathcal{V}$ and $\mathcal{V} \succcurlyeq \mathcal{U}$.

The *convex closure* $\mathcal{C}(\mathcal{U})$ of a (finite or infinite) subset $\mathcal{U} \subseteq \mathbb{R}^p$ is defined as the set of all the elements in $\mathbb{R}^p$ that are expressed as convex combination of the elements of $\mathcal{U}$, i.e., $\mathcal{C}(\mathcal{U})$ is defined to consist of every element of the form $\sum_{i=1}^{k}(q_i \times \vec{u}_i)$, where k is any arbitrary natural number, each $\vec{u}_i$ in $\mathcal{U}$, each $q_i \in \mathbb{R}^+ \cup \{0\}$, where $\sum_{i=1}^{k} q_i = 1$.

The equivalence relation $\equiv$ between the finite sets of utility vectors in $\mathbb{R}^p$ is defined as follows:

Definition 31 » Relation $\equiv$

For given any $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$, $\mathcal{U} \equiv \mathcal{V}$ if and only if $\mathcal{C}(\mathcal{U}) \approx \mathcal{C}(\mathcal{V})$.

That means two sets of utility vectors are considered to be equivalent if their convex closures are equivalent. In other words, the two sets of utility vectors are equivalent if, for every convex combination of elements of one, there exists the convex combination of elements of the other which is at least as good with respect to the partial order $\succcurlyeq$ on $\mathbb{R}^p$.

The following result from [Wilson and Marinescu, 2012] shows that operations on sets of utility vectors respect the $\equiv$ relation.

Proposition 10 » Operations $+, \times$ and $\max$ respect $\equiv$

Let $\mathcal{U}, \mathcal{V}, \mathcal{W} \subseteq \mathbb{R}^p$ be finite sets and let $q \geq 0$. The following properties hold: (1) $\mathcal{U} \equiv \max_{\succcurlyeq}(\mathcal{U})$; (2) if $\mathcal{U} \equiv \mathcal{V}$ then $q \times \mathcal{U} \equiv q \times \mathcal{V}$, $\mathcal{U} + \mathcal{W} \equiv \mathcal{V} + \mathcal{W}$ and $\max(\mathcal{U}, \mathcal{W}) \equiv \max(\mathcal{V}, \mathcal{W})$.

The following result from [Wilson and Marinescu, 2012] shows that for any finite sets of (partially ordered) utility values in $\mathbb{R}^p$ the (right) distributive prop-

erty that we require holds with respect to the $\equiv$ relation. It also gives other important properties that we require for the variable elimination procedure.

Theorem 2 » Operations on sets of utility values

Let $\succcurlyeq$ be a partial order on $\mathbb{R}^p$ satisfying Independence and Scale-Invariance. Then, for all $q, q_1, q_2 \geq 0$ and for all finite sets $\mathcal{U}, \mathcal{V}, \mathcal{W} \subseteq \mathbb{R}^p$, we have that: (i) $q \times (\mathcal{U} + \mathcal{V}) = q \times \mathcal{U} + q \times \mathcal{V}$; *(ii)* $(q_1 + q_2) \times \mathcal{U} \equiv (q_1 \times \mathcal{U}) + (q_2 \times \mathcal{U})$; *(iii)* $q_1 \times (q_2 \times \mathcal{U}) = (q_1 \times q_2) \times \mathcal{U}$; *(iv)* $\max(q \times \mathcal{U}, q \times \mathcal{V}) = q \times \max(\mathcal{U}, \mathcal{V})$; *(v)* $\max(\mathcal{U} + \mathcal{W}, \mathcal{V} + \mathcal{W}) \equiv \max(\mathcal{U}, \mathcal{V}) + \mathcal{W}$.

5.2.4 Variable Elimination

A *legal elimination sequence* [Wilson and Marinescu, 2012] for an influence diagram is a permutation $Y_1, \ldots, Y_n$ of the variables $\mathbf{X} \cup \mathbf{D}$, which extends the relation $<$ given by: $\mathbf{I}_0 < D_1 < \mathbf{I}_1 \ldots < D_m < \mathbf{I}_m$, so that, for $i = 0, \ldots, m$, each chance variable X of $\mathbf{I}_i$ comes after D_i (if $i \geq 1$) and before D_{i+1} (if $i < m$). For instance, if $Y_j \in D_1$ and $Y_k \in \mathbf{I}_1$ then we must have $j < k$.

Given a legal elimination ordering of the variables, Theorem 2 allows us to eliminate the variables using iterative variable elimination procedure ELIM-MOID described by Algorithm 3. It eliminates chance variables by $+'$, which is defined as $\mathcal{U} +' \mathcal{V} = \max_{\succcurlyeq}(\mathcal{U} + \mathcal{V})$, and eliminates the decision variables by $\max$. The probability and (set-valued) utility values are combined by the operations $\times$ and $+$, respectively. The maximal expected utility values will then be equivalent to:

$$\sum_{\mathbf{I}_0} \max_{D_1} \sum_{\mathbf{I}_1} \cdots \sum_{\mathbf{I}_{m-1}} \max_{D_m} \sum_{\mathbf{I}_m} \left(\prod_{i=1}^{n} P_i \times \sum_{j=1}^{r} U_j \right).$$

The algorithm 3 is based on Dechter's bucket elimination framework [Dechter, 2000b]. Given a legal elimination ordering it finds the maximal set $\max_{\succcurlyeq}\{EU_\Delta | \text{ policies } \Delta\}$ and the associated optimal policies. If $\tau = Y_1, \ldots, Y_t$ is the legal elimination ordering, it partitions the input utility functions into a bucket structure called as *buckets*. Each variable Y_l has exactly a bucket associated with it, and it contains all input utility functions and probability distributions whose highest variable in their scope is Y_l.

Algorithm ELIM-MOID operates in two phases namely, top-down (lines 3-11)

Algorithm 3: ELIM-MOID

Data: A MOID $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U} \rangle$ with $p > 1$ objectives, a legal elimination ordering of the variables $\tau = (Y_1, \ldots, Y_t)$
Result: An optimal policy Δ

```
// partition the functions into buckets
```
1 **for** $l = t$ *downto 1* **do**
2 place in $buckets[l]$ all remaining components in $\mathbf{P}$ and $\mathbf{U}$ that contain variable Y_l in their scope;

```
// top-down step
```
3 **for** $l = t$ *downto 1* **do**
4 let $\Phi^l = \{\phi_1, \ldots, \phi_j\}$ and $\Psi^l = \{\psi_1, \ldots, \psi_k\}$ be the probability and utility components in $buckets[l]$;
5 **if** Y_l *is a chance variable* **then**
6 $\phi^l \leftarrow \sum_{Y_l} \prod_{i=1}^{j} \phi_i$;
7 $\psi^l \leftarrow (\phi^l)^{-1} \times \sum'_{Y_l}((\prod_{i=1}^{j} \phi_i) \times (\sum_{j=1}^{k} \psi_j))$;
8 **else if** Y_l *is a decision variable* **then**
9 $\phi^l \leftarrow \max_{Y_l} \prod_{i=1}^{j} \phi_i$;
10 $\psi^l \leftarrow \max_{Y_l}((\prod_{i=1}^{j} \phi_i) \times (\sum_{j=1}^{k} \psi_j))$;
11 place each ϕ^l and ψ^l in the bucket of the highest-index variable in its scope;

```
// bottom-up step
```
12 **for** $l = 1$ *to* t **do**
13 **if** Y_l *is a decision variable* **then**
14 $\delta_l \leftarrow \arg\max_{Y_l}((\prod_{i=1}^{j} \phi_i) \times (\sum_{j=1}^{k} \psi_j))$;
15 $\Delta \leftarrow \Delta \cup \delta_l$;
16 **return** Δ

and bottom-up (lines (12-16). During the top-down approach it applies from last to first, a iterative variable elimination procedure to compute the new probability (denoted by ψ) and utility (denoted by ϕ) components. These are then placed into the corresponding lower buckets. If Y_l is a chance variable then the algorithm computes the ψ-message by multiplying all the probability components in the corresponding bucket of Y_l. The variable Y_l is then eliminated by summation operation. Similarly, the ϕ-message is computed adding all the utility functions in that bucket and then normalize it by buckets compiled ψ-message. The variable Y_l is then eliminated using $\sum'$ operation. If Y_l is a decision variable, then ψ and ϕ-messages are computed in the same manner and the variable Y_l is eliminated using the maximization ($\max$) operation. The product of all probability components of the bucket is a constant when viewed as a function of the bucket's decision variable [Jensen et al., 1994, Wilson and Marinescu, 2012], i.e., ψ-message is a constant.

During the bottom-up phase, the algorithm computes optimal policies (lines 12-16). It operates the decision buckets in the reverse order, i.e., from first to

the last. For each decision variable, the algorithm computes the decision rule by applying the maximization operation on the combination of probability and utility components of each configuration of the variables in the bucket's scope, where the bucket's scope is the union of all the scopes of the functions in that bucket. The algorithm records the values assigned to the previous decisions.

Since the utility values are only partially ordered (i.e., the utility vectors are in $\mathbb{R}^p$), it is not possible to predict the size of the undominated set of expected utility values.

5.3 Approximating the Pareto Set

Since the weak Pareto ordering ($\geq$) on $\mathbb{R}^p$ is not very discriminating the maximal set with respect to the Pareto ordering, $\max_{\geq}\{EU_\Delta : \text{ policies } \Delta\}$ often gets quite large, and consequently the number of optimal policies greatly increases. In this section we present a method that approximates the Pareto set. Our approximation method largely depends on the Pareto ordering and constructs a set which contains all the elements that *approximately* dominate all other elements belong to the set $\{EU_\Delta : \text{ policies } \Delta\}$.

We take the approach based on ϵ-*dominance* between utility vectors given by [Papadimitriou and Yannakakis, 2000] and it is defined as follows (also see Section 3.4.4 in Chapter 3):

Definition 32 » ϵ-dominance

If $\epsilon > 0$ be any small positive real number, we define the ϵ-dominance *relation on positive vectors of $\mathbb{R}^p_+$ by $\vec{u} \geq_\epsilon \vec{v} \iff (1+\epsilon)\vec{u} \geq \vec{v}$.*

We now define the ϵ-covering of a finite subset of $\mathbb{R}^p_+$ based on the notion of ϵ-dominance defined above.

Definition 33 » ϵ-covering

Let $\mathcal{U} \subseteq \mathbb{R}^p_+$ and $\epsilon > 0$. Then a set $\mathcal{U}_\epsilon \subseteq \mathcal{U}$ is called an ϵ-approximate Pareto set *or an* ϵ-covering*, if every vector $\vec{v} \in \mathcal{U}$ is ϵ-dominated by at least one vector $\vec{u} \in \mathcal{U}_\epsilon$, i.e., $\forall \vec{v} \in \mathcal{U}\ \exists \vec{u} \in \mathcal{U}_\epsilon$ such that $\vec{u} \geq_\epsilon \vec{v}$.*

The ϵ-covering $\mathcal{U}_\epsilon$ of the set $\mathcal{U} \subseteq \mathbb{R}^p$ is constructed using the mapping $\varphi : \mathbb{R}^p_+ \to \mathbb{Z}^p_+$, defined by $\varphi(\vec{u}) = (\varphi(u_1), \ldots, \varphi(u_p))$ where $\forall i,\ \varphi(\vec{u_i}) = \lceil \log u_i / \log(1 +$

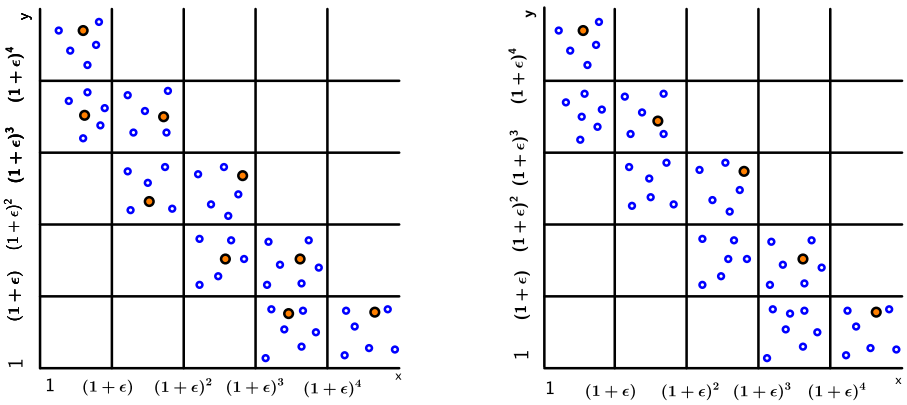


Figure 5.2: Examples of ϵ-covering.

$\epsilon)\rceil$ [Papadimitriou and Yannakakis, 2000]. The mapping φ transforms every element of $\mathcal{U}$ onto a logarithmic grid, for every component $\vec{u_i}$ of utility vector $\vec{u}$ it returns an integer k such that $(1+\epsilon)^k \leq u_i \leq (1+\epsilon)^{k+1}$. Each cell on the grid represents a class of utility vectors having the same image through φ and every utility vector in the cell ϵ-dominates all the other utility vectors in that cell. This means selecting one vector from each cell yields ϵ-covering of the entire set.

In figure 5.2 the left-hand-side graph is an example of this method for bi-objective case. The dotted lines form the logarithmic grid and ϵ-covering of the Pareto frontier can be obtained by selecting one utility vector (red dots) from each non-empty cell of the grid. We can further process the resulting ϵ-covering by removing the dominated vectors from the covering, the red dots (undominated utility vectors) in the right-hand-side graph shows the refined ϵ-covering [Marinescu, 2011].

The following result shows that any two elements of a cell ϵ-dominates each other. Therefore, for given any smallest positive real ϵ we can construct the ϵ-covering of a finite set $\mathcal{U} \subseteq \mathbb{R}^p$ by choosing exactly one element from each cell and keeping only undominated cells occupied. Moreover, if we restrict each $\vec{u} \in \mathcal{U}$ to have the components bounded between 1 and a positive integer B then the size of $\mathcal{U}_\epsilon$ is polynomial in $\log B$ and $1/\epsilon$ [Papadimitriou and Yannakakis, 2000]. In addition, we can easily see that any cell of the grid represents a different class of vectors having the same image through φ.

Proposition 11 » Sufficient condition for ϵ-dominance

$\forall \vec{u}, \vec{v} \in \mathbb{R}^p_+$, $\varphi(\vec{u}) \geq \varphi(\vec{v}) \Rightarrow \vec{u} \geq_\epsilon \vec{v}$.

Proof. Suppose $\vec{u}, \vec{v} \in \mathbb{R}^p_+$, by definition, for $\varphi(\vec{u})$ there exists an integer k such that $(1+\epsilon)^k \leq u_i \leq (1+\epsilon)^{k+1}$ for all $i \in \{1, \ldots, p\}$. Similarly, for $\varphi(\vec{v})$ there exists an integer h such that $(1+\epsilon)^h \leq v_i \leq (1+\epsilon)^{h+1}$ for all $i \in \{1, \ldots, p\}$. Consider $\varphi(\vec{u}) \geq_\epsilon \varphi(\vec{v})$, this implies $k \geq h$. Then $(1+\epsilon)^{k+1} \geq (1+\epsilon)^{h+1}$, also we have $u_i \geq (1+\epsilon)^k$ which implies $u_i(1+\epsilon) \geq (1+\epsilon)^{k+1}$, i.e., $u_i(1+\epsilon) \geq (1+\epsilon)^{h+1}$, it follows $u_i(1+\epsilon) \geq v_i$, for all $i \in \{1, \ldots, p\}$, implying that $u \geq_\epsilon v$.

□

The following example gives the clear picture of the above result.

Example 20 » ϵ-dominance

Let $\mathcal{U} = \{\vec{u}, \vec{v}\}$ where $\vec{u} = (3.1, 2.9)$ and $\vec{v} = (3, 3.05)$. Clearly, neither $\vec{u} \not\geq \vec{v}$ nor $\vec{v} \not\geq \vec{u}$. Choose $\epsilon = 0.1$ then we have $\varphi(\vec{u}) = \varphi(\vec{v}) = (12, 12)$, and it is easy to verify that $\vec{u} \geq_\epsilon \vec{v}$ and $\vec{v} \geq_\epsilon \vec{u}$. Therefore, $\mathcal{U}_\epsilon = \{\vec{u}\}$ (or $\mathcal{U}_\epsilon = \{\vec{v}\}$) is a valid ϵ-covering of $\mathcal{U}$.

We next extend the algorithm ELIM-MOID to compute an ϵ-covering of the expected utility set $\{EU_\Delta |$ policies $\Delta\}$. However, it is not possible to just replace the Pareto dominance with ϵ-dominance because ϵ-dominance does not satisfy the key transitive property. For instance, if $\vec{u} = (10, 50)$, $\vec{v} = (9, 54)$ and $\vec{w} = (7, 58)$ then for $\epsilon = 0.1$ we have $\vec{u} \geq_\epsilon \vec{v}$ and $\vec{v} \geq_\epsilon \vec{w}$ but $\vec{u} \not\geq_\epsilon \vec{w}$. Nevertheless, [Dubus et al., 2009] defined a finer ϵ-dominance relation that satisfies the transitivity, which is defined as:

Definition 34 » (ϵ, λ)-dominance

For any reals $\epsilon > 0$ and $\lambda \in (0, 1)$, the (ϵ, λ)-dominance relation on $\mathbb{R}^p_+$ is defined as $\vec{u} \geq^\lambda_\epsilon \vec{v} \Leftrightarrow (1+\epsilon)^\lambda \vec{u} \geq \vec{v}$. Given a set $\mathcal{U} \subseteq \mathbb{R}^p_+$, a subset $\mathcal{U}_{(\epsilon,\lambda)} \subseteq \mathcal{U}$ is called an (ϵ, λ)-covering, if $\forall \vec{v} \in \mathcal{U}$ $\exists \vec{u} \in \mathcal{U}_{(\epsilon,\lambda)}$ such that $\vec{u} \geq^\lambda_\epsilon \vec{v}$.

For given $\epsilon > 0$ and any $\lambda \in (0, 1)$ we can define a mapping $\varphi_\lambda : \mathbb{R}^p_+ \to \mathbb{Z}^p_+$ which is defined by $\varphi_\lambda(\vec{u}) = (\varphi_\lambda(u_1), \ldots, \varphi_\lambda(u_p))$ where $\forall i$, $\varphi_\lambda(u_i) = \lceil \log u_i / \log(1+\epsilon)^\lambda \rceil$. We can see that the mapping φ_λ transforms the elements of $\mathbb{R}^p_+$ onto the logarithmic grid.

Proposition 12 » Sufficient condition for (ϵ, λ)-dominance

$\forall \vec{u}, \vec{v} \in \mathbb{R}^p_+$, $\varphi_\lambda(\vec{u}) \geq \varphi_\lambda(\vec{v}) \Rightarrow \vec{u} \geq^\lambda_\epsilon \vec{v}$.

Proof. The proof is analogous to the proof of Proposition 11.

□

The following result shows that the (ϵ, λ)-dominance relation satisfies the properties that we need for the variable elimination algorithm to compute the ϵ-covering of the expected utility set $\{EU_\Delta : \text{ policies } \Delta\}$.

Proposition 13 » Properties of (ϵ, λ)-dominance

Let $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^p_+$ and $\lambda, \lambda' \in (0, 1)$. The following properties hold: (i) if $\vec{u} \geq^\lambda_\epsilon \vec{v}$ then $\vec{u} + \vec{w} \geq^\lambda_\epsilon \vec{v} + \vec{w}$, and if $\vec{u} \geq^\lambda_\epsilon \vec{v}$ and $q \geq 0$ then $q\vec{u} \geq^\lambda_\epsilon q\vec{v}$; (ii) if $\vec{u} \geq^\lambda_\epsilon \vec{v}$ and $\vec{v} \geq^{\lambda'}_\epsilon \vec{w}$ then $\vec{u} \geq^{\lambda+\lambda'}_\epsilon \vec{w}$.

Proof. Suppose that $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^p_+$ and $\epsilon > 0$.

(i): Assume that $\vec{u} \geq^\lambda_\epsilon \vec{v}$; then we have $(1+\epsilon)^\lambda \vec{u} \geq \vec{v}$. For any $\vec{w}$ we have, $(1+\epsilon)^\lambda \vec{w} \geq \vec{w}$. By adding two inequalities we get, $(1+\epsilon)^\lambda(\vec{u}+\vec{w}) \geq (\vec{v}+\vec{w})$, which implies $\vec{u} + \vec{w} \geq^\lambda_\epsilon \vec{v} + \vec{w}$. Suppose $q \geq 0$ is any real, multiplying it on both sides of $(1+\epsilon)^\lambda \vec{u} \geq \vec{v}$, we get $(1+\epsilon)^\lambda(q\vec{u}) \geq (q\vec{v})$, this implies $q\vec{u} \geq^\lambda_\epsilon q\vec{v}$.

(ii): Suppose that $\vec{u} \geq^\lambda_\epsilon \vec{v}$ and $\vec{v} \geq^{\lambda'}_\epsilon \vec{w}$, then for any $\lambda, \lambda' \in (0, 1)$ we have $(1+\epsilon)^\lambda \vec{u} \geq \vec{v}$ and $(1+\epsilon)^{\lambda'} \vec{v} \geq \vec{w}$. Multiplying $(1+\epsilon)^{\lambda'}$ on both sides of the first inequality we get, $(1+\epsilon)^{\lambda+\lambda'} \vec{u} \geq (1+\epsilon)^{\lambda'} \vec{v}$. Transitivity of $\geq$ implies that $(1+\epsilon)^{\lambda+\lambda'} \vec{u} \geq \vec{w}$, thus $\vec{u} \geq^{\lambda+\lambda'}_\epsilon \vec{w}$.

□

For given any finite set $\mathcal{U} \subseteq \mathbb{R}^p_+$ Algorithm 4 computes its (ϵ, λ)-covering by removing (ϵ, λ)-dominated elements from $\mathcal{U}$ (steps $2-6$).

In order to extend the ELIM-MOID for (ϵ, λ)-dominance case we need the operations maximization and addition of finite sets of utility values. If $\max^*$ is

Algorithm 4: $(\epsilon, \lambda)-$COVERING$(\mathcal{U})$

```
1 Γ ← ∅; V ← ∅;
2 foreach u⃗ ∈ U do
3     if φ_λ(u⃗) ∉ Γ then
4         remove from Γ all φ_λ(v⃗) such that φ_λ(u⃗) ≥ φ_λ(v⃗);
5         Γ ← Γ ∪ {φ_λ(u⃗)};
6 foreach φ_λ(u⃗) ∈ Γ do  V ← V ∪ {u⃗};
7 return V;
```

the maximization and $+^*$ is the addition operation then we defined them as $\max^*(\mathcal{U}, \mathcal{V}) = \max_{\geq_\epsilon^\lambda}(\mathcal{U} \cup \mathcal{V})$ and $\mathcal{U} +^* \mathcal{V} = \max_{\geq_\epsilon^\lambda}(\mathcal{U} + \mathcal{V})$, where $\max_{\geq_\epsilon^\lambda}(\mathcal{U})$ is an (ϵ, λ)-covering of the finite set $\mathcal{U} \subseteq \mathbb{R}_+^p$ computed using Algorithm 4.

We replace $\max$ and $\sum'$ by $\max^*$ and $\sum^*$ in Algorithm 3 and call the refined algorithm as ELIM-MOID$_\epsilon$, which computes the ϵ-covering of the expected utility set. During the top-down phase, the algorithm executes t buckets, one for each variable, and eliminates the bucket variable via $\max^*$ operation. That means it needs to compute (ϵ, λ_i)-covering for t number of times. To obtain a valid (ϵ, λ)-covering the sufficient condition is to choose each λ_i in $(0, 1)$ such that sum of all λ's is equal to 1. Since we have t variables, we choose each $\lambda_i = 1/t$, $i = 1, \ldots, t$ such that $\sum_{i=1}^t \lambda_i = 1$. The algorithm ELIM-MOID$_\epsilon$ is now guaranteed to compute the valid ϵ-covering of the expected utility set and the following theorem gives the correctness of the algorithm.

Theorem 3 » Correctness of algorithm ELIM-MOID$_\epsilon$

Given a MOID instance $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U} \rangle$ with L variables, $p > 1$ objectives and any finite $\epsilon > 0$, algorithm ELIM-MOID$_\epsilon$ *computes an ϵ-covering of the expected utility set.*

Proof. We prove this by induction on t, the number of variables (or buckets) in the problem and the main focus will be on the ψ-messages which are combined in chance buckets by summation $(+)$ and decision buckets by set union $(\cup)$.

Suppose $t = 1$, then the proof is immediate by the definition of the $\max$ operator, namely the ψ-message generated by eliminating the variable in line 7 is an ϵ-covering.

Let A be the source bucket containing $\Psi^A = \{\vec{u}_1, \ldots, \vec{u}_n\}$ utility vectors and

assume that $\psi_A = \{\vec{u}'_1, \ldots, \vec{u}'_t\}$, the ψ-message generated from the bucket is an $(\epsilon, \frac{t}{L})$-covering of Ψ^A. Let B be the target bucket where ψ_A will be placed, and let $\Psi^B = \{\vec{v}_1, \ldots, \vec{v}_m\}$ be the utility values residing in bucket B. We need to show that ψ_B, the ψ-message generated by bucket B, is an $(\epsilon, \frac{t+1}{L})$-covering of $\Psi^A + \Psi^B$, where $\Psi^A + \Psi^B = \{\vec{u}_i + \vec{v}_j | \vec{u}_i \in \Psi^A, \vec{v}_j \in \Psi^B\}$ and similarly we can define $\psi_A + \Psi^B = \{\vec{u}' + \vec{v}_j | \vec{u}' \in \psi_A, \vec{v}_j \in \Psi^B\}$.

Clearly, $\Psi_A \subseteq \Psi^A$ and by definition, for every $\vec{u_i} \in \Psi^A$, $\exists \vec{u}' \in \psi_A$ such that $\vec{u}' \geq_\epsilon^{\frac{t}{L}} \vec{u_i}$, and Proposition 13(i) implies $(\vec{u}' + \vec{v}_j) \geq_\epsilon^{\frac{t}{L}} (\vec{u}_i + \vec{v}_j)$.

By definition, $\psi_B = \max_{\geq_{(\epsilon, \frac{1}{L})}} \{\psi_A + \Psi^B\}$ such that $\psi_B \subseteq \psi_A + \Psi^B$ and $\forall (\vec{u}' + \vec{v}_j) \in \psi_A + \Psi^B$, $\exists \vec{v}'' \in \psi_B$ such that $\vec{v}'' \geq_\epsilon^{\frac{1}{L}} (\vec{u}' + \vec{v}_j)$.

We have $\vec{v}'' \geq_\epsilon^{\frac{1}{L}} (\vec{u}' + \vec{v}_j)$ and $(\vec{u}' + \vec{v}_j) \geq_\epsilon^{\frac{t}{L}} (\vec{u}_i + \vec{v}_j)$ then Proposition 13(ii) implies $\vec{v}'' \geq_\epsilon^{\frac{1}{L} + \frac{t}{L}} (\vec{u}_i + \vec{v}_j)$, i.e., $\vec{v}'' \geq_\epsilon^{\frac{t+1}{L}} (\vec{u}_i + \vec{v}_j)$. Therefore, ψ_B is an $(\epsilon, \frac{t+1}{L})$-covering of $\Psi^A + \Psi^B$.

□

The time and space complexity of the ELIM-MOID$_\epsilon$ is bounded by the induced width of the legal elimination ordering. However, the experimental results in section 5.5 show that in many cases the cardinality of the ϵ-coverings of the corresponding Pareto sets are significantly smaller.

5.4 Handling Imprecise Trade-offs

In this section we summarize our approach for handling the imprecise trade-offs, which is described in the previous chapter. We also discuss how a variable elimination can be used to compute the set of maximal values of expected utility with respect to these input preferences.

As described in the previous chapter (for MOCOP case) and in Section 5.1, very often the decision maker allows some trade-offs between the objectives. For instance, in Example 19, imagine that the decision maker is presented with a situation in which the content of the oil is known to be '*wet*', and asked whether they have any preference for drilling. In response, suppose that the decision maker reveals a preference of $(50, 12)$ over $(0, 0)$, i.e., they will be happy to gain $\$50$ at the cost of allowing 12 percent environmental damage.

Such elicited input preferences are represented by the set Θ consisting the elements of the form $(\vec{u}, \vec{v})$ such that $\vec{u}$ is preferred to $\vec{v}$, where $\vec{u}, \vec{v} \in \mathbb{R}^p$. As described in Section 4.4 (of Chapter 4), if $\succcurlyeq_\Theta$ is the induced preference relation then we assume that it extends Θ and extends the Pareto on $\mathbb{R}^p$ (i.e., if $\vec{u} \geq \vec{v}$ then $\vec{u} \succcurlyeq_\Theta \vec{v}$). Also $\succcurlyeq_\Theta$ satisfies the Independence (i.e., if $\vec{u} \succcurlyeq_\Theta \vec{v}$ and $\vec{w} \in \mathbb{R}^p$ then $\vec{u} + \vec{w} \succcurlyeq_\Theta \vec{v} + \vec{w}$) and Scale-Invariance (i.e., if $\vec{u} \succcurlyeq_\Theta \vec{v}$ and $q \in \mathbb{R}^+$ then $q\vec{u} \succcurlyeq_\Theta q\vec{v}$) properties.

If $\Theta = \{(\vec{u_i}, \vec{v_i}) \mid \vec{u_i}, \vec{v_i} \in \mathbb{R}^p, i = 1, \ldots, k\}$ be the set of input preferences and $W_\Theta = \{\vec{u_i} - \vec{v_i} \mid (\vec{u_i}, \vec{v_i}) \in \Theta, i = 1, \ldots, k\}$ the set of (difference) vectors then we have shown (in Lemma 6, Chapter 4) that the set defined by:

$$\mathbf{C}(W_\Theta) = \{\vec{u} \in \mathbb{R}^p \mid \exists q_1, \ldots, q_k \in \mathbb{R}^+ \cup \{0\}, \text{ such that } \vec{u} \geq \sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i})\}$$

is a *positive convex cone*. We defined (in Definition 28, Chapter 4) a pre-order relation $\succcurlyeq^\Theta$ on $\mathbb{R}^p$ as:

$$\vec{u} \succcurlyeq^\Theta \vec{v} \Leftrightarrow \vec{u} - \vec{v} \in \mathbf{C}(W_\Theta) \tag{5.1}$$

In Proposition 4 (Section 4.4, Chapter 4), we have shown that $\succcurlyeq^\Theta$ extends the Pareto and satisfies Scale-Invariance and Independence properties. Proposition 6 (Section 4.4, Chapter 4) gives that the relation $\succcurlyeq^\Theta$ and the induced preference relation $\succcurlyeq_\Theta$ are equal. That means, for given any $\vec{u}, \vec{v} \in \mathbb{R}^p$ to test whether $\vec{u} \succcurlyeq_\Theta \vec{v}$ it is equivalent to check if $\vec{u} \succcurlyeq^\Theta \vec{v}$. In Section 4.4.3, we presented three different approaches for testing $\vec{u} \succcurlyeq^\Theta \vec{v}$. The first is based on the linear programming approach, where we we solve the system of p linear inequalities. The second is using the distance algorithm, which measures the distance between point and a convex cone. The third is based on the matrix multiplication, where the matrix is constructed using the input preferences.

The dominance between finite sets of utility values is defined in similar fashion, i.e., for given $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^p$ we define

$$\mathcal{U} \succcurlyeq_\Theta \mathcal{V} \Leftrightarrow \forall \vec{v} \in \mathcal{V}\ \exists \vec{u} \in \mathcal{U} \text{ such that } \vec{u} \succcurlyeq_\Theta \vec{v}.$$

For given $\mathcal{U} \subset \mathbb{R}^p$, the maximal set of $\mathcal{U}$ is defined as $\max_{\succcurlyeq_\Theta}(\mathcal{U}) = \{\vec{u} \in \mathcal{U} \mid \nexists \vec{v} \in \mathcal{U}, \vec{v} \succcurlyeq_\Theta \vec{u}\}$.

We obtain the algorithm ELIM-MOID-TOF from Algorithm 3, which makes use

of the decision maker's imprecise trade-offs. We just replace the operations $+'$ and $\max$ with the operations $+^{\Theta}$ and $\max^{\Theta}$, which are defined as follows: $\max^{\Theta}(\mathcal{U},\mathcal{V}) = \max_{\succcurlyeq_{\Theta}}(\mathcal{U}\cup\mathcal{V})$ and $\mathcal{U}+^{\Theta}\mathcal{V} = \max_{\succcurlyeq_{\Theta}}(\mathcal{U}+\mathcal{V})$, where the $\max_{\succcurlyeq_{\Theta}}(\mathcal{U})$ is the undominated set of elements of the finite set $\mathcal{U}\subseteq\mathbb{R}^p$ with respect to the relation $\succcurlyeq_{\Theta}$.

The algorithm ELIM-MOID-TOF eliminates $\succcurlyeq_{\Theta}$-dominated utility vectors during the computation leading to the set of maximal values of expected utility. Alternatively, one can first generate the Pareto optimal set of the utility values and then eliminate $\succcurlyeq_{\Theta}$-dominated elements. However, the experimental results in Section 5.5 (Table 5.2) indicate that this will typically be much less computationally.

Approximating the Maximal Values of Expected Utility Set

We extend the algorithm ELIM-MOID-TOF to compute the (ϵ,λ)-covering (given in Algorithm 4, Section 5.3) of the maximal (or undominated) values of expected utility with respect to $\succcurlyeq_{\Theta}$. The extended algorithm ELIM-MOID-TOF$_{\epsilon}$ is obtained from Algorithm 3, by replacing the $+'$ and $\max$ operators with $+^{\Theta}_{\epsilon}$ and $\max^{\Theta}_{\epsilon}$, respectively, which are defined as follows: $\mathcal{U}+^{\Theta}_{\epsilon}\mathcal{V} = \max_{\geq^{\lambda}_{\epsilon}}(\max^{\Theta}(\mathcal{U}+\mathcal{V}))$, $\max^{\Theta}_{\epsilon}(\mathcal{U},\mathcal{V}) = \max_{\geq^{\lambda}_{\epsilon}}(\max^{\Theta}(\mathcal{U}\cup\mathcal{V}))$, where $\max^{\Theta}_{\epsilon}(\mathcal{U})$ is the (ϵ,λ)-covering of the set of $\succcurlyeq^{\Theta}$-undominated utility values.

The results in Section 5.5 (Tables 5.2 and 5.1) indicate that the ϵ-covering and $\succcurlyeq^{\Theta}$-undominated utility values are significantly smaller than the corresponding Pareto set. The algorithm ELIM-MOID-TOF$_{\epsilon}$ is the combination of these two, so we can expect that the maximal values of expected utility generated by ELIM-MOID-TOF$_{\epsilon}$ is even smaller than that of ELIM-MOID-TOF. This indicates that ELIM-MOID-TOF$_{\epsilon}$ will eliminate some maximal expected utility values generated with respect to $\succcurlyeq^{\Theta}$. In order to see how far the sets of maximal values of expected utility generated by ELIM-MOID-TOF and ELIM-MOID-TOF$_{\epsilon}$ we compute the closeness between them using the following procedure:

If meu and meu_{ϵ} be the maximal expected utility value sets generated by ELIM-MOID-TOF and ELIM-MOID-TOF$_{\epsilon}$, respectively, then we define the function $dist$, which measures the closeness of the set meu to the meu_{ϵ}, and it is

defined as

$$dist(meu, meu_\epsilon) = \max_{\vec{u} \in meu} \min_{\vec{v} \in meu_\epsilon} ed(\vec{u}, \vec{v})$$

where ed is the Euclidean distance between $\vec{u}$ and $\vec{v}$, which is defined as,

$$ed(\vec{u}, \vec{v}) = \sqrt{\sum_{i=1}^{p} (u_i - v_i)^2}.$$

$dist(meu, meu_\epsilon)$ gives the closeness of the set meu to the set meu_ϵ, which is computed by taking the maximum of the minimum Euclidean distances from every $\vec{u} \in meu$ to each $\vec{v} \in meu_\epsilon$.

5.5 Experiments

In this section, we present the experimental results obtained with the proposed variable elimination algorithms on randomly generated multi-objective influence diagrams. Experiments are run on a 2.6GHz quad-core processor with 4GB of RAM. The algorithms ELIM-MOID (Section 5.2.4), ELIM-MOID$_\epsilon$ (Section 5.3) and ELIM-MOID-TOF (Section 5.4) respectively, were implemented in C++ (32 bit). For the trade-offs case, firstly we implemented the algorithm ELIM-MOID-TOF using both the linear program based technique and the approach of [Zheng and Chew, 2009] that measures the distance between point and the convex cone, and we present only the results obtained with the former because the individual performances of both the methods are very much comparable. We denote the results obtained using this approach with ELIM-MOID-TOF (LP). Secondly, the algorithm ELIM-MOID-TOF is implemented for the faster $\succcurlyeq_\Theta$-based dominance checks compiled by use of a matrix. We name ELIM-MOID-TOF (MATRIX) for the results obtain using matrix-based $\succcurlyeq_\Theta$ dominance checks.

Problem Instances

We generate a class of random influence diagrams defined by the tuple $\langle C, D, k, par, r, art, p \rangle$, where C is the number of chance variables, D is the number of decision variables, k is the maximum domain size of the utility functions, par is the number of parents for each variable in the graph, r is the number of root nodes, art is the arity, i.e., the number of arguments of the utility functions

and finally p is the number of objectives for each decision.

The graph of the influence diagram is created by picking $C + D - r$ number of variables out of $C + D$ number of variables. For each selected variable we then assign par parents from all the variables which precede it. The parent variables are selected relative to some ordering and we ensure that the decision variables are connected by a direct path. In the end we add D utility nodes where each one is assigned art parent variables selected randomly from the chance and decision variables.

In all our experiments we generate random influence diagram instances with the following parameter settings: $C \in \{15, 25, 35, 45, 55\}$, $D \in \{5, 10\}$, $k = 2$, $par = 2$, $r = 5$, and $art = 3$, respectively. For each generated instance around 25% of the chance variables are assigned deterministic conditional probability tables (CPTs), i.e., they contain 0 and 1 entries in their tables. The remaining CPTs are filled randomly using a uniform distribution. The utility vectors with p objectives are generated randomly, where each objective value is allocated uniformly at random between 1 and 30.

For analysing the results we focus on the average CPU time (in seconds) taken for solving each problem instance along with its average size, standard deviation and median of the maximal expected utility sets. In addition to this, we also report the average induced width (w^*) of the instances, obtained using a minfill elimination ordering [Dechter, 2000b].

Random Trade-offs

The crucial part of the experimental evaluation is generating consistent random trade-offs. We make use of the random generator from the previous chapter (Section 4.6) for generating consistent random trade-offs. As we have already seen, the random generator is characterized by the two parameters, namely the number of two-way or binary trade-offs K and the number of three-way trade-offs T, respectively, where each of them is defined as follows.

Two-way trade-offs: Select randomly a pair of objectives (i, j), where $i \neq j$, out of p objectives. Choose randomly a, b and c in $[0.1, 1)$ with a uniform distribution. Generate the binary trade-offs $ae_i - be_j$ and $be_j - ace_i$, where e_i and e_j are the i-th and j-th unit vectors in $\mathbb{R}^p$, respectively. These two vectors represents that the decision maker is willing to allow a decrease in one objective in order to gain in the other.

Three-way trade-offs: Choose randomly three objectives (i, j, k) out of p objectives and generate a trade-off vector $ae_i + be_j - ce_k$. It represents that the decision maker is willing to allow c units of decay in objective k to gain a units of the i-th objective plus b units of the j-th objective.

Impact of the ϵ-covering

Table 5.1 is the summary of the results obtain with algorithms ELIM-MOID$_\epsilon$ where $\epsilon \in \{0.1, 0.2, 0.3\}$ and ELIM-MOID. The algorithms are run on the problem instances with 5 and 10 decision variables. For each problem class (i.e., the combination (C, D, p)), we record the number of instances solved (#) out of 20 instances within the time (20 minutes) or memory limit, average time (time) taken for solved instances, average size (avg) of the Pareto frontier or ϵ-covering, standard deviation (std) of the Pareto frontier or ϵ-covering and median (med) of the Pareto frontier or ϵ-covering. We can notice that ELIM-MOID$_\epsilon$ dominates ELIM-MOID across all problem instances, further, ELIM-MOID solves instances with relatively small number of variables and easily runs out of time or memory for the instances with large number of variables. Whereas, the algorithm ELIM-MOID$_\epsilon$ solves relatively large problem instances while generating smaller ϵ-coverings. As ϵ gets bigger we can see that a large number of problems are being solved; the reason is that the corresponding logarithmic grid gets coarser (i.e., fewer cells) as the value of ϵ increases. For example, ELIM-MOID manages to solve only 55% of the problem instance in an average of about 60 seconds resulting the Pareto sets with approximately 2,100 elements on problem class $\langle 35, 5, 2\rangle$. On the same problem class, ELIM-MOID$_\epsilon$ solves all the problem instances for $\epsilon = 0.01, 0.1, 0.3$ and we can see that both solving time and size of the Pareto sets reduce as ϵ increases.

Figure 5.3 shows the mean and standard deviation distributions of the size of the ϵ-coverings generated for problem classes $\langle 35, 5, 2\rangle$, $\langle 35, 5, 3\rangle$, $\langle 35, 5, 5\rangle$ and $\langle 35, 5, 10\rangle$, respectively, as a function of ϵ. The results clearly indicate that, as ϵ increases the size of the ϵ-covering decreases significantly. In addition, as ϵ increases the algorithm solves considerably more number of problems.

Table 5.1: Results with algorithms ELIM-MOID and ELIM-MOID$_\epsilon$ on random influence diagrams. Time limit 20 minutes.

size	w^*	ELIM-MOID					$\epsilon = 0.01$					$\epsilon = 0.1$					$\epsilon = 0.3$				
(C,D,O)		#	time	avg	stdev	med	#	time	avg	stdev	med	#	time	avg	stdev	med	#	time	avg	stdev	med
(15,5,2)	9	16	10.77	3,601	7,422	1,330	20	18.48	2,051	4,811	92	20	0.06	87	150	14	20	0.03	18	24	5
(25,5,2)	11	12	17.28	3,663	6,952	1,623	16	58.47	1,340	2,835	137	20	0.76	157	321	13	20	0.17	24	42	6
(35,5,2)	14	11	60.63	2,104	2,092	2,046	20	141.86	4,554	8,979	344	20	1.49	112	167	23	20	1.12	16	20	5
(45,5,2)	16	5	304.08	7,131	6,086	6,917	18	56.57	1,791	3,183	804	20	24.44	121	199	69	20	2.53	21	27	12
(55,5,2)	18	5	267.74	8,227	11,166	4,266	18	58.91	3,428	6,363	1,270	20	17.59	80	113	23	20	16.60	10	14	5
(15,5,3)	9	13	21.12	3,827	9,993	429	8	8.52	1,247	1,477	973	17	51.31	7,220	12,355	140	19	1.89	470	675	16
(25,5,3)	11	2	163.84	4,638	4,518	4,578	8	103.75	5,167	8,122	2,063	18	77.98	2,641	4,409	876	20	1.43	139	229	62
(35,5,3)	14	0					1	49.56	2,200	0	2,200	13	83.25	6,113	8,614	390	20	37.26	1,326	2,713	200
(45,5,3)	16	1	0.74	907	0	907	3	317.34	30,025	22,643	35,248	15	57.58	5,689	15,495	55	20	76.48	1,256	3,746	20
(55,5,3)	18	0					3	19.95	711	794	220	14	165.69	898	1,918	106	20	85.02	1,434	4,715	43
(15,5,5)	9	7	183.32	19,973	21,437	9,659	3	111.53	7,267	3,789	6,628	7	80.26	2,368	4,541	586	13	10.97	156	235	59
(25,5,5)	11	1	199.13	25,021	0	25,021	0					6	222.08	5,142	5,993	6,499	9	106.95	6,432	17,449	133
(35,5,5)	14	0					0					1	36.04	636	0	636	6	305.37	21	9	27
(45,5,5)	16	0					0					0					7	51.94	6,620	5,898	8,250
(55,5,5)	18	0					0					0					4	77.25	1,556	2,488	3,091
(15,10,2)	12	11	221.24	5,516	6,653	1,946	17	192.55	11,783	27,315	173	20	10.74	1,074	3,122	23	20	6.01	153	470	9
(25,10,2)	17	1	0.62	688	0	688	14	208.05	4,391	12,479	235	19	49.35	186	544	16	20	14.18	42	125	6
(35,10,2)	20	0					2	490.39	3,788	1,601	2,695	15	95.18	266	475	73	16	79.63	68	152	18
(45,10,2)	22	0					1	512.51	4,136	0	4,136	9	196.46	493	647	189	9	125.28	87	111	54
(55,10,2)	26	0					0					1	590.92	20	0	20	1	148.03	7	0	7
(15,10,3)	12	0					0					2	53.32	312	17	164	12	118.71	4,429	9,303	47
(25,10,3)	17	0					0					2	104.96	2,822	2,584	2,703	8	215.39	1,960	2,005	2,678
(35,10,3)	20	0					0					1	15.38	49	0	49	4	272.04	3,496	5,659	6,918
(45,10,3)	22	0					0					2	280.15	956	809	882	2	98.59	87	70	78
(55,10,3)	26	0					0					0					0				
(15,10,5)	12	0					0					0					1	112.00	429	0	429
(25,10,5)	17	0					0					0					1	790.34	584	0	584
(35,10,5)	20	0					0					0					0				
(45,10,5)	22	0					0					0					0				
(55,10,5)	26	0					0					0					0				

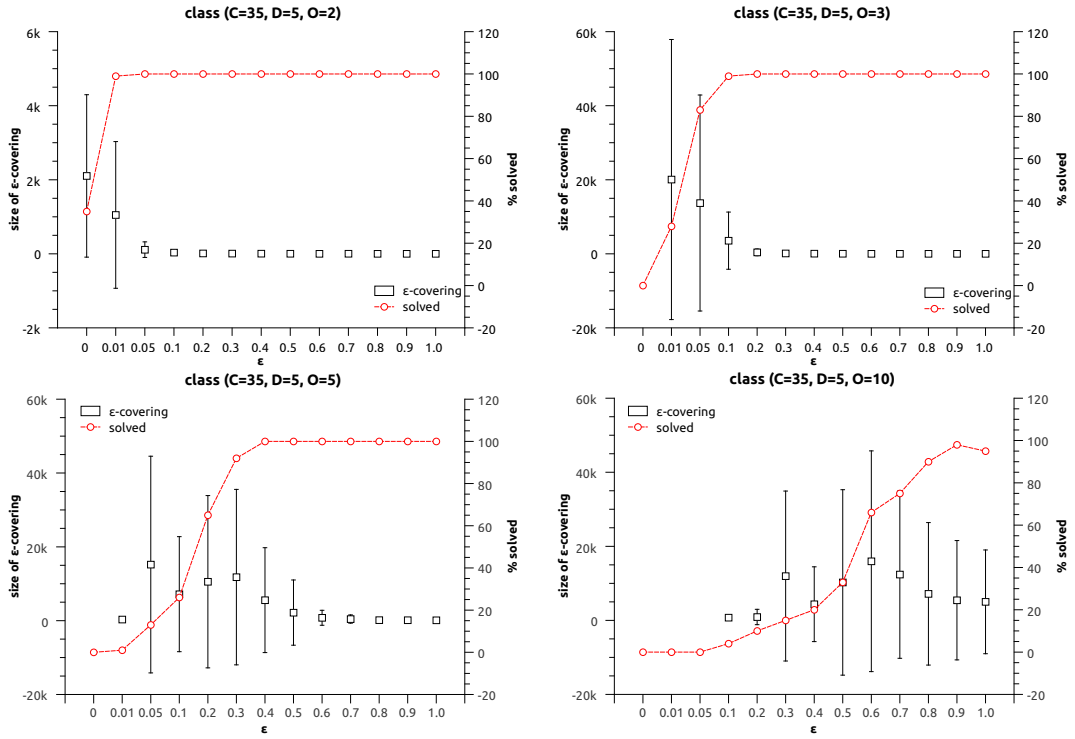


Figure 5.3: Distribution (mean and standard deviation) of the ϵ-covering size as a function of the ϵ value and plots showing the percentage of instances solved out of 100. Time limit 20 minutes.

Impact of Imprecise Trade-offs

Table 5.2 shows the results obtained with the algorithms ELIM-MOID and ELIM-MOID-TOF. The algorithm ELIM-MOID-TOF is compiled using both

linear programming (ELIM-MOID-TOF(LP)) and matrix-based (ELIM-MOID-TOF(MATRIX)) approaches. For each problem class $\langle C, D, p\rangle$, we generated 10 random influence diagram instances. Each instance is then solved using 10 sets of random trade-offs generated using the algorithm RANDOM-TRADEOFFS. The setting of the parameters K, T, a, b, c is given in the header of each horizontal block. The column labeled by # represents the number of problem instances solved by the algorithms out of 10 and 100 instances, respectively. Clearly, we can see that there is a significant difference in results obtained with both the algorithms. First, the sizes of the expected utility sets computed by ELIM-MOID-TOFs are orders of magnitude smaller in comparison with the Pareto optimal sets generated by ELIM-MOID. Second, the median sizes obtained with ELIM-MOID-TOFs are much smaller than the ELIM-MOID ones and in some cases median values of the size of the expected utility sets are 1, implying that the decision maker has strong preferences between the objectives which makes the induced preference relation $\succcurlyeq_\Theta$ to be equivalent to a total order. Third, the amount of time taken to solve a large instance using ELIM-MOID-TOFs are orders of magnitude smaller than the time taken by ELIM-MOID.

We can clearly see that the algorithm ELIM-MOID-TOF (MATRIX) using the matrix-based dominance check between the utility vectors is much faster than the linear programming based approach ELIM-MOID-TOF (LP) and is able to solve noticeably more number of problems.

Table 5.3 reports the results obtained on bi-objective influence diagrams with 5 decision variables by varying the parameter c (from 0 to 1) that determines the strength of the trade-offs. If $c = 0$ then we get a single tradeoff between the pair $(1, 2)$ of objectives generated using RANDOM-TRADEOFFS, as c increases from 0 to 1 the preference relation $\succcurlyeq_\Theta$ gets stronger and we can see this from the results obtained with $c = 0.1$ and by keeping $c > 0.1$.

Figures 5.4 and 5.5 show the distribution of the size of the maximal sets generated by ELIM-MOID-TOF as a function of the number of two-way trade-offs K on problem classes $\langle 35, 5, 5\rangle$ and $\langle 35, 5, 10\rangle$, respectively. For these experiments, we generated 10 random influence diagrams for each problem class and then solved each instance with 10 randomly generated consistent trade-offs. That means the data points on the graphs and error bars represents the mean and standard deviation over 100 instances. We can see from the results that as the number of trade-offs increase the relation $\succcurlyeq_\Theta$ gets stronger. Consequently, greater number of problems are solved quickly and the sizes of the maximal

Table 5.2: Results comparing algorithms ELIM-MOID and ELIM-MOID-TOFs on random influence diagrams with random trade-offs. Time limit 20 minutes.

size (C,D,O)	w^*	ELIM-MOID #	time	avg	stdev	med	ELIM-MOID-TOF (LP) #	time	avg	stdev	med	ELIM-MOID-TOF (MATRIX) #	time	avg	stdev	med
$K=1; T=0; a,b,c \in [0.1,1)$																
(15,5,2)	9	9	139.58	2,714	2,864	1,673	90	40.77	68	180	2	96	5.14	189	631	3
(25,5,2)	11	7	18.25	2,344	2,614	269	90	22.23	30	105	1	95	7.01	114	400	1
(35,5,2)	14	2	283.29	9,115	8,934	9,024	73	76.01	67	173	1	81	35.60	186	634	1
(45,5,2)	16	2	397.72	7,596	7,536	7,566	70	28.54	34	107	1	75	12.98	87	243	1
(55,5,2)	18	4	717.68	10,422	5,851	14,896	70	36.13	28	68	1	80	75.39	249	682	2
$K=2; T=1; a,b,c \in [0.1,1)$																
(15,5,3)	9	6	10.69	4,889	4,069	5,830	85	34.62	48	135	2	87	5.72	255	801	4
(25,5,3)	11	2	4.42	4,000	3,938	3,969	70	18.20	41	95	2	71	39.61	445	1265	7
(35,5,3)	14	0					50	89.51	119	198	13	75	66.85	687	1813	26
(45,5,3)	16	2	242.68	15,431	1,729	8,580	52	28.18	41	73	4	57	82.37	361	1140	5
(55,5,3)	18	0					51	94.63	75	154	4	65	57.66	150	493	4
$K=6; T=3; a,b,c \in [0.1,1)$																
(15,5,5)	9	3	65.73	15,062	12,229	14,741	84	19.70	41	104	4	88	1.92	102	315	4
(25,5,5)	11	1	50.35	21,074	0	21,074	74	83.30	97	217	4	85	32.55	361	1119	6
(35,5,5)	14	0					59	63.69	101	225	8	68	23.34	616	1781	9
(45,5,5)	16	0					61	96.59	107	216	8	69	20.87	215	522	13
(55,5,5)	18	0					41	84.32	51	101	12	55	117.91	742	1867	33
$K=1; T=0; a,b,c \in [0.1,1)$																
(15,10,2)	12	5	91.82	6,856	8,280	3,175	65	53.28	31	95	1	76	28.47	387	1443	2
(25,10,2)	17	1	808.94	4,964	0	4,964	26	82.89	21	50	1	39	83.62	436	1158	1
(35,10,2)	20	0					12	283.73	2	1	1	21	231.13	89	236	1
(45,10,2)	22	0					7	206.47	78	175	7	11	73.83	387	588	16
(55,10,2)	26	0					0					0				
$K=2; T=1; a,b,c \in [0.1,1)$																
(15,10,3)	12	0					45	157.48	86	191	12	58	15.87	178	510	6
(25,10,3)	17	0					11	204.78	74	83	73	28	136.96	994	1995	67
(35,10,3)	20	0					3	303.42	8	8	4	12	231.19	313	669	3
(45,10,3)	22	0					3	158.17	4	4	1	4	234.08	11	12	5
(55,10,3)	26	0					0					0				
$K=6; T=3; a,b,c \in [0.1,1)$																
(15,10,5)	12	0					40	106.50	27	64	5	57	50.60	704	2157	11
(25,10,5)	17	0					21	104.70	67	114	10	26	24.13	493	1449	55
(35,10,5)	20	0					5	244.45	72	80	48	8	26.71	353	413	153
(45,10,5)	22	0					0					0				
(55,10,5)	26	0					0					0				

Table 5.3: Impact of the quality of the random trade-offs on bi-objective influence diagrams. Time limit 20 minutes.

size (C,D,O)	w^*	ELIM-MOID #	time	avg	stdev	med	ELIM-MOID-TOF #	time	avg	stdev	med	ELIM-MOID-TOF #	time	avg	stdev	med	#	time	avg	stdev	med
		$K=1; c=0$					$K=1; c=0$					$K=1; c=0.1$					$K=1; c>0.1$				
(15,5,2)	9	9	139.58	2,714	2,864	1,673	84	48.30	243	542	23	84	37.10	167	304	9	94	24.93	98	232	1
(25,5,2)	11	7	18.25	2,344	2,614	269	61	16.83	79	260	10	72	80.25	127	377	4	92	7.12	33	126	1
(35,5,2)	14	2	283.29	9,115	8,934	9,024	42	94.89	190	344	11	36	46.01	88	223	18	83	59.18	147	378	1
(45,5,2)	16	2	397.72	7,596	7,536	7,566	42	76.83	130	297	5	47	116.08	155	459	2	76	23.08	86	1	1
(55,5,2)	18	4	717.68	10,422	5,851	14,896	41	144.64	268	312	78	41	117.88	182	328	29	76	48.02	90	233	2

sets reduced dramatically. Moreover, as the number of trade-offs increase the relation $\succcurlyeq_\Theta$ gets closer to the total order and therefore the maximal expected utility sets contain only one element.

Imprecise Trade-offs and ϵ-covering

Table 5.4 summarizes the results obtained with the algorithms ELIM-MOID-TOF and ELIM-MOID-TOF$_\epsilon$. The algorithms are compiled using the matrix-based dominance approach. For ELIM-MOID-TOF$_\epsilon$ the ϵ value is set to 0.01

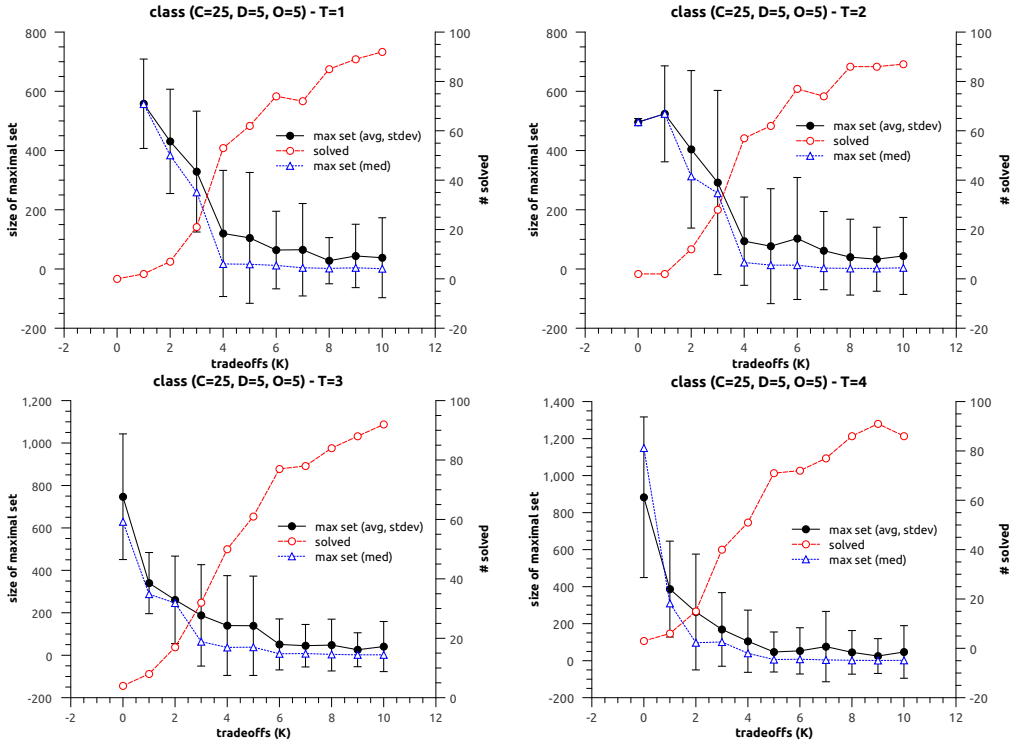


Figure 5.4: Distribution of the maximal sets as a function of the number of pairwise trade-offs K and fixed 3-way trade-offs (T). Mean and standard deviation are shown as well as the percentage of instances solved. Time limit 20 minutes.

and 0.1. We can notice that ELIM-MOID-TOF$_\epsilon$ outperforms ELIM-MOID-TOF in all problem classes. In addition, the algorithm ELIM-MOID-TOF$_\epsilon$ solves significantly more problems as ϵ value increases, generating even smaller (ϵ, λ)-coverings. For instance, for the problem class $(55, 5, 2)$ with $\epsilon = 0.01$ it managed to solve only 85% of the problem instances, whereas it solved 100% of the problem instances with $\epsilon = 0.1$. The column $(dist)$ (see Section 5.4) indicates the closeness of the set of maximal expected utility values generated by ELIM-MOID-TOF and ELIM-MOID-TOF$_\epsilon$. Clearly, as ϵ increases the distance, $dist$ between ELIM-MOID-TOF and ELIM-MOID-TOF$_\epsilon$ is also increase, which indicates that the the maximal expected utility value sets generated by ELIM-MOID-TOF$_\epsilon$ move farther away from the original maximal expected utility sets generated by ELIM-MOID-TOF.

5.6 Summary and Conclusion

We extended the variable elimination algorithm for influence diagrams to the case where there are more than one objective. Since the Pareto ordering for

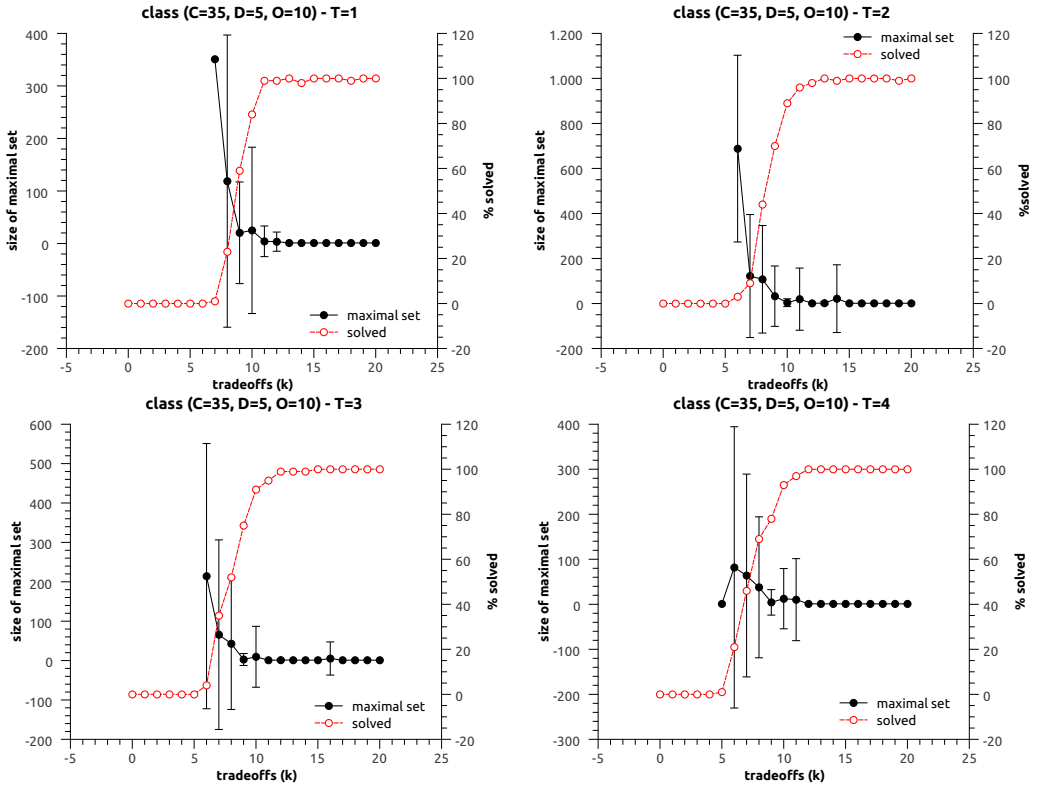


Figure 5.5: Distribution of the maximal sets as a function of the number of pairwise trade-offs K and fixed 3-way trade-offs (T). Mean and standard deviation are shown as well as the percentage of instances solved. Time limit 20 minutes.

multi-objective utility vectors is usually weak, as a result the expected utility sets become extremely large. We presented an efficient method to approximate the expected utility sets based on the notion of ϵ-covering which is much more practical computational approach than the exact method.

We introduced a formalism for bringing the imprecise preferences of the decision maker during the computation and presented a method for testing the dominance condition. We experimentally demonstrated that the proposed formalism generates maximal expected utility sets with relatively small cardinality and in some cases they contain only one element. Results indicate that adding even a small number of trade-offs can greatly reduce the maximal expected utility values. Also we showed that as the number of trade-offs increases the induced preference relation acts almost like a total order on the multi-objective utility values.

Table 5.4: Results comparing algorithms ELIM-MOID-TOF (MATRIX) and ELIM-MOID-TOF$_\epsilon$ (MATRIX) with varying ϵ values on random influence diagrams with random tradeoffs. Time limit 20 minutes.

size	w^*	ELIM-MOID-TOF (MATRIX)					ELIM-MOID-TOF$_\epsilon$ (MATRIX, $\epsilon = 0.01$)						ELIM-MOID-TOF$_\epsilon$ (MATRIX, $\epsilon = 0.1$)					
(C,D,O)		#	time	avg	stdev	med	#	time	avg	stdev	med	*dist*	#	time	avg	stdev	med	*dist*
$K = 1; T = 0; a, b, c \in [0.1, 1)$																		
(15,5,2)	9	96	5.14	189	631	3	100	0.17	51	157	4	0.13	100	0.018	9	20	1	1.37
(25,5,2)	11	95	7.01	114	400	1	100	0.56	13	36	1	0.1	100	0.16	2	4	1	0.8
(35,5,2)	14	81	35.60	186	634	1	100	6.65	24	75	1	0.1	100	0.62	3	4	1	0.9
(45,5,2)	16	75	12.98	87	243	1	100	28.18	356	1270	2	0.06	100	3.16	6	12	1	0.38
(55,5,2)	18	80	75.39	249	682	2	98	20.08	164	670	1	0.14	100	13.17	10	36	1	0.85
$K = 2; T = 1; a, b, c \in [0.1, 1)$																		
(15,5,3)	9	87	5.72	255	801	4	99	8.72	120	436	6	0.26	100	0.11	23	160	2	2.56
(25,5,3)	11	71	39.61	445	1265	7	98	18.29	336	981	9	0.21	100	0.53	20	55	2	2.02
(35,5,3)	14	75	66.85	687	1813	26	92	25.43	248	715	8	0.27	100	6.20	13	33	2	2
(45,5,3)	16	57	82.37	361	1140	5	85	47.18	203	796	10	0.13	100	16	30	217	3	1.38
(55,5,3)	18	65	57.66	150	493	4	85	38.87	210	713	5	0.2	100	30.57	43	185	2	1.5
$K = 6; T = 3; a, b, c \in [0.1, 1)$																		
(15,5,5)	9	88	1.92	102	315	4	93	19.50	223	998	4	0.13	100	0.19	38	156	2	1.66
(25,5,5)	11	85	32.55	361	1119	6	92	26.26	288	1100	6	0.14	98	14.20	19	57	2	1.54
(35,5,5)	14	68	23.34	616	1781	9	75	17.56	328	1022	7	0.14	98	5.86	111	731	3	1.16
(45,5,5)	16	69	20.87	215	522	13	80	30.50	525	2150	11	0.29	96	14.79	96	498	2	2.26
(55,5,5)	18	55	117.91	742	1867	33	80	71.68	552	1868	12	0.18	96	33.14	265	1255	4	1.4
$K = 1; T = 0; a, b, c \in [0.1, 1)$																		
(15,10,2)	12	76	28.47	387	1443	2	99	30.52	11	22	2	0.42	100	0.46	2	2	1	4.27
(25,10,2)	17	39	83.62	436	1158	1	91	53.83	48	197	2	0.33	97	18.20	3	6	1	2.08
(35,10,2)	20	21	231.13	89	236	1	48	59.63	57	238	1	0.12	77	81.03	17	58	3	0.72
(45,10,2)	22	11	73.83	387	588	16	24	176.63	38	104	7	0.45	36	138.74	2	3	1	3.43
(55,10,2)	26	0					13	235.84	9	24	1	-	17	225.03	2	3	1	-
$K = 2; T = 1; a, b, c \in [0.1, 1)$																		
(15,10,3)	12	58	15.87	178	510	6	89	34.46	67	205	7	0.5	100	2.14	10	50	2	5.68
(25,10,3)	17	28	136.96	994	1995	67	58	106.94	53	138	12	0.68	90	50.65	9	41	2	4.36
(35,10,3)	20	12	231.19	313	669	3	42	188.94	286	1057	8	0.48	77	81.03	17	58	3	1.92
(45,10,3)	22	4	234.08	11	12	5	19	224.65	1134	3431	13	0.39	31	108.60	19	32	4	0.94
(55,10,3)	26	0					0						0					
$K = 6; T = 3; a, b, c \in [0.1, 1)$																		
(15,10,5)	12	57	50.60	704	2157	11	80	24.74	282	828	16	0.53	97	18.72	112	471	4	4.89
(25,10,5)	17	26	24.13	493	1449	55	53	93.48	315	1470	13	0.27	91	31	63	223	4	2.03
(35,10,5)	20	8	26.71	353	413	153	37	123.33	242	880	9	0.6	41	55.44	54	127	3	3.19
(45,10,5)	22	0					15	172.98	19	28	5	-	20	224.75	3	3	2	-
(55,10,5)	26	0					0						0					

Chapter 6

Conclusion and Future Directions

In this chapter, we summarize the achievements of this thesis and briefly discuss a number of possible future directions we would like to look at in the future.

6.1 Summary and conclusion

The aim of this thesis is to help the decision maker to make better decisions in a multi-objective optimization context, under certainty and uncertainty. We considered multi-objective constraint optimization framework for representing the decision making problems with multiple objectives under certainty. We looked at the algorithms to solve multi-objective constraint optimization problems, including branch-and-bound and variable elimination algorithms. The branch-and-bound algorithms that we considered were recently developed in [Marinescu, 2009], which perform a depth-first traversal of an AND/OR search tree that captures the underlying structure of the problem instance, and use a mini-buckets algorithm to generate an upper bound, which is a set of utility vectors, at each node of the search tree.

The utility vectors associated with the solutions are compared using the Pareto ordering. However, the Pareto ordering does not discriminate the multi-objective utility vectors, consequently, the Pareto-undominated sets often become too large for the decision maker to handle. To overcome this issue, we defined a simple formalism for representing the imprecise trade-offs of the decision maker, which allows the decision maker, during the elicitation stage, to specify a preference for one multi-objective utility vector over another, and use

such input preferences to infer other preferences. In addition, we gave a sufficient condition for testing whether the input preferences are consistent. The consistent set of input preferences then induce a preference relation, which eliminates the dominated utility vectors during the computation. We showed that our trade-offs inference technique can be given an alternative semantics based on Multi-Attribute Utility Theory, where it is assumed that the decision maker compares utility vectors by a weighted sum of the individual values.

We developed a computational method based on the linear programming approach for checking the dominance between multi-objective utility vectors, which can be determined by using a linear programming solver and we showed that the (incomplete) algorithm of [Zheng and Chew, 2009] can be an alternative approach for testing the dominance between multi-objective utility vectors. Since the multi-objective constraint optimization algorithms need to make many dominance checks with respect to the preference relation induced by the imprecise trade-offs, for computational efficiency we constructed a matrix that represents the preferences of the decision maker. We then compiled the dominance check by use of this matrix and showed that it can achieve almost an order of magnitude speed up over the linear programming approach.

During the search, the guiding upper bound sets can become large and therefore can have a dramatic impact on the performance of the search algorithms. We developed efficient methods for both the Pareto and trade-offs case, which reduce the upper bound sets by incrementally replacing a selection of the elements by an upper bound of them, ensuring that the new reduced upper bound sets still maintain the key upper bound property. Our experimental results for the Pareto case indicated that using a singleton upper bound sets is best, and this considerably improved the current state-of-the-art. For the trade-offs case, the results suggested that it is usually best to use non-singleton upper bound set but of quite small cardinality.

We considered the framework of multi-objective influence diagrams, for representing the decision making problems with multiple objectives under uncertainty. We mainly focused on extending the variable elimination algorithm for solving these models, which results in a set of maximal values of expected utility, for the trade-offs case. Our experimental results showed that the proposed formalism for trade-offs generates maximal expected utility sets with relatively small cardinality and in some cases even a small number of consistent input preferences can greatly reduce the maximal expected utility values.

In addition, we explored the notion of ϵ-covering for approximating the Pareto frontier, which heavily depends on the Pareto ordering. Since the ϵ-dominance does not satisfy the transitive property, we used a finer dominance relation, known as (ϵ, λ)-dominance, where $\lambda \in (0, 1)$. We showed that the (ϵ, λ)-dominance satisfies the transitivity and other properties which are needed for the variable elimination algorithm. We experimentally demonstrated that this new approximation technique allows us to scale up to much larger problems than the original Pareto dominance.

6.2 Future Directions

In this section, we discuss some possible future directions of the work in this thesis. The first possible future work is distributed constraint optimization problems, which are a general model for distributed problem solving that have a wide range of applications in multi-agent systems. The second is limited memory influence diagrams which are the generalization of standard influence diagrams.

6.2.1 Distributed Constraint Optimization Problem

A Distributed Constraint Satisfaction Problem (DisCSP) is a constraint satisfaction problem, which gives a way to model and reason about the interactions between agents local decisions. In a DisCSP each agent has some variables and needs to choose their values. Variables owned by different agents are connected by constraints, these constraints are known as inter-agent constraints. However, agents must coordinate their choice of values to their variables so that all inter-agent constraints are satisfied [Yokoo et al., 1998]. Finding a value assignment to variables that satisfy inter-agent constraint is viewed as consistency among agents. In order to achieve this, agents check value assignment to their variables for local consistency and exchange messages among them to check consistency of their proposed assignment against constraints that contain variables which belong to other agents [Wahbi, 2012].

A Distributed Constraint Optimization Problem (DCOP) is a DisCSP with an additional objective function which is modelled as a set of constraints. Each agent in DCOP is only assumed to have knowledge about the constraints in which their variables are involved. The goal is to find the value assignment

of the variables such that the objective function is maximized or minimized. Formally, a DCOP is defined in [Modi et al., 2005] as a quadruple $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, where:

- $\mathbf{A} = \{A_1, \ldots, A_p\}$ is a set of p agents.
- $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of n variables such that each variable X_i is controlled by one agent in **A**.
- $\mathbf{D} = \{D_1, \ldots, D_n\}$ is a set of finite domains of variables in **X**.
- $\mathbf{F} = \{f_1, \ldots, f_r\}$ is a set of utility constraints, where each f_i is defined over some subset of variables in **X**.

Several algorithms have been proposed for solving DCOPs in the last decade, which include BnB-Adopt [Modi et al., 2005] that uses a depth-first branch-and-bound strategy, the synchronous branch-and-bound (SyncBB) [Hirayama and Yokoo, 1997] that performs assignments sequentially and synchronously, non-commitment branch-and-bound [Chechetka and Sycara, 2006] which is another synchronous polynomial-space search algorithm, asynchronous forward bounding [Gershman et al., 2006] that can be seen as an improvement of SyncBB.

We propose, in brief, as a possible area of future work to consider DCOPs extended to the case where utility functions take values in $\mathbb{R}^p$, for instance see [Okimoto et al., 2013], in this case DCOPs as multi-objective DCOPs and develop different algorithms, such as AND/OR branch-and-bound described in Section 4.3 to solve multi-objective DCOPs. We plan to, develop formalisms, such as the formalism presented in Section 4.4.1 for combining and comparing information about agents preferences, and using this information to solve MO-DOSPs.

6.2.2 Limited Memory Influence Diagrams

In this thesis, we presented the framework of influence diagrams for reasoning about decision making under uncertainty. The two important assumptions of the influence diagram representation are the no-forgetting assumption, that implies a perfect recall of the past decisions, and the assumption of a total order on the decisions. The limited memory influence diagram (LIMID) [Nilsson and Lauritzen, 2000] relaxes these two assumptions. That means, with a LIMID

representation, it is possible to model multi-stage decision problems with an unordered sequence of decisions and decision problems in which the no-forgetting assumption is not appropriate.

LIMIDs can also be used for modelling decision making problems under uncertainty involving multiple decision makers with limited information. In addition, every decision problem under uncertainty that can be represented as an influence diagram can also be represented as a LIMID. The structure of a LIMID is identical to an influence diagram but due to the relaxation of the two fundamental assumptions the solution process of LIMID is more complex than the solution process of influence diagrams.

As a possible future work, we plan to look at extending the LIMIDs to include multi-objective utility functions, in this case we call LIMIDs as multi-objective LIMIDs. We will investigate the development of the algorithms to solve when there is no information available about the preferences of the decision maker, and also the case when we have some partial information of the preferences of the decision maker.

Bibliography

[Al-Najjar and Alsyouf, 2003] Al-Najjar, B. and Alsyouf, I. (2003). Selecting the most efficient maintenance approach using fuzzy multiple criteria decision making. *International Journal of Production Economics*, 84(1):85–100.

[Arnborg, 1985] Arnborg, S. (1985). Efficient algorithms for combinatorial problems on graphs with bounded decomposability? A survey. *BIT Numerical Mathematics*, 25(1):1–23.

[Ashby, 2000] Ashby, M. (2000). Multi-objective optimization in material design and selection. *Acta Materialia*, 48(1):359–369.

[Bacchus and Grove, 1995] Bacchus, F. and Grove, A. (1995). Graphical models for preference and utility. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 1995)*, pages 3–10. Morgan Kaufmann Publishers Inc.

[Bana et al., 2005] Bana, C. A., De Corte, J.-M., Vansnick, J.-C., et al. (2005). On the mathematical foundation of MACBETH. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 409–437. Springer.

[Bayardo and Miranker, 1995] Bayardo, R. J. and Miranker, D. P. (1995). On the space-time trade-off in solving constraint satisfaction problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 14, pages 558–562. Morgan Kaufmann.

[Berkelaar et al., 2006] Berkelaar, M., Eikland, K., and Notebaert, P. (2006). lpsolve, version 5.5. *Eindhoven University of Technology, Design Automation Section, Eindhoven, The Netherlands*.

[Berman et al., 1993] Berman, V. P., Naumow, G. Y., and Podinovskii, V. V. (1993). Interval value tradeoffs methodology and techniques of multicriteria decision analysis. In *User-Oriented Methodology and Techniques of Decision Analysis and Support*, pages 144–149. Springer.

[Bertele and Brioschi, 1972] Bertele, U. and Brioschi, F. (1972). *Nonserial Dynamic Programming*. Academic Press, Inc.

[Binder et al., 1997] Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2-3):213–244.

[Boutilier et al., 2001] Boutilier, C., Bacchus, F., and Brafman, R. I. (2001). UCP-networks: A directed graphical representation of conditional utilities. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 2001)*, pages 56–64. Morgan Kaufmann Publishers Inc.

[Boutilier et al., 1997] Boutilier, C., Brafman, R., Geib, C., and Poole, D. (1997). A constraint-based approach to preference elicitation and decision making. pages 19–28.

[Boutilier et al., 2003] Boutilier, C., Patrascu, R., Poupart, P., and Schuurmans, D. (2003). Constraint-based optimization with the minimax decision criterion. In *Principles and Practice of Constraint Programming (CP 2003)*, pages 168–182. Springer.

[Boutilier et al., 2005] Boutilier, C., Patrascu, R., Poupart, P., and Schuurmans, D. (2005). Regret-based utility elicitation in constraint-based decision problems. In *International Joint Conference on Artificial Intelligence (IJCAI 2005)*, volume 9, pages 929–934.

[Bouyssou, 1989] Bouyssou, D. (1989). *Modelling Inaccurate Determination, Uncertainty, Imprecision using Multiple Criteria*. Springer.

[Boyd and Vandenberghe, 2004] Boyd, S. P. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Brafman and Domshlak, 2009] Brafman, R. and Domshlak, C. (2009). Preference handling – An introductory tutorial. *AI magazine*, 30(1).

[Brafman et al., 2006] Brafman, R. I., Domshlak, C., and Shimony, S. E. (2006). On graphical modeling of preference and importance. *Journal of Artificial Intellgence Ressearch (JAIR)*, 25:389–424.

[Brans and Mareschal, 2005] Brans, J.-P. and Mareschal, B. (2005). Promethee methods. In *Multiple criteria decision analysis: State of the Art Surveys*, pages 163–186. Springer.

[Braziunas, 2011] Braziunas, D. (2011). *Decision-theoretic elicitation of generalized additive utilities*. PhD thesis, University of Toronto.

[Braziunas and Boutilier, 2006] Braziunas, D. and Boutilier, C. (2006). Preference elicitation and Generalized Additive Utility. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1573. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Braziunas and Boutilier, 2007] Braziunas, D. and Boutilier, C. (2007). Minimax regret based elicitation of generalized additive utilities. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 2007)*, pages 25–32, Vancouver.

[Braziunas and Boutilier, 2012] Braziunas, D. and Boutilier, C. (2012). Local utility elicitation in GAI models. *arXiv preprint arXiv:1207.1361*.

[Casas-Garriga and Balcázar, 2004] Casas-Garriga, G. and Balcázar, J. L. (2004). Coproduct transformations on lattices of closed partial orders. In *Graph Transformations*, pages 336–351. Springer.

[Cavin et al., 2004] Cavin, L., Fischer, U., Glover, F., and Hungerbühler, K. (2004). Multi-objective process design in multi-purpose batch plants using a tabu search optimization algorithm. *Computers & chemical engineering*, 28(4):459–478.

[Chechetka and Sycara, 2006] Chechetka, A. and Sycara, K. (2006). No-commitment branch and bound search for distributed constraint optimization. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1427–1429. ACM.

[Cooper, 1988] Cooper, G. (1988). A method for using belief networks as influence diagrams. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 1988)*, pages 9–16.

[Crowley, 2004] Crowley, M. (2004). Evaluating influence diagrams. *University of British Columbia*.

[Dattorro, 2005] Dattorro, J. (2005). *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA.

[de Campos and Ji, 2008] de Campos, C. and Ji, Q. (2008). Strategy selection in influence diagrams using imprecise probabilities. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 2008)*, pages 121–128.

[Dechter, 1999] Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85.

[Dechter, 2000a] Dechter, R. (2000a). A new perspective on algorithms for optimizing policies under uncertainty. In *International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 72–81.

[Dechter, 2000b] Dechter, R. (2000b). A new perspective on algorithms for optimizing policies under uncertainty. In *Artificial Intelligence Planning Systems (AIPS 2000)*, pages 72–81.

[Dechter, 2003] Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

[Dechter and Mateescu, 2004] Dechter, R. and Mateescu, R. (2004). Mixtures of deterministic-probabilistic networks and their AND/OR search space. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 2004)*, pages 120–129. AUAI Press.

[Dechter and Mateescu, 2007a] Dechter, R. and Mateescu, R. (2007a). AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2):73–106.

[Dechter and Mateescu, 2007b] Dechter, R. and Mateescu, R. (2007b). AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106.

[Dechter and Pearl, 1989] Dechter, R. and Pearl, J. (1989). Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366.

[Dechter and Rish, 2003] Dechter, R. and Rish, I. (2003). Mini-buckets: A general scheme of approximating inference. *Journal of ACM*, 50(2):107–153.

[Dechter and Rish, 1998] Dechter, R. and Rish, I. (October 1998). Mini-buckets: A general scheme for approximating inference. Technical report.

[Dehnokhalaji et al., 2011] Dehnokhalaji, A., Korhonen, P. J., Köksalan, M., Nasrabadi, N., and Wallenius, J. (2011). Convex cone-based partial order for multiple criteria alternatives. *Decision Support Systems*, 51(2):256–261.

[Diehl and Haimes, 2004] Diehl, M. and Haimes, Y. (2004). Influence diagrams with multiple objectives and tradeoff analysis. *IEEE Transactions On Systems, Man, and Cybernetics Part A*, 34(3):293–304.

[Domshlak, 2002] Domshlak, C. (2002). *Modeling and Reasoning about Preferences with CP-nets*. PhD thesis, Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel.

[Domshlak et al., 2011] Domshlak, C., Hüllermeier, E., Kaci, S., and Prade, H. (2011). Preferences in AI: An overview. *Artificial Intelligence*, 175(7):1037–1052.

[Dubus et al., 2009] Dubus, J.-P., Gonzales, C., and Perny, P. (2009). Multiobjective optimization using GAI models. In *International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1902–1907.

[Dyer et al., 1992] Dyer, J. S., Fishburn, P. C., Steuer, R. E., Wallenius, J., and Zionts, S. (1992). Multiple criteria decision making, multiattribute utility theory: the next ten years. *Management Science*, 38(5):645–654.

[Eckenrode, 1965] Eckenrode, R. T. (1965). Weighting multiple criteria. *Management Science*, 12(3):180–192.

[Ehrgott, 1999] Ehrgott, M. (1999). *Multicriteria Optimization*. Springer.

[Engau, 2008] Engau, A. (2008). Variable preference modeling with ideal-symmetric convex cones. *Journal of Global Optimization*, 42(2):295–311.

[Ezawa, 1986] Ezawa, K. J. (1986). *Efficient Evaluation of Influence Diagrams*. University Microfilms.

[Figueira et al., 2005] Figueira, J., Greco, S., and Ehrgott, M. (2005). *Multiple Criteria Decision Analysis—State of the Art Surveys*. Springer International Series in Operations Research and Management Science Volume 76.

[Fishburn, 1967] Fishburn, P. C. (1967). Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, 8(3):335–342.

[Fishburn, 1970] Fishburn, P. C. (1970). Utility theory for decision making. Technical report, Defense Technical Information Center Document.

[Fishburn, 1974] Fishburn, P. C. (1974). Lexicographic orders, utilities and decision rules: A survey. *Management Science*, pages 1442–1471.

[French, 1986] French, S. (1986). *Decision Theory: An Introduction to the Mathematics of Rationality*. Halsted Press, New York, NY, USA.

[Freuder et al., 2010] Freuder, E. C., Heffernan, R., Wallace, R. J., and Wilson, N. (2010). Lexicographically-ordered constraint satisfaction problems. *Constraints*, 15(1):1–28.

[Freuder and Quinn, 1985] Freuder, E. C. and Quinn, M. J. (1985). Taking advantage of stable sets of variables in constraint satisfaction problems. In *International Joint Conference on Artificial Intelligence (IJCAI 1985)*, volume 85, pages 1076–1078.

[Gavanelli, 2002] Gavanelli, M. (2002). An algorithm for multi-criteria optimization in CSPs. In *European Conference on Artificial Intelligence (ECAI 2002)*, volume 2, pages 136–140.

[Gennert and Yuille, 1988] Gennert, M. A. and Yuille, A. L. (1988). Determining the optimal weights in multiple objective function optimization. In *Second International Conference on Computer Vision*, pages 87–89.

[Gershman et al., 2006] Gershman, A., Meisels, A., and Zivan, R. (2006). Asynchronous forward-bounding for distributed constraints optimization. *Frontiers in Artificial Intelligence and Applications*, 141:103.

[Gonzales and Perny, 2004] Gonzales, C. and Perny, P. (2004). GAI networks for utility elicitation. *Principles of Knowledge Representation (KR 2004)*, 4:224–234.

[Gonzales et al., 2011] Gonzales, C., Perny, P., and Dubus, J.-P. (2011). Decision making with multiple objectives using GAI networks. *Artificial Intelligence*, 175(1):1153–1179.

[Güler, 2010] Güler, O. (2010). *Foundations of Optimization*, volume 258. Springer Science+ Business Media.

[Hansson, 1994] Hansson, S. O. (1994). Decision theory. *A brief introduction. Department of Philosophy and the History of Technology. Royal Institute of Technology. Stockholm*.

[Harju, 2006] Harju, T. (2006). Ordered sets. *Collection of lecture notes available from http://users. utu. fi/harju/orderedsets/Mainorder. pdf*.

[Henrion et al., 1991] Henrion, M., Breese, J. S., and Horvitz, E. J. (1991). Decision analysis and expert systems. *AI magazine*, 12(4):64.

[Hirayama and Yokoo, 1997] Hirayama, K. and Yokoo, M. (1997). Distributed partial constraint satisfaction problem. In *Principles and Practice of Constraint Programming (CP 1997)*, pages 222–236. Springer.

[Holland, 2005] Holland, A. (2005). *Risk Management in Combinatorial Auctions*. PhD thesis, University College Cork.

[Howard, 1984] Howard, Ronald A., M. J. E. (1984). *Readings on the Principles and Applications of Decision Analysis*. Strategic Decisions Group.

[Howard and Matheson, 1984a] Howard, R. and Matheson, J. (1984a). Influence diagrams. In *Readings on the Principles and Applications of Decision Analyis*, pages 721–762.

[Howard and Matheson, 1984b] Howard, R. and Matheson, J. (1984b). *Influence Diagrams. The Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, CA, USA.

[Howard, 1967] Howard, R. A. (1967). Value of information lotteries. *IEEE Trans. Systems Science and Cybernetics*, 3(1):54–60.

[Howard, 2007] Howard, R. A. (2007). The foundations of decision analysis revisited. *Advances in decision analysis: From foundations to applications*, pages 32–56.

[Howard and Matheson, 1981] Howard, R. A. and Matheson, J. E. (1981). *Influence Diagrams*. Stanford Research Institute.

[Howard and Matheson, 2005] Howard, R. A. and Matheson, J. E. (2005). Influence diagram retrospective. *Decision Analysis*, 2(3):144–147.

[Hüllermeier and Fürnkranz, 2010] Hüllermeier, E. and Fürnkranz, J. (2010). On loss functions in label ranking and risk minimization by pairwise learning. *Journal of Computer and System Sciences*, 76(1):49–62.

[Hunt et al., 2010] Hunt, B. J., Wiecek, M. M., and Hughes, C. S. (2010). Relative importance of criteria in multiobjective programming: A cone-based approach. *European Journal of Operational Research*, 207(2):936–945.

[Jacquet-Lagreze and Siskos, 1982] Jacquet-Lagreze, E. and Siskos, J. (1982). Assessing a set of additive utility functions for multicriteria decision-making, the UTA method. *European Journal of Operational Research*, 10(2):151–164.

[Jensen et al., 1994] Jensen, F., Jensen, V., and Dittmer, S. (1994). From influence diagrams to junction trees. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 1994)*, pages 367–363.

[Jensen and Jensen, 1994] Jensen, F. V. and Jensen, F. (1994). Optimal junction trees. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 1994)*, pages 360–366. Morgan Kaufmann Publishers Inc.

[Kahneman and Tversky, 1979] Kahneman, D. and Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, pages 263–291.

[Kask and Dechter, 2001] Kask, K. and Dechter, R. (2001). A general scheme for automatic search heuristics from specification dependencies. *Artificial Intelligence*.

[Kask et al., 2005] Kask, K., Dechter, R., Larrosa, J., and Dechter, A. (2005). Unifying cluster-tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166 (1-2):165–193.

[Keeney and Raiffa, 1993] Keeney, R. and Raiffa, H. (1993). *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Cambridge University Press.

[Keeney, 1982] Keeney, R. L. (1982). Decision analysis: An overview. *Operations Research*, 30(5):803–838.

[Keeney, 1993] Keeney, R. L. (1993). *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Cambridge University Press.

[Kikuti and Cozman, 2007] Kikuti, D. and Cozman, F. (2007). Influence diagrams with partially ordered preferences. In *Proceedings of 3rd Multidisciplinary Workshop on Advances in Preference Handling*.

[Kikuti et al., 2011] Kikuti, D., Cozman, F., and Filho, R. (2011). Sequential decision making with partially ordered preferences. *Artificial Intelligence*, 175(7-8):1346–1365.

[Kjærulff and Madsen, 2005] Kjærulff, U. B. and Madsen, A. L. (2005). Probabilistic networks – An introduction to Bayesian networks and Influence diagrams. *Aalborg University*.

[Koksalan, 1984] Koksalan, M. M. (1984). Multiple criteria decision making with discrete alternatives. *Dissertation Abstracts International Part B: Science and Engineering*, 45(6).

[Koksalan et al., 1984] Koksalan, M. M., Karwan, M. H., and Zionts, S. (1984). An improved method for solving multiple criteria problems involving discrete alternatives. *Systems, Man and Cybernetics, IEEE Transactions on*, (1):24–34.

[Köksalan and Sagala, 1995] Köksalan, M. M. and Sagala, P. N. (1995). Interactive approaches for discrete alternative multiple criteria decision making with monotone utility functions. *Management Science*, 41(7):1158–1171.

[Korhonen et al., 1984] Korhonen, P., Wallenius, J., and Zionts, S. (1984). Solving the discrete multiple criteria problem using convex cones. *Management Science*, 30(11):1336–1345.

[Leyton-Brown et al., 2000] Leyton-Brown, K., Pearson, M., and Shoham, Y. (2000). Towards a universal test suite for combinatorial auction algorithms. In *ACM Electronic Commerce*, pages 66–76.

[Lin, 2010] Lin, M. (2010). *Multi-objective constrained optimization for decision making and optimization for system architectures*. PhD thesis, Massachusetts Institute of Technology.

[López-Díaz and Rodríguez-Muñiz, 2007] López-Díaz, M. and Rodríguez-Muñiz, L. (2007). Influence diagrams with super value nodes involving imprecise information. *European Journal of Operational Research*, 179(1):203–219.

[Madsen, 1996] Madsen, A. L. (1996). *Deep Green: A Bridge Playing System Based on Bayesian Networks*. PhD thesis, Aalborg University, Institute of Electronic Systems, Department of Mathematics and Computer Science.

[Malakooti, 1989] Malakooti, B. (1989). Ranking multiple criteria alternatives with half-space, convex, and non-convex dominating cones: quasi-concave and quasi-convex multiple attribute utility functions. *Computers & Operations Research*, 16(2):117–127.

[Marinescu, 2008] Marinescu, R. (2008). *And/OR Search Strategies for Combinatorial Optimization in Graphical Models*. PhD thesis, Long Beach, CA, USA. AAI3318708.

[Marinescu, 2009] Marinescu, R. (2009). Exploiting problem decomposition in multi-objective constraint optimization. In *Principles and Practice of Constraint Programming (CP 2009)*, pages 592–607. Springer.

[Marinescu, 2010] Marinescu, R. (2010). A new approach to influence diagrams evaluation. In *Research and Development in Intelligent Systems XXVI*, pages 107–120. Springer.

[Marinescu, 2011] Marinescu, R. (2011). Efficient approximation algorithms for multi-objective constraint optimization. In *Algorithmic Decision Theory*, pages 150–164. Springer.

[Marinescu and Dechter, 2005] Marinescu, R. and Dechter, R. (2005). AND/OR branch-and-bound for graphical models. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 224–229.

[Marinescu and Dechter, 2006] Marinescu, R. and Dechter, R. (2006). Dynamic orderings for AND/OR branch-and-bound search in graphical models. In *European Conference on Artificial Intelligence (ECAI 2006)*, pages 138–142.

[Marinescu and Dechter, 2009a] Marinescu, R. and Dechter, R. (2009a). AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491.

[Marinescu and Dechter, 2009b] Marinescu, R. and Dechter, R. (2009b). Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1492–1524.

[Marinescu et al., 2012] Marinescu, R., Razak, A., and Wilson, N. (2012). Multi-objective influence diagrams. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 2012)*, pages 574–583.

[Mateescu and Dechter, 2005] Mateescu, R. and Dechter, R. (2005). AND/OR cutset conditioning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 230–235.

[Mateescu and Dechter, 2012] Mateescu, R. and Dechter, R. (2012). The relationship between AND/OR search and variable elimination. *arXiv preprint arXiv:1207.1407*.

[Maua and de Campos, 2011] Maua, D. and de Campos, C. (2011). Solving decision problems with limited information. In *Advances in Neural Information Processing Systems (NIPS 2011)*, pages 603–611.

[Modi et al., 2005] Modi, P. J., Shen, W.-M., Tambe, M., and Yokoo, M. (2005). ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180.

[Murat Köksalan and Sagala, 1995] Murat Köksalan, M. and Sagala, P. N. (1995). An approach to and computational results on testing the form of a decision maker's utility function. *Journal of Multi-Criteria Decision Analysis*, 4(3):189–202.

[Murat Köksalan and Taner, 1992] Murat Köksalan, M. and Taner, O. V. (1992). An approach for finding the most preferred alternative in the presence of multiple criteria. *European Journal of Operational Research*, 60(1):52–60.

[Nilsson and Höhle, 2001] Nilsson, D. and Höhle, M. (2001). *Computing Bounds on Expected Utilities for Optimal Policies based on Limited Information*. The Royal Veterinary and Agricultural University, DINA Rsearch Report.

[Nilsson and Lauritzen, 2000] Nilsson, D. and Lauritzen, S. L. (2000). Evaluating influence diagrams using limids. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 2000)*, pages 436–445. Morgan Kaufmann Publishers Inc.

[Noghin and Tolstykh, 2000] Noghin, V. and Tolstykh, I. (2000). Using quantitative information on the relative importance of criteria for decision making. *Computational Mathematics and Mathematical Physics*, 40(11):1529–1536.

[Noghin, 1997] Noghin, V. D. (1997). Relative importance of criteria: a quantitative approach. *Journal of Multi-Criteria Decision Analysis*, 6(6):355–363.

[Noghin, 2001] Noghin, V. D. (2001). What is the relative importance of criteria and how to use it in mcdm. In *Multiple Criteria Decision Making in the New Millennium*, pages 59–68. Springer.

[Okimoto et al., 2013] Okimoto, T., Clement, M., and Inoue, K. (2013). Aof-based algorithm for dynamic multi-objective distributed constraint optimization. In *Multi-disciplinary Trends in Artificial Intelligence*, pages 175–186. Springer.

[Olmsted, 1983] Olmsted, S. M. (1983). *On representing and solving decision problems*. PhD thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

[Papadimitriou and Yannakakis, 2000] Papadimitriou, C. and Yannakakis, M. (2000). On the approximability of trade-offs and optimal access to web sources. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 86–92.

[Papadimitriou, 1991] Papadimitriou, C. H. (1991). On selecting a satisfying truth assignment. *Proceedings of the Conference on the Foundations of Computer Science*.

[Pareto, 1964] Pareto, V. (1964). *Cours d'economie Politique*. Librairie Droz.

[Pearl, 1982] Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *Associate for the Advancement of Artificial Intelligence (AAAI 1982)*, pages 133–136.

[Pearl, 1986] Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[Pearl, 1996] Pearl, J. (1996). Decision making under uncertainty. *ACM Computing Surveys (CSUR)*, 28(1):89–92.

[Qi, 1994] Qi, R. (1994). *Decision Graphs: algorithms and applications to influence diagram evaluation and high level path planning with uncertainty*. PhD thesis, Department of Computer Science, University of British Columbia.

[Qi and Poole, 1993] Qi, R. and Poole, D. (1993). *Decision Graph Search*. Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada.

[Qi and Poole, 1995] Qi, R. and Poole, D. (1995). A new method for influence diagram evaluation. *Computational Intelligence*, 11(3):498–528.

[Qi et al., 1994] Qi, R., Zhang, L., and Poole, D. (1994). Solving asymmetric decision problems with influence diagrams. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 1994)*, pages 491–497. Morgan Kaufmann Publishers Inc.

[Raiffa, 1993] Raiffa, H. (1993). Decision analysis: Introductory lectures on choices under uncertainty. 1968. *MD Computing: Computers in Medical Practice*, 10(5):312.

[Raiffa and Keeney, 1976] Raiffa, H. and Keeney, R. (1976). Decisions with multiple objectives: Preferences and value tradeoffs. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*.

[Rockafellar, 1997] Rockafellar, R. T. (1997). *Convex analysis*. Number 28. Princeton university press.

[Rollon, 2008] Rollon, E. (2008). *Multi-Objective Optimization for Graphical Models*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

[Rollón and Larrosa, 2006a] Rollón, E. and Larrosa, J. (2006a). Bucket elimination for multiobjective optimization problems. *Journal of Heuristics*, 12(4-5):307–328.

[Rollón and Larrosa, 2006b] Rollón, E. and Larrosa, J. (2006b). Bucket elimination for multiobjective optimization problems. *Journal of Heuristics*, 12(4-5):307–328.

[Rollon and Larrosa, 2006] Rollon, E. and Larrosa, J. (2006). Constraint optimization techniques for exact multiobjective optimization. In *Proceedings of the Seventh International Conference on Multi-Objective Programming and Goal Programming*.

[Rollon and Larrosa, 2007] Rollon, E. and Larrosa, J. (2007). Multi-objective russian doll search. In *Proceedings Of The National Conference On Artificial Intelligence*, volume 22, page 249. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Rollon and Larrosa, 2011] Rollon, E. and Larrosa, J. (2011). On mini-buckets and the min-fill elimination ordering. In *Principles and Practice of Constraint Programming–CP 2011*, pages 759–773. Springer.

[Roy, 1989] Roy, B. (1989). Main sources of inaccurate determination, uncertainty and imprecision in decision models. *Mathematical and Computer Modelling*, 12(10):1245–1254.

[Roy, 1996] Roy, B. (1996). *Multicriteria Methodology for Decision Analysis*. Kluwer Academic Press.

[Rudas and Fodor, 2008] Rudas, I. J. and Fodor, J. (2008). Intelligent systems. *International Journal of Computers, Communications & Control*, 3:132–138.

[Saaty, 1977] Saaty, T. L. (1977). A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15(3):234–281.

[Saaty, 1988] Saaty, T. L. (1988). *What is the Analytic Hierarchy Process?* Springer.

[Saaty, 2003] Saaty, T. L. (2003). Decision-making with the AHP: Why is the principal eigenvector necessary? *European Journal of Operational Research*, 145(1):85–91.

[Saaty, 2005] Saaty, T. L. (2005). The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 345–405. Springer.

[Saaty and Hu, 1998] Saaty, T. L. and Hu, G. (1998). Ranking by eigenvector versus other methods in the analytic hierarchy process. *Applied Mathematics Letters*, 11(4):121–125.

[Shachter, 1986] Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, 34(6):871–882.

[Shachter, 1988] Shachter, R. D. (1988). Probabilistic inference and influence diagrams. *Operations Research*, 36(4):589–604.

[Shachter, 1990] Shachter, R. D. (1990). An ordered examination of influence diagrams. *Networks*, 20:535–563.

[Shachter and Peot, 1992] Shachter, R. D. and Peot, M. A. (1992). Decision making using probabilistic inference methods. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 1992)*, pages 276–283. Morgan Kaufmann Publishers Inc.

[Shafer and Shenoy, 1990] Shafer, G. R. and Shenoy, P. P. (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2(1-4):327–351.

[Shenoy, 1992] Shenoy, P. (1992). Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(1):463–484.

[Siskos et al., 2005] Siskos, Y., Grigoroudis, E., and Matsatsinis, N. F. (2005). UTA methods. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 297–334. Springer.

[Tamura, 1976] Tamura, K. (1976). A method for constructing the polar cone of a polyhedral cone, with applications to linear multicriteria decision problems. *Journal of Optimization Theory and Applications*, 19(4):547–564.

[Tatman and Shachter, 1990] Tatman, J. and Shachter, R. (1990). Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(1):365–379.

[Tversky, 1996] Tversky, A. (1996). Contrasting rational and psychological principles of choice. *Wise choices*, pages 5–21.

[Von Neumann and Morgenstern, 1945] Von Neumann, J. and Morgenstern, O. (1945). Theory of games and economic behavior. *Bulletin of the Amererican Mathematical Society*, 51:498–504.

[Von Winterfeldt et al., 1986] Von Winterfeldt, D., Edwards, W., et al. (1986). *Decision Analysis and Behavioral Research*, volume 604. Cambridge University Press Cambridge.

[Wahbi, 2012] Wahbi, M. (2012). *Algorithms and ordering heuristics for distributed constraint satisfaction problems*. PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc.

[Walsh, 2007] Walsh, T. (2007). Representing and reasoning with preferences. *AI Magazine*, 28(4):59.

[Wang et al., 2009] Wang, H., Shao, S., Zhou, X., Wan, C., and Bouguettaya, A. (2009). Web service selection with incomplete or inconsistent user preferences. pages 83–98.

[Wiecek, 2007] Wiecek, M. M. (2007). Advances in cone-based preference modeling for decision making with multiple criteria. *Decision Making in Manufacturing and Services*, 1(1-2):153–173.

[Wilson and Marinescu, 2012] Wilson, N. and Marinescu, R. (2012). An axiomatic framework for influence diagram computation with partially ordered utilities. In *International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 210–220.

[Xia and Wu, 2005] Xia, W. and Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 48(2):409–425.

[Xiang and Ye, 2001] Xiang, Y. and Ye, C. (2001). A simple method to evaluate influence diagrams. In *Proceedings of the Third International Conference on Cognitive Science*.

[Yaman et al., 2011] Yaman, F., Walsh, T. J., Littman, M. L., and Desjardins, M. (2011). Democratic approximation of lexicographic preference models. *Artificial Intelligence*, 175(7):1290–1307.

[Yokoo et al., 1998] Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K. (1998). The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 10(5):673–685.

[Yoon and Hwang, 1995] Yoon, K. P. and Hwang, C.-L. (1995). *Multiple Attribute Decision Making: An Introduction*. Number 104. London: SAGE Publications.

[Yu, 1974] Yu, P. (1974). Cone convexity, cone extreme points, and nondominated solutions in decision problems with multiobjectives. *Journal of Optimization Theory and Applications*, 14(3):319–377.

[Yuan and Hansen, 2009] Yuan, C. and Hansen, E. A. (2009). Efficient computation of jointree bounds for systematic map search. In *International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1982–1989.

[Yuan et al., 2012] Yuan, C., Wu, X., and Hansen, E. A. (2012). Solving multistage influence diagrams using branch-and-bound search. *arXiv preprint arXiv:1203.3531*.

[Zhang, 1998] Zhang, N. L. (1998). Probabilistic inference in influence diagrams. *Computational Intelligence*, 14(4):475–497.

[Zhang and Poole, 1994] Zhang, N. L. and Poole, D. (1994). A simple approach to Bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*.

[Zhang and Poole, 1992] Zhang, N. L. and Poole, D. L. (1992). Stepwise-decomposable influence diagrams. In *Principles of Knowledge Representation (KR 1992)*, pages 141–152.

[Zheng and Chew, 2009] Zheng, Y. and Chew, C. (2009). Distance between a point and a convex cone in n-dimensional space: computation and applications. *IEEE Transactions on Robotics*, 25(6):1397–1412.

[Zhou et al., 2009] Zhou, L., Liu, W., and Wang, L. (2009). Influence diagram model with interval-valued utilities. In *IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 601–605.

[Zionts and Wallenius, 1980] Zionts, S. and Wallenius, J. (1980). Identifying efficient vectors: Some theory and computational results. *Operations Research*, 28(3-Part-II):785–793.