WASHINGTON UNIVERSITY IN ST.LOUIS

School of Engineering & Applied Science
Department of Computer Science and Engineering

# Preference Elicitation in Constraint-Based Models: Models, Algorithms, and Applications

by

Atena M. Tabakhi

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2021
St. Louis, Missouri

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

I express my appreciation to all those that helped me throughout the pursuit of my Ph.D. degree – Family, friends, colleagues, professors, collaborators, and staff. Below, I would like to extend my special thanks to some of those who contributed to this dissertation in some way.

My first appreciation goes to my advisor Dr. William Yeoh for his marvelous supervision, guidance, patience, motivation during these years. These past years have been a fantastic journey, and I could not have imagined having a better advisor throughout this journey. I had the honor to share with you the best moment of my life – my wedding – when none of my family could be there for me. Thank you, William, for being such an outstanding researcher, an excellent teacher, and a true friend, just like a family to me. It has been an absolute pleasure working with you.

I am very grateful to my committee members, Dr. Roman Garnett, Dr. Chien-Ju Ho, Dr. Alvitta Ottley, and Dr. Judy Goldsmith, for asking insightful questions, providing valuable suggestions and constructive feedback throughout my research. Special thanks to Roman for encouraging me to explore exciting research areas such as Bayesian optimization and active learning and integrate them into my research projects.

I greatly appreciate the successful women who I had the privilege to know over the past couple of years, Dr. Judy Goldsmith, Dr. Marion Neumann, and Dr. Alvitta Ottley, for their time, advice, and moral support. Judy, I will never forget how we met at AAAI16. Since then, you have been such a great friend and inspiration to me. I am truly grateful for your time answering my random questions, encouraging, and advising me during my job hunting.

Many thanks to the best lab-mates I could ever ask for, Khoi Hoang, Christabel Wayllace, Stylianos Vasileiou, and Ashwin Kumar, for providing fruitful comments and kind support during my practice talks. Special thanks to my best friend and lab-mate, Khoi, for your tremendous support during the most overwhelming time of my Ph.D. life. Thanks to many other friends not listed here for the fun times we had together.

Last but not least, I extend my gratitude to the love of my life, Dr. Reza Tourani, a supportive friend, loving husband, and sometimes a strict advisor. Thank you, Reza for your constant support, fruitful discussions and extremely helpful suggestions that kept me going through the way. I am very fortunate to have you by my side – in all the ups and downs of our life – the happy and sad moments. Finally, to my parents, sisters, and brothers, thank you for your unconditional love, always believing in me, and encouraging me from far, far away, and around the world. Without your support and strength, I would not stand where I am standing now. I love you all!

Atena M. Tabakhi

*Washington University in Saint Louis*

*August 2021*

Dedicated to my parents.

ABSTRACT OF THE DISSERTATION

Preference Elicitation in Constraint-Based Models:

Models, Algorithms, and Applications

by

Atena M. Tabakhi

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2021

Research Advisor: Professor William Yeoh

Constraint-based models offer powerful approaches for describing and resolving many combinatorial optimization problems in a centralized or distributed environment. In such models, the goal is to find a value assignment to a set of variables given a set of preferences expressed by means of cost functions such that the sum over all costs is optimized. The importance of constraint-based models is outlined by the impact of their applications in a wide range of agent-based systems. Many real-life combinatorial problems can be naturally formalized using constraint-based models. Examples of such applications are supply-chain management, roster scheduling, meeting scheduling, combinatorial auctions, bioinformatics, and smart home automation.

The majority of these constraint-based models assume that all constraint costs are specified or known a priori. Unfortunately, such an assumption is impractical, especially in many human-in-the-loop applications, such as scheduling problems where constraints encode the preferences of human users. These constraints may not be fully specified because it is unrealistic to accurately know the preferences of users for all possible scenarios in an application. These constraint costs are only known after they are queried or elicited from domain experts or human users.

This dissertation proposes solutions for further improving the applicability of constraint-based models. These solutions aim for better formalizing and solving optimization problems, requiring human interactions to address the above limitation. Our core contributions are listed as follows:

We begin by introducing the uncertain constraint-based model that represents the uncertainty in users' preferences (i.e., constraint costs) as Gaussian distributions. To solve such a constraint-based model with uncertainty, we propose probabilistic heuristics that select a subset of constraints to elicit and choose those that significantly impact the overall solution quality. The elicitation of these preferences occurs prior to the execution of the search algorithm for an optimal solution.

Human users are likely bothered by repeated elicitations and will refuse to provide an unbounded number of preferences. Hence, as our next contribution, we propose the incomplete weighted constraint satisfaction problems with elicitation costs (IWCSPs+EC) that takes into consideration how much users are bothered by queries. To solve IWCSPs+EC, we offer three parameterized heuristics that allow users to trade off solution quality for fewer elicited preferences and faster computation times. Further, they provide theoretical quality guarantees for problems where elicitations are free.

Finally, we extend IWCSPs to distributed problems and introduce incomplete distributed constraint optimization problems (I-DCOPs). To solve I-DCOPs, we propose an extended version of SyncBB – a complete search algorithm – with two parameterized heuristics. These heuristics interleave the elicitation process with the search for an optimal solution. Our proposed heuristics for SyncBB allow users to trade off solution quality for fewer elicited preferences and faster computation times. To improve the scalability of our proposed framework, we offer an extended version of ALS-MGM – a local search algorithm – which can solve much larger I-DCOPs. Local search algorithms are computationally much faster and provide sub-optimal or close to optimal solutions. The number of elicited preferences is significantly smaller than SyncBB with heuristics, and its solution quality is not far from optimal.

We apply our proposed model to smart home device scheduling and distributed meeting scheduling applications with partial users' preferences. The empirical results show the significance of our contributions against random methods and the previous state-of-the-art approaches. Our models and heuristics thus extend the state-of-the-art in constraint reasoning to better model and solve agent-based applications with user preferences.

# Chapter 1

# Introduction

Centralized *Constraint Satisfaction Problems* (CSPs) [3, 19, 44, 135] and its variant *Weighted Constraint Satisfaction Problems* (WCSPs) [47, 56, 59], as well as its decentralized variant *Distributed Constraint Optimization Problems* (DCOPs) [14, 137, 145], are powerful paradigms for formulating many combinatorial and optimization problems. WCSPs are also known as *Constraint Optimization Problems* (COPs). When resources are distributed among a set of autonomous agents, COPs take the form of DCOPs, which are used to model cooperative combinatorial optimization problems. In this dissertation, we refer to these paradigms as "*Constraint-Based Models*."

The Constraint-Based Models are mathematical paradigms defined as a set of objects whose states must satisfy a number of constraints or limitations. Such models typically represent the entities in a problem as a homogeneous collection of finite constraints over (decision) variables. In Chapter 2, we will describe each of these frameworks formally in more details. In classical CSPs, the objective is to find a value assignment for a set of variables that satisfies a set of constraints. The assignments satisfying the constraints are called *solutions*. In WCSPs and DCOPs, the objective is to find an optimal solution, given a set of preferences expressed through cost functions [6, 101, 102]. And often, in these models, constraints can encode human users' preferences. Consequently, constraint costs corresponds to preferences that can be estimated from historical data or queried from domain experts or human users.

The importance of constraint-based models is outlined by the impact of their applications in a wide range of agent-based systems. Many real-life combinatorial problems can be naturally modeled using constraint-based models. It is just a matter of identifying the decision variables of the problem and how they are related. Typically the relations between the decision variables are described in the form of constraints. For example, there might be as many variables as tasks in a scheduling problem, each specifying its starting time, and constraints can model the relations among the tasks, such as "the beginning of task 2 must occur after the end of task 1." Similar models have been designed for many agent-based applications such as supply-chain management [34, 95], roster scheduling [1, 11], meeting scheduling [69], combinatorial auctions [99], bioinformatics [2, 12, 28], and smart home automation [31, 52, 98, 120].

A key drawback of these constraint-based models is the assumption that *all* the constraint costs are specified or known a priori. For instance, in several applications (e.g., a scheduling problem), some constraints encode the preferences of human users. Such constraints may not be fully specified because it is unrealistic to accurately know the preferences of users for all possible scenarios in an application. These constraint costs are only known after they are queried or elicited from domain experts or human users. Preference elicitation is crucial in such situations, which is a process of asking questions about the users' preferences. This process allows users to intelligently interact with the constraint-based solver (i.e., resolution algorithms) without being forced to state all their constraints, or preferences, at the beginning of the interaction. More specifically, preference elicitation is more pronounced in scenarios where the users want to avoid revealing all of their preferences at the beginning of the interaction due to privacy and security [52, 117, 119] reasons.

Such an assumption in constraint-based models restrains their capabilities to model and solve many combinatorial optimization problems in a centralized or decentralized manner. In this dissertation, to address this limitation, we investigate two approaches for eliciting constraint costs or preferences in constraint-based models. In the first approach, the elicitation process occurs before executing an algorithm to solve the underlying constraint-based problem. In the second approach, the elicitation process occurs while executing an algorithm to solve the underlying constraint-based problem. Researchers have partially investigated the latter by introducing new formulations for CSPs, which allow a set of constraints to be unspecified. Gelain *et al.* proposed the *Incomplete WCSP* (IWCSP) formulation [35], which extends WCSPs by allowing some constraints to be partially specified (i.e., the costs for some constraints are unknown). To solve IWCSPs, they introduced a series of algorithms that interleave a resolution algorithm's search process with the elicitation process. The search process seeks to find a good solution, while the preference elicitation process seeks to obtain some subset of cost functions from the user.

Despite the fact that the proposed models and resolution algorithms by Gelain *et al.* address the key drawback of the CSP models, it suffers from another assumption – they assume that the elicitation of preferences does not incur any cost. This assumption is not realistic as human users are likely bothered by repeated elicitations and will refuse to provide an unbounded number of preferences. In the next sections, we propose models and resolution algorithms that not only address this assumption but also tackle the key drawback in constraint-based models.

## 1.1 Hypotheses

Our interests lie in improving the practical applicability of constraint-based models to better model applications that involve users' preferences so that constraint-based models can be used to model a more general class of applications. In the previous section, we discussed the key drawback of constraint-based models and their limitations. To adequately address this drawback in constraint-based models, we hypothesize that:

> One can improve the applicability of constraint-based models by developing new formulations where constraint costs can be uncertain or unknown and applying elicitation strategies to constraint-based algorithms to solve such models.

As mentioned earlier, this dissertation's focus is on elicitation strategies in constraint-based models. Hence, we first classify elicitation approaches into "pre-execution" and "intra-execution" elicitation categories. Then, within each category, we discuss how to extend the constraint-based models when constraint costs are uncertain or unknown in a centralized or distributed manner.

1. Pre-execution elicitation approach:

   We decouple the elicitation process and the algorithm to solve the constraint-based model. As such, the elicitation process happens before solving the constraint-based model problem. The elicitation strategies can be applied to both centralized and decentralized constraint-based models in this approach: WCSPs/COPs and DCOPs. We hypothesize that: One can allow all constraint costs of a constraint-based model (i.e., centralized or decentralized) to be uncertain and assume that random variables represent them following Normal distributions. Then, one can employ probabilistic strategies to identify the constraints that highly affect the solution quality and realize the remaining from their corresponding distributions. Since the actual constraint costs are not retrieved from the users, we assume that the realization of constraints does not incur any penalty (e.g., any cognitive cost that represents how much a user is annoyed by those queries).

2. Intra-execution elicitation approach:

   We interleave the elicitation process and the algorithm to solve the constraint-based model. As such, the elicitation process happens while executing the algorithm to retrieve the actual constraint costs from the users needed to compute the problem's solution quality. To make the model realistic for a broader class of applications, we associate a penalty (e.g., cognitive cost) for eliciting a constraint cost and incorporate it in the solution quality. Typically, the algorithms to solve constraint-based problems modeled in a centralized manner are independent and different from constraint-based problems modeled in a decentralized manner. Consequently, the elicitation strategies applied to centralized algorithms differ from those applied to decentralized algorithms. Therefore, we carry out our investigation for centralized and decentralized approaches separately:

   - On the centralized approach, Gelain *et al.* extend WCSPs by allowing some constraint costs/preferences to be partially specified (i.e., the costs/preferences for some constraints are unknown) [35] (i.e., IWCSPs). However, it is unrealistic to assume the costs of elicitation of all unknown constraints are identical. To address this assumption, we hypothesize that: One can extend IWCSPs by associating costs to the elicitation of the unknown constraints. We assume that elicitation costs are not uniform. Then, to solve IWCSPs with elicitation costs, one can develop resolution algorithms that interleave the elicitation process with search strategies to optimize both constraint and elicitation costs.

- On the decentralized approach, we hypothesize that: One can extend DCOPs by allowing some constraint costs/preferences to be partially specified (i.e., the costs/preferences for some constraints are unknown) and associating costs to the elicitation of the unknown constraints. We assume that elicitation costs are not uniform. Then, to solve such incomplete DCOPs one can develop resolution algorithms that interleave elicitation process with distributed search strategies to optimize both constraint and elicitation costs.

## 1.2 Contributions

Constraint-based models have been well-studied since their perception and can model a wide range of combinatorial and optimization problems. However, still, some assumptions restrain their practical applicabilities. One of the critical drawbacks in constraint-based models is the assumption that almost all the constraints are specified or known a priori, which does not hold in many applications, for instance, in a meeting scheduling problem, where constraints encode human users' preferences. It is unrealistic to accurately know users' preferences for all possible scenarios in such a scheduling problem. These constraint costs or preferences are only known after being queried or elicited from domain experts or users. Thus, in this dissertation, we address this limitation using the approaches discussed in Section 1.1. Our contributions are outlined as follows:

1. To assess the hypothesis that allows all constraint costs of a constraint-based model (i.e., centralized or decentralized models) to be uncertain and employ Normal distributions to represent the uncertainty of the constraint costs, we introduce the "Uncertain

DCOPs/COPs" framework [107, 109]. The uncertain DCOP framework extends DCOPs where constraint costs are uncertain and represented as random variables following Normal distributions. To solve uncertain DCOPs, we develop heuristics to execute the preference elicitation prior to the search for an optimal solution of a DCOP. As the proposed heuristics are performed before searching for an optimal solution, we reside the proposed constraint-based model and its heuristics in the pre-execution elicitation approach category. The proposed elicitation strategies can be applied to both COPs and DCOPs since they are independent of their underlying resolution algorithms. We introduce an expected error as the evaluation metric. The expected error is the absolute difference between the solution quality of an oracle DCOP and an uncertain DCOP that only reveals a limited number of constraints. Our heuristics aim at minimizing the expected error by eliciting the constraints that highly affect the solution quality. To evaluate the significance of our heuristics, we carry out extensive experiments using multiple benchmarks such as random graphs and smart home device scheduling problems. The results of our experiments indicate that the expected error decreases with increasing the number of elicited constraints. Furthermore, by analyzing these results, we validate our hypothesis that one can allow the constraint costs to be uncertain and develop strategies to only reveal the constraints that highly affect the solution quality of the underlying constraint-based model.

2. To assess the hypotheses that one can allow some constraint costs to be partially specified in the constraint-based models and develop resolution algorithms to solve the constraint-based models with unknown constraints, we introduce the following frameworks:

   - Gelain *et al.* introduced the incomplete WCSP (IWCSP) framework, which extends WCSPs by allowing some of the constraint costs to be partially specified [35]. To solve IWCSPs, they proposed several resolution algorithms by interleaving elicitation with the search for an optimal solution. Even though their work addressed the key drawback of WCSPs – constraints are specified a priori – they assumed that eliciting constraints costs from human users do not incur any cognitive costs. Human users often get annoyed by an unlimited number of queries and interactions of the system with users. Therefore, they may hesitate to provide their actual feedback/preferences. To address this limitation, we assume that the eliciting constraint costs from users incur some cognitive costs that we refer to them as "elicitation costs" (i.e., abbreviated as EC) in this dissertation. More specifically, we introduced the "IWCSPs+EC" framework in which we associate penalties (i.e., elicitation costs) for eliciting the unknown constraint costs [106, 114]. We propose heuristics in conjunction with a variant of the depth-first search algorithm for WCSPs. Our proposed search algorithm embodies elicitation strategies and incorporates the elicitation costs into the solution quality.

Consequently, the proposed resolution algorithm finds the best possible solution that minimizes both the constraint and elicitation costs when elicitation costs are non-zero. We also parameterize our algorithm such that it takes in a user-defined threshold to tradeoff solution quality for fewer elicitations and faster runtimes. Thus, the solution quality can be bounded from above by the user-defined threshold. We carry out extensive experiments using multiple benchmarks such as random graphs and smart grids to evaluate the significance of the algorithm. Our experimental results indicate that the proposed algorithm can find the optimal solution when elicitation costs are zero eliciting only necessary constraints. When elicitation costs are non-zero, our algorithm can guarantee solution optimality. These results validate our hypothesis that one can allow the constraint costs to be partially specified and develop elicitation strategies to reveal the constraint costs required to find the best solution with smaller elicitation costs.

- We introduce the incomplete DCOP (I-DCOP) framework, which extends DCOPs by allowing some of the constraint costs to be partially specified [111, 134]. Similar to IWCSP+EC, we associate penalties for eliciting the unknown constraint costs. We propose heuristics in conjunction with a distributed branch-and-bound complete search algorithm for DCOPs. Our proposed algorithm embodies elicitation strategies and incorporates the elicitation costs into the solution quality.

  Consequently, the proposed resolution algorithm finds the best possible solution that minimizes both the constraint and elicitation costs (assuming elicitation costs are non-zero). In addition to the proposed distributed complete search algorithm to solve I-DCOPs, we combine elicitation strategies with an anytime local search algorithm. Our extended version of the anytime local search algorithm can solve larger scale I-DCOP problems and provide sub-optimal solutions to minimize both constraint and elicitation costs. We also parameterize our algorithms such that it takes in a user-defined threshold to tradeoff solution quality for fewer elicitations and faster runtimes. Thus, the solution quality provided by these algorithms can be bounded from above by a user-defined threshold. We carry out extensive experiments in multiple benchmarks such as random graphs and meeting scheduling problems to evaluate the significance of our algorithms. The experimental results indicate that our distributed branch-and-bound algorithm can find the optimal solution when elicitation costs are zero (i.e., elicitation is free), only elicits those constraint costs that are required to find the optimal solution. Thus, when elicitation is free, the proposed complete search algorithm can guarantee solution optimality. Additionally our proposed distributed local search algorithm provides sub-optimal solutions and using its anytime property it improves the solution quality at every iteration. These results validate our hypothesis that one can allow the constraint costs to be partially specified in a distributed environment and develop elicitation strategies to explicitly reveal the constraint costs required to find the best solution.

In the frameworks mentioned above (in a centralized and decentralized environment), we interleave search with elicitation. Thus elicitation occurs during the execution of a resolution algorithm. We reside the proposed centralized and decentralized constraint-based models and their resolution algorithms in the intra-execution elicitation approach category. Since the above frameworks are entirely independent of each other despite being in the same category, we dedicate a complete chapter to each framework.

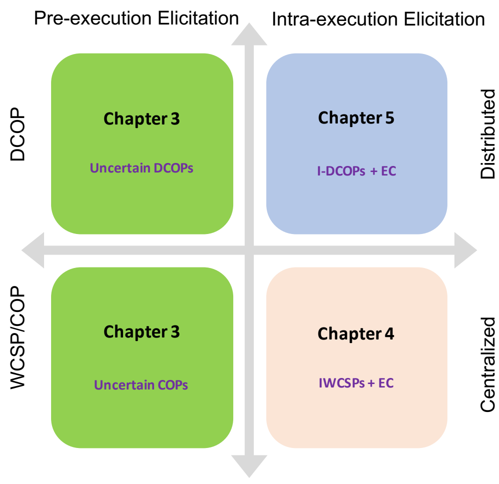


Figure 1.1: Structure of Contributions in this Dissertation

## 1.3 Dissertation Structure

This dissertation is the result of collaborating with other researchers and most of the work of this dissertation has been published in peer-reviewed conferences. We structured it as follows: In the next chapter, we give an overview of the centralized and decentralized/distributed constraint-based models such as CSPs, WCSPs, and DCOPs, respectively. In the same chapter, we briefly introduce some of the resolution algorithms of centralized and decentralized frameworks. Figure 1.1 shows the structure of this dissertation. The x-axis corresponds to the elicitation approach that is divided into pre-execution and intra-execution categories. The y-axis corresponds to the constraint-based models that are divided into two main categories of centralized and distributed approaches. In what follows, we will detail the original work, with references to those previous publications, and explain the contributions of each author.

- **Chapter 3:** The work in this chapter appears in

M. Tabakhi, A., Le, T., Fioretto, F., and Yeoh, W., Preference elicitation for DCOPs. In Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP), 2017.

Yeoh proposed the idea of the Uncertain DCOP model. Fioretto, Le, and Tabakhi proposed several elicitation heuristics to solve the uncertain model. Tabakhi and Fioretto modeled Smart Home Device Scheduling (SHDS) using DCOPs. Tabakhi developed, evaluated the model along with its heuristics, and conducted experiments under different benchmarks of random graphs and SHDS graphs. Yeoh, Fioretto, Tabakhi, and Le collaboratively wrote the paper.

- **Chapter 4:** The work in this chapter appears in

  M. Tabakhi, A., Yeoh, W., and Yokoo, M., Parameterized Heuristics for Incomplete Weighted CSPs with Elicitation Costs. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2019.

  The idea of the Incomplete WCSPs+EC model was conceived by Tabakhi and Yeoh originally. Yeoh and Yokoo proposed the use of the depth-first search technique to solve the incomplete model. Then, Tabakhi and Yeoh iteratively proposed heuristics to enhance the performance of the algorithm. Tabakhi developed the model, resolution algorithm, elicitation heuristics and conducted the experiments under different benchmarks of random graphs and SHDS graphs. Most of the paper was written by Yeoh, and Tabakhi added the heuristic and experimental sections to the paper.

- **Chapter 5:** The work in this chapter appears as short papers in

  Xiao, Y., M. Tabakhi, A., and Yeoh, W., Embedding Preference Elicitation Within the Search for DCOP Solutions. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2020.

Tabakhi proposed the idea of the Incomplete DCOP (I-DCOP) model. Yeoh and Tabakhi iteratively proposed two distributed heuristics to improve the performance of the synchronous branch-and-bound distributed algorithm. Xiao and Tabakhi collaboratively developed synchronous branch-and-bound with two heuristics. Tabakhi conducted the experiments under different benchmarks of random graphs and meeting scheduling graphs. Tabakhi and Yeoh collaboratively wrote the paper.

> M. Tabakhi, A., Xiao, Y., Yeoh, W., and Zivan, R., Branch-and-Bound Heuristics for Incomplete DCOPs. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2021.

Later, Tabakhi proposed a distributed local search algorithm to solve I-DCOPs and Zivan proposed applying anytime local search framework for each agent to keep track of the state of the best solution found so far. Tabakhi proposed the heuristics for enhancing the algorithm. Tabakhi and Xiao collaboratively developed the local search algorithm. Tabakhi conducted experiments under different benchmarks of random graphs and meeting scheduling graphs. Most of the paper was written by Tabakhi and revised by Yeoh. The complete version of this work has yet to be published, and the candidate intends to submit the work at the International Conference on Distributed Artificial Intelligence (DAI), 2022.

- **Chapter 6:** We discuss potential future directions and briefly mention our collaborative work that are not part of this dissertation but part of the future work, which can be further extended and improved. Our collaborative work appears in:

> Le, T., M. Tabakhi, A., Long, T., Yeoh, W., and Son, T., Preference Elicitation with Interdependency and User Bother Cost. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2018.

Long, Yeoh, and Son proposed the idea of preference elicitation using the matrix completion technique. Le proposed the use of convex optimization, developed the algorithm. Tabakhi and Le proposed applying the model on an open-source dataset in the domain of smart grids. Le and Tabakhi collaboratively conducted the experiments under random graphs and smart grids benchmarks. Tabakhi prepared the dataset to be used in experimental evaluations. Long, Yeoh, Le, Son, and Tabakhi collaboratively wrote the paper.

> M. Tabakhi, Yeoh, W., and Fioretto, F., The Smart Appliance Scheduling Problem: A Bayesian Optimization Approach. In Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA), 2020.

The idea behind this work was proposed by Yeoh and Fioretto primarily and later on revised by Tabakhi. Tabakhi modeled the Smart Appliance Scheduling Problem using CSPs and proposed several acquisition functions for Bayesian optimization. Tabakhi implemented the idea, developed tailored acquisition functions, and conducted experimental evaluations. The paper is mainly written by Tabakhi and revised by Yeoh and Fioretto.

# Chapter 2

# Background

This chapter provides an overview of several constraint-based models, their definitions, and the relation among these models. We provide an overview of different existing resolution algorithms to solve such constraint-based models. As some of these existing algorithms are the starting platforms for the work in this dissertation, we provide detailed descriptions along with examples. Figure 2.1 illustrates the relations among the constraint-based models. In addition, we explain the meaning of preferences and elicitation in the context of this dissertation and clarify all the assumptions of our contributions.

## 2.1 Weighted Constraint Satisfaction Problems

**Constraint Satisfaction Problems (CSPs):** Centralized CSPs [44, 96] are decision problems that assign values to variables under a set of pre-specified constraints. The set of constraints defines how variables should be related to each other. CSPs have been well studied in artificial intelligence [21]. CSPs can formulate and model a number of combinatorial

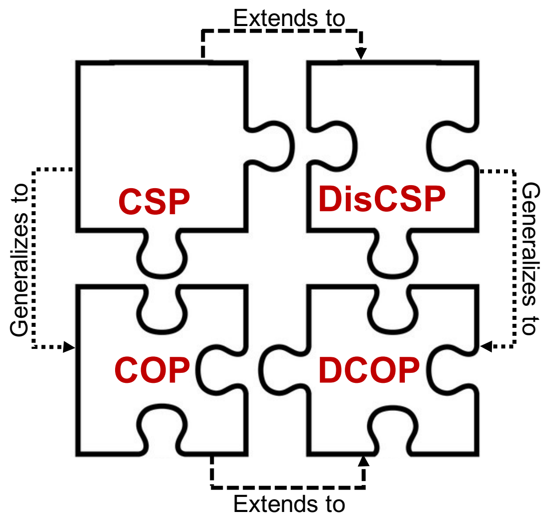


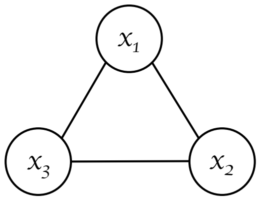
Figure 2.1: Relation Among the Constraint-Based Models

problems, including scheduling, vehicle routing, and resource allocation problems.

A CSP is formally defined as a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F} \rangle$, where:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$, is a set of finite variables.
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite domains for each variable $x_i \in \mathcal{X}$, with $D_i$ being the set of possible values for $x_i$.
- $\mathcal{F} = \{f_1, \ldots, f_m\}$ is a set of $m$ constraints, restricting the values that the variables can take simultaneously. A constraint $f_j$, defined over $k$ variables $x_{j1}, \ldots, x_{jk}$, is a relation $f_j \subseteq \times_{i=1}^{k} D_{j_i}$, where $\{j_1, \ldots, j_k\} \subseteq \{1, \ldots, n\}$. The set of variables $\mathbf{x}^{f_j} = \{x_{j1}, \ldots, x_{jk}\}$ is referred to as the scope of $f_j$. If $k = 1$, then $f_j$ is called a *unary* constraint, and if $k = 2$, then $f_j$ is called a *binary* constraint, otherwise $f_j$ is called $k$-ary constraint. The number of variables involved in $f_j$ is called the constraint's arity.

A *partial assignment* is a value assignment for a proper subset of variables that must be consistent with their respective domains. A value assignment is defined as a partial function $\sigma : \mathcal{X} \to \bigcup_{i=1}^{n} D_i$ such that, for each $x_j \in \mathcal{X}$, if $\sigma(x_j)$ is defined, then $\sigma(x_j) \in D_j$. An



(a) Constraint Graph

| $x_1$ | $x_2$ | $f_1$ |
|---|---|---|
| 0 | 0 | 15 |
| 0 | 1 | 11 |
| 1 | 0 | 21 |
| 1 | 1 | 30 |

| $x_1$ | $x_3$ | $f_2$ |
|---|---|---|
| 0 | 0 | 4 |
| 0 | 1 | 45 |
| 1 | 0 | 8 |
| 1 | 1 | 10 |

| $x_2$ | $x_3$ | $f_3$ |
|---|---|---|
| 0 | 0 | 50 |
| 0 | 1 | 5 |
| 1 | 0 | 7 |
| 1 | 1 | 6 |

(b) Constraint Cost Tables

Figure 2.2: Example of a Weighted CSP

assignment is *complete* if it assigns a value to all variables in $\mathcal{X}$. In a CSP, the objective is to find a complete solution $\sigma$ such that, for each $f_j \in \mathcal{F}$, $\sigma_{\mathbf{x}^{f_j}}$ satisfies all the constraints. Such a complete assignment is called a solution of the CSP.

As mentioned above, a solution of a CSP must satisfy all of its constraints. However, it is desirable to consider complete assignments whose constraints can be violated in many real-world scenarios, according to a violation/satisfaction degree. The *Weighted Constraint Satisfaction Problem* (WCSP) [59, 102] was introduced to capture this property. WCSPs are problems whose constraints represent preferences that specify the extent of violation (or satisfaction) of the corresponding constraints.

A WCSP is defined as a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F} \rangle$, where:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a finite set of variables.
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite domains for the variables in $\mathcal{X}$, with $D_i$ being the set of possible values for the variable $x_i$.
- $\mathcal{F} = \{f_1, \ldots, f_m\}$ is a set of *weighted constraints* (or *cost tables*). Each weighted constraint is a function $f_i : \bigtimes_{x_j \in \mathbf{x}^{f_i}} D_j \to \mathbb{R}_0^+ \cup \{\perp\}$, where $\mathbf{x}^{f_i} \subseteq \mathcal{X}$ is the set of variables relevant to $f_i$, referred to as the *scope* of $f_i$, and $\perp$ is a special element denoting that a given combination of value assignments is not allowed.

A *solution* $\sigma$ is a value assignment to a set of variables $\mathbf{x}_\sigma \subseteq \mathcal{X}$ that is consistent with the variables' domains. The cost $\mathcal{F}_\mathcal{P}(\mathbf{x}_\sigma) = \sum_{f \in \mathcal{F}, \mathbf{x}^f \subseteq \mathbf{x}_\sigma} f(\mathbf{x}_\sigma)$ is the sum of the costs of all the applicable cost functions in $\mathbf{x}_\sigma$. A solution $\sigma$ is said to be *complete* if $\mathbf{x}_\sigma = \mathcal{X}$ and is a partial solution otherwise. The goal is to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmin}_\mathbf{x} \mathcal{F}_\mathcal{P}(\mathbf{x})$. Figure 2.2 illustrates a WCSP example. Typically, the problem is visualized as a constraint graph shown in Figure 2.2(a) and the constraints are specified in constraint tables shown in Figure 2.2(b).

As shown in Figure 2.1, a WCSP is a generalization of a CSP which can be seen as a WCSP whose constraints utilize the costs 0 and $\perp$. A WCSP is also known as a *Constraint Optimization Problem* (COP). These terms are often used interchangeably in the literature.

### 2.1.1 Algorithms

As the field of classical CSPs has become mature, researchers have proposed many different resolution algorithms to solve CSPs and WCSPs. In order to find solutions for a CSP and its variant WCSPs, we generally need to conduct some form of search. There are multiple categories of algorithms that can be mainly classified into two groups: *Complete* and *incomplete* algorithms. The complete algorithms can guarantee the optimal solution or determine unsolvability, while incomplete algorithms are often faster as they trade optimality for shorter execution time and only provide sub-optimal solutions. As representatives of the class of complete algorithms, we can name the Bucket Elimination algorithm [20] and the Depth-First Branch-and-Bound algorithm [71]. As representatives of the class of incomplete algorithms, we can name the Weighted Arc Consistency algorithm [60] and a local search algorithm called the Tabu algorithm [36, 77]. To provide a brief overview, we describe one algorithm from each category that are used to solve WCSPs/COPs.

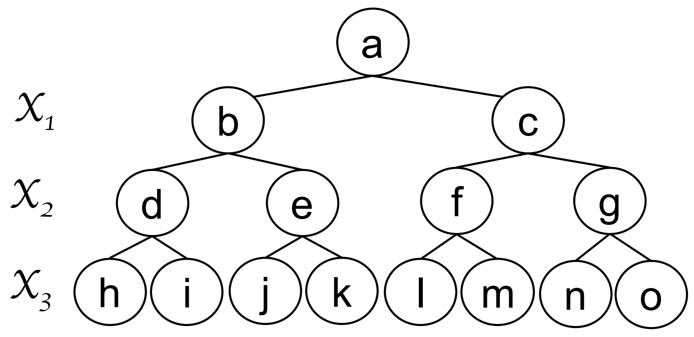


Figure 2.3: The Search Tree with Labeled Nodes

**Depth-First Branch-and-Bound Algorithm:**

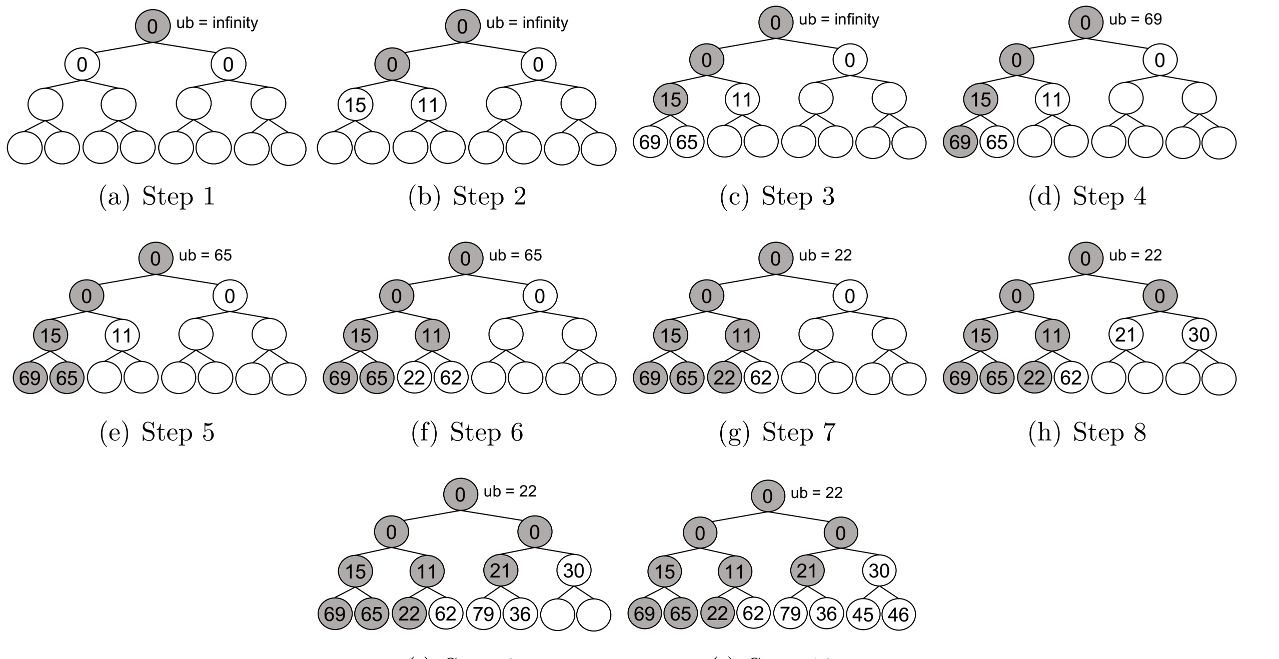
In complete search algorithms, the state space of the problem is explored as a tree which is called the *search tree*. The search tree consists of a set of nodes and edges connecting the nodes. Each node represents a subproblem. The root node represents the whole problem. Node successors are produced by selecting an unassigned variable and generating as many successors as the number of values in the domain of the corresponding variable. Each edge connecting a node to its successor is labeled with one of the values in the domain, which is now assigned to the selected variable. Each path from the root node to any node in the search tree represents an assignment of values to variables. Figure 2.3 is the state space of the WCSP in Figure 2.2(a) and represents the search tree, where levels 1, 2, and 3 correspond to variables $x_1$, $x_2$, and $x_3$, respectively, and all nodes are labeled. Left branches correspond to the variable being assigned the value 0, and right branches correspond to the variable being assigned the value 1.

*Depth-First Branch-and-Bound Algorithm* (DFBnB) performs a depth-first traversal of the search tree. It keeps two bounds – lower bound ($lb$) and upper bound ($ub$) – during the search. At any node $n$, $lb(n)$ is an underestimate of the total constraint costs (violation degree) of any complete assignment. And $ub$ is the cost of the best complete assignment so far. When $lb(n) \geq ub$, the subtree rooted at node $n$ can be pruned because it contains no solution with a cost less than $ub$. If it finds a complete assignment with a cost less than $ub$, it updates the $ub$ with the new cost. After exhausting the search tree, the algorithm returns the solution and the most updated $ub$. The time complexity of the DFBnB algorithm is $O(d^n)$, where $n$ is the number of variables and $d$ is the maximum number of values in the domain. The space complexity of the algorithm is $O(nd)$. The efficiency of DFBnB depends largely on its pruning capacity, which relies on the quality of its bounds: the higher the $lb$ and the lower the $ub$, the better DFBnB performs since it does more pruning, exploring a smaller part of the search tree. Many efforts have been made to improve (that is, to increase) the lower bound [44].

Figure 2.4 shows a simplified execution trace of DFBnB, where the search starts by assigning the value 0 to the variables first. The shaded nodes are the expanded nodes meaning that the corresponding variable is already assigned a value. The $ub$ is updated after finding a complete assignment at the end of each leaf node. The DFBnB algorithm finds the optimal solution at step 7 and prunes subtrees rooted at $k$, $l$, $m$, and $g$. The optimal solution is when $x_1 = 0$, $x_2 = 1$, and $x_3 = 0$, with the minimum constraint cost of 22. In this example, we imposed a value ordering for the variables, such that the value 0 is assigned first. However, this value ordering might not be the most efficient approach. Thus, choosing a different value ordering as well as a variable ordering will change the performance of the algorithm.

**Tabu Local Search Algorithm:**

The key idea behind a local search is to start from an initial search position where all variables in a WCSP are assigned a value randomly. This variable assignment might be sub-optimal and to improve this assignment iteratively the algorithm makes use of minor modification. In each step, the search process moves to a position selected from the local neighborhood based on a heuristic evaluation function. This iterative process continues until a termination criterion is satisfied. This termination criterion is usually the fact that a solution is found or



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6 (g) Step 7 (h) Step 8

(i) Step 9 (j) Step 10

Figure 2.4: Simplified Execution Trace of DFBnB

a predefined number of iterations/steps is reached. In almost all local search algorithms some form of randomization is used to avoid any stagnation of the search process. *Tabu* is one of the variants of the Breakout algorithm [36, 77], where the search history is used to make the remaining search process more efficient. Typically, the tabu search is combined with random moves to improve the search and avoid stagnation. Hence, the Tabu search algorithm takes in two parameters: $p$, that is the probability of the random moves and $t$, that is the tabu tenure parameter. When there is a need to choose a variable to re-assign, the variable will be chosen randomly with probability $p$ or, with probability $(1 - p)$. We perform the above local search procedure. When all new assignments in the neighborhood are worse than or equal to the current one, this implies no improving move is possible. In such situation, the chosen variable is marked as tabu and will not be used for $t$ steps.

We take the WCSP example in Figure 2.2 and run the tabu local search for 3 steps. As the initial step, we start off the algorithm with a random assignment of values to all variables which is when $x_1 = 0$, $x_2 = 0$, and $x_3 = 1$ with the cost of 65 which is not the optimal solution. In the first iteration, the algorithm chooses the variable with the smallest partial cost among all other neighboring variables and change its current value to the new value in its domain that makes the solution cost to be smaller. In this example, the algorithm chooses $x_2$, and changes its current value 0 to value 1. With the new value assignment, the new improved solution is $x_1 = 0$, $x_2 = 1$, and $x_3 = 1$ with the cost of 62. In the next iteration, the algorithm continues the above procedure and chooses $x_3$ and changes its current value 1 to the new value 0. Thus, the new improved solution is $x_1 = 0$, $x_2 = 1$, and $x_3 = 0$ with the cost of 22. The algorithm continues and move to the next variable to change its value, however, there is no improving solution, and in the next iteration, the algorithm terminates as its reaches 3 iterations. The final solution is $x_1 = 0$, $x_2 = 1$, and $x_3 = 0$ with the cost of 22. For this simple algorithm, we set $p = 1$, $t = 0$, and the number of steps to 3 iterations.

## 2.2 Distributed Constraint Optimization Problems

When elements of a CSP are distributed among a set of autonomous agents, the resulting model takes the form of a *Distributed Constraint Satisfaction Problem* (DisCSP) [140, 141]. As shown in Figure 2.1, DisCSPs can be seen as an extension of CSPs where agents communicate with each other to take on values for their variables that satisfy all constraints. A DisCSP is also defined as a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F} \rangle$, where $\mathcal{X}$, $\mathcal{D}$, and $\mathcal{F}$ are exactly the same as in a CSP; $\mathcal{A} = \{a_1, \dots, a_p\}$ is the set of finite agents; and finally $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ is a *mapping function* that associates each variable to one or more than agents. The goal in a DisCSP is to find a complete assignment that satisfies all the constraints of the problem.

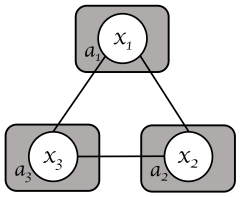
Similar to the generalization of CSPs to WCSPs (COPs), the *Distributed Constraint Optimization Problem* (DCOP) model [76, 86, 139] emerges as a generalization of DisCSPs, where constraints specify a degree of preference over their violation, rather than a Boolean satisfaction metric. DCOPs can be seen as an extension of COPs, where agents control variables and coordinate with each other to take on values for their variables so as to optimize a global objective function. The DCOP framework is formally defined as a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \alpha \rangle$, where:

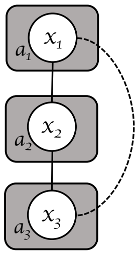
- $\mathcal{A} = \{a_1, \ldots, a_p\}$, is a set of agents.
- $\mathcal{X} = \{x_1, \ldots, x_n\}$, is a finite set of variables, with $n \geq p$.
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite domains for each variable $x_i \in \mathcal{X}$, with $D_i$ being the set of possible values for $x_i$.
- $\mathcal{F} = \{f_1, \ldots, f_m\}$ is a set of $m$ constraints, with $f_j$, being defined over a set of variables $f_j = \prod_x \in \mathbf{x}^{f_j} D_x \rightarrow \mathbb{R} \cup \{\perp\}$, and $\perp$ is a special element denoting that a given combination of value assignments is not allowed. Similar to WCSP, $\mathbf{x}^{f_j} \subseteq \mathcal{X}$ is the set of variables relevant to $f_j$, referred to as the scope of $f_j$. The arity of a constraint function is the number of variables in its scope.
- $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ is a mapping function associates each variable to one agent.

A solution $\sigma$ is a value assignment for a set $\mathbf{x}_\sigma \subseteq \mathcal{X}$ of variables that is consistent with their respective domains. The cost $\mathcal{F}(\mathbf{x}_\sigma) = \sum_{f \in \mathcal{F}, \mathbf{x}^f \subseteq \mathbf{x}_\sigma} f(\mathbf{x}_\sigma)$ is the sum of the costs across all the applicable constraints in $\mathbf{x}_\sigma$ . A solution $\sigma$ is a complete solution if $\mathbf{x}_\sigma = \mathcal{X}$ and is a partial solution otherwise. The goal is to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \mathcal{F}(\mathbf{x})$.

**Representation of DCOPs:**

Typically in the DCOP framework, the problem representation plays a key role from an agent coordination perspective as well as an algorithmic perspective [29]. We start with describing some widely adopted assumptions in this framework regarding agent knowledge and coordination, typically in a DCOP, it is assumed that:





| $x_1$ | $x_2$ | $f_1$ |
|---|---|---|
| 0 | 0 | 15 |
| 0 | 1 | 11 |
| 1 | 0 | 21 |
| 1 | 1 | 30 |

| $x_1$ | $x_3$ | $f_2$ |
|---|---|---|
| 0 | 0 | 4 |
| 0 | 1 | 45 |
| 1 | 0 | 8 |
| 1 | 1 | 10 |

| $x_2$ | $x_3$ | $f_3$ |
|---|---|---|
| 0 | 0 | 50 |
| 0 | 1 | 5 |
| 1 | 0 | 7 |
| 1 | 1 | 6 |

(a) Constraint Graph (b) Pseudo-Tree (b) Constraint Cost Tables

Figure 2.5: Example of a DCOP

- Each agent/variable knows constraint costs values involving at least one of its local variables a priori.
- A variable and its domain are solely known to the agent controlling it and its neighboring agents.
- Each agent/variable solely communicates with its own neighboring agents.
- It is assumed that communication [94, 118] between all pairs of agents are identical [108] .

This assumption of a priori knowledge on constraint costs – each agent/variable knows constraint costs values involving at least one of its local variables a priori – are common in centralized constraint-based models such as WCSPs/COPs.

As mentioned in previous sections, one of the common ways to represents a DCOP as well as other centralized and decentralized constraint-based models explained earlier in this chapter is to use a graphical representation, called *constraint graph*, to visualize a DCOP (or any other constraint-based model such as CSP, WCSP, and DisCSP). In such a representation, nodes in the graph correspond to variables in the DCOP and edges connect pairs of variables appearing in the same constraint.

A *constraint graph* shown in Figure 2.5(a) visualizes a DCOP, where nodes in the graph correspond to variables in the DCOP and edges connect pairs of variables appearing in the same constraint. The constraint tables are shown in Figure 2.5(c). In the rest of this

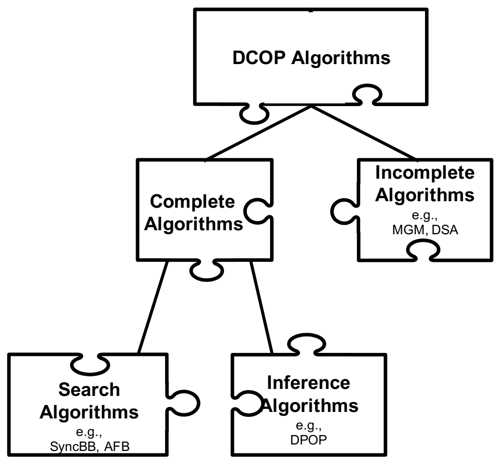


Figure 2.6: Taxonomy of DCOP Algorithms

dissertation, for the sake of simplicity of the examples, we assume that each agent controls exactly one variable and thus use the terms “agent” and “variable” interchangeably.

Several DCOP algorithms require a partial ordering among the agents. When such an order is derived from a depth-first search (DFS) exploration, the resulting structure is known as a (DFS) pseudo-tree. A *pseudo-tree* arrangement has the same nodes as the constraint graph and includes all the edges of the constraint graph. The edges in the pseudo-tree are divided into *tree edges*, which connect parent-child nodes and all together form a rooted tree, and *backedges*, which connect a node with its *pseudo-parents* and *pseudo-children.* Finally, two variables that are constrained together in the constraint graph must appear in the same branch of the pseudo-tree. When the pseudo-tree has only a single branch, it is called a *pseudo-chain.* One can also view a pseudo-chain as a complete ordering of all the variables in a DCOP, which is used by a number of DCOP algorithms. Figure 2.5(b) shows the pseudo-tree (pseudo-chain) of the DCOP example.

### 2.2.1 Algorithms

The field of DCOP has matured significantly over the past decade since its inception [76]. DCOP researchers have proposed a wide variety of solution approaches which can be classified into two categories namely complete and incomplete resolution algorithms. Complete DCOP algorithms find cost-minimal solutions while incomplete DCOP algorithms are often faster and more scalable to larger problems but they usually find suboptimal solutions. As representatives of the class of complete DCOP algorithms we can name synchronous branch-and-bound (SyncBB) [48] and asynchronous forward bounding (AFB) [38]. As representatives of the class of incomplete DCOP algorithms we can name Max-Sum [27] and Maximum Gain Message (MGM) [69]. Figure 2.6 illustrates the relationship of DCOP search algorithms to other algorithms and their categories. To provide a brief overview, we describe one algorithm from each category that are used to solve DCOPs.

**Synchronous Branch-and-Bound Algorithm:**

The Synchronous Branch-and-Bound (SyncBB) [48] is a complete, synchronous, search-based algorithm that can be considered as a distributed version of a depth-first branch-and-bound algorithm. It uses a complete ordering of the agents to extend a *Current Partial Assignment* (CPA) via a synchronous communication process. The CPA holds the assignments of all the variables controlled by all the visited agents, and, in addition, functions as a mechanism to propagate bound information. The algorithm prunes those parts of the search space whose solution quality is sub-optimal by exploiting the bounds that are updated at each step of the algorithm. In other words, an agent backtracks when the cost of its CPA is no smaller than the cost of the best complete solution found so far. The algorithm terminates when the root backtracks (i.e., the algorithm has explored or pruned the entire search space).

SyncBB agents perform $O(d^n)$ number of operations since the lowest priority agent needs to enumerate through all possible value combinations for all variables, where $d$ is the size of the largest domain, and $n$ is the number of agents/variables. The memory requirement per SyncBB agent is $O(n)$ since the lowest priority agent stores the value assignment of all problem variables. The number of messages that are exchanged between agents is $O(d^n)$. The communication model of SyncBB depends on the given agents' complete ordering. Therefore, agents may communicate with non-neighboring agents.

We take the DCOP example in Figure 2.5 and show a simplified execution trace of SyncBB. The operation of SyncBB is visualized with search trees. Let us use Figure 2.3 and label each node of the search tree with an identifier so that we can refer to them easily below. The root agent $a_1$ first expands node $a$ followed by node $b$. This is done when it assigns its variable $x_1$ the value 0. It then sends a CPA with this value assignment and the cost of this CPA $(= 0)$ to its child $a_2$. Upon receipt of the message, agent $a_2$ needs to decide whether to expand nodes $d$ or $e$, which correspond to assigning its variable $x_2$ the values 0 or 1, respectively. Agent $a_2$ should expand the node with the smaller cost and $f_1(x_1 = 0, x_2 = 1) = 11$ is the smaller cost after expanding node $e$. It sends a CPA with this value assignment and the cost of this CPA$(= 11)$ to its child $a_3$. Upon receipt of this message, agent $a_3$ needs to decide whether to expand nodes $j$ or $k$. Using the same rationale as above, agent $a_3$ expands node $j$ and $f_1(x_1 = 0, x_2 = 1) = 11$ and $f_2(x_1 = 0, x_3 = 0) = 4$ and $f_3(x_2 = 1, x_3 = 0) = 7$ with cost of 22, since the cost of the solution is smaller than the upper bound, it updates the upper bound to 22. Then, it evaluates node $k$ whether it should be expanded or pruned. Since the cost of node $k$ is no smaller than the upper bound, agent $a_3$ prunes this node and backtracks to its parent $a_2$ by sending a BACKTRACK message that contains its best complete solution, assignments and CPA. Upon receipt of the BACKTRACK message, agent $a_2$ updates its CPA based on the cost received in the message. Agent $a_2$ now can expand node $d$ as the cost is smaller than the solution found so far, it sends its CPA and its updated upper bound to its child $a_3$, and prunes the branches as the cost of the solution is no smaller than the cost of the solution found so far. Continuing the same rationale above, subtrees rooted at nodes $l$, $m$, and $g$ are also pruned from the search space as their costs are larger than the upper bound. Finally, the algorithm terminates after exhausting the search space and pruning branches. The optimal solution is $x_1 = 0$, $x_2 = 1$, and $x_3 = 0$ with the minimum cost 22.

**Maximum Gain Message Algorithm:**

The Maximum Gain Message (MGM) algorithm [69] is an incomplete, synchronous, search-based algorithm that performs a distributed local search. Each agent starts by assigning a random value to each of its variables. Then, it sends this information to all its neighbors. Upon receiving the values of its neighbors, it calculates the maximum gain (i.e., the maximum decrease in cost) if it changes its value and sends this information to all its neighbors. Upon receiving the gains of its neighbors, the agent changes its value if its gain is the largest among those of its neighbors. This process repeats until a termination condition is met. MGM provides no quality guarantees on the returned solution.

MGM agents perform $O(ld)$ number of operations in each iteration, as each agent needs to compute the cost for each of its values by taking into account the values of all its neighbors, where $l$ is the largest number of neighboring agents and $d$ is the size of the largest domain. MGM is anytime since agents only change their values when they have a non-negative gain. The memory requirement per MGM agent is $O(l)$. Each agent needs to store the values of all its neighboring agents. In terms of communication requirements, each MGM agent sends $O(l)$ messages, one to each of its neighboring agents. Thus, the total number of messages sent across all agents is $O(\ell nl)$, where $\ell$ is the number of iterations performed by the algorithm. Each message is of constant size $O(1)$ as it contains either the agent's current value or the agent's current gain. Finally, the agents communicate exclusively with their neighboring agents.

**Anytime Local Search Algorithms:**

An algorithm is said anytime if it can return a valid solution even if the DCOP agents are interrupted at any time before the algorithm terminates. Anytime algorithms are expected to seek for solutions of increasing quality as they keep running. *Local search* DCOP algorithms (e.g., MGM, DSA) [68, 143] are synchronous iterative processes, where, in each step of the algorithm, each agent sends its value assignment to all its neighbors in the constraint graph and waits to receive the value assignments of all its neighbors before deciding whether to change its value. In local search algorithms, agents are only aware of the cost of their own assignments and their neighbors' assignments. Therefore, no agent knows when a globally good solution is found. The *Anytime Local Search* (ALS) framework [146] enhances the local search algorithms by allowing them to detect when a globally better solution is found and return that solution upon termination (i.e., the anytime property). It uses a *Breadth-First Search spanning tree* (BFS-tree) of the constraint graph to aggregate costs up the tree to the root agent such that it is able to detect when a better solution is found. When such a solution is found, the root agent propagates the step number in which that solution is found down to its descendants. Therefore, upon termination, all the agents have a consistent view on when the best solution is found and take on their corresponding values.

Taking our example DCOP in Figure 2.5, we use the anytime property in MGM algorithm to provide a simple execution trace of our MGM local search algorithm. The BFS-tree is similar to its constraint graph shown in Figure 2.5(a), when $x_1$ is the root, and the height of the tree is 1. The height and distance parameters in $x_2$ and $x_3$ are initialized to $h = 0$ and $d = 1$, respectively. Let's assume that all agents choose value 1 for their variables and we only run the algorithm for one step. In the initialization step of the algorithm, all agents have already selected a value for their variables randomly. In this step, agent $a_3$ assigns its variable $x_3$ the value 1 and receives the values of its neighbors ($x_2 = 1$) and parent ($x_1 = 1$) and calculates the cost of this partial assignment $f_2(x_1 = 1, x_3 = 1) + f_3(x_2 = 1, x_3 = 1) = 16$

Agent $a_3$ also calculates the value of improvement (loss) by checking if variable $x_3$ changes its value to 0 assuming all neighbors keep their current values. The total cost of this change is $f_2(x_1 = 1, x_3 = 0) + f_3(x_2 = 1, x_3 = 0) = 15$. The gain at this step for agent $a_3$ is $16 - 15 = 1$. Similarly, in agent $a_2$, the total cost of the partial assignment is $f_1(x_1 = 1, x_2 = 1) + f_3(x_2 = 1, x_3 = 1) = 36$. The total estimated cost for agent $a_2$ changing to a new value is $f_1(x_1 = 1, x_2 = 0) + f_3(x_2 = 0, x_3 = 1) = 26$ and the gain is $36 - 26 = 10$. The root agent $a_1$ calculates the cost of its assignment based on the information received from its children (at the initial step, the costs received from children are still 0). The total cost of the assignment $f_1(x_1 = 1, x_2 = 1) + f_2(x_1 = 1, x_3 = 1) = 40$. The total estimated cost for agent $a_1$ changing to a new value is $f_1(x_1 = 0, x_2 = 1) + f_2(x_1 = 0, x_3 = 1) = 2.5$, and the loss $40 - 56 = -16$.

Each agent sends messages to its neighboring agents, the value message agents send to their parent and children includes the cost calculation. Thus, in the next step of the algorithm the root agent has received the cost calculation from its children and can calculate the cost of a complete assignment $f_1(x_1 = 1, x_2 = 1) + f_2(x_1 = 1, x_3 = 1) + f_3(x_2 = 1, x_3 = 1) = 46$. Since the calculated cost is smaller than initial value that we initially set to $\infty$, the root agent saves the information for this assignment as the best one found so far. In this step, all agents have received the best index from the root agent, where the best cost found so far is 46. If we increase the number of iterations, the algorithm may find a better solution, however, finding the optimal solution even for this simple example is not guaranteed.

## 2.3 Preferences and Preference Elicitation

As mentioned earlier, constraint-based models are promising frameworks for formalizing many problems, such as scheduling, planning, and resource allocations. Such problems are often represented by a set of variables, their domains, and constraints. A solution of the problem is an assignment to all the variables in their domains such that all constraints are satisfied. Preferences or cost functions have been used to extend the formalism of constraint satisfaction models and allow for the modeling of constraint optimization rather than satisfaction problems. In such frameworks, the data (variables, domains, and constraints) are presumed to be known before the solving process starts. With the increasing use of agent-based applications, many of such applications demand for the formalization and handling of data that is only partially known or specified, when the solving process works, and that can be added later, for instance through elicitation [130, 131] or estimation from historical data. In many multi-agent settings, it occurs that multiple agents may hide their data due to privacy reasons and only be released if needed by resolution algorithms to find a solution to the problem.

Gelain *et al.* address this issue by focusing on constraint optimization problems while seeking an optimal solution. They introduced the *Incomplete Weighted Constraint Satisfaction Problems* (IWCSPs) model, where constraints are replaced by soft constraints, in which each assignment to the variables of the constraint has an associated preference coming from a preference set [6]. We assume that variables, domains, and constraint topology are given initially, while the preferences are partially specified and are elicited during the solving process. The proposed model is effective in scenarios where one regards the fact that quantitative preferences, needed in soft constraints, may be difficult and tedious to provide for a user. In other scenarios, where one regards multi-agent settings, that agents agree on the structures of the problem but they may provide their preferences on different parts of the problem at different times. Finally, some preferences can be initially hidden due to several privacy reasons.

IWCSPs allow for some preferences to be unspecified. In this setting, it is assumed that users may know all the preferences but are willing to reveal only some of them at the beginning. Even though some of the preferences can be missing, it could still be feasible to find an optimal solution, and if it is not, some of the missing preferences are elicited from users. They introduced two notions of optimal solution: possibly optimal solutions are assignments to all the variables that are optimal in at least one way in which the currently unspecified references can be revealed. Necessarily optimal solutions are assignments to all the variables that are optimal in all ways in which the currently unspecified preferences can be revealed. The proposed notions are taken from multi-agent preference aggregation [58, 89, 90], where, in the context of voting theory, some preferences are missing, but still, one would like to declare a winner.

There are some lines of work that are closer to work By Gelain *et al.* in terms of addressing similar issues such as *Open CSPs* [26] and *Interactive CSPs* [57], where domains are partially specified. An Open CSP is a possibly unbounded, partially ordered set of constraint satisfaction problems, each containing at least one more domain value than its predecessor. The goal is to solve larger and larger problems until a solution is found. An interactive CSP is a *Dynamic CSP* [67], where variables, domains, and constraints may change over time. The interactive CSPs can be seen as a particular case of both dynamic and open CSPs with the goal of minimizing the run time of the solver. These works are different from IWCSPs, and this dissertation's work as variable values are assumed to be known from the beginning, while some of the preferences might be missing or uncertain. Open CSPs exploit a monotonicity assumption that each agent provides variable values in a strictly non-decreasing order of preference. Even when there are no preferences, each agent gives only variable values that are feasible. Since the agent that provides new values/costs for a variable must know the bounds on the remaining possible costs, this is not desirable in the setting where bound computation is costly or time-consuming.

An example of another approach for elicitation in incomplete CSPs is the one presented in [8], where the user provides a classical CSP and a partially unknown utility function over its solutions. The system then performs elicitation queries to select a specific utility function by a regret-based technique. The elicitation is used to ease the computation of the minimax regret function. More specifically, the elicitation considers bounds on the parameters of the utility function. Moreover, quasi-optimal decisions may be obtained since often they require much less effort than finding optimal ones. This approach is different from the work of this dissertation and the work by Gelain *et al.* since the preferences are explicitly elicited or estimated.

The work in this dissertation employs constraint-based frameworks to model real-world combinatorial optimization problems. In constraint-based models, cost functions express preferences or degree of satisfaction/violation of the constraints. Preferences can be qualitative or quantitative. There are some lines of work on qualitative preferences. For example, *Conditional-Preference Networks* (CP-nets) [7, 97] also lie in this category. CP-nets are a graphical representation model for qualitative preferences and reflect conditional dependencies between sets of preference statements. In contrast, IWCSPs focus more on the notion of *conditional additive independence* [4], which requires that the cost of an outcome be the sum of the "costs" of the different variable values of the outcome.

Cost functions represent the soft constraints in our constraint-based models. We later refer to these constraint costs as preferences and use constraint costs and preferences interchangeably in the remainder of this dissertation. In the context of this dissertation, we only concern quantitative preferences. Such quantitative preferences in the context of this work can be estimated from historical data or elicited from domain experts or human users. Preference elicitation is the process of asking questions about the preferences of users or domain experts.

As mentioned in Chapter 1, one of the limitations of constraint-based models is that all preferences must be specified or know a priori, which is unrealistic to accurately know the preferences for all possible scenarios in an application. To address this limitation, Chapter 4 extends IWCSPs, proposed IWCSPs+EC, which associates penalties to the elicitation of missing preferences to represent how much users are bothered by queries. Chapter 3 and 5 introduce uncertain and incomplete constraint-based models, where constraint costs are uncertain and partially specified, respectively. Such costs/preferences must be elicited from domain experts or estimated from historical data. In these chapters, as we introduce models in a distributed manner and multi-agent scenarios, we assume all agents are truthful and cooperative. Therefore, each agent is acting to help its neighboring agents solve the problem to minimize the cost over all constraints and consequently achieve an optimal solution.

While our contributions assume truthful agents in a cooperative environment, agents might lie about their preferences and try to game the system in the real world. There is much work on designing protocols and mechanisms so that agents will be truthful and not game the system [17, 65, 82]. For example, *Vickrey-Clarke-Groves* (VCG) schemes [16, 42, 127] provide a method for charging agents so that each is motivated to tell the truth about its preferences. Studying the game-theoretic and mechanism design perspective of motivating agents to be truthful is out of the scope of this dissertation.

As people increasingly rely on interactive systems to make decisions on their behalf, building effective interfaces for such systems becomes challenging due to the initial incomplete users' preferences and users' cognitive costs. How to accurately elicit users' preferences has become the main concern of some of such decision making systems [15]. Within the last decades, researchers have studied the preference elicitation problem in different fields of research from various perspectives: starting from the traditional utility function elicitation [9, 132] and CP-nets [55, 97] to preferences clustering and matching [23, 144] as well as collaborative filtering in recommendation systems [51, 85]. In this dissertation, we only address preference elicitation from the traditional utility function elicitation perspective and leave other aspects to the future, as they are out of the scope of this dissertation's work. Later, in Sections 3.6, 4.6, and 5.6, we discuss the research work that is more closely related to our work in detail and clarify the similarities and differences.

# Chapter 3

# Pre-execution Elicitation Approach in Constraint-Based Models

To address the key drawback in constraint-based models, we assume that constraint costs (i.e., preferences) are uncertain, in this chapter. Thus, we extend the constraint-based models under the assumption that preferences are unknown and represented by Normal distributions. Prior to solving such constraint-based models, we propose strategies for eliciting unknown preferences. This chapter introduces the uncertain constraint-based models and preference elicitation strategies to reveal a subset of constraint costs. In what follows, we briefly introduce the intuition behind our model and motivate our approach with a real-world application. Finally, we will discuss our model, elicitation heuristic strategies, and our evaluation analyzes in more detail.

## 3.1 Introduction

The importance of constraint optimization is outlined by the impact of its application in a range of *Weighted Constraint Satisfaction Problems* (WCSPs), also known as *Constraint Optimization Problems* (COPs), such as supply chain management [95] and roster scheduling [1]. When resources are distributed among a set of autonomous agents and communication among the agents are restricted, COPs take the form of *Distributed Constraint Optimization Problems* (DCOPs) [29, 76, 86, 139]. In this context, agents coordinate their value assignments to minimize the overall sum of resulting constraint costs. DCOPs are suitable to model problems that are distributed in nature and where a collection of agents attempts to optimize a global objective within the confines of localized communication. They have been employed to model various distributed optimization problems, such as meeting scheduling [136, 146], sensor networks [27], coalition formation [123], and smart grids [72, 73, 113].

The field of DCOP has matured significantly over the past decade since its inception [76]. DCOP researchers have proposed a wide variety of solution approaches, from complete approaches that use distributed search-based techniques [76, 78, 136] to distributed inference-based techniques [86, 128]. There is also a significant body of work on incomplete methods that can be similarly categorized into local search-based methods [27, 68], inference-based techniques [128], and sampling-based methods [30, 80, 83]. Researchers have also proposed the use of other off-the-shelf solvers such as logic programming solvers [61, 62] and mixed-integer programming solvers [46].

One of the core limitations of all these approaches is that they assume that the constraint costs in a DCOP are known a priori. Unfortunately, in some application domains, these costs are only known after they are queried or elicited from experts or users in the domain. One such application is the *Smart Home Device Scheduling* (SHDS) problem [31]. In this problem, agents have to coordinate with each other to schedule smart devices (e.g., smart thermostats, smart lightbulbs, smart washers, etc.) distributed across a network of smart homes. The goal is to schedule these devices to optimize the preferences of occupants in those homes subject to a larger constraint that the peak energy demand in the network does not exceed an energy utility defined limit. By introducing several smart devices in the commercial market, they are becoming ubiquitous in today's very interconnected environment, consistent with the Internet-of-Things paradigm [74]. Therefore, we suspect that this SHDS problem will become more important in the future.

DCOPs are a natural framework to represent this problem as each home can be represented as an agent, and occupants' preferences can be represented as constraints. Furthermore, due to privacy reasons, it is preferred that the preferences of each occupant are not revealed to other occupants. The DCOP formulation allows the preservation of such privacy since agents are only aware of constraints that they are involved in. We further describe this motivating application and its mapping to DCOPs in more detail in Section 5.2.

A priori knowledge on the constraint costs is infeasible in our motivating SHDS application. A key challenge is thus in the elicitation of users' preferences to populate the constraint cost tables. Due to the infeasibility of eliciting preferences to populate *all* preferences, in this chapter, we introduce the *uncertain constraint-based models* namely uncertain COPs and uncertain DCOPs. Uncertain constraint-based models study *how to select a subset of $k$ cost tables to elicit from each agent* intending to choose those constraints that have a significant impact on the overall solution quality. We propose several methods to select this subset of cost tables to elicit, based on the notion of *partial orderings.* We extend the SHDS problem modeled with the uncertain DCOP to allow for the encoding and elicitation of soft preferences. Moreover, we evaluate our methods on this extended SHDS problem and on random graphs to show generality. Our results illustrate the effectiveness of our approach in contrast to a baseline evaluator that randomly selects cost tables to elicit. While the description of our solution focuses on DCOPs, our framework is also suitable to solve WCSPs/COPs.

## 3.2 Motivation

To motive our work and the need for introducing uncertain constraint-based models, let us first describe our motivating application domain, the *Smart Home Device Scheduling* (SHDS) problem [31], in this section. An SHDS problem is composed of a neighborhood $\mathcal{H}$ of smart homes $h_i \in \mathcal{H}$ that are able to communicate with one another and whose energy demands are served by an energy provider. The energy prices are set according to a real-time pricing schema specified at regular intervals $t$ within a finite time horizon $H$. We use $\mathbf{T} = \{1, \ldots, H\}$ to denote the set of time intervals and $\theta : \mathbf{T} \rightarrow \mathbb{R}^+$ to represent the price function associated with the pricing schema adopted, which expresses the cost per kWh of energy consumed by a consumer.

Within each smart home $h_i$, there is a set of (smart) electric devices $\mathcal{Z}_i$ networked together and controlled by a home automation system. We assume all the devices are uninterruptible (i.e., they cannot be stopped once they are started) and use $s_{z_j}$ and $\delta_{z_j}$ to denote, respectively, the start time and duration (expressed in multiples of time intervals) of device $z_j \in \mathcal{Z}_i$. The energy consumption of each device $z_j$ is $\rho_{z_j}$ kWh for each hour that it is *on*. It will not consume any energy if it is *off*. We use the indicator function $\phi^t_{z_j}$ to indicate the state of the device $z_j$ at time step $t$:

$$\phi^t_{z_j} = \begin{cases} 1 & \text{if} \quad s_{z_j} \leq t \wedge s_{z_j} + \delta_{z_j} \geq t \\ 0 & \text{otherwise} \end{cases}$$

Additionally, the usage of a device $z_j$ is characterized by a *cost*, representing the monetary expense to schedule $z_j$ at a given time. The aggregated cost of the home $h_i$ at time step $t$ is denoted with $C_i^t$ and expressed as:

$$C_i^t = E_i^t \cdot \theta(t) \tag{3.1}$$

where $E_i^t = \sum_{z_j \in \mathcal{Z}_i} \phi_{z_j}^t \cdot \rho_{z_j}$ is the aggregated energy consumed by home $h_i$ at time step $t$.

The SHDS problem seeks a schedule for the devices of each home in the neighborhood in a coordinated fashion to minimize the monetary costs and, at the same time, ensure that user-defined scheduling constraints (called *active scheduling rules* in [31]) are satisfied. The SHDS problem is also subject to the following constraints:

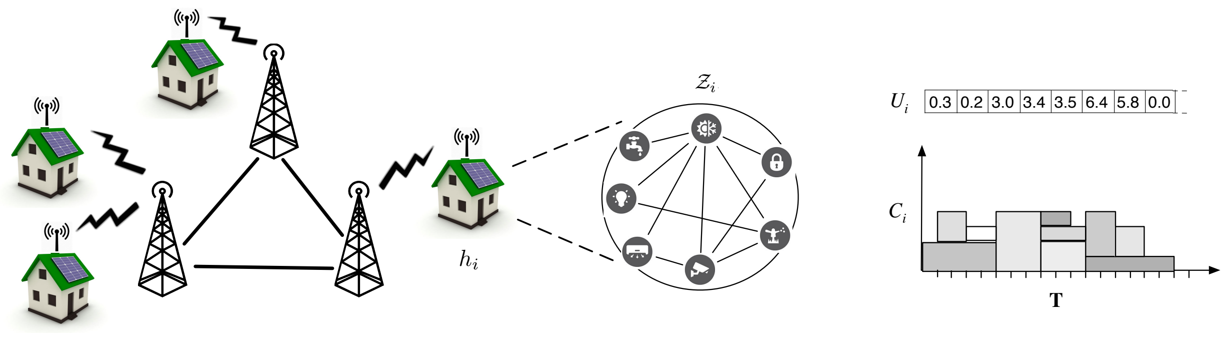
$$1 \leq s_{z_j} \leq T - \delta_{z_j} \qquad \forall h_i \in \mathcal{H}, z_j \in \mathcal{Z}_i \tag{3.2}$$

$$\sum_{t \in \mathbf{T}} \phi_{z_j}^t = \delta_{z_j} \qquad \forall h_i \in \mathcal{H}, z_j \in \mathcal{Z}_i \tag{3.3}$$

$$\sum_{h_i \in \mathcal{H}} E_i^t \leq \ell^t \qquad \forall t \in \mathbf{T} \tag{3.4}$$

where $\ell^t \in \mathbb{R}^+$ is the maximum allowed total energy consumed by all the homes in the neighborhood at time step $t$. This constraint is typically imposed by the energy provider and is adopted to guarantee reliable electricity delivery. Constraint (3.2) expresses the lower and upper bounds for the start time associated to the schedule of each device. Constraint (3.3) ensures the devices are scheduled and executed for exactly their duration time. Constraint (3.4) ensures the total amount of energy consumed by the homes in the neighborhood does not exceed the maximum allowed threshold.

Fioretto *et al.* introduced a mapping of the SHDS problem to a DCOP [31]. At a high level, each home $h_i \in \mathcal{H}$ is mapped to an autonomous agent in the DCOP. For each home, the start



(a) (b) (c)

Figure 3.1: Smart Home Device Scheduling Illustration

times $s_{z_j}$, indicator variables $\phi^t_{z_j}$, and aggregated energy in the home are mapped to DCOP variables, which are controlled by the agent for that home. Constraints (3.2) to (3.4) are enforced by the DCOP constraints. Finally, the objective function of the SHDS is expressed through agents' preferences.

Figure 3.1 (a) provides an illustration of a network of smart home, Figure 3.1 (b) demonstrates a set of smart devices $\mathcal{Z}_i$ controlled within a smart home $h_i$, and Figure 3.1 (c) shows an example of discomfort values (top) and costs (bottom) for a schedule of the devices $\mathcal{Z}_i$ within home $h_i$.

**Encoding and Eliciting Preferences in SHDS:** The above SHDS problem thus far includes exclusively hard constraints and has no soft constraints (i.e., preferences for when devices are scheduled). Therefore, we will describe in this section how to integrate such preferences as soft constraints into SHDS.

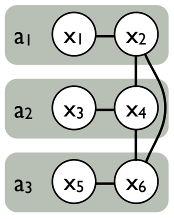
We consider the scenario in which a single home $h_i$ may host multiple users $u \in \mathbf{U}_{h_i}$, with $\mathbf{U}_{h_i}$ denoting the set of users in $h_i$. In modeling agents' preferences, we introduce *discomfort values* $d^t_{z_j,u} \in \mathbb{R}^+_0$ describing the degree of dissatisfaction for a user $u$ to schedule the device $z_j$ at a given time step $t$. Note that the monetary cost is the same for all users while the degree of dissatisfaction is user-dependent. Thus, to avoid conflicting users' decisions over the control of the device, we assume that there is one user who has exclusive access to a device $z \in \mathcal{Z}_i$ at any point in time. For each device $z_j \in \mathcal{Z}_i$ in home $h_i$ and each time step $t$, we assume the likelihood for a user to gain exclusive access on a device $z_j$ is expressed though a probability $Pr^t_{z_j}$ (i.e., $\forall u \in \mathbf{U}_{h_i}, Pr^t_{z_j}(u) \in [0, 1]$ and $\sum_{u \in \mathbf{U}_{h_i}} Pr^t_{z_j}(u) = 1$).

Additionally, we use $\mathbf{d}^t_i = \sum_{z_j \in \mathcal{Z}_i} \phi^t_{z_j} \cdot d^t_{z_j}$ to denote the aggregated discomfort in home $h_i$ at time step $t$, where $d^t_{z_j}$ is the discomfort value of the user who has exclusive access to the device $z_j$ at time step $t$. We can update the SHDS objective in a way that it considers the users' preferences and minimizes the monetary costs. While this is a multi-objective problem, we combine the two objectives into a single one through the use of a weighted sum:

$$\textbf{minimize} \sum_{t \in \mathbf{T}} \sum_{h_i \in \mathcal{H}} \alpha_c \cdot C^t_i + \alpha_u \cdot \mathbf{d}^t_i \tag{3.5}$$

where $\alpha_c$ and $\alpha_u$ are weights in the open interval $(0, 1) \subseteq \mathbb{R}$ such that $\alpha_c + \alpha_u = 1$. While, in general, the real-time pricing schema $\theta$ that defines the cost per kWh of energy consumed and the energy consumption $\rho_{z_j}$ of each device $z_j$ are well-defined concepts and can be easily acquired or modeled, the preferences on the users' discomfort values $d^t_{z_j,u}$ on scheduling a device $z_j$ at time step $t$ are subjective and, thus, more difficult to model explicitly.

We envision two approaches to acquire these preferences: *(1)* eliciting them directly from the users and *(2)* estimating them based on historical preferences or from preferences of similar users. While the former method will be more accurate and reliable, it is cumbersome for the user to enter their preference for every device $z_j$ and every time step $t$ of the problem. Therefore, we assume that a combination of the two approaches will be used, where a subset of preferences will be elicited, and the remaining preferences will be estimated from historical sources or similar users. We believe that this strategy is especially important in application



(a) Constraint Graph

for $i < j$

| $x_i$ | $x_j$ | Costs |
|---|---|---|
| 0 | 0 | 20 |
| 0 | 1 | 8 |
| 1 | 0 | 10 |
| 1 | 1 | 3 |

(b) Cost Table

for $i = 2, j = 4$

| $x_i$ | $x_j$ | Costs |
|---|---|---|
| 0 | 0 | $\mathcal{N}(65{,}8)$ |
| 0 | 1 | $\mathcal{N}(71{,}8)$ |
| 1 | 0 | $\mathcal{N}(41{,}6)$ |
| 1 | 1 | $\mathcal{N}(29{,}5)$ |

for $i = 2, j = 6$

| $x_i$ | $x_j$ | Costs |
|---|---|---|
| 0 | 0 | $\mathcal{N}(10{,}6)$ |
| 0 | 1 | $\mathcal{N}(9{,}5)$ |
| 1 | 0 | $\mathcal{N}(8{,}5)$ |
| 1 | 1 | $\mathcal{N}(19{,}6)$ |

(c) Uncertain Cost Tables

Figure 3.2: Example of a classical DCOP and an Uncertain COP/DCOP Cost Tables

domains such as the SHDS problem, where users' preferences may be learned over time, thus, ensuring a continuous elicitation process of the unknown users preferences.

## 3.3 Uncertain Constraint-Based Models

A key drawback of existing WCSPs (a.k.a COPs) and DCOP approaches is the underlying assumption of a *total* knowledge of the model, which is not the case for a number of applications involving users' preferences, including the SHDS problem. Due to the infeasibility of eliciting *all* users' preferences – and, thus, their associated complete cost tables – in this chapter, we study how to choose a subset of $k$ cost tables to elicit. We first cast this problem as an optimization problem, before describing our proposed techniques.

**Definition 1** *For each agent* $a_i \in \mathcal{A}$, $\mathbf{L}_i = \{x_j \in \mathcal{X} \mid \alpha(x_j) = a_i\}$ *is the set of its* local *variables.* $\mathbf{I}_i = \{x_j \in \mathbf{L}_i \mid \exists x_k \in \mathcal{X} \wedge \exists f_s \in \mathcal{F} : \ \alpha(x_k) \neq a_i \wedge \{x_j, x_k\} \subseteq \mathbf{x}^{f_s}\}$ *is the set of its* interface *variables.*

**Definition 2** *For each agent* $a_i \in \mathcal{A}$*, its* local constraint graph $G_i = (\mathbf{L}_i, \mathcal{E}_{\mathcal{F}_i})$ *is a subgraph of the constraint graph, where* $\mathcal{F}_i = \{f_j \in \mathcal{F} \mid \mathbf{x}^{f_j} \subseteq \mathbf{L}_i\}$.

### 3.3.1 Uncertain COPs

Let $\hat{\mathcal{P}} = \langle \mathcal{X}, \mathcal{D}, \hat{\mathcal{F}} \rangle$ denotes a WCSP/COP whose constraints have partial knowledge on the cost tables, where:

- $\mathcal{X}$ is the set of variables.
- $\mathcal{D}$ is the set of domains.
- $\hat{\mathcal{F}} = \mathcal{F}_r \cup \mathcal{F}_u$ is the set of constraints that are composed of *revealed constraints* $\mathcal{F}_r$, whose cost tables are accurately revealed, and *uncertain constraints* $\mathcal{F}_u$, whose cost tables are unrevealed and must be either estimated from historical sources or elicited.

We refer to this problem as the *uncertain COP*.

### 3.3.2 Uncertain DCOPs

Let $\hat{\mathcal{P}} = \langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \hat{\mathcal{F}}, \alpha \rangle$ denotes a WCSP/COP whose constraints have partial knowledge on the cost tables, where:

- $\mathcal{A}$ is the set of agents.
- $\mathcal{X}$ is the set of variables.
- $\mathcal{D}$ is the set of domains.
- $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ is a surjective function, from variables to agents.
- $\hat{\mathcal{F}} = \mathcal{F}_r \cup \mathcal{F}_u$ is the set of constraints that are composed of *revealed constraints* $\mathcal{F}_r$, whose cost tables are accurately revealed, and *uncertain constraints* $\mathcal{F}_u$, whose cost tables are unrevealed and must be either estimated from historical sources or elicited.

We refer to this problem as the *uncertain DCOP*.

In both uncertain COPs and uncertain DCOPs, a *solution* $\mathbf{x}$ is a value assignment to a set of variables $X_{\mathbf{x}} \subseteq \mathcal{X}$ that is consistent with the variables' domains. The cost $\mathbf{F}_{\mathcal{P}}(\mathbf{x}) = \sum_{f \in \mathcal{F}, \mathbf{x}^f \subseteq X_{\mathbf{x}}} f(\mathbf{x})$ is the sum of the costs of all the applicable cost functions in $\mathbf{x}$. A solution $\mathbf{x}$ is said *complete* if $X_{\mathbf{x}} = \mathcal{X}$. The goal is to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \mathbf{F}_{\mathcal{P}}(\mathbf{x})$. To find a solution, the unrevealed cost tables must be be either estimated from historical sources or elicited from users. In the following sections, we discuss in more detail the elicitation process of the uncertain costs tables.

We assume that the costs of the uncertain constraints are sampled from Normal distributions that can be estimated from historical sources.[1] Further, we assume that the distribution for each cost value is independent from the distribution of all other cost values. Figure 3.2 (c) illustrates an uncertain cost table whose costs are modeled via random variables obeying Normal distributions, and $u_1$ and $u_2$ denote two distinct users that can control the associated device.

**The Preference Elicitation Problem:**

The preference elicitation problem in DCOPs is formalized as follows: Given an oracle DCOP $\mathcal{P}$ and a value $k \in \mathbb{N}$, construct an uncertain DCOP $\hat{\mathcal{P}}$ that reveals only $k$ constraints per agent (i.e., $|\mathcal{F}_r| = k \cdot |\mathcal{X}|$) and minimizes the error:

$$\epsilon_{\hat{\mathcal{P}}} = \mathbb{E}\big[\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}^*) - \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*)\big] \tag{3.6}$$

[1]Other forms of distributions can also be used, but our minimax regret heuristics require that the form of the distributions have the following property: The sum of two distributions has the same form as their individual distributions.

where $\hat{\mathbf{x}}^*$ is the optimal solution for a *realization* of the uncertain DCOP $\hat{\mathcal{P}}$, and $\mathbf{x}^*$ is the optimal complete solution for the oracle DCOP $\mathcal{P}$. A realization of an uncertain DCOP $\hat{\mathcal{P}}$ is a DCOP (with no uncertainty), whose values for the cost tables are sampled from their corresponding Normal distributions. Note that the possible numbers of uncertain DCOPs that can be generated is $\binom{|\mathcal{F}|}{k \cdot |\mathcal{X}|}$. Since solving each DCOP is NP-hard [75], the preference elicitation problem is a particularly challenging one. Thus, we propose a number of heuristic methods to determine the subset of constraints to reveal, and to construct an uncertain problem $\hat{\mathcal{P}}$.

## 3.4 Resolution Algorithms and Heuristics

Let us first introduce a general concept of dominance between cost tables of uncertain constraints. Given two cost tables of uncertain constraints $f_{z_i}, f_{z_j} \in \mathcal{F}_u \subset \hat{\mathcal{F}}$, let $\succeq_\circ$ denote the dominance between the two cost tables according to partial ordering criteria $\circ$. In other words, $f_{z_i} \succeq_\circ f_{z_j}$ means that $f_{z_i}$ *dominates* $f_{z_j}$ according to criteria $\circ$. We now introduce the heuristic methods for different possible ordering criteria $\circ$.

### 3.4.1 Heuristic Strategies

**Minimax Regret:** *Minimax regret* is a well-known strategy that minimizes the maximum regret, and it is particularly suitable in a risk-neutral environment. At a high level, the minimax regret approach seeks to approximate and minimize the impact of the worst-case scenario. The idea of using minimax regret in our domain of interest is derived by the desire of taking into account the possible different outcomes occurring when eliciting the preferences of different users for a single device. Further, we assume that constraints that can be elicited are either unary or binary constraints. We leave to future work the extension to higher arity constraints. We now describe how to compute the regret for a single user $u$, and later how to combine the regrets across multiple users. We use $Pr_{x_i}(d)$ to estimate the likelihood of an assignment $d \in D_i$ to a variable $x_i$:

$$Pr_{x_i,u}(d) = \Pi_{d' \in D_i \setminus \{d\}} Pr(\psi^d_{x_i,u} \leq \psi^{d'}_{x_i,u}) \tag{3.7}$$

where $\psi^d_{x_i,u}$ is the random variable representing the total cost incurred by $x_i$ if it is assigned value $d$ from its domain under user $u$. Then, the value:

$$d^*_{x_i,u} = \operatorname*{argmax}_d Pr_{x_i,u}(d) \tag{3.8}$$

with the largest probability is the one that is most likely to be assigned to $x_i$.

The probability $Pr(\psi^d_{x_i,u} \leq \psi^{d'}_{x_i,u})$ can be computed using:

$$Pr(\psi^d_{x_i,u} \leq \psi^{d'}_{x_i,u}) = \int_{c'=0}^{\infty} \int_{c=0}^{c'} Pr^d_{x_i,u}(c) Pr^{d'}_{x_i,u}(c')\, dc\, dc' \tag{3.9}$$

where $Pr^d_{x_i,u}$ is the *probability distribution function* (PDF) for random variable $\psi^d_{x_i,u}$. Unfortunately, the PDF $Pr^d_{x_i,u}$ is not explicitly defined in the uncertain DCOP. There are two challenges that one needs to address to obtain or estimate this PDF:

*i.* First, the total cost incurred by an agent is the summation of the costs over all constraints of that agent. Thus, the PDF for the total cost needs to be obtained by summing over the PDFs of all the individual constraint costs. Since we assume that these PDFs are all Normally distributed, one can efficiently construct the summed PDF, which is also a Normal distribution. Specifically, if $\mathcal{N}(\mu_i, \sigma_i^2)$ is the PDF for random variables $c_i$ ($i = 1, 2$), then $\mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$ is the PDF for $c_1 + c_2$.

*ii.* Second, the cost associated to a variable for each constraint is not only dependent on its value but also on the value of the other variables constrained with it. In turn, the value of those variables depend on the variables that they are constrained with, and so on. As a result, estimating the true PDF requires the estimation of all the constraint costs in the entire DCOP. To simplify the computation process and introduce an independence between the costs of all variables, we propose the three following variants, each of which *estimates* the true PDF $Pr_{x_i,u}^{d,f}$ of a random variable $\psi_{x_i,u}^{d,f}$, representing the cost incurred by $x_i$ from constraint $f$ if assigned value $d$ when its control is under user $u$:

- OPTIMISTIC: In this variant, the agent will optimistically choose the PDF with smallest mean among all the PDFs for all possible values of variables $x_j \in \mathbf{x}^f \setminus \{x_i\}$ in the scope of constraint $f$:

$$Pr_{x_i,u}^{d,f} = \mathcal{N}(\mu^*, \sigma_{\hat{d}}^2) \tag{3.10}$$

$$\mu^* = \min_{\hat{d} \in D_j} \mu_{\hat{d}} \tag{3.11}$$

  where $\mathcal{N}(\mu_{\hat{d}}, \sigma_{\hat{d}}^2)$ is the PDF of the constraint cost if $x_i = d$ and $x_j = \hat{d}$ under user $u$. For example, in the uncertain cost tables in Figure 3.2 (c), the estimated PDF of the cost incurred for the choice $x_2 = 0$ from constraint $f_{24}$ is $Pr_{x_2,u}^{0,f_{24}} = \mathcal{N}(65, 8^2)$, which *optimistically* assumes that $x_4$ will be assigned value 0 to minimize the incurred cost.

- PESSIMISTIC: In this variant, the agent chooses the PDF with largest mean among all the PDFs for all possible values of $x_j \in \mathbf{x}^f \setminus \{x_i\}$:

$$Pr_{x_i,u}^{d,f} = \mathcal{N}(\mu^*, \sigma_{\hat{d}}^2) \tag{3.12}$$

$$\mu^* = \max_{\hat{d} \in D_j} \mu_{\hat{d}} \tag{3.13}$$

In Figure 3.2 (c), the estimated PDF of the cost incurred by $x_2 = 0$ from constraint $f_{24}$ is $Pr^{0,f_{24}}_{x_2,u} = \mathcal{N}(71, 8^2)$, which *pessimistically* assumes that $x_4$ will be assigned value 1 to maximize the incurred cost.

- Expected: In this variant, the agent chooses the PDF with the "average" value of all the PDFs for all possible values of $x_j \in \mathbf{x}^f \setminus \{x_i\}$:

$$Pr^{d,f}_{x_i,u} = \mathcal{N}\left(\frac{1}{|D_j|}\sum_{\hat{d}\in D_j}\mu_{\hat{d}}, \frac{1}{|D_j|^2}\sum_{\hat{d}\in D_j}\sigma^2_{\hat{d}}\right) \tag{3.14}$$

In Figure 3.2 (c), the estimated PDF of the cost incurred by $x_2 = 0$ from constraint $f_{24}$ is $Pr^{0,f_{24}}_{x_2,u} = \mathcal{N}(68, 8^2)$, assuming that $x_4 = 0$ or $x_4 = 1$ with equal probability.

The *regret* $R^d_{x_i,u}$ of variable $x_i$ being assigned value $d$ is defined as:

$$R^d_{x_i,u} = 1 - Pr_{x_i,u}(d) \tag{3.15}$$

Each variable $x_i$ will most likely be assigned the value $d^*_{x_i}$ with the smallest regret by definition (see Equations 3.7 and 3.8). We thus define the regret $R_{x_i,u}$ for each variable $x_i$ to be the regret for this value:

$$R_{x_i,u} = R^{d^*_{x_i}}_{x_i,u} = \min_{d\in D_i} R^d_{x_i,u} \tag{3.16}$$

To generalize our approach to also handle multiple users in each house, where the PDFs differ across users, we take the maximum regret over all users $u$ for each variable $x_i$ and its value $d$ before taking the minimum over all values. More precisely,

$$R_{x_i} = \min_{d\in D_i}\max_{u} R^d_{x_i,u} \tag{3.17}$$

Therefore, the minimax regret approach seeks to approximate the impact of the worst-case scenario. Finally, we define the regret $R_{f_i}$ for a constraint $f_i$ to be the absolute difference between the regrets of the variables in the scope of the function:

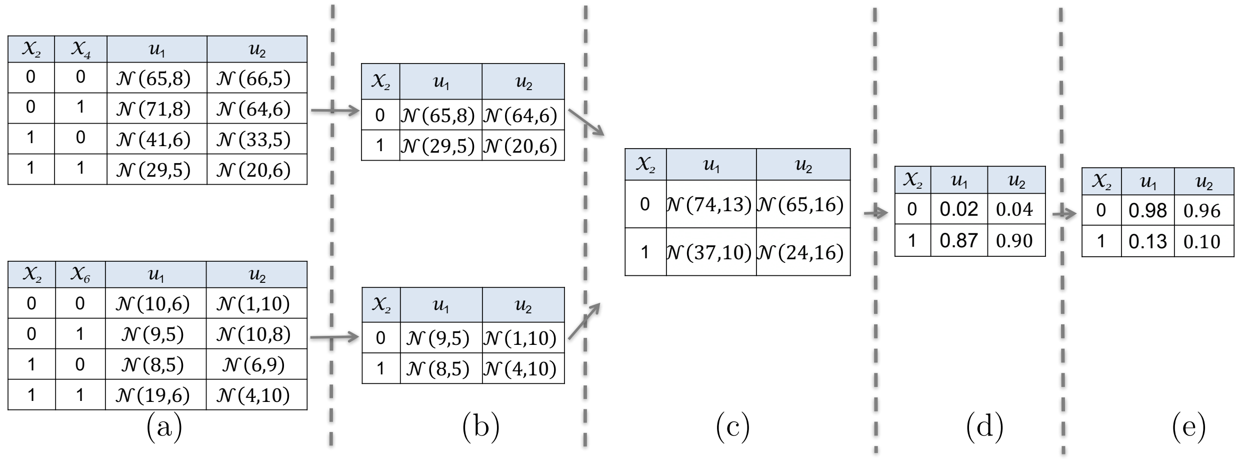
$$R_{f_i} = |R_{x_{i_1}} - R_{x_{i_2}}| \tag{3.18}$$

where $\mathbf{x}^{f_i} = \{x_{i_1}, x_{i_2}\}$.

While defining the regret to be the sum of the two variables' regrets may be more intuitive, our experimental results show that the above definition provides better results. Intuitively, if the regret of a variable $x_i$ is large, then there is little confidence that it will take on value $d^*_{x_i}$ with the smallest regret because the PDFs for all its values are very similar and have significant overlaps. Thus, eliciting a constraint between two variables with large regrets will likely not help in improving the overall solution quality since the PDFs for all value combinations for that constraint are likely to be similar.

Similarly, if the regret of a variable is small, then there is a high confidence that it will be assigned value with the smallest regret because the PDFs for its values are sufficiently distinct that regardless of the actual realizations of the random variables (costs in the cost table), the value with the smallest regret will be the one with the smallest cost. Therefore, eliciting a constraint between two agents with small regrets will also not help. Therefore, we define the regret of a constraint to be the difference in the regrets of the variables in its scope (see Equation 3.18).

If we order the constraints using the ordering criteria $\circ = MR[\cdot]$, that is, according to the minimax regret criterion, then, given two uncertain constraints $f_i, f_j \in \mathcal{F}_u$, we say that $f_i \succeq_{MR} f_j$ *iff* $MR[f_i] \geq MR[f_j]$, where $MR[f_j] = R_{f_j}$ is the regret as defined in Equation 3.18.



(a)

| $x_2$ | $x_4$ | $u_1$ | $u_2$ |
|---|---|---|---|
| 0 | 0 | $\mathcal{N}(65,8)$ | $\mathcal{N}(66,5)$ |
| 0 | 1 | $\mathcal{N}(71,8)$ | $\mathcal{N}(64,6)$ |
| 1 | 0 | $\mathcal{N}(41,6)$ | $\mathcal{N}(33,5)$ |
| 1 | 1 | $\mathcal{N}(29,5)$ | $\mathcal{N}(20,6)$ |

| $x_2$ | $x_6$ | $u_1$ | $u_2$ |
|---|---|---|---|
| 0 | 0 | $\mathcal{N}(10,6)$ | $\mathcal{N}(1,10)$ |
| 0 | 1 | $\mathcal{N}(9,5)$ | $\mathcal{N}(10,8)$ |
| 1 | 0 | $\mathcal{N}(8,5)$ | $\mathcal{N}(6,9)$ |
| 1 | 1 | $\mathcal{N}(19,6)$ | $\mathcal{N}(4,10)$ |

(b)

| $x_2$ | $u_1$ | $u_2$ |
|---|---|---|
| 0 | $\mathcal{N}(65,8)$ | $\mathcal{N}(64,6)$ |
| 1 | $\mathcal{N}(29,5)$ | $\mathcal{N}(20,6)$ |

| $x_2$ | $u_1$ | $u_2$ |
|---|---|---|
| 0 | $\mathcal{N}(9,5)$ | $\mathcal{N}(1,10)$ |
| 1 | $\mathcal{N}(8,5)$ | $\mathcal{N}(4,10)$ |

(c)

| $x_2$ | $u_1$ | $u_2$ |
|---|---|---|
| 0 | $\mathcal{N}(74,13)$ | $\mathcal{N}(65,16)$ |
| 1 | $\mathcal{N}(37,10)$ | $\mathcal{N}(24,16)$ |

(d)

| $x_2$ | $u_1$ | $u_2$ |
|---|---|---|
| 0 | 0.02 | 0.04 |
| 1 | 0.87 | 0.90 |

(e)

| $x_2$ | $u_1$ | $u_2$ |
|---|---|---|
| 0 | 0.98 | 0.96 |
| 1 | 0.13 | 0.10 |

Figure 3.3: Simplified Execution Trace of the Minimax Regret Hueristic

Figure 3.3 illustrates a partial trace of this approach on the example DCOP of Figure 3.2 with two users $u_1$ and $u_2$. Figure 3.3 (a) shows the uncertain cost tables for constraint $f_{24}$ between variables $x_2$ and $x_4$ and constraint $f_{26}$ between variables $x_2$ and $x_6$. Figure 3.3 (b) shows the estimated PDFs $Pr^{d,f}_{x_2,u}$ of the constraint costs incurred by variables $x_2$ from constraint $f$ under user $u$ if it takes on value $d$. In this trace, we use the "optimistic" variant of the algorithm, and the PDFs are estimated using Equations 3.10 and 3.11. Figure 3.3 (c) shows estimated summed PDFs $Pr^{d,f}_{x_2,u}$ of the *total* constraint costs incurred by the agent, summed over all of its constraints. Here, we only sum the PDFs for the two constraints $f_{24}$ and $f_{26}$. Figure 3.3 (d) shows the probabilities $Pr_{x_2,u}(d)$ of $x_2 = d$ under user $u$, computed using Equation 3.7, and Figure 3.3 (e) shows the regrets $R^d_{x_2,u}$, computed using Equation 3.15. Thus, the regret $R_{x_2}$ for $x_2$ is 0.13, computed using Equation 3.17. Assume that the regret $R_{x_1}$ of $x_1$ is 0.50. Then, the regret $R_{f_{12}}$ of constraint $f_{12} = |R_{x_1} - R_{x_2}| = |0.50 - 0.13| = 0.37$.

**Maximum Standard Deviation:** We now propose a different heuristic that makes use of the degree of uncertainty in the constraint costs $\chi^v_{f,u}$ for constraint $f$, value combination $v = \langle x_{i_1} = d_{i_1}, \ldots, x_{i_k} = d_{i_k} \rangle$, and user $u$, where $\mathbf{x}^f = \{x_{i_1}, \ldots, x_{i_k}\}$, $d_{i_1} \in D_{i_1}$, …, and $d_{i_k} \in D_{i_k}$. Assume that there is only a single user $u$. Then, using the same motivation described for the minimax regret heuristic, and assuming that *variables be assigned a different value for different constraints*, the value combination chosen for a constraint $f$ will be the $v^* = \text{argmin}_v \chi^v_{f,u}$ that has the smallest cost. Unfortunately, the actual constraint costs are not known and only their PDFs $\mathcal{N}(\mu^v_{f,u}, (\sigma^v_{f,u})^2)$ are known.

Since the constraint costs' distribution means are known, we assume that the value combination chosen for a constraint $f$ will be the value $v^* = \text{argmin}_v \mu^v_{f,u}$ that has the smallest mean. The degree of uncertainty in the constraint cost for that constraint $f$ is thus the standard deviation associated $\sigma^{v^*}_{f,u}$ with that value combination $v^*$.

To generalize this approach to multiple users, we take the maximum standard deviations over all users $u$. More precisely, the degree of uncertainty in the constraint costs for a constraint $f$ is:

$$\sigma_f = \max_u \sigma^{v^*}_{f,u} \tag{3.19}$$

One can then use this maximum standard deviation criterion to order the constraints. In other words, if the ordering criteria $\circ = MS[\cdot]$ is done according to the maximum standard deviation criterion, then, given two unknown functions $f_i, f_j \in \mathcal{F}_u$, we say that $f_i \succeq_{MS} f_j$ *iff* $MS[f_i] \geq MS[f_j]$, where $MS[f_j] = \sigma_{f_j}$ is the maximum standard deviation as defined in Equation 3.19.

### 3.4.2 Off-the-Shelf Solvers

As mentioned before, to solve the proposed uncertain constraint-based model, we must elicit a subset of constraint tables (preferences) using one of the heuristics introduced in previous section. The elicitation of the unknown constraints (preferences) occurs before executing an algorithm to solve the uncertain COPs/DCOPs. Therefore, we can use an off-the-shelf constraint programming solver, MiniZinc [104], to solve all COPs/DCOPs in centralized manner. In this work, we only measure the solution quality of COPs/DCOPs which is independent of the choice of solver we used, hence, any algorithm or solver can be used to find a solution for the constraint-based model after eliciting its unknown preferences.

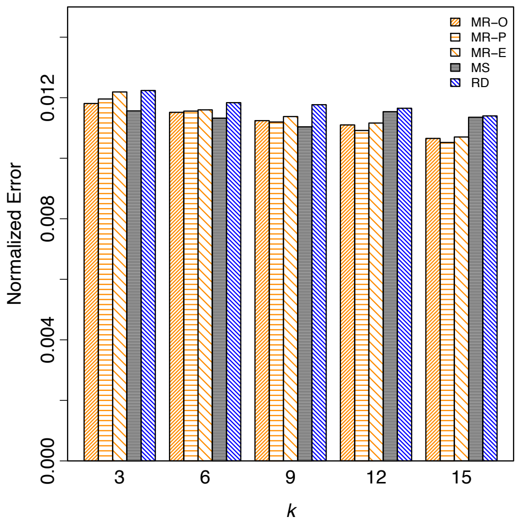
MiniZinc is a language for specifying constraint optimization problems. MiniZinc models consist of variable declarations and constraint definitions as well as a definition of the objective function if the problem is an optimization problem.

## 3.5 Experimental Evaluations

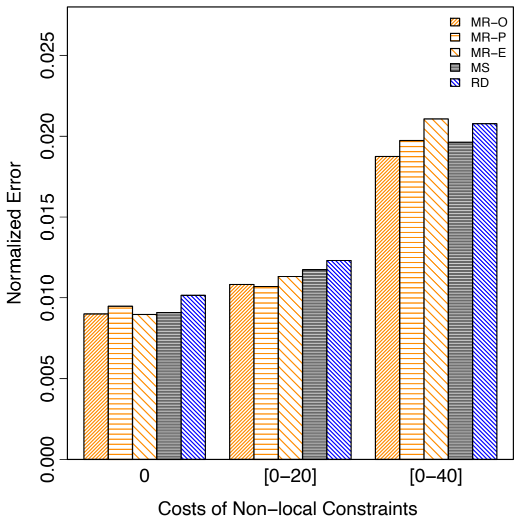
We evaluate our preference elicitation framework on distributed random binary graphs and smart home device scheduling (SHDS) problems [31, 108], where we compared our four heuristics – minimax regret with the three variants: optimistic (MR-O), pessimistic (MR-P), and expected (MR-E) and maximum standard deviation (MS) – against a random baseline (RD) that chooses the constraints to elicit randomly. All the problems are modeled and solved optimally on multiple computers with Intel Core i7-3770 CPU 3.40GHz and 16 GB of RAM.

### 3.5.1 Metrics

In our experiments, the preference elicitation heuristics are evaluated in terms of the *normalized error* $\frac{\epsilon_{\hat{\mathcal{P}}}}{\mathbf{F}_{\mathcal{P}}(\mathbf{x}^*)}$, where $\epsilon_{\hat{\mathcal{P}}}$ is the error as defined by Equation 3.6. An accurate computation of this error requires us to generate all possible realizations for the uncertain DCOPs. Due to the complexity of such task, we create $m = 50$ realizations of the uncertain DCOPs and compute the error $\epsilon_{\hat{\mathcal{P}}}$ in this reduced sampled space.



(a) Varying $k$ (Number of Elicited Constraints)



(b) Varying Costs of Non-Local Constraints

Figure 3.4: Random Graphs: Preference Elicitation Empirical Results

### 3.5.2 Random Graphs

We create 100 random graphs whose topologies are based on the Erdős and Rényi model [24] with the following parameters: $|\mathcal{X}| = 50$, $|\mathcal{A}| = 5$, and $|D_i| = 2$ for all variables $x_i \in \mathcal{X}$. Each agent $a_i$ has $|\mathbf{L}_i| = 10$ local variables with density $p_1 = 0.8$ that produces $|\mathcal{F}_i| = 36$ local constraints per agent. These constraints are unknown (uncertain constraints) and we set two scenarios for all uncertain cost tables. All constraint costs are modeled as random variables following a Normal distribution $\mathcal{N}(\mu, \sigma^2)$, where $\sigma$ is uniformly sampled from the range $[5, 10]$ and the means $\mu$ are uniformly sampled from one of the following six ranges $[5, 70]$, $[5, 80]$, $[5, 90]$, $[5, 100]$, $[5, 110]$, and $[5, 120]$. The different ranges are to introduce some heterogeneity into the constraints. We set the non-local constraints (i.e., inter-agent constraints) to be uncertain constraints as well, where we vary the mean $\mu$ of their Normal distributions to be from different distributions: $\mu = 0$; $\mu \in [0, 20]$; and $\mu \in [0, 40]$. Finally, we allow only local constraints (or preferences) to be elicited.

Figure 3.4 (a) illustrates the normalized errors of our heuristics and that of the random baseline heuristic, where the mean $\mu$ of the non-local constraints are uniformly sampled from the range $[0, 20]$. We control $k$ so that the number of the constraints per agent elicited from the oracle DCOP varies from 3 to 15 with increment of 3. We make the following observations:

- As the number of constraints ($k$) to elicit increases, the errors of the MR-P and MR-O heuristic decrease for all values of $k$ as opposed to the random heuristic which is approximately the same for all values of $k$. The reason is, as we increase $k$, the random heuristic randomly selects $k$ constraint to elicit with high likelihood of choosing the wrong constraints. However, since the regret-based heuristic (e.g., MR-P) takes into account the uncertain cost of the constraints it chooses those minimizing the regret.
- The MS heuristic performs slightly better than random heuristic. The reason is that MS orders the uncertain constraints by their degrees of uncertainty (i.e., $\sigma$) corresponding to the most likely value combinations to be assigned (i.e., the ones with the smallest $\mu$). In contrast, the random heuristic chooses constraint randomly without taking into account the degree of uncertainty.
- All regret-based heuristics outperform the baseline heuristic, especially for larger values of $k$, indicating that they are able to effectively take the regrets of the constraints into account.

Figure 3.4 (b) illustrates the normalized errors for the random problems, where we vary the mean values $\mu$ of the non-local constraints, sampled from different distributions; we set $k = 15$ for all cases. The same trends observed above apply here. However, the normalized error increases as the range of the mean increases for all heuristics. The reason is because the magnitude in the error (when variables are assigned wrong value due to wrong guesses in the cost of the constraints) increases when the range increases. However, generally, the

| Dish-washer | Washer | Dryer | Hob | Oven | Micro-wave | Lap-top | Desk-top | Vacuum Cleaner | Fridge | Electrical Vehicle |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.75 | 1.20 | 2.50 | 3.00 | 5.00 | 1.70 | 0.10 | 0.30 | 1.20 | 0.30 | 3.50 |

Table 3.1: Smart Devices and their Energy Consumption (in kWh)

optimistic and pessimistic variants of the minimax regret heuristics still perform better in all three cases.

### 3.5.3 Smart Home Device Scheduling

**SHDS Problem Construction:** We now describe how we construct SHDS problems. As the only uncertain element in the uncertain constraints are the discomfort values $d^t_{z_j,u}$ (defined in Section 5.2) for devices $z_j$, time steps $t$, and users $u$, we model these values as random variables following a Normal distribution (e.g., one could fit a Normal distribution to the historical data). As the distribution for one user may be different from the distribution for a different user in a home, for each user $u$, we generate a discomfort table composed of a Normal distribution $\mathcal{N}(\mu^t_{z_j,u}, (\sigma^t_{z_j,u})^2)$ for each device $z_j$ and time step $t$. Each user $u$ can gain the exclusive access to a device $z_j$ with the probability $Pr^t_{z_j}$, and the Normal distribution of the discomfort of device $z_j$ at time step $t$ is the Normal distribution of the user that has exclusive access for that device and time step.

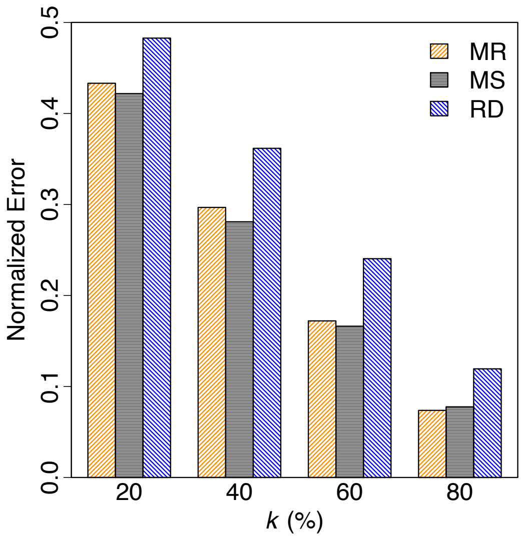
Next, let $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A}, \alpha \rangle$ denote the DCOP whose constraints $\mathcal{F}$ have *accurate* cost tables that depend only on external parameters and are easily obtained (e.g., price function $\theta$ and energy consumption of devices $\rho_{z_j}$) or they depend on user preferences that are accurately obtained through an oracle. Using the same process described above, we combine the discomfort tables for multiple users into a single aggregated discomfort table $\mathcal{U}$. Note that this aggregated discomfort table may be different than the one $\hat{\mathcal{U}}$ for the uncertain DCOP if there are multiple users. Then, the actual discomfort value $d^t_{z_j}$ for each device $z_j$

and time step $t$ is sampled from the Normal distribution $\mathcal{N}(\mu^t_{z_j}, (\sigma^t_{z_j})^2)$ for that device and time step in the aggregated discomfort table $\mathcal{U}$. We refer to this problem as the *oracle DCOP*. In summary, when a constraint is elicited, the actual discomfort values are retrieved from the oracle DCOP.

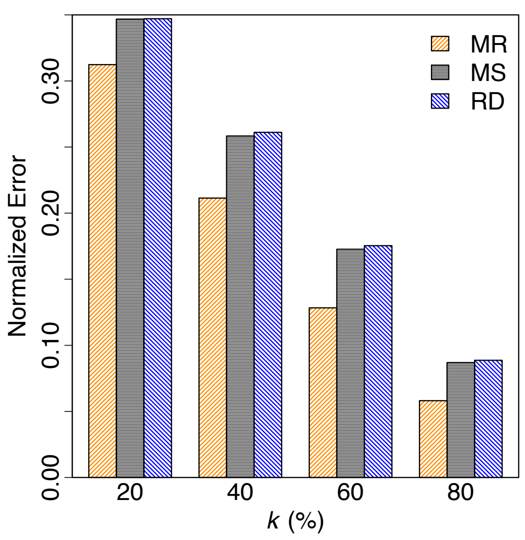
**Experimental Setup:** In our experiments, we consider $|\mathcal{H}| = 10$ homes, each controlling $|\mathcal{Z}_i| = 10$ smart devices, listed in Table 3.1 along with their energy consumption. We populate the set of smart devices $\mathcal{Z}_i$ of each home by randomly sampling 10 elements from $\mathcal{Z}$. Thus, a home might control multiple devices of the same type. We set a time horizon $H = 6$ with increments of 4 hours. We use the same real-time pricing schema as proposed by Fioretto *et al.* [31], which is the one used by the Pacific Gas and Electric Company for their Californian consumers during peak summer months.[2]

To generate the discomfort table for each user, we assume that there is a weak correlation between the price of energy and the level of discomfort of the user. Specifically, we assume that users will prefer (i.e., they are more comfortable) using their devices when prices are low to save money. Therefore, the higher the price, the more uncomfortable the user will be at using the device at that time. Based on this assumption, for each home, user, and device, the mean $\mu^t$ at each time step $t$ is an integer that is uniformly sampled from the range $[\max\{1, \theta(t) - 50\}, \theta(t) + 50]$, where $\theta(t)$ is the real-time pricing at time step $t$ used by the Pacific Gas and Electric Company. Therefore, the range of the means differ across time steps but are the same for all devices as the discomfort level is primarily motivated by the pricing schema. The weights $\alpha_c$ and $\alpha_u$ of the objective function defined in Equation 3.5 are both set to 0.5. These settings are employed to create both an oracle DCOP and the corresponding uncertain DCOP, except that the values of the constraints of the uncertain

[2]`https://www.pge.com/en_US/business/rate-plans/rate-plans/peak-day-pricing/peak-day-pricing.page`. Retrieved in November 2016.



(a) SHDS Single User



(b) SHDS Multiple Users

Figure 3.5: Smart Home Device Scheduling: Preference Elicitation Empirical Results

DCOPs are not realized (i.e., they are distributions). Finally, since all uncertain constraints in an SHDS problem are unary constraints, all three variants of the minimax regret heuristics are identical, and we use "MR" to label this heuristic.

**Single User Experiments:** In the first set of experiments, we set each home to have only one user. Figure 3.5 (a) plots the error for our heuristics compared against the random baseline heuristic. The results are averaged over 100 randomly generated SHDS problem instances. We make the following observations:

- As expected, for all elicitation heuristics, the error decreases as the number of cost tables to elicit increases.
- Both the MR and MS heuristics consistently outperform the random heuristic for all values of $k$. Like the results in random graphs, the random heuristic has a higher likelihood of choosing the wrong constraint to elicit, while MR and MS choose better constraints.

- Interestingly, we observe that MS selects the constraints slightly better than MR, indicating that despite the fact that MS is a simpler heuristic, it is well-suited in problems with single users. The reason is that the key feature of MR – maximizing the regret over all users – is ignored when there is only one user.

**Multiple User Experiments:** In the second set of experiments, we set each home to have two users, where both users have equal likelihood of controlling the devices (i.e., $P_{u_1} = P_{u_2} = 0.5$). Figure 3.5 (b) shows the results. The trends for this experiment is similar to that shown for Figure 3.5 (a), where our MR heuristic outperforms the random heuristic. However, the MS heuristic performs poorly in this experiment, with similar performance as the random baseline. In general, our results show that the regret-based method outperforms other heuristics in multiple users scenarios, as it takes into account the discomfort values of all users, orders the constraints to elicit based on their minimum regrets. Similar to random graph results, MR performs better in the scenarios that multiple users take control of the devices in a building.

Finally, the SHDS and random graph experiment results demonstrate that the regret-based elicitation heuristics achieve approximately 30% and 11% improvement over the baseline random heuristic in minimizing the error, respectively. The improvements in SHDS problems are larger than those in random graphs because variables are highly connected ($p_1 = 0.8$) in random graph problems. In contrast, variables in SHDS problems are mostly independent as they mostly have unary constraints. The higher dependency between variables in random graphs reduces the improvements of our heuristics over the baseline random heuristic.

## 3.6 Related Work

There is an extensive body of work on the topic of modeling preferences [40]. In particular, Rossi *et al.* discussed *conditional-preference networks* (CP-nets) for handling preferences [97], which provide a qualitative graphical representation of preferences reflecting the conditional dependence of the problem variables. Differently from CP-nets, our proposal focuses on the notion of *conditional additive independence* [5], which requires the utility of an outcome to be the sum of the "utilities" of the different variable values of the outcome.

In terms of preference elicitation, two major approaches are studied in the literature: A Bayesian approach [10] and a minimax regret approach [9, 37, 132]. The former is typically adopted when the uncertainty can be quantified probabilistically, and preference elicitation is often formalized as a *partially-observable Markov decision process* (POMDP) [53] that assumes each query to a user is associated with a finite set of possible responses. In contrast, our proposal follows the minimax regret approach [9, 37, 132]. The proposed framework differs from other proposals in the literature in the following ways: We assume the unknown costs are sampled from a Normal distribution and compute the regret based on such distributions. In contrast, other minimax regret based methods have different assumptions. For example, Boutilier *et al.* assumes that a set of (hard) constraints together with a graphical utility model captures user preferences [9]. While the structure of the utility model is known, the parameters of this utility model are imprecise, given by upper and lower bounds. The notion of regret is computed based on those upper and lower bounds. Differently, Wang and Boutilier compute regrets under the assumption that constraints over unknown utility values are linear [132].

Moreover, Gelain *et al.* computes regrets by taking the minimum among the known utilities associated to the projections of an assignment, that is, of the appropriated sub-tuples in the constraints [37].

Finally, preference elicitation has never been applied directly on DCOPs before. The closest DCOP-related problem is a class of DCOPs where agents have partial knowledge on the costs of their constraints and, therefore, they may discover the unknown costs via exploration [115, 147]. In this context, agents must balance the coordinated *exploration* of the unknown environment and the *exploitation* of the known portion of the rewards, in order to optimize the global objective [103]. Another orthogonal related DCOP model is the problem where costs are sampled from probability distribution functions [79]. In such a problem, agents seek to minimize either the worst-case regret [133] or the expected regret [61].

## 3.7 Conclusions

DCOPs have been used to model a number of multi-agent coordination problems including the smart home device scheduling (SHDS) problem. However, one of the key limitations in DCOPs – that constraint costs are known a priori – do not apply to many applications including SHDS. To address this limitation, in this chapter, we proposed the uncertain constraint-based model where constraint costs ( i.e preferences) are represented as Normal distributions; introduced the problem of preference (i.e., constraint cost) elicitation for DCOPs; designed minimax regret-based heuristics to elicit the preferences; evaluated our proposed framework and its heuristic strategies on random binary DCOPs as well as SHDS problems. Our results demonstrated that our methods outperform the baseline method that elicits preferences randomly. By introducing the uncertain constraint-based models, we make the foundational contributions necessary in deploying DCOP algorithms on practical applications, where preferences or constraint costs must be elicited or estimated.

In future chapters of this dissertation, we will discuss other approaches to model constraint-based problems where constraints are partially defined. We will propose heuristic strategies to elicit a subset of constraint costs while interleaving elicitation and resolution algorithms to find optimal and sub-optimal solutions.

# Chapter 4

# Intra-execution Elicitation Approach in Centralized Constraint-Based Models

To address the key drawback in centralized constraint-based models, we design the model so that constraint costs (i.e., preferences) are allowed to be unknown a priori. Thus, we extend the constraint-based models under the assumption that some of the preferences are not specified. The unknown preferences will be elicited during the search for optimal/sub-optimal solutions. Revealing the actual values of the unknown preferences throughout the search will incur accumulative elicitation costs. This chapter introduces incomplete constraint-based models and preference elicitation strategies with the objective of finding solutions that minimize both constraint and elicitation costs. In what follows, we briefly introduce the intuition behind our model and motivate our approach with a real-world application. Then, we will discuss our model, elicitation heuristic strategies, and our evaluation analyses in more detail.

## 4.1 Introduction

The importance of constraint reasoning in agent-based systems is outlined by the impact of its application in a wide range of agent-based applications, such as supply-chain management [34, 95], roster scheduling [1, 11], meeting scheduling [69], combinatorial auctions [99], bioinformatics [2, 12, 28], and smart home automation [31, 98]. In *Constraint Satisfaction Problems* (CSPs), the goal is to find a value assignment for a set of variables that satisfies a set of constraints [3, 44]. The assignments satisfying the problem constraints are called *solutions.* In *Weighted Constraint Satisfaction Problems* (WCSPs), the goal is to find an optimal solution, given a set of preferences expressed by means of cost functions [6, 101, 102].

A key assumption in all these constraint-based models is that *all* the constraints are specified or known a priori. In some applications, such as roster and meeting scheduling problems, some constraints encode the preferences of human users. As such, they may not be fully specified simply because it is unrealistic to accurately know the preferences of users for all possible scenarios in an application. Motivated by such applications, researchers proposed the *Incomplete WCSP* (IWCSP) problem formulation [35], which extends WCSPs by allowing some constraints to be partially specified (i.e., the costs for some constraints are unknown). To solve IWCSPs, they introduced a series of algorithms that interleave the search process, which seeks to find a good solution, and the preference elicitation process, which seeks to obtain some subset of cost functions from the user. Unfortunately, existing approaches suffer from a key limitation – they assume that the elicitation of preferences does not incur any additional cost. This assumption is not realistic as human users are likely bothered by repeated elicitations and will refuse to provide an unbounded number of preferences.

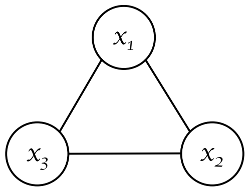
To address this limitation, we make the following contributions:

- We first propose the *IWCSP with Elicitation Costs* (IWCSP+EC) model, which extends the IWCSP model to include the notion of *elicitation costs*. This problem aims at finding a solution that minimizes the sum of both the constraint costs and elicitation costs.
- We then introduce three *parameterized* heuristics – Least Unknown Cost (LUC), Least Known Cost (LKC) and their combination heuristic (COM) – that allow users to trade off solution quality for fewer elicited preferences and faster computation times. Further, in settings where elicitations are free (i.e., the elicitation costs are zero), these heuristics also provide *theoretical quality guarantees* on the solutions found.

Our experimental results show that COM finds solutions with larger constraint costs than LKC and LUC, but finds them faster and with fewer elicitations than LKC and LUC. Therefore, COM is the preferred heuristic in critical time-sensitive domains. COM also does a better job at trading off solution quality for smaller runtimes, especially when runtimes are large, through the use of user-defined weights. Our model and heuristics thus improve the practical applicability of IWCSPs as they now take into account elicitation costs and provide control knobs, in the form of user-defined weights, to perform tradeoffs along three key dimensions – solution quality, runtime, and number of elicited preferences.

## 4.2 Motivation

We have described the *smart home scheduling* problem [110, 112] as a motivating application for our work in Chapter 3. In this chapter, we formulate this problem in a centralized manner,



(a) Constraint Graph

| $x_1$ | $x_2$ | $\tilde{f}_1$ | $f_1$ | $e_1$ |
|---|---|---|---|---|
| 0 | 0 | 15 | 15 | 0 |
| 0 | 1 | ? | 11 | 20 |
| 1 | 0 | 21 | 21 | 0 |
| 1 | 1 | ? | 30 | 9 |

| $x_1$ | $x_3$ | $\tilde{f}_2$ | $f_2$ | $e_2$ |
|---|---|---|---|---|
| 0 | 0 | 4 | 4 | 0 |
| 0 | 1 | ? | 45 | 3 |
| 1 | 0 | ? | 8 | 29 |
| 1 | 1 | 10 | 10 | 0 |

| $x_2$ | $x_3$ | $\tilde{f}_3$ | $f_3$ | $e_3$ |
|---|---|---|---|---|
| 0 | 0 | ? | 50 | 9 |
| 0 | 1 | 5 | 5 | 0 |
| 1 | 0 | 7 | 7 | 0 |
| 1 | 1 | ? | 6 | 81 |

(b) Incomplete Constraint Costs and Elicitation Costs

Figure 4.1: Example of an Incomplete Weighted CSP with Elicitation Costs

considering the scheduling problem within one smart home rather than a neighborhood of smart homes. Thus, we assume, an autonomous software agent is deployed in a smart home that is able to automate and schedule the smart IoT devices within the home. To do so effectively, it needs to know the preferences and constraints of users in the home in order to find a schedule that satisfies all the constraints and optimizes the preferences of the users. While some of these preferences may be known, some preferences may be unknown and must be elicited. This problem can be modeled as an IWCSP:

- Variables correspond to smart IoT devices in the home.
- Domains for the variables correspond to the different possible schedules of the devices.
- Weighted constraints correspond to the preferences of users.

The objective of this problem is to find a schedule for each device such that the cumulative preference of the user is optimized while eliciting as few partially-known preferences as possible.

However, in a smart home scheduling problem, users will likely be bothered by the elicitation of their preferences. Therefore, there is a need for a model and approaches that explicitly consider such elicitation costs. The existing IWCSP model only takes into account such elicitation costs implicitly – through its goal of minimizing the number of preferences elicited during the search. Further, such an assumption also assumes that the elicitation cost is uniform for all preferences, which may be unrealistic. Therefore, we describe in the next section an IWCSP extension that models elicitation costs explicitly and approaches that explicitly take them into account during the search.

## 4.3 Incomplete Centralized Constraint-Based Model

An *Incomplete WCSP* (IWCSP) [35] extends WCSPs by allowing some constraints to be *partially specified.* It is defined by a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \tilde{\mathcal{F}} \rangle$,[1] where $\mathcal{X}$, $\mathcal{D}$, and $\mathcal{F}$ are exactly the same as WCSPs. The key difference is that the set of *fully-specified* constraints $\mathcal{F}$ are not known to an IWCSP algorithm. Instead, only the set of *partially-specified* constraints $\tilde{\mathcal{F}} = \{\tilde{f}_1, \ldots, \tilde{f}_m\}$ are known. Each partially-specified constraint is a function $\tilde{f}_i : \bigtimes_{x_j \in \mathbf{x}^{f_i}} D_j \rightarrow \mathbb{R}_0^+ \cup \{\infty, ?\}$, where ? is a special element denoting that the cost for a given combination of value assignment is not specified. The costs $\mathbb{R}_0^+ \cup \{\infty\}$ that are specified are exactly the costs of the corresponding specified constraints $f_i \in \mathcal{F}$.[2] The goal is still to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \mathbf{F}_{\mathcal{P}}(\mathbf{x})$, while specifying as few ? of the partially-specified constraints as possible.

### 4.3.1 Incomplete WCSPs with Elicitation Costs

We propose the *IWCSP with Elicitation Cost* (IWCSP+EC) model, which, as the name implies, extends IWCSPs with elicitation costs. It is defined by a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \tilde{\mathcal{F}}, \mathcal{E} \rangle$, where $\mathcal{X}$, $\mathcal{D}$, $\mathcal{F}$, and $\tilde{\mathcal{F}}$ are exactly the same as IWCSPs. $\mathcal{E} = \{e_1, \ldots, e_m\}$ is the set of elicitation costs, where each elicitation cost $e_i : \bigtimes_{x_j \in \mathbf{x}^{f_i}} D_j \rightarrow \mathbb{R}_0^+$ specifies the cost of specifying

[1]The original definition by Gelain *et al.* [35] does not include the set of weighted constraints $\mathcal{F}$. We include it here to ease the understanding of the different types of solutions.

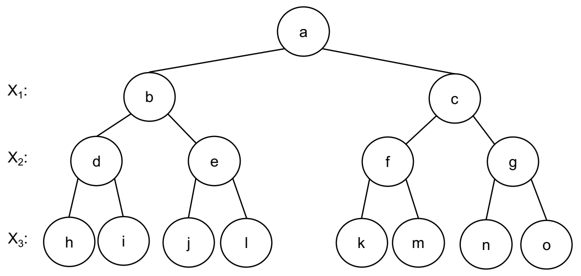
[2]A constraint $\tilde{f}_i \in \tilde{\mathcal{F}}$ can be fully specified, in which case $\tilde{f}_i = f_i \in \mathcal{F}$.

the weighted constraint $f_i$ (i.e., it is the cost of eliciting the preferences corresponding to constraint $f_i$).

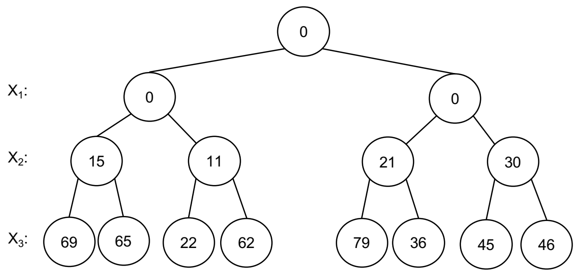
An *explored solution space* $\tilde{\mathbf{x}}$ is the union of all solutions explored thus far by a particular algorithm. The cost $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) = \sum_{e \in \mathcal{E}} e(\tilde{\mathbf{x}})$ is the sum of the costs of all applicable elicitation cost functions in $\tilde{\mathbf{x}}$. In other words, it is the cumulative elicitation cost for specifying all ? of partially-specified constraints in the explored solution space.

The total cost $\mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$ is the sum of both the cumulative constraint cost $\mathbf{F}_{\mathcal{P}}(\mathbf{x})$ of solution $\mathbf{x}$ and the cumulative elicitation cost $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$ of explored solutions $\tilde{\mathbf{x}}$. Naturally, the solution $\mathbf{x} \subseteq \tilde{\mathbf{x}}$ must be within the space of explored solutions thus far. The goal is to find an optimal complete solution $\mathbf{x}^*$ while eliciting preferences from a minimal set of solutions $\tilde{\mathbf{x}}^*$ only. More formally, $(\mathbf{x}^*, \tilde{\mathbf{x}}^*) = \text{argmin}_{(\mathbf{x},\tilde{\mathbf{x}})} \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$. We would like to note that it is likely impossible to find such an optimal complete solution *and* elicit preferences related to that solution *only*, except in some very special cases (e.g., only the optimal complete solution has partially-specified constraints).

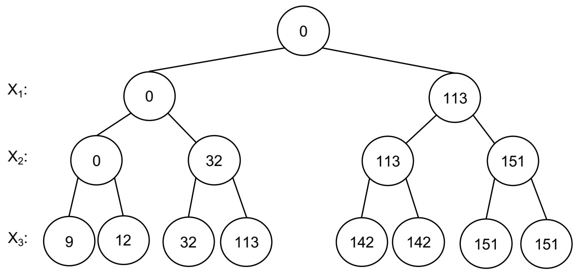
Figure 5.1(a) shows the constraint graph of an example IWCSP+EC with three variables $x_1$, $x_2$, and $x_3$, where all variables are constrained with each other. The domains are $D_1 = D_2 = D_3 = \{0, 1\}$. Figure 5.1(b) shows both the partially-specified and fully-specified costs as well as the elicitation costs for all constraints. In this example, the optimal complete solution is $\mathbf{x}^* = \langle x_1 = 1, x_2 = 0, x_3 = 1 \rangle$, and only that solution is explored (i.e., $\tilde{\mathbf{x}}^* = \mathbf{x}^*$). The constraint cost of that solution is 36 ($=$ 21 from $f_1(\langle x_1 = 1, x_2 = 0 \rangle)$ + 10 from $f_2(\langle x_1 = 1, x_3 = 1 \rangle)$ + 5 from $f_3(\langle x_2 = 0, x_3 = 1 \rangle)$). As no unknown costs are elicited, the cumulative elicitation cost is 0. Thus, the total cost is 36.



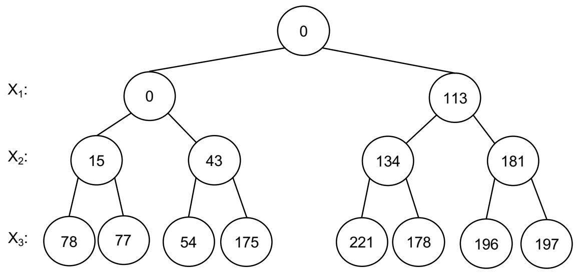
(a) Labels



(b) Constraint Costs



(c) Cumulative Elicitation Costs



(d) Total Costs

Figure 4.2: Search Trees

## 4.4 Resolution Algorithms and Heuristics

Existing IWCSP solvers [35] are all based on the *depth-first branch-and-bound* (DFBnB) search algorithm. As our solvers also use it as the underlying framework, we briefly describe it here. The operations of DFBnB can be visualized with search trees. Figure 4.2 shows search trees for our example IWCSP+EC shown in Figure 5.1, where levels 1, 2, and 3 correspond to the variables $x_1$, $x_2$, and $x_3$, respectively. Left branches correspond to the variable being assigned the value 0 and right branches correspond to the variable being assigned the value 1. Each non-leaf node thus corresponds to a partial solution of the IWCSP+EC.

Figure 4.2(a) shows the identifiers of the nodes that allow us to refer to them easily; Figure 4.2(b) shows the constraint costs of the partial solutions if all unknown constraint costs are elicited; Figure 4.2(c) shows the cumulative elicitation costs of the explored solution space if one were to expand nodes in a depth-first order and preferring left branches over right branches; and Figure 4.2(d) shows the total costs (= sum of constraint and cumulative elicitation costs). For example, node $f$ corresponds to the partial solution $\langle x_1 = 1, x_2 = 0\rangle$ with cost 134 (= constraint cost of 21 + cumulative elicitation cost of 113).

### 4.4.1 Depth First Branch-and-Bound Algorithm

Depth First Branch-and-Bound (DFBnB) expands nodes in the search tree in a depth-first order and prunes nodes whose costs are no smaller than a threshold $\mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$, where $\mathbf{x}$ is the best complete solution found so far and $\tilde{\mathbf{x}}$ is the solution space explored thus far. It backtracks once all children of a node have been expanded or pruned.

A simple and straightforward extension of DFBnB to solve IWCSP is as follows, where we use A* notations [45] in some of our definitions: Before expanding a node $n$, it elicits all the *unknown constraint costs* associated with that node and adds those costs to the *known constraint costs* associated with that node. We refer to these costs as $g(n)$. Additionally, it also adds all the *elicitation costs* associated with the unknown constraint costs elicited to the *cumulative elicitation costs* $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$, where $\tilde{\mathbf{x}}$ is the union of all partial solutions corresponding to the set of nodes expanded by DFBnB thus far. Finally, one can use heuristics, referred to as $h(n)$, to estimate the sum of constraint and elicitation costs needed to complete the partial solution at node $n$ and if those heuristics are underestimates on the true cost, then they can be used to better prune the search space, that is, when $f(n, \tilde{\mathbf{x}}) = g(n) + h(n) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) \geq \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$, where $\mathbf{x}$ is the best complete solution found so far. We provide more formal definitions below, where we use $\mathbf{x}_n$ to refer to the partial solution corresponding to node $n$.

**Definition 3 (Unknown Constraint Costs)** *The unknown constraint costs of a node $n$ are all the partially-specified constraint costs $\tilde{f}(\mathbf{x}) =\ ?$ that are unknown, where $\mathbf{x} \subseteq \mathbf{x}_n$ is a subset of the partial solution corresponding to node $n$.*

**Definition 4 (Elicited Constraint Costs)** *The elicited constraint costs of a node $n$ are all the constraint costs $f(\mathbf{x})$, where $\mathbf{x} \subseteq \mathbf{x}_n$ is a subset of the partial solution corresponding to node $n$ and the partially-specified cost $\tilde{f}(\mathbf{x}) = \ ?$ of that subset is unknown.*

**Definition 5 (Known Constraint Costs)** *The known constraint costs of a node $n$ are all the constraint costs $\tilde{f}(\mathbf{x}) \neq \ ?$, where $\mathbf{x} \subseteq \mathbf{x}_n$ is a subset of the partial solution corresponding to node $n$.*

**Definition 6 (Cumulative Elicitation Cost)** *The cumulative elicitation cost of a set of nodes $\tilde{\mathbf{x}}$ is $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$.*

**Definition 7 ($g$-Value)** *The $g$-value of a node $n$ is $\mathbf{F}_{\mathcal{P}}(\mathbf{x}_n)$.*

**Definition 8 ($h$-Value)** *The $h$-value of a node $n$ is a lower bound estimate of the minimal sum of constraint and elicitation costs to complete the partial solution $\mathbf{x}_n$.*

**Definition 9 ($f$-Value)** *The $f$-value of a node $n$ and the set of nodes $\tilde{\mathbf{x}}$ expanded thus far is the sum of its $g$-value, $h$-value, and cumulative elicitation costs.*

For example, for node $i$ in Figure 4.2(a), its unknown constraint cost is $\tilde{f}_2(\langle x_1 = 0, x_3 = 1\rangle) = \ ?$; its elicited constraint cost is $f_2(\langle x_1 = 0, x_3 = 1\rangle) = 45$; its known constraint costs are $\tilde{f}_1(\langle x_1 = 0, x_2 = 0\rangle) = 15$ and $\tilde{f}_3(\langle x_2 = 0, x_3 = 1\rangle) = 5$; and its $g$-value is $g(i) = 45 + 15 + 5 = 65$. If nodes are explored in depth-first order from left to right, then the explored solution space when node $i$ is expanded is $\tilde{\mathbf{x}} = \{\langle x_1 = 0, x_2 = 0, x_3 = 0\rangle, \langle x_1 = 0, x_2 = 0, x_3 = 1\rangle\}$, which includes two unknown constraint costs that were elicited – $f_3(\langle x_2 = 0, x_3 = 0\rangle)$ and $f_2(\langle x_1 = 0, x_3 = 1\rangle)$. The costs of these elicitations are $e_3(\langle x_2 = 0, x_3 = 0\rangle) = 9$ and $e_2(\langle x_1 = 0, x_3 = 1\rangle) = 3$, summing up to a cumulative elicitation cost of 12. Consequently, if we use a zero heuristics (i.e., the $h$-value for all nodes is 0), then the $f$-value of node $i$ and the corresponding explored solution space is $f(i, \tilde{\mathbf{x}}) = 65 + 12 = 77$.

### 4.4.2 Heuristic Strategies

We now describe our three heuristic functions that can be used in conjunction with DFBnB to solve our IWCSP-EC problem. These heuristics make use of an estimated lower bound $\mathcal{L}$ on the cost of all constraints $f \in \mathcal{F}$. Such a lower bound can usually be estimated through domain expertise or can be set to 0 in the worst case since all costs are non-negative. The more informed the lower bound, the more effective the heuristics will be in pruning the search space.

Additionally, these heuristics are parameterized by two parameters – a relative weight $w \geq 1$ and an additive weight $\epsilon \geq 0$. Users can define these parameters a priori allowing them to trade off solution quality for fewer elicited preferences and faster computation times. Further, in settings where elicitations are free (i.e., the elicitation costs are zero), the costs of solutions found are guaranteed to be at most $w \cdot OPT + \epsilon$, where $OPT$ is the optimal solution cost (Theorems 1, 2, and 3).

**Least Unknown Cost (LUC) Heuristic:** Our first heuristic function is called the *Least Unknown Cost* (LUC) heuristic. Let $S_n$ be the set of all possible variable-value assignments needed to complete the partial solution corresponding to node $n$. For each variable-value assignment $\varsigma \in S_n$, let $\varphi_\varsigma$ denote the number of yet-to-be-elicited unknown constraint costs that must be elicited to complete the partial solution, and $E_\varsigma$ denote the sum of elicitation costs of those constraint costs. LUC computes the $h$-value for each node $n$ as follows:

$$h(n) = \min_{\varsigma \in S_n} (\varphi_\varsigma \cdot \mathcal{L} + E_\varsigma) \tag{4.1}$$

Therefore, the $f$-value for node $n$, under the assumption that $\tilde{\mathbf{x}}$ is the set of nodes expanded thus far, is as follows:

$$f(n, \tilde{\mathbf{x}}) = g(n) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) + \min_{\varsigma \in S_n} (\varphi_\varsigma \cdot \mathcal{L} + E_\varsigma) \tag{4.2}$$

Thus, when using this heuristic, DFBnB prunes a node $n$ if

$$w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon \geq \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) \tag{4.3}$$

where $\mathbf{x}$ is the best complete solution found so far. Note that users can increase the weights $w$ and $\epsilon$, which will prune a larger portion of the search space. Consequently, it will reduce the computation time as well as the number of preferences elicited. However, the downside is that it will also degrade the quality of solutions found.

**Least Known Cost (LKC) Heuristic:** Our second heuristic function is called the *Least Known Cost* (LKC) heuristic. Instead of returning $h$-values like LUC, LKC directly returns the $f$-values, but taking into account only known and elicited constraint costs. To compute the $f$-value for each node $n$, LKC computes the estimated $g$-values of the leafs in the subtree rooted at $n$ (i.e., all the leaf variables in $S_n$) – it is an estimate because it does not take into account unknown constraint costs that are yet to be elicited:

$$f(n, \tilde{\mathbf{x}}) = \min_{\varsigma \in S_n} \tilde{g}(l_\varsigma) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) \tag{4.4}$$

where $\tilde{g}(l_\varsigma)$ is the estimated $g$-value of node $l_\varsigma$ and $l_\varsigma$ is the leaf node along the branch $\varsigma \in S_n$ that completes the partial solution at node $n$. Thus, when using this heuristic, DFBnB uses the same pruning condition as Equation 5.1, except that the $f$-values are computed using Equation 4.4.

**Combination (COM) Heuristic:** When estimating the minimal cost to complete the partial solution at a node $n$, the LUC heuristic takes into account yet-to-be-elicited unknown constraint costs and their corresponding elicitation costs. However, it ignores the known and elicited constraint costs needed to complete the partial solution. In contrast, the LKC heuristic does take into account such known and elicited constraint costs. However, it ignores the yet-to-be elicited unknown constraint costs and their corresponding elicitation costs. Therefore, the LUC and LKC heuristics complement each other as they are estimating non-intersecting components of the minimal cost to complete partial solutions. To take advantage of both heuristics, we combined them into a new *Combination* (COM) heuristic. The new $f$-values are thus a combination of both the $f$-values of the LUC and LKC heuristics:

$$f(n, \tilde{\mathbf{x}}) = \min_{\varsigma \in S_n} \left( \tilde{g}(l_\varsigma) + \varphi_\varsigma \cdot \mathcal{L} + E_\varsigma \right) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) \tag{4.5}$$

Thus, when using this heuristic, DFBnB uses the same pruning condition as Equation 5.1, except that the $f$-values are computed using Equation 4.5.

**Value-Ordering Heuristic:** Finally, instead of choosing a random order to explore the children of a node, for each heuristic, we order the nodes according to their $f$-values. DFBnB will then expand the child with the smallest $f$-value first as that is the most promising child.

### 4.4.3 Theoretical Results

We now discuss some theoretical results that are applicable only in the original IWCSP setting, i.e., in problems with zero elicitation costs. Therefore, in this setting, $\mathbf{F}_{\mathcal{P}}(\mathbf{x}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$ for all solutions $\mathbf{x}$ and explored search spaces $\tilde{\mathbf{x}}$.

**Theorem 1** *DFBnB with the LUC heuristic parameterized by a user-defined relative weight $w \geq 1$ and a user-defined additive weight $\epsilon \geq 0$ will return an IWCSP solution whose cost is bounded from above by $w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon$, where $\mathbf{x}^*$ is an optimal complete IWCSP solution.*

*Proof* :Assume that DFBnB returns a complete solution $\hat{\mathbf{x}}$ with cost $\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}})$. There are the following two cases:

- **Case 1:** $\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*)$. It is trivial to see that

$$\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) \leq w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon \tag{4.6}$$

  since $w \geq 1$ and $\epsilon \geq 0$.

- **Case 2:** $\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) > \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*)$. It must be the case that the subtree rooted at some node $n$ along the branch to the optimal solution $\mathbf{x}^*$ was pruned at some point during the search. Otherwise, the algorithm would have returned $\mathbf{x}^*$ since $\mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) < \mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}})$. Therefore, when the subtree was pruned, the following pruning condition from Equation 5.1 must have held:

$$w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon \geq \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) \tag{4.7}$$

where $\mathbf{x}$ is the best solution found so far and $\tilde{\mathbf{x}}$ is the search space explored so far. Further, since the RHS of the pruning condition above is non-increasing, it must be the case that:

$$\mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) \geq \mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) \tag{4.8}$$

Next, expanding on the LHS of Equation 4.7:

$$\begin{aligned}
w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon &= w \cdot (g(n) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) + \min_{\varsigma \in S_n} (\varphi_{\varsigma} \cdot \mathcal{L} + E_{\varsigma})) + \epsilon && (4.9)\\
&= w \cdot (g(n) + \min_{\varsigma \in S_n} \varphi_{\varsigma} \cdot \mathcal{L}) + \epsilon && (4.10)\\
&= w \cdot (g(n) + h(n)) + \epsilon && (4.11)\\
&\leq w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon && (4.12)
\end{aligned}$$

Finally, combining Equations 4.7 to 4.12, we get:

$$\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) \leq w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon \tag{4.13}$$

which concludes the proof. ■

**Theorem 2** *DFBnB with the LKC heuristic parameterized by a user-defined relative weight $w \geq 1$ and a user-defined additive weight $\epsilon \geq 0$ will return an IWCSP solution whose cost is bounded from above by $w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon$, where $\mathbf{x}^*$ is an optimal complete IWCSP solution.*

**Theorem 3** *DFBnB with the COM heuristic parameterized by a user-defined relative weight $w \geq 1$ and a user-defined additive weight $\epsilon \geq 0$ will return an IWCSP solution whose cost is bounded from above by $w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon$, where $\mathbf{x}^*$ is an optimal complete IWCSP solution.*

The proofs for Theorems 2 and 3 are similar to the proof for Theorem 1.

Note that these error bounds do not apply to the IWCSP+EC setting with non-zero elicitation costs. The reason is because the RHS of the pruning condition used by DFBnB is not guaranteed to be non-increasing, thereby invalidating our proof.

## 4.5 Experimental Evaluations

We evaluate DFBnB using our three heuristics – LUC, LKC, and COM – against a random (RND) heuristic on both IWCSP+ECs and IWCSPs (without elicitation costs). For IWCSPs, we also compare against the best algorithm proposed by Gelain *et al.* [35] – the LU.WW.BRANCH algorithm (labeled LWB).

### 4.5.1 Metrics

We evaluate the algorithms on two benchmarks – random graphs and smart home scheduling problems [109], where we measure the various costs of the solutions found – the cumulative constraint costs, cumulative elicitation costs, and their aggregated total cost – the number of unknown costs elicited to find those solutions, and the runtime of the algorithms. All experiments were performed on an Intel Core i7, 3.4GHz machine with 16GB of RAM. Each data point shown is an average of over 100 instances.

### 4.5.2 Random Graphs

We generate 100 random (binary) graphs [24], where we vary the number of variables $|\mathcal{X}|$ from 5 to 12; the constraint density $p_1$ from 0.2 to 0.8; the fraction of unknown costs $i$ in each constraint from 0.2 to 1.0;[3].the user-defined relative weight $w$ from 1 to 10; and the

[3]In other words, for each constraint $\tilde{f} \in \tilde{\mathcal{F}}$, $i$ is the number of value assignments with unspecified costs as a fraction of the total number of value assignments in that constraint.

(a) Varying Number of Variables $|\mathcal{X}|$, $i = 0.6$, $p_1 = 0.4$

| $\lvert\mathcal{X}\rvert$ | # unknown | # of elicited costs | | | runtime (sec) | | | total cost | | | cumulative constraint cost | | | cumulative elicitation cost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | costs | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM |
| 5 | 20.75 | 14.32 | 12.16 | 11.83 | 0.03 | 0.03 | 0.03 | 167.93 | 151.50 | 160.30 | 98.41 | 115.97 | 109.45 | 65.37 | 53.09 | 50.85 |
| 6 | 30.60 | 21.71 | 18.23 | 17.06 | 0.11 | 0.10 | 0.09 | 264.17 | 241.08 | 248.97 | 171.97 | 161.85 | 177.42 | 92.20 | 79.23 | 71.55 |
| 7 | 40.90 | 30.44 | 26.51 | 23.31 | 0.39 | 0.33 | 0.34 | 366.68 | 339.77 | 354.28 | 242.34 | 243.01 | 260.36 | 124.34 | 96.76 | 93.92 |
| 8 | 55.30 | 41.73 | 36.39 | 30.31 | 1.63 | 1.64 | 1.35 | 515.34 | 480.56 | 483.67 | 352.83 | 347.82 | 365.52 | 162.51 | 132.74 | 118.15 |
| 9 | 70.00 | 53.16 | 46.26 | 39.27 | 7.74 | 7.26 | 5.35 | 673.10 | 630.47 | 637.40 | 485.87 | 467.85 | 499.14 | 187.23 | 162.62 | 138.26 |
| 10 | 90.05 | 68.24 | 60.52 | 52.01 | 34.11 | 33.73 | 20.96 | 867.03 | 833.77 | 861.34 | 646.32 | 656.12 | 688.39 | 220.71 | 177.65 | 172.95 |
| 11 | 110.05 | 84.39 | 75.77 | 62.77 | 192.98 | 182.93 | 101.12 | 1086.84 | 1047.23 | 1058.42 | 845.08 | 831.48 | 872.95 | 241.76 | 215.75 | 185.47 |
| 12 | 130.00 | 100.41 | 89.31 | 73.95 | 1505.35 | 1301.48 | 552.12 | 1295.68 | 1248.14 | 1258.51 | 1003.53 | 1014.30 | 1047.98 | 292.15 | 233.84 | 210.53 |

(b) Varying Constraint Density $p_1$, $|\mathcal{X}| = 10$, $i = 0.6$

| $p_1$ | # unknown | # of elicited costs | | | runtime (sec) | | | total cost | | | cumulative constraint cost | | | cumulative elicitation cost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | costs | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM |
| 0.2 | 56.70 | 44.67 | 38.92 | 33.15 | 34.08 | 26.56 | 23.15 | 546.26 | 501.72 | 505.75 | 365.32 | 359.93 | 374.56 | 180.94 | 141.79 | 131.19 |
| 0.4 | 90.05 | 68.24 | 60.52 | 52.01 | 34.11 | 33.73 | 20.96 | 867.03 | 833.77 | 861.34 | 646.32 | 656.12 | 688.39 | 220.71 | 177.65 | 172.95 |
| 0.6 | 135.00 | 97.74 | 89.41 | 74.15 | 34.36 | 40.78 | 27.35 | 1333.73 | 1331.18 | 1335.97 | 1053.82 | 1086.79 | 1116.19 | 279.91 | 244.39 | 219.78 |
| 0.8 | 180.00 | 124.55 | 117.58 | 92.29 | 37.59 | 49.13 | 33.57 | 1813.18 | 1806.55 | 1817.44 | 1493.86 | 1522.51 | 1534.46 | 319.32 | 284.04 | 282.98 |

(c) Varying Fraction of Unknown Costs $i$, $|\mathcal{X}| = 10$, $p_1 = 0.4$

| $i$ | # unknown | # of elicited costs | | | runtime (sec) | | | total cost | | | cumulative constraint cost | | | cumulative elicitation cost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | costs | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM |
| 0.2 | 36.02 | 30.15 | 25.40 | 13.71 | 48.58 | 41.77 | 16.97 | 666.87 | 662.86 | 718.70 | 544.06 | 563.76 | 645.59 | 122.81 | 99.10 | 73.11 |
| 0.4 | 72.04 | 57.75 | 49.63 | 37.14 | 43.22 | 36.15 | 19.40 | 809.65 | 790.16 | 802.20 | 615.10 | 637.36 | 665.96 | 194.55 | 152.80 | 136.24 |
| 0.6 | 90.05 | 68.24 | 60.52 | 52.01 | 34.11 | 33.73 | 20.96 | 867.03 | 833.77 | 861.34 | 646.32 | 656.12 | 688.39 | 220.71 | 177.65 | 172.95 |
| 0.8 | 126.07 | 85.40 | 80.64 | 73.95 | 25.09 | 28.14 | 22.66 | 964.39 | 941.53 | 928.84 | 712.97 | 712.14 | 707.86 | 251.42 | 229.39 | 220.98 |
| 1.0 | 162.09 | 90.34 | 91.56 | 82.16 | 20.51 | 23.70 | 21.79 | 1019.10 | 1019.87 | 1018.80 | 772.25 | 775.46 | 768.83 | 246.85 | 244.41 | 249.97 |

Table 4.1: Random Graphs: IWCSP+EC Empirical Results

user-defined additive weight $\epsilon$ from 0 to 1000. The domain size $|D_i|$ for all variables $x_i \in \mathcal{X}$ is set to 3. In our experiments below, we only vary one parameter at a time, setting the rest at their default values: $|\mathcal{X}| = 10$, $p_1 = 0.4$, $i = 0.6$, $w = 1$, and $\epsilon = 0$. All constraint costs are randomly sampled from $[2, 100]$ and all elicitation costs are randomly sampled from $[0, 20]$.

**IWCSP+ECs:** Table 4.1 tabulates the empirical results for our IWCSP+EC experiments, where we vary number of variables $|\mathcal{X}|$, the density $p_1$, and the fraction of unknown costs $i$. We make the following observations with regards to the runtime of the algorithms and the number of unknown costs that they elicit:

- As expected, the runtimes of all algorithms increase with increasing number of variables $|\mathcal{X}|$ and constraint density $p_1$. This trend is also reflected in the number of unknown costs elicited. The reason is that the size of the problem, in terms of the number of constraints in the problem, increases with increasing $|\mathcal{X}|$ and $p_1$. And all algorithms need to elicit more unknown costs and evaluate the costs of more constraints before terminating.

- Interestingly, while the number of unknown costs elicited also increases for all algorithms with increasing fraction of unknown costs $i$, the runtimes of LKC and LUC decrease instead. The reason is that the ratio of unknown costs elicited compared to the total number of unknown costs actually decreases with increasing $i$. Since the size of the problem, in terms of the number of constraints in the problem, is the same for all values of $i$ (as they are dependent only on $|\mathcal{X}|$ and $p_1$, which remain unchanged), the ratio of unknown costs elicited roughly reflects the ratio of search space explored. Therefore, as this ratio decreases with $i$, so does the runtime of LKC and LUC. The exception to this observation is COM, whose runtimes increase slightly from when $i = 0.2$ to $i = 0.4$, after which it plateaus. However, the reason is similar to the one above – the ratio of unknown costs elicited compared to the total number of unknown costs increased slightly when $i = 0.2$ to $i = 0.4$ and remained relatively unchanged for larger values of $i$.
- Finally, COM is generally faster than both LKC and LUC across all parameters. The reason is because COM is able to prune a larger portion of the search space compared to LKC and LUC, as reflected by the observation that the number of unknown costs elicited by COM is smaller than that of LKC and LUC.

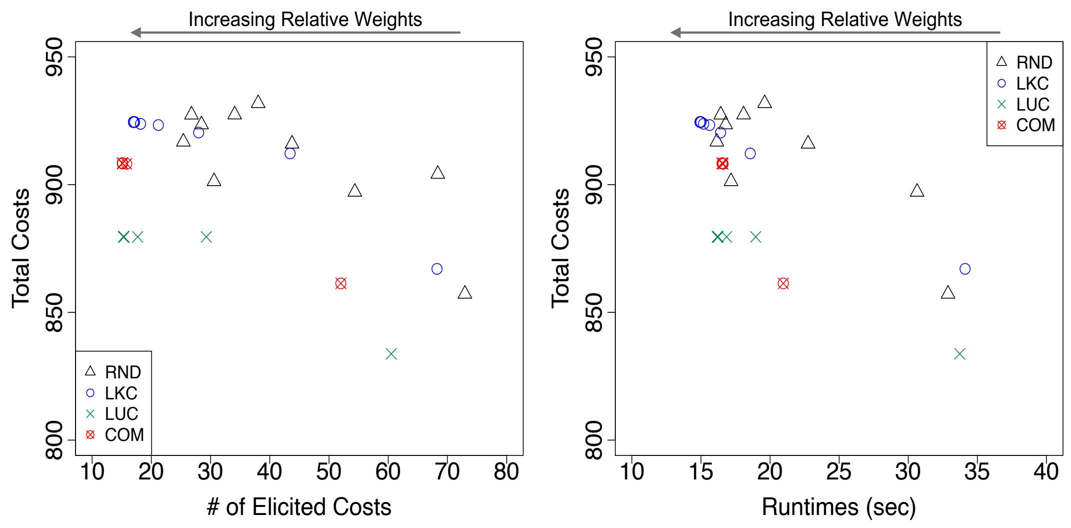
We now discuss the costs of the solutions found in terms of their cumulative constraint costs, cumulative elicitation costs, and their aggregated total costs:

- As expected, the cumulative elicitation cost of the solutions found, which is proportional to the number of unknown costs elicited, by all algorithms increase with increasing number of variables $|\mathcal{X}|$, constraint density $p_1$, and fraction of unknown costs $i$. The reason is that the number of unknown costs in the problem increases with increasing $|\mathcal{X}|$, $p_1$, and $i$.

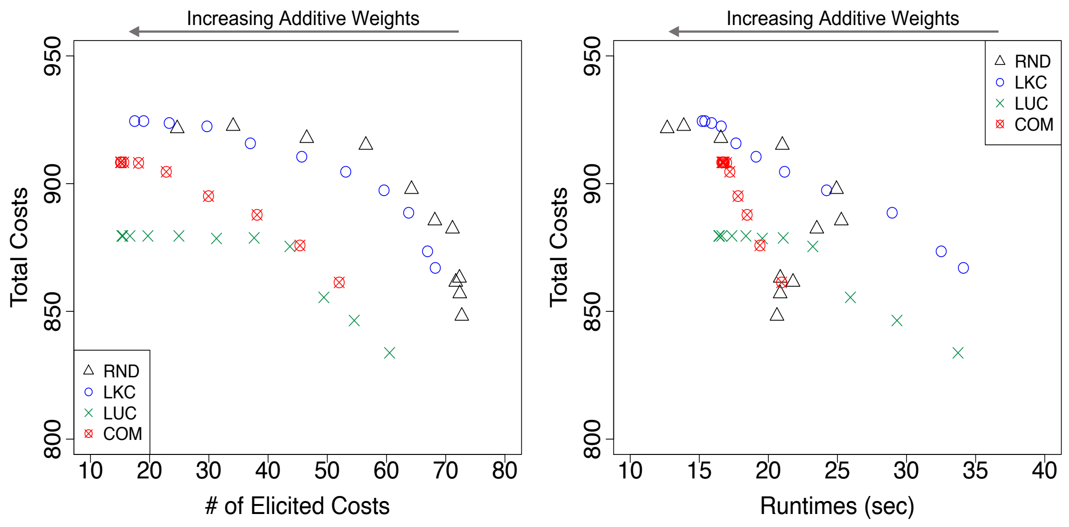
- The cumulative constraint cost of the solutions found by all algorithms also increase with increasing $|\mathcal{X}|$, $p_1$, and $i$. However, the reasons for why they increase is different: The constraint costs increase for increasing $|\mathcal{X}|$ and $p_1$ because the number of constraints in the problem increases. As such, the average constraint cost of solutions increases with increasing $|\mathcal{X}|$ and $p_1$.

  In contrast, the average constraint cost of solutions should remain relatively unchanged with increasing $i$ because the number of constraints remain unchanged. In this case, the constraint costs of solutions found increase because the algorithms find increasingly worse solutions with increasing $i$. As the algorithms do not know the unknown constraint costs until they are elicited, they will prefer to elicit unknown costs that have smaller elicitation costs. And since the elicitation cost and unknown constraint cost are not correlated, the behavior of the algorithms become increasingly random with increasing $i$, thereby resulting in solutions with larger constraint costs.
- Finally, since both the cumulative elicitation costs and cumulative constraint costs increase with increasing $|\mathcal{X}|$, $p_1$, and $i$, the aggregated total costs also increase similarly.
- In general, COM finds solutions with larger constraint costs than LKC and LUC. However, its elicitation costs are smaller than those of LKC and LUC. (Our results are statistically significant with $p < 0.05$). This implies that COM is able to find relatively good solutions quickly and uses that solution to prune a large portion of the search space. In contrast, LKC and LUC explores a larger portion of the search space to find better solutions, but at the cost of increasing elicitation cost. This behavior is also reflected in the runtimes of COM, which are smaller than the runtimes of LKC and LUC. The total cost of solutions found by all three algorithms are all approximately the same.

Figure 4.3 plots the empirical results for our IWCSP+EC experiments, where we vary the relative weight $w$ from 1 to 10 in increments of 1 and the additive weight $\epsilon$ from 0 to 1000



(a) Varying Relative Weight $w$

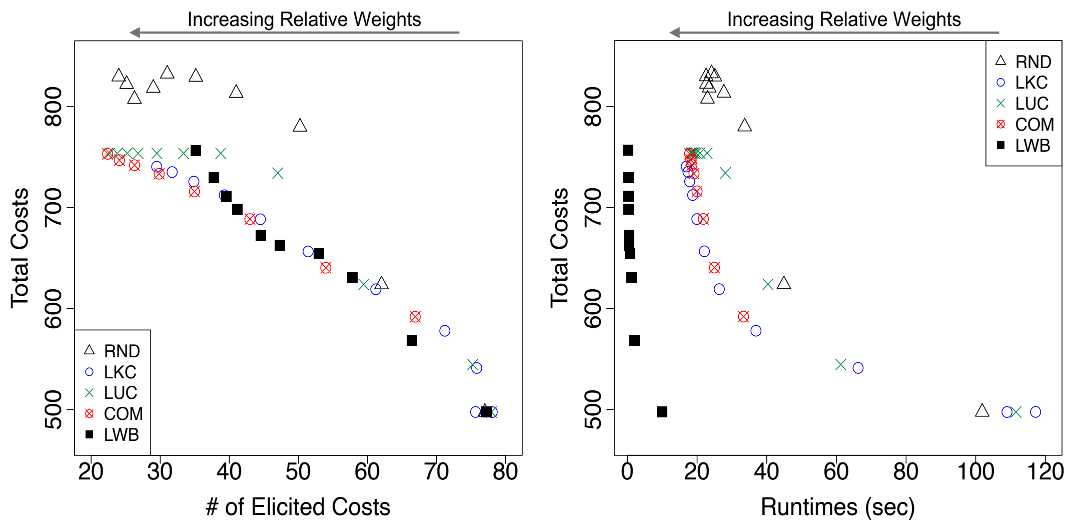


(b) Varying Additive Weight $\epsilon$

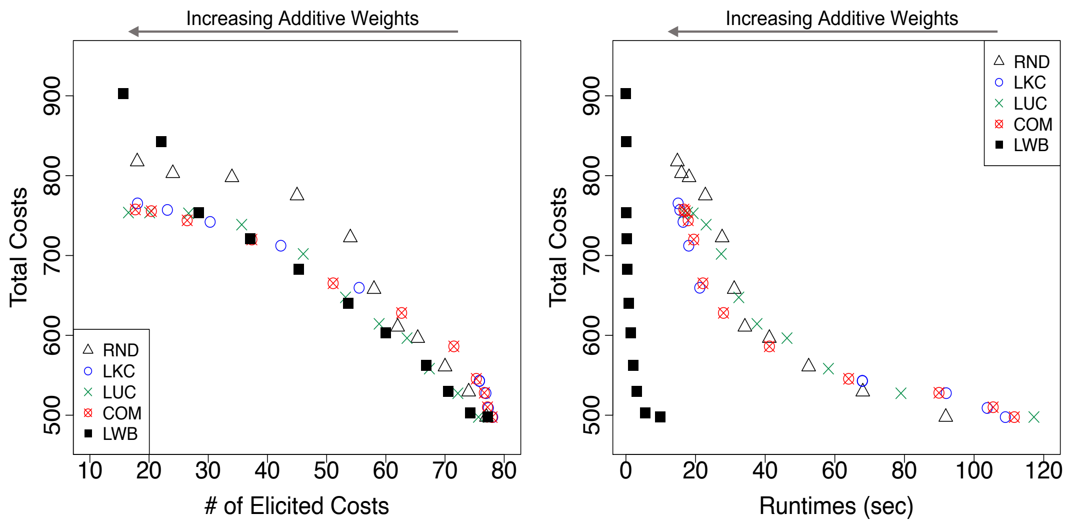
Figure 4.3: Random Graphs: IWCSP+EC Empirical Results with $|\mathcal{X}| = 10$, $p_1 = 0.4$, and $i = 0.6$

in increments of 100. Each data point in the figures thus shows the result for one of the heuristic with one of the values of $w$ or $\epsilon$. Data points for smaller values of $w$ and $\epsilon$ are in the bottom right of the figures and data points for larger values are in the top left of the figures.

We plot the tradeoffs between total costs (= cumulative constraint and elicitation costs) and number of elicited costs as well as the tradeoffs between total costs and runtimes. As



(a) Varying Relative Weight $w$



(b) Varying Additive Weight $\epsilon$

Figure 4.4: Random Graphs: IWCSP Empirical Results with $|\mathcal{X}| = 10$, $p_1 = 0.4$, and $i = 0.6$

expected, as the relative and additive weights increase, the total costs increase, the number of elicited costs decreases, and the runtimes decrease. The key difference between both weights is that the tradeoffs are much more uniform when using additive weights compared to relative weights. Therefore, using additive weights may allow users to better control the granularity of the tradeoffs. Compared to our three heuristics, the random heuristic performs poorly and randomly.

When comparing the tradeoffs between total costs and number of elicited costs, LUC is better than COM, which is better than LKC. In other words, to find solutions of the same cost, LUC elicits fewer costs than COM, which elicits fewer costs than LKC. However, when comparing the tradeoffs between total costs and runtimes, the same trend applies when the runtimes are small, but COM is better than LUC when the runtimes are large (larger than 20s in our experiments). Therefore, when the weights are sufficiently large, COM provides the best tradeoff between total costs and runtimes.

**IWCSPs:** We now present our empirical results for our IWCSP experiments. Note that these are problems without elicitation costs and the weights define the theoretical error bounds on the quality of solutions found in these problems (Theorems 1, 2, and 3).

While the original LWB algorithm does not allow users to define error bounds, it could be extended to do so since it uses its own specific heuristic function. We thus parameterize it the same way using Equation 5.1 and compare against it in our experiments below.

Figure 4.4 plots the empirical results, where we vary the relative weight $w$ and additive weight $\epsilon$. The key difference for these plots compared to the earlier ones is that the results for LWB is also included. We make the following observations:

When comparing the tradeoffs between total costs and number of elicited costs, LUC is better than COM, which is better than LKC. In other words, to find solutions of the same cost, LUC elicits fewer costs than COM, which elicits fewer costs than LKC. However, when comparing the tradeoffs between total costs and runtimes, the same trend applies when the runtimes are small, but COM is better than LUC when the runtimes are large (larger than 20s in our experiments). Therefore, when the weights are sufficiently large, COM provides the best tradeoff between total costs and runtimes.

- Similar to the trends in IWCSP+EC problems, as the additive and relative weights increase (from the bottom right of the figures to the top left of the figures), the total costs of solutions increase, the number of elicited costs decreases, and the runtimes decrease. Additionally, the random heuristic also performs poorly in this setting as expected.
- When comparing the tradeoffs between total costs and number of elicited costs, COM and LKC behave similarly, and they are both better than LUC. Interestingly, this trend is the opposite of that in IWCSP+EC problems, where LUC was the best. The reason is that the heuristic function used by LUC to estimate the cost to complete the current partial solution is poor when elicitation costs are not taken into account. In such a case, the only contribution to the estimate is the number of yet-to-be elicited unknown costs, which is then multiplied by a lower bound $\mathcal{L}$ on the cost of all constraints. In our experiments, $\mathcal{L} = 2$, which is a poor estimate as the costs can be as large as 100.
- This trend is also repeated when comparing the tradeoffs between total costs and runtimes – to find solutions of the same cost, COM and LKC requires a smaller runtime than LUC.
- When searching for optimal solutions (i.e., $w = 1$ and $\epsilon = 0$), all three of our algorithms elicit similar number of costs compared to LWB. LWB is also about one order of magnitude faster than our three algorithms. The reason is because, unlike our algorithms, LWB does not use any heuristics to estimate the cost to complete the solution.
- When searching for suboptimal solutions (i.e., $w > 1$ or $\epsilon > 0$), all three of our algorithms find solutions of similar costs compared to LWB but with smaller number of elicited costs once $w$ is sufficiently large. They also find solutions with smaller costs but with similar number of elicited costs once $\epsilon$ is sufficiently large.

(a) Varying Number of Variables $|\mathcal{X}|$, $i = 0.6$

| $\lvert\mathcal{X}\rvert$ | # unknown | # of elicited costs | | | runtime (sec) | | | total cost | | | cumulative constraint cost | | | cumulative elicitation cost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | costs | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM |
| 2 | 8.00 | 3.51 | 3.74 | 2.25 | 0.01 | 0.01 | 0.01 | 294.58 | 302.93 | 273.78 | 220.12 | 225.22 | 226.54 | 74.46 | 77.72 | 47.24 |
| 3 | 12.00 | 5.37 | 6.94 | 3.17 | 0.01 | 0.02 | 0.02 | 457.47 | 491.07 | 414.92 | 344.82 | 348.52 | 347.78 | 112.65 | 142.55 | 67.15 |
| 4 | 16.00 | 6.58 | 10.06 | 4.03 | 0.07 | 0.19 | 0.08 | 640.94 | 722.07 | 590.52 | 493.89 | 497.78 | 498.75 | 147.05 | 224.28 | 91.78 |
| 5 | 20.00 | 8.37 | 13.28 | 4.97 | 0.51 | 1.76 | 0.57 | 754.18 | 862.73 | 684.67 | 573.22 | 574.81 | 574.42 | 180.96 | 287.92 | 110.25 |
| 6 | 24.00 | 10.11 | 17.06 | 5.95 | 4.38 | 17.29 | 4.34 | 911.89 | 1067.14 | 829.99 | 692.96 | 696.15 | 696.15 | 218.93 | 370.98 | 133.84 |

(b) Varying Fraction of Unknown Costs $i$, $|\mathcal{X}| = 5$

| $i$ | # unknown | # of elicited costs | | | runtime (sec) | | | total cost | | | cumulative constraint cost | | | cumulative elicitation cost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | costs | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM | LKC | LUC | COM |
| 0.2 | 5.00 | 4.62 | 4.64 | 4.12 | 0.46 | 3.94 | 0.53 | 618.67 | 660.66 | 651.38 | 517.84 | 558.07 | 561.45 | 100.84 | 102.58 | 89.93 |
| 0.4 | 10.00 | 6.98 | 7.96 | 4.73 | 0.47 | 3.13 | 0.54 | 698.28 | 745.37 | 674.69 | 549.25 | 572.77 | 571.86 | 149.02 | 172.60 | 102.83 |
| 0.6 | 20.00 | 8.37 | 13.28 | 4.97 | 0.51 | 1.76 | 0.57 | 754.18 | 862.73 | 684.67 | 573.22 | 574.81 | 574.42 | 180.96 | 287.92 | 110.25 |
| 0.8 | 25.00 | 8.85 | 15.47 | 5.15 | 0.52 | 1.47 | 0.58 | 773.75 | 915.20 | 695.97 | 583.63 | 583.25 | 582.22 | 190.12 | 331.95 | 113.75 |
| 1.0 | 30.00 | 5.30 | 14.03 | 5.00 | 0.47 | 0.83 | 0.58 | 622.66 | 805.35 | 616.47 | 504.20 | 504.20 | 504.20 | 118.46 | 301.15 | 112.27 |

Table 4.2: Smart Home Device Scheduling: IWCSP+EC Empirical Results

### 4.5.3 Smart Home Device Scheduling

We also evaluate our algorithms on smart home device scheduling problems (described in Chapter 3), where each home consists of a set of smart devices that can be turned on or off according to the preferences of the home occupants [109]. We assume that some of the preferences are unknown to the system and associate an elicitation cost to the unknown preference so that important devices to the home occupants have smaller elicitation costs.

We generate 100 instances where we vary the number of devices (variables) $|\mathcal{X}|$ from 2 to 6 at each home; the fraction of unknown costs $i$ in each constraint from 0.2 to 1.0; the user-defined relative weight $w$ from 1 to 5; and the user-defined additive weight $\epsilon$ from 0 to 500. All elicitation costs are randomly sampled from $[2, 30]$. The constraints are set to be unary, and the domain size $|D_i|$ for all variables $x_i \in \mathcal{X}$ is set to 6. To generate the constraint costs and preferences we follow our experimental setup described in Chapter 3. Table 4.2 shows the empirical results, where the trends are mostly similar to those in random graphs.

## 4.6 Related Work

As our work lies in the intersection of constraint-based models, preference elicitation, and heuristic search, we will first focus on related work in this intersection before covering the three broader areas. The body of work that is most related to ours is the work by Gelain *et al.* [35], where they introduced IWCSPs as well as a family of DFBnB-based algorithms to solve them. They parameterized their algorithm based on *what* preferences must be elicited, *when* the elicitation should take place, and *who* decides the value ordering followed by the algorithm, resulting in 32 combinations of parameter values. They found that the combination that works best is LU.WW.BRANCH. In this combination, preferences with the worst unknown costs are elicited first (what = WW), preferences are elicited once the search reaches a leaf of the search tree (when = BRANCH), and the value ordering is based on a lazy user who prefers values with smaller costs without considering the constraints involving the current variable (who = LU). We compare against this algorithm in our empirical evaluations in the next section. In addition to Incomplete WCSPs, Gelain *et al.* [35] also introduced *Incomplete Fuzzy CSPs* and generalized both models to *Incomplete Soft CSPs.* Their parameterized algorithms described above are also generalized to solve the general problem.

Another body of work in this intersection is the use of weighted heuristics in *Distributed Constraint Optimization Problems* (DCOPs), which can be viewed as decentralized versions of WCSPs [29, 76, 86, 139]. For example, additive and relative weights were introduced for some algorithms [76, 136, 137], which provide additive and relative quality guarantees as those in Theorem 1. Similarly, relative weights were also introduced for ADOPT and other search-based algorithms [137], which provide relative quality guarantees as those in Theorem 1.

Finally, *Conditional-Preference Networks* (CP-nets) [7, 97] also lie in this intersection. CP-nets are a graphical representation model for qualitative preferences and reflects conditional dependencies between sets of preference statements. In contrast, IWCSPs focuses more on the notion of *conditional additive independence* [4], which requires that the cost of an outcome to be the sum of the "costs" of the different variable values of the outcome.

In the context of the broader constraint-based models where constraints may not be fully specified, there are a number of such models, including *Uncertain CSPs* [142], where the outcomes of constraints are parameterized; *Open CSPs* [25], where the domains of variables and constraints are incrementally discovered; *Dynamic CSPs* [22, 129], where the CSP can change over time; as well as distributed variants of these models [49, 50, 64, 81, 87, 88, 138].

In the context of the broader preference elicitation area, there is a very large body of work [41], and we focus on those that are most closely related to our work. They include techniques that ask users a number of preset questions [109, 121] and send alerts and notification messages to interact with users [18], techniques that ask users to rank alternative options or user-provided option improvements to learn a (possibly approximately) user preference function [8, 13, 116, 126], and techniques that associate costs to eliciting preferences and takes these costs into account when identifying which preference to elicit as well as when to stop eliciting preferences (e.g., when the cost outweighs the expected gain in utility from eliciting any preference) [63, 122]. The key difference between these approaches and ours is that they identify preferences to elicit a priori before the search while we interleave preference elicitation and search.

Finally, in the context of the broader heuristic search area, starting with Weighted A* [91], researchers have long used weighted heuristics to speed up the search process in general search problems. Often, solutions found by these weighted approaches also have similar quality guarantees as those in Theorem 1. Researchers have also investigated the use of dynamically-changing weights [92, 105]; using weighted heuristic with other heuristic search algorithms like DFBnB [32], RBFS [54], and AND/OR search [70, 71]; as well as extending them to provide anytime characteristics [43, 66].

## 4.7 Conclusions

*Incomplete Weighted Constraint Satisfaction Problems* (IWCSPs) are an elegant paradigm for modeling combinatorial optimization problems with partially-specified constraints. To fully specify such constraints, one must elicit preferences of users. Unfortunately, existing IWCSP approaches assume that the elicitation of preferences does not incur any additional cost. This is unrealistic as human users are likely bothered by repeated elicitations and will refuse to provide an unbounded number of preferences.

To overcome this limitation, we proposed the *IWCSP with Elicitation Costs* (IWCSP+EC) model, which extends the IWCSP model to include the notion of *elicitation costs.* The objective in this problem is to find a solution that minimizes the sum of both the constraint costs and elicitation costs. We also introduced three *parameterized* heuristics – Least Unknown Cost (LUC), Least Known Cost (LKC) and their combination heuristic (COM) – that allow users to trade off solution quality for fewer elicited preferences and faster computation times. Further, in settings where elicitations are free, these heuristics also provide *theoretical quality guarantees* on the solutions found.

Our empirical results show that COM finds solutions with larger constraint costs than LKC and LUC, but finds them faster and with fewer elicitations than LKC and LUC. Therefore, COM is the preferred heuristic in critical time-sensitive domains. COM also does a better job at trading off solution quality for smaller runtimes, especially when runtimes are large, through the use of user-defined weights. In conclusion, our heuristics improve the practical applicability of IWCSPs as they now not only take into account elicitation costs but also provide control knobs, in the form of user-defined weights, to perform tradeoffs along three key dimensions – solution quality, runtime, and number of elicited preferences.

So far, to address the key drawback of constraint-based models – a priori knowledge on all constraints – we have only introduced IWCSP+EC, where problems such as scheduling of smart devices within a smart home are modeled in a centralized approach. As there are many naturally distributed problems, in the next chapter, we will introduce a novel framework and algorithms that are more suited to formalize such distributed problems, where constraints encode users' preferences and are allowed to be partially specified. An example of such distributed problems is the problem of distributed meeting scheduling in which not all users' preferences of all participants are specified a priori. The next chapter introduces the *Incomplete Distributed Constraint Optimization Problems* (I-DCOPs) framework and algorithms to solve such problems with elicitation strategies in more detail.

# Chapter 5

# Intra-execution Elicitation Approach in Distributed Constraint-Based Models

To address the key drawback in distributed constraint-based models, we assume that constraint costs (i.e., preferences) are allowed to be unknown a priori. Thus, we extend the distributed constraint-based models under the assumption that some of the preferences are not specified. The unknown preferences will be elicited during the distributed search for optimal/sub-optimal solutions. Revealing the actual values of the unknown preferences throughout the search will incur cumulative elicitation costs. In this chapter, we introduce the incomplete constraint-based models and preference elicitation strategies, where the objective is to find the solutions that minimize both constraint and elicitation costs in a distributed environment. In what follows, we briefly introduce the intuition behind our model and motivate our approach with a real-world application, then we will discuss our model, algorithms and elicitation heuristic strategies, and our evaluation analyses in more detail.

## 5.1 Introduction

The *Distributed Constraint Optimization Problem* (DCOP) [76, 86] formulation is a powerful tool to model cooperative multi-agent problems. DCOPs are well-suited to model many problems that are distributed by nature, where agents need to coordinate their value assignments to minimize the aggregated constraint costs. This model is widely employed for representing distributed problems such as meeting scheduling [69], sensor and wireless networks [27, 139], multi-robot teams coordination [147], smart grids [73], and smart homes [31, 98, 112], coalition structure generation [123].

The study and use of DCOP have matured significantly over more than a decade since its inception [76]. DCOP researchers have proposed a wide variety of solution approaches, from complete approaches that use distributed search-based techniques [76, 136] to distributed inference-based techniques [86]. There is also a significant body of work on incomplete methods that can be similarly categorized into local search based methods [27], inference GDL-based techniques [128], and sampling-based methods [84]. Researchers have also proposed the use of other off-the-shelf solvers such as logic programming solvers [61, 62] and mixed-integer programming solvers [46].

One of the core limitations of all these approaches is that they assume that the constraint costs in a DCOP are specified or known a priori. In some applications, such as meeting scheduling problems, constraints encode the preferences of human users. As such, some of the constraint costs may be unspecified and must be elicited from human users.

To address this limitation, researchers have proposed the *preference elicitation problem for DCOPs* [107, 109]. In this preference elicitation problem, some constraint costs are initially unknown, and they can be accurately elicited from human users. The goal is to identify which subset of constraints to elicit in order to minimize a specific form of expected error in solution quality. Unfortunately, it suffers from two limitations: First, it assumes that the cost of eliciting constraints is uniform across all constraints, which is unrealistic as providing the preferences for some constraints may require more cognitive effort than the preferences for other constraints. Second, it decouples the elicitation process from the DCOP solving process since the elicitation process must be completed before one solves the DCOP with elicited constraints. As both the elicitation and solving process are actually coupled, this two-phase decoupled approach prohibits the elicitation process from relying on the solving process.

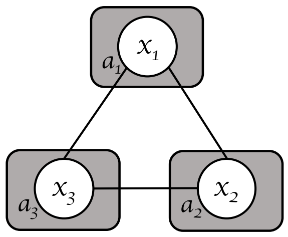
Therefore, in this chapter, we propose the *Incomplete DCOP* (I-DCOP) model, which *integrates* both the elicitation and solving problems into a single integrated optimization problem. In an I-DCOP, some constraint costs are unknown and can be elicited. Elicitation of unknown constraint costs will incur elicitation costs, and the goal is to find a solution that minimizes the sum of constraint and elicitation costs incurred. To solve this problem, we adapt a complete algorithm – Synchronous Branch-and-Bounds (SyncBB) [48] – and an incomplete algorithm – an *Anytime Local Search* (ALS) [146] variant of *Maximum Gain Message* (MGM) [68], which we call ALS-MGM. We also introduce parameterized heuristics that can be used by SyncBB and ALS-MGM to trade off solution quality for faster runtimes and fewer elicitations, and provides quality guarantees for I-DCOPs without elicitation costs when the underlying DCOP algorithm is correct and complete.

## 5.2 Motivation

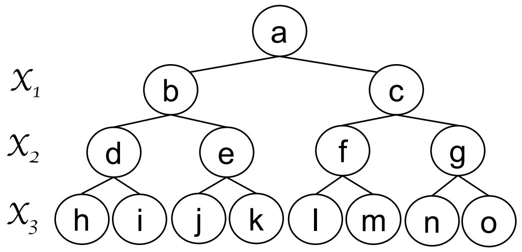
In a *distributed meeting scheduling problem*, an organization wishes to schedule a set of meetings in a distributed manner, where meeting participants have constraints for the different time slots that they are available as well as preferences over those time slots. This problem has been one of the first and more popular motivating applications for DCOPs since its inception [69, 86, 136]. While there are a number of possible formulations, we use the *Private Events as Variables* (PEAV) formulation proposed by Maheswaran *et al.* [69] in this chapter. In the PEAV formulation, the agents are meeting participants, their variables correspond to the different meetings that they must attend, and their values correspond to the different time slots of the meetings.[1] Equality constraints are imposed on variables of all agents involved in the same meeting – this enforces that all participants of a meeting agree on the time of that meeting – and inequality constraints are imposed on all variables of a single agent – this enforces that each participant cannot attend two meetings at the same time. Finally, unary constraints are imposed on each of the agent's variables where the costs correspond to the preferences of the participant on the different time slots.

To solve this problem, existing work has assumed that all the costs of such constraints are all known [69, 86, 136]. However, since these costs correspond to preferences of human users, it is unrealistic to assume that all the preferences are known a priori. These unknown preferences must thus be elicited if they are needed. Further, the elicitation of such preferences will incur elicitation costs that correspond to the degree at which a user is bothered by the elicitation process. As the existing canonical DCOP model is unable to capture these two features, we describe in the next section a DCOP extension that models unknown constraint costs that must be elicited as well as the cost of performing such elicitations.

[1]The description in this section assumes that each agent can control multiple variables.



(a) Constraint Graph



(b) Labels

| $x_1$ | $x_2$ | $\tilde{f}_1$ | $f_1$ | $e_1$ |
|---|---|---|---|---|
| 0 | 0 | ? | 1 | 3 |
| 0 | 1 | ? | 2 | 2 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

| $x_1$ | $x_3$ | $\tilde{f}_2$ | $f_2$ | $e_2$ |
|---|---|---|---|---|
| 0 | 0 | ? | 3 | 1 |
| 0 | 1 | ? | 3 | 1 |
| 1 | 0 | ? | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| $x_2$ | $x_3$ | $\tilde{f}_3$ | $f_3$ | $e_3$ |
|---|---|---|---|---|
| 0 | 0 | 3 | 3 | 0 |
| 0 | 1 | 4 | 4 | 0 |
| 1 | 0 | ? | 1 | 1 |
| 1 | 1 | ? | 2 | 1 |

(c) Incomplete Constraint Costs and Elicitation Costs

Figure 5.1: Example of an Incomplete DCOP with Elicitation Costs with its Labeled Search Tree

## 5.3 Incomplete Distributed Constraint-Based Model

We extend the distributed constraint-based model to *Incomplete DCOPs* (I-DCOPs), where some constraints can be partially specified. User preferences for these partially-specified constraints can be elicited during the execution of I-DCOP algorithms, but they incur some elicitation costs. Additionally, we extend SyncBB, a complete DCOP algorithm, and ALS-MGM, an incomplete DCOP algorithm, to solve I-DCOPs. We also propose parameterized heuristics that can be used by those algorithms to trade off solution quality for faster runtimes and fewer elicitations.

### 5.3.1 Incomplete DCOPs with Elicitation Costs

An *Incomplete DCOP* (I-DCOP) extends a DCOP by allowing some constraints to be partially specified. It is defined by a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \tilde{\mathcal{F}}, \mathcal{E}, \alpha \rangle$, where $\mathcal{A}$, $\mathcal{X}$, $\mathcal{D}$, $\mathcal{F}$, and $\alpha$ are exactly the same as in a DCOP. There are two key differences:

- The set of constraints $\mathcal{F}$ are not known to agents in an I-DCOP. Instead, only the set of partially-specified constraints $\tilde{\mathcal{F}} = \{\tilde{f}_i\}_{i=1}^m$ are known. Each partially-specified constraint is a function $\tilde{f}_i : \prod_{x \in \mathbf{x}^{f_i}} D_x \rightarrow \mathbb{R} \cup \{\infty, ?\}$, where ? is a special element denoting that the cost for a given combination of value assignment is not specified. The costs $\mathbb{R} \cup \{\infty\}$ that are specified are exactly the costs of the corresponding constraints $f_i \in \mathcal{F}$.
- $\mathcal{E} = \{e_i\}_{i=1}^m$ is the set of elicitation costs, where each elicitation cost $e_i : \prod_{x \in \mathbf{x}^{f_i}} D_x \rightarrow \mathbb{R}$ specifies the cost of eliciting the constraint cost of a particular ? in $\tilde{f}_i$.

An *explored solution space* $\tilde{\mathbf{x}}$ is the union of all solutions explored so far by a particular algorithm. The *cumulative elicitation cost* $\mathcal{E}(\tilde{\mathbf{x}}) = \sum_{e \in \mathcal{E}} e(\tilde{\mathbf{x}})$ is the sum of the costs of all elicitations conducted while exploring $\tilde{\mathbf{x}}$.

The *total cost* $\mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}}) = \alpha_f \cdot \mathcal{F}(\mathbf{x}) + \alpha_e \cdot \mathcal{E}(\tilde{\mathbf{x}})$ is the weighted sum of both the cumulative constraint cost $\mathcal{F}(\mathbf{x})$ of solution $\mathbf{x}$ and the cumulative elicitation cost $\mathcal{E}(\tilde{\mathbf{x}})$ of the explored solution space $\tilde{\mathbf{x}}$, where $\alpha_f \in (0, 1]$ and $\alpha_e \in [0, 1]$ such that $\alpha_f + \alpha_e = 1$. The weights represent the tradeoffs between the importance of solution quality and the cumulative elicitation cost.

The goal is to find an optimal complete solution $\mathbf{x}^*$ while eliciting only a cost-minimal set of preferences from a solution space $\tilde{\mathbf{x}}^*$. More formally, the goal is to find $(\mathbf{x}^*, \tilde{\mathbf{x}}^*) = \operatorname{argmin}_{(\mathbf{x}, \tilde{\mathbf{x}})} \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}})$.

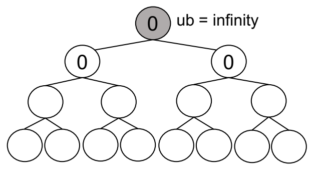
Figure 5.1(a) shows the constraint graph of an example I-DCOP that we will use as a running example in this section. It has three variables $x_1$, $x_2$, and $x_3$ with identical domains $D_1 = D_2 = D_3 = \{0, 1\}$. All three variables are constrained with one another and Figure 5.1(b) shows the partially-specified constraints $\tilde{f}_i$, their corresponding fully-specified constraints $f_i$, and the elicitation costs $e_i$. For simplicity, assume that $\alpha_f = \alpha_e = 0.5$ throughout this chapter. Therefore, in this example, the optimal complete solution is $\mathbf{x}^* = \langle x_1 = 1, x_2 = 1, x_3 = 0\rangle$ and only that solution is explored (i.e., $\tilde{\mathbf{x}} = \mathbf{x}^*$). The constraint cost of that solution is 3 $(= f_1(\langle x_1 = 1, x_2 = 1\rangle) + f_2(\langle x_1 = 1, x_3 = 0\rangle) + f_3(\langle x_2 = 1, x_3 = 0\rangle))$. The cumulative elicitation cost is 2 $(= e_2(\langle x_1 = 1, x_3 = 0\rangle) + e_3(\langle x_2 = 1, x_3 = 0\rangle))$. Thus, the total cost is $\alpha_f \cdot 3 + \alpha_e \cdot 2 = 0.5 \cdot 3 + 0.5 \cdot 2 = 2.5$.

## 5.4 Resolution Algorithms and Heuristics

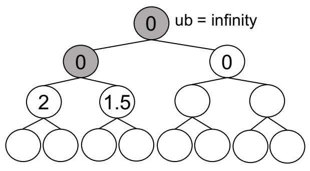
To solve I-DCOPs, one can easily adapt existing DCOP algorithms by allowing them to elicit unknown costs whenever those costs are needed by the algorithm. We describe below how to adapt SyncBB, a complete search algorithm, and ALS-MGM, a variant of the MGM [68] local search algorithm using the ALS framework [146], to solve I-DCOPs. We will use also SyncBB and ALS-MGM as the underlying algorithms that use our proposed heuristics later.

### 5.4.1 Synchronous Branch-and-Bound Algorithm

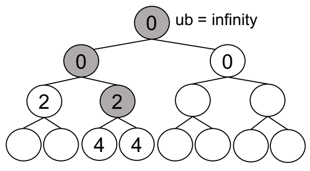
The operations of SyncBB can be visualized with search trees. Figure 5.1(c) shows the search tree for our example I-DCOP shown in Figures 5.1(a) and Figures 5.1(b), where levels 1, 2, and 3 correspond to variable $x_1$, $x_2$, and $x_3$, respectively. Left branches correspond to the variable being assigned the value 0 and right branches correspond to the variable being assigned the value 1. Each non-leaf node thus corresponds to a partial solution and each leaf



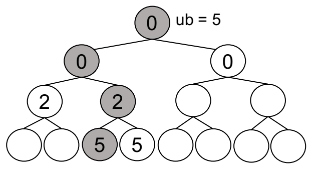
(a) Step 1



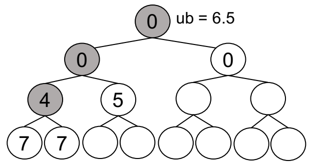
(b) Step 2



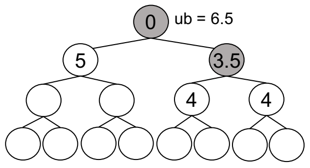
(c) Step 3



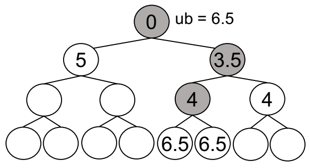
(d) Step 4



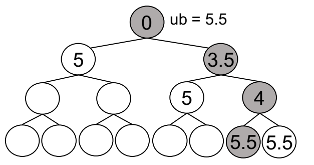
(e) Step 5



(f) Step 6



(g) Step 7



(h) Step 8

Figure 5.2: Simplified Execution Trace of SyncBB Without Heuristics

node corresponds to a complete solution. These nodes also correspond to unique CPAs of agents when they run SyncBB. We label each node of the search tree with an identifier so that we can refer to them easily below.

When SyncBB evaluates a node $n$ after exploring search space $\tilde{\mathbf{x}}$, it considers only the cumulative elicitation cost so far $\mathcal{E}(\tilde{\mathbf{x}})$ and the constraint costs of the CPA at node $n$, which we will refer to as $g$-values (defined in Chapter 4, Definition 7), denoted by $g(n)$. We refer to the weighted sum of these values as $f$-values (defined in Chapter 4, Definition 9), denoted by $f(n, \tilde{\mathbf{x}}) = \alpha_f \cdot g(n) + \alpha_e \cdot \mathcal{E}(\tilde{\mathbf{x}})$.

Assume that all the agents know that there is a lower bound $\mathcal{L}$ on all the constraint costs. Before calculating $f(n, \tilde{\mathbf{x}})$ at node $n$, the algorithm *estimates* the total cost (i.e., constraint cost + elicitation cost) by replacing unknown constraint costs with $\mathcal{L}$ and summing them up with the elicitation cost thus far. If the estimated total cost is no smaller than the cost of the best solution found so far, SyncBB prunes node $n$. Otherwise, it elicits the unknown costs of node $n$ and calculates its true total cost. By estimating the total costs, SyncBB only elicits unknown constraints when their costs are needed.

Figure 5.2 shows a simplified execution trace of SyncBB, where the CPA at each step of the algorithm corresponds to the shaded branch of search tree. For example, the CPA is $\langle x_1 = 1, x_2 = 0\rangle$ in Step 7. The numbers in the shaded nodes correspond to their $f$-values when those nodes were expanded by SyncBB. Unshaded nodes with numbers correspond to nodes whose estimated weighted $f$-values are maintained by the agents. For example, agent $a_3$ maintains estimated weighted $f$-values of nodes $l$ and $m$ in Step 7 as it needs to figure out whether those nodes should be expanded or pruned in that step. The upper bound $ub$, which corresponds to the cost of the best complete solution found so far, is shown in the figure beside the root node.

We now describe the trace in more detail. The root agent $a_1$ first expands node $a$ followed by node $b$ in Steps 1 and 2. This is done when it assigns its variable $x_1$ the value 0. It then sends a CPA with this value assignment as well as the cost of this CPA (= 0) and the cumulative elicitation cost so far (= 0) to its child $a_2$. Upon receipt of the message, agent $a_2$ needs to decide whether to expand nodes $d$ or $e$, which correspond to assigning its variable $x_2$ the values 0 or 1, respectively. If the cost of both partial solutions are known, then it should expand the node with the smaller cost. However, in this case, the costs of both nodes are unknown. These costs correspond to the unknown constraints $\tilde{f}_1(\langle x_1 = 0, x_2 = 0\rangle)$ and $\tilde{f}_1(\langle x_1 = 0, x_2 = 1\rangle)$. Therefore, it estimates the cost of both nodes and expand the node with the smallest estimated cost.

Let's assume the lower bound $\mathcal{L} = 1$ on all the constraint costs. Using this knowledge, agent $a_2$ computes an *estimated* cost of node $d$ to be 2, which is the sum of the cost of the received CPA (= 0), the cumulative elicitation cost so far (= 0), the lower bound on the constraint cost ($= \alpha_f \cdot 1$), and the weighted elicitation cost ($= e_1(\langle x_1 = 0, x_2 = 0\rangle) = \alpha_e \cdot 3$). Similarly, the agent computes the estimated cost of node $e$ to be 1.5. These costs are shown in the corresponding nodes in Step 2.

The estimated cost of node $e$ is the smaller of the two. So, in Step 3, agent $a_2$ expands node $e$ by eliciting the unknown cost $f_1(\langle x_1 = 0, x_2 = 1\rangle) = 2$, updating the weighted cumulative elicitation cost to $\alpha_e \cdot 2 = 1$, updating the cost of node $e$ to 2 (= cost of received CPA of 0 + weighted constraint cost of 1 + weighted cumulative elicitation cost of 1), updating its CPA to include this new value assignment, and sending the updated CPA together with the weighted cost of the CPA (= 1) and the weighted cumulative elicitation cost so far (= 1) to its child $a_3$.

Upon receipt of this message, agent $a_3$ needs to decide whether to expand nodes $j$ or $k$. Using the same rationale as above, in Step 4, agent $a_3$ expands node $j$ by eliciting the unknown costs $f_2(\langle x_1 = 0, x_3 = 0\rangle) = 3$ and $f_3(\langle x_2 = 1, x_3 = 0\rangle) = 1$, incurring a weighted total elicitation cost of $\alpha_e \cdot 2 = 1$, 0.5 for each elicitation. It then updates the weighted cumulative constraint cost to $\alpha_f \cdot 4 = 2$, updates the cost of node $j$ to 5 (= cost of received CPA of 2 + weighted constraint costs of 2 + weighted cumulative elicitation cost of 1), and updates its CPA to include this new value assignment. Since agent $a_3$ is a leaf node, it knows that its CPA is a *complete solution.* Since the cost of the solution is smaller than the upper bound, it updates the upper bound to 5.

Then, it evaluates node $k$ whether it should be expanded or pruned. Note that the estimated weighted $f$-value of the node increased from 4 in Step 3 to 5 in Step 4. The reason for this increase is because the weighted cumulative elicitation cost increased by 1 between Steps 3 and 4. Since the weighted estimated $f$-value of node $k$ is no smaller than the upper bound, agent $a_3$ prunes this node and backtracks to its parent $a_2$ by sending a BACKTRACK message that contains its best complete solution, the weighted cumulative constraint cost of that solution, and the weighted cumulative elicitation cost so far.

Upon receipt of the BACKTRACK message, agent $a_2$ then updates its weighted cumulative elicitation cost to 2 based on the cost received in the message, and updates the weighted estimated $f$-value of node $d$ to 4 (= constraint cost of 0 + weighted lower bound of 0.5 + weighted elicitation cost of 1.5 from $e_1(\langle x_1 = 0, x_2 = 0 \rangle)$ + weighted cumulative elicitation cost of 2). If this cost is no smaller than the upper bound in the message, it will prune this branch and backtrack. Since the cost is smaller, it will expand the node by eliciting the unknown cost, updates its CPA to include this new value assignment, and sends the updated CPA together with the weighted cost of the CPA and the weighted cumulative elicitation cost so far to its child $a_3$. Since the estimated costs of both nodes $h$ and $i$ are no smaller than the upper bound, the algorithm prunes the branches and backtracks to agent $a_1$. Then it updates the weighted cost of the CPA to 6.5 and the weighted cumulative elicitation cost so far to 3.5. The algorithm in Step 8 finds a solution with a cost smaller than the current upper bound 6.5. Thus, it updates the upper bound to 5.5. The process continues until the root agent backtracks and returns the best complete solution found.

### 5.4.2 Anytime Local Search - Maximum Gain Message Algorithm

Just like for regular DCOPs, ALS-MGM here also uses a *Breadth-First Search spanning tree* (BFS-tree) of the constraint graph to aggregate costs up the tree to the root agent such that it is able to detect when a better solution is found. When such a solution is found, the root agent propagates the step number in which that solution is found down to its descendants. Therefore, upon termination, all the agents have a consistent view on when the best solution is found and take on their corresponding values.

In more detail, Algorithm 1 presents the pseudo-code for each agent in ALS-MGM. In lines 1-7, each agent initializes the parameters required by the ALS framework. The root agent initializes an additional parameter to keep track of the cost of the best solution. In line 8, each agent chooses a value randomly or based on some heuristic (we introduce one such heuristic in Section 5.4.3). Every agent performs $m + d + h$ synchronous steps (line 9), where $m \geq d + h$ is the number steps that a regular MGM algorithm would run, $d$ is the distance between the root and the agent, $h$ is the height of the agent in the BFS-tree and, thus, $d + h$ is the height of the BFS-tree. Since the ALS framework requires that $m \geq d + h$, it runs an additional $d + h$ steps compared to MGM because those additional steps are needed to ensure that the best solution found is propagated by the root to every agent.

In lines 10-19, each agent sends messages, calculates the cost $cost_{step}$ of its current assignments $value_{step}$ given the assignments of its neighbors, calculates the best assignment *best_value* given the assignments of its neighbors and the corresponding cost $cost_{new}$, and calculates its loss $loss_{step}$, given the corresponding costs $cost_{new}$. The root agent checks if the cost it calculated is smaller than the best cost found thus far *best_cost*. If so, it saves or updates the cost, value, and index of the best assignment so far. A non-root agent checks whether the best index from its parent is new. If so, it updates the best index *best_index* and value assignment *best* of the step of the best assignment (lines 20-27). In lines 28-30, each agent checks its own calculated loss and those that it received from the neighbors. If the agent's loss is the smallest among all the neighbors' losses, then the agent changes its current value *value_step* to its best value *best_value* given the assignments of its neighbors and elicits any unknown constraint costs involved with that value.

In lines 31-32, each agent deletes information that no longer needs to keep since the information has been propagated to the root, and if the best solution is found during one of those previous steps, the root would have propagated that information to the agent. In line 33, each agent

**Algorithm 1:** Pseudo-code for each ALS-MGM Agent

```
h ← height of the agent in the BFS-tree
d ← distance of the agent from the root
best ← null
best_index ← null
step ← 0
if the agent is the root node in the BFS-tree then
    best_cost ← $\infty$
end
$value_{step}$ ← a random value from the agent's domain
while $step < m$ do
    send $value_{step}$, $loss_{step}$, and $cost_{step}$ to parent
    send $value_{step}$ and $loss_{step}$ to non-tree neighbors
    send $value_{step}$, $loss_{step}$ and best_index to children
    wait to receive messages from neighbors
    $cost_{step}$ ← estimated cost with $value_{step}$
    for all values d from the agent's domain do
        $cost(d)$ ← estimated cost with d
    end
    best_value ← $\text{argmin}_d\, cost(d)$
    $cost_{new}$ ← estimated cost with best_value
    $loss_{step}$ ← ($cost_{new}$ - $cost_{step}$)
    if the agent is the root node in the BFS-tree then
        if $cost_{step}$ < best_cost then
            best_cost ← $cost_{step}$
            best ← $value_{step}$
            best_index ← step
        end
    end
    if message from parent includes new best_index j then
        best ← $value_j$
        best_index ← j
    end
    if $loss_{step} \leq$ losses of all neighbors then
        $value_{step}$ ← best_value
        Elicit unknown constraints costs
    end
    Delete $value_{(step-2*d)}$
    Delete $cost_{(step-h)}$
    step ++
end
```

```
for $d + h$ iterations do
    wait to receive messages from parent
    if message from parent includes a new $best\_index$ $j$ then
        $best \leftarrow value_j$
        $best\_index \leftarrow j$
    send $best\_index$ to children
```

increments its step counter and this process repeats until it has ran for $m$ steps. Then, each agent performs $d + h$ additional steps to propagate the index of the best step down the tree (lines 34-39).

Taking our example I-DCOP in Figure 5.1, the BFS-tree is similar to its constraint graph shown in Figure 5.1(a), when $x_1$ is the root, and the height of the tree is 1. The height and distance parameters in $x_2$ and $x_3$ are initialized to $h = 0$ and $d = 1$, respectively. Let's assume that all agents choose value 1 for their variables and $m = 1$. In step 0 of the algorithm, all agents have already selected a value for their variables randomly. We let each agent elicit all unknown constraint that they are involved in *only* in step 0 to find the first complete solution. In this step, agent $a_3$ assigns its variable $x_3$ the value 1 and receives the values of its neighbors ($x_2 = 1$) and parent ($x_1 = 1$).

To calculate the cost of its assignment, it elicits the unknown cost $\tilde{f}_3(\langle x_2 = 1, x_3 = 1\rangle)$. With a weighted elicitation cost of $e_3(\langle x_2 = 1, x_3 = 1\rangle) = \alpha_e \cdot 1$, the total cost of the partial assignment is $f_2(\langle x_1 = 1, x_3 = 1\rangle) + f_3(\langle x_2 = 1, x_3 = 1\rangle) + e_3(\langle x_2 = 1, x_3 = 1\rangle) = \alpha_f \cdot (1 + 2) + \alpha_e \cdot 1 = 2$ (assuming $\alpha_f = \alpha_e = 0.5$). Then, it updates the cumulative elicitation cost to 0.5. Agent $a_3$ also calculates the value of improvement (loss) by checking if variable $x_3$ changes its value to 0 assuming all neighbors keep their current values. The total estimated cost of this change is $\tilde{f}_2(\langle x_1 = 1, x_3 = 0\rangle) + \tilde{f}_3(\langle x_2 = 1, x_3 = 0\rangle) + e_3(\langle x_2 = 1, x_3 = 0\rangle) = \alpha_f \cdot (1+1) + \alpha_e \cdot (1+1) = 2$, when all agents know that a lower bound $\mathcal{L}$ on all the constraint costs is 1. The loss at step

0 for agent $a_3$ is $2 - 2 = 0$. Similarly, in agent $a_2$, the total cost of the partial assignment is $f_1(\langle x_1 = 1, x_2 = 1\rangle) + f_3(\langle x_2 = 1, x_3 = 1\rangle) + e_3(\langle x_2 = 1, x_3 = 1\rangle) = \alpha_f \cdot (1 + 2) + \alpha_e \cdot 1 = 2$. The total estimated cost for agent $a_2$ changing to a new value is $f_1(\langle x_1 = 1, x_2 = 0\rangle) + f_3(\langle x_2 = 0, x_3 = 1\rangle) = \alpha_f \cdot (1 + 4) = 2.5$ and the loss is $2.5 - 2 = 0.5$. The root agent $a_1$ calculates the cost of its assignment based on the information received from its children (at step 0, the costs received from children are still 0). The total cost of the assignment $f_1(\langle x_1 = 1, x_2 = 1\rangle) + f_2(\langle x_1 = 1, x_3 = 1\rangle) = \alpha_f \cdot (1 + 1) = 1$. The total estimated cost for agent $a_1$ changing to a new value is $\tilde{f}_1(\langle x_1 = 0, x_2 = 1\rangle) + \tilde{f}_2(\langle x_1 = 0, x_3 = 1\rangle) + e_1(\langle x_1 = 0, x_2 = 1\rangle) + e_2(\langle x_1 = 0, x_3 = 1\rangle) = \alpha_f \cdot (1 + 1) + \alpha_e \cdot (2 + 1) = 2.5$, and the loss $2.5 - 1 = 1.5$.

In Step 1, the value message agents send to their parent and children include the cost calculation. Thus, the root agent has received the cost calculation from its children and can calculate the cost of a complete assignment $f_1(\langle x_1 = 1, x_2 = 1\rangle) + f_2(\langle x_1 = 1, x_3 = 1\rangle) + f_3(\langle x_2 = 1, x_3 = 1\rangle) = \alpha_f \cdot (1 + 1 + 2) + \alpha_e \cdot 1 = 2.5$. Since the calculated cost is smaller than initial value of the best cost, the root agent saves the information for this assignment as the best one found so far. In Step 2, all agents have received the best index from the root agent, where the best cost found so far is 2.5.

### 5.4.3 Heuristic Strategies

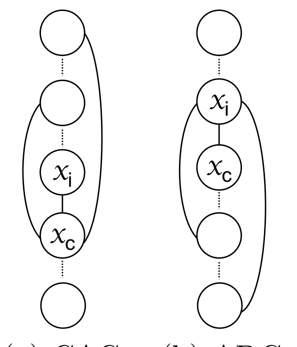
To speed up SyncBB, one can use *cost-estimate heuristics* $h(n)$ to estimate the sum of the constraint and elicitation costs needed to complete the CPA at a particular node $n$. And if those heuristics are underestimates of the true cost, then they can be used to better prune the search space, that is, when $f(n, \tilde{\mathbf{x}}) = \alpha_f \cdot g(n) + h(n) + \alpha_e \cdot \mathcal{E}(\tilde{\mathbf{x}}) \geq \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}})$, where $\mathbf{x}$ is the best complete solution found so far and $\tilde{\mathbf{x}}$ is the current explored solution space.

We now describe below a cost-estimate heuristic that can be used in conjunction with SyncBB to solve I-DCOPs. This heuristic makes use of an estimated lower bound $\mathcal{L}$ on the cost of all constraints $f \in \mathcal{F}$. Such a lower bound can usually be estimated through domain expertise. In the worst case since all costs are non-negative, for our running example we set the lower bound ($\mathcal{L}$) to 1. The more informed the lower bound, the more effective the heuristics will be in pruning the search space.

Additionally, this heuristic is parameterized by two parameters – a relative weight $w \geq 1$ and an additive weight $\epsilon \geq 0$. When using these parameters, SyncBB will prune a node $n$ if:

$$w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon \geq \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}}) \tag{5.1}$$

where $\mathbf{x}$ is the best complete solution found so far and $\tilde{\mathbf{x}}$ is the current explored solution space. Users can increase the weights $w$ and $\epsilon$ to prune a larger portion of the search space and, consequently, reduce the computation time as well as the number of preferences elicited. However, the downside is that it will also likely degrade the quality of solutions found. Further, in I-DCOPs where elicitations are free (i.e., the elicitation costs are all zero), we theoretically show that the cost of solutions found are guaranteed to be at most $w \cdot OPT + \epsilon$, where $OPT$ is the optimal solution cost.



(a) CAC (b) ADC

Figure 5.3: Constraints Estimated Directly by Agent $x_i$ for the CAC and ADC Heuristics

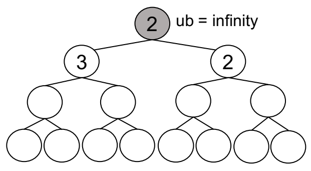
**Child's Ancestors' Constraints (CAC) Heuristic:** This heuristic is defined recursively from the leaf of the pseudo-chain (i.e., last agent in the variable ordering) used by SyncBB up to the root of the pseudo-chain (i.e., first agent in the ordering). Agent $x_i$ in the ordering computes a heuristic value $h(x_i = d_i)$ for each of its values $d_i \in D_i$ as follows: $h(x_i = d_i) = 0$ if $x_i$ is the leaf of the pseudo-chain. Otherwise:

$$
\begin{aligned}
h(x_i{=}d_i) = &\\
&\min_{d_c \in D_c}\Big[\alpha_f \cdot \hat{f}(x_i{=}d_i, x_c{=}d_c) + \alpha_e \cdot e(x_i{=}d_i, x_c{=}d_c) + h(x_c{=}d_c)\Big]\\
&+ \sum_{x_k \in Anc(x_c)\setminus\{x_i\}} \min_{d_k \in D_k}\Big[\alpha_f \cdot \hat{f}(x_c{=}d_c, x_k{=}d_k) + \alpha_e \cdot e(x_c{=}d_c, x_k{=}d_k)\Big]
\end{aligned}
\tag{5.2}
$$

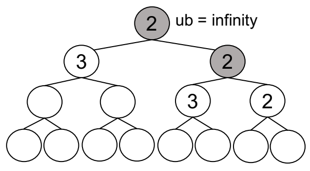
where $x_c$ is the next agent in the ordering (i.e., child of $x_i$ in the pseudo-chain), $Anc(x_c)$ is the set of variables higher up in the ordering that $x_c$ is constrained with, and each estimated cost function $\hat{f}$ corresponds exactly to a partially-specified function $\tilde{f}$, except that all the unknown costs ? are replaced with the lower bound $\mathcal{L}$. Therefore, the estimated cost $\hat{f}(\mathbf{x})$ is guaranteed to be no larger than the true cost $f(\mathbf{x})$ for any solution $\mathbf{x}$.

For a parent $x_p$ of a leaf agent $x_l$, the heuristic value $h(x_p{=}d_p)$ is then the minimal constraint and elicitation cost between the two agents, under the assumption that the parent takes on value $d_p$, and the sum of the minimal constraint cost of the leaf agent with its ancestors. As the heuristic of a child agent is included in the heuristic of the parent agent, this summation of costs are recursively aggregated up the pseudo-chain. Figure 5.3(a) illustrates an example pseudo-chain, where the constraints whose costs are *directly* estimated by agent $x_i$ to compute the CAC heuristic are bolded in solid lines. Constraints whose costs are *indirectly* estimated through heuristic values that are propagated up through the pseudo-chain are not shown.

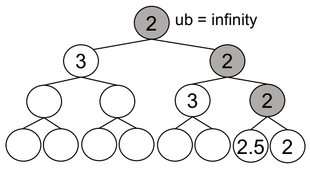
It is fairly straightforward to see that this heuristic can be computed in a distributed manner – the leaf agent $x_l$ initializes its heuristic values $h(x_l{=}d_l) = 0$ for all its values $d_l \in D_l$ and



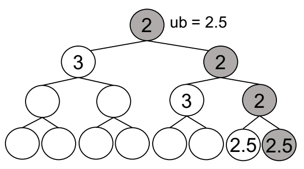
(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Figure 5.4: Simplified Execution Trace of SyncBB with CAC Heuristic

computes the latter term in Equation 5.2:

$$\sum_{x_k \in Anc(x_l)} \min_{d_k \in D_k} \left[ \alpha_f \cdot \hat{f}(x_l = d_l, x_k = d_k) + \alpha_f \cdot e(x_l = d_l, x_k = d_k) \right] \tag{5.3}$$

for each of its values $d_l \in D_l$. It then sends these heuristic values and costs to its parent. Upon receiving this message, the parent agent $x_p$ uses the information in the message to compute its own heuristic values $h(x_p = d_p)$ using Equation 5.2, computes the latter term similar to Equation 5.3 above, and sends these heuristic values and costs to its parent. This process continues until the root agent computes its own heuristic values, at which point it starts the SyncBB algorithm.

Figure 5.4 shows the order of node expansions conducted by SyncBB using the CAC heuristic on our example I-DCOP. Note that the algorithm needs to only expand 4 nodes and elicit 1 unknown constraint cost before returning a solution of weighted cost 2.5. In contrast, SyncBB without heuristics expanded 9 nodes elicited 4 unknown constraint costs before returning a solution of weighted cost 5.5 (see Figure 5.2).

**Agent's Descendants' Constraints (ADC) Heuristic:** Our second heuristic is called *Agent's Descendants' Constraints* (ADC) heuristic. Like the CAC heuristic, it is also defined recursively from the leaf of the pseudo-chain used by SyncBB up to the root of the pseudo-chain. Agent $x_i$ in the ordering computes a heuristic value $h(x_i = d_i)$ for each of its values

$d_i \in D_i$ as follows: $h(x_i = d_i) = 0$ if $x_i$ is the leaf of the pseudo-chain. Otherwise,

$$\begin{aligned} h(x_i{=}d_i) = & \\ & \min_{d_c \in D_c}\Big[\alpha_f \cdot \hat{f}(x_i{=}d_i, x_c{=}d_c) + \alpha_e \cdot e(x_i{=}d_i, x_c{=}d_c) + h(x_c{=}d_c)\Big] \\ & + \sum_{x_k \in Des(x_i)\setminus\{x_c\}} \min_{d_k \in D_k}\Big[\alpha_f \cdot \hat{f}(x_i{=}d_i, x_k{=}d_k) + \alpha_e \cdot e(x_i{=}d_i, x_k{=}d_k)\Big] \end{aligned} \tag{5.4}$$

where $x_c$ is the next agent in the ordering, $Des(x_i)$ is the set of variables lower down in the ordering that $x_i$ is constrained with, and each estimated cost function $\hat{f}$ is as defined for the CAC heuristic above. Figure 5.3(b) illustrates an example pseudo-chain, where the constraints whose costs are *directly* estimated by agent $x_i$ to compute the ADC heuristic are bolded in solid lines. Constraints whose costs are *indirectly* estimated through heuristic values that are propagated up through the pseudo-chain are not shown.

Like CAC, it is also straightforward to see that this heuristic can be computed in a distributed manner – the leaf agent $x_l$ initializes its heuristic values $h(x_l = d_l) = 0$ for all its values $d_l \in D_l$ and sends these heuristic values to its parent. Upon receiving this message, the parent agent $x_p$ uses the information in the message to compute its own heuristic values $h(x_p{=}d_p)$ using Equation 5.4 and sends them to its parent. This process continues until the root agent computes its own heuristic values, at which point it starts the SyncBB algorithm.

**Neighbors' Constraints (NHC) Heuristic:** Instead of having each agent in ALS-MGM choose its initial value randomly from its domains, one can also use cost-estimate heuristics to estimate costs for each value and have the agent choose the value that minimizes the estimated costs. Using cost-estimate heuristics helps ALS-MGM to find solutions with smaller costs faster since it starts with a better initial solution, which is more pronounced when there is not enough time to let the algorithm run until convergence. In NHC heuristic, each agent

$x_i$ computes a heuristic value for each of its values $d_i \in D_i$ as follows:

$$h(x_i = d_i) = \sum_{x_c \in Nh(x_i)} \min_{d_c \in D_c} \Big[ \alpha_f \cdot \hat{f}(x_i = d_i, x_c = d_c) + \alpha_e \cdot e(x_i = d_i, x_c = d_c) \Big] \tag{5.5}$$

where $x_c$ is a neighboring variable, $Nh(x_i)$ is the set of neighboring variables that $x_i$ is constrained with, and each estimated cost function $\hat{f}$ corresponds exactly to a partially-specified function $\tilde{f}$, except that all the unknown costs ? are replaced with the lower bound $\mathcal{L}$. Therefore, the estimated cost $\hat{f}(\mathbf{x})$ is guaranteed to be no larger than the true cost $f(\mathbf{x})$ for any solution $\mathbf{x}$.

### 5.4.4 Variable and Value Ordering

Instead of choosing a random order to explore the different values of an agent, we order their values according to the best-available cost function $f(n, \tilde{\mathbf{x}}) = \alpha_f \cdot g(n) + h(n) + \alpha_e \cdot \mathcal{E}(\tilde{\mathbf{x}})$, where $n$ is the node corresponding to the value of the agent and $\tilde{\mathbf{x}}$ is the current explored solution space.

Instead of choosing a random ordering of variables for SyncBB, we order the variables based on the number of their constraints that has unknown costs – the variable with the fewest number of constraints with unknown costs as the root and the variable with the most number of constraints with unknown costs as the leaf.

The rationale for this heuristic is the following: When an agent is higher up in the search tree (i.e., closer to the root), it will likely need to explore all of its values since the partial cost of its CPA (i.e., the partial solution from the root to the agent) is likely to be small as the CPA only contains the value assignments of few agents. As a result, if any of its constraints contain unknown costs, those costs will likely need to be elicited. In contrast, when an agent is lower down in the search tree (i.e., closer to the leaf), it is more likely to be able to prune many of its values since the partial cost of its CPA is likely to be larger as the CPA contains value assignments of more agents. As a result, it is more likely that many of its unknown costs will not be elicited.

### 5.4.5 Theoretical Results

**Theorem 4** *The computation of the CAC and ADC heuristics require $O(|\mathcal{A}|)$ number of messages. The computation of the NHC heuristic requires no messages.*

*Proof* : The CAC and ADC heuristics are recursively computed starting from the leaf to the root and will take exactly $|\mathcal{A}| - 1$ number of messages. The NHC heuristic is not computed recursively and does not send any messages to compute its heuristic cost. ■

**Lemma 1** *When all elicitation costs are zero, the CAC, ADC and NHC heuristics are admissible.*

*Proof* : We first prove the admissibility of the CAC heuristic. We prove that $h(n) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n)$, where $\mathbf{x}_n$ is the best complete solution in the subtree rooted at node $n$, for all nodes $n$ in the search tree. We prove this by induction from the leaf agent up the pseudo-chain:

- **Leaf Agent:** For a leaf agent $x_i$, $h(x_i = d_i) = 0$ for each of its values $d_i \in D_i$. Therefore, the inequality $h(n) = 0 \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n)$ trivially applies for all nodes $n$ corresponding the agent $x_i$ taking on its values $d_i \in D_i$.
- **Induction Assumption:** Assume that the lemma holds for all agents up to the $(k-1)$-th agent up the pseudo-chain.

- **The $k$-th Agent:** For the $k$-th agent $x_k$ from the leaf:

$$\begin{aligned} h(x_k{=}d_k) &= \\ &\min_{d_c \in D_c} \Big[\alpha_f \cdot \hat{f}(x_k{=}d_k, x_c{=}d_c) + \alpha_e \cdot e(x_k{=}d_k, x_c{=}d_c) + h(x_c{=}d_c)\Big] \\ &+ \sum_{x_m \in Anc(x_c) \setminus \{x_k\}} \min_{d_m \in D_m} \Big[\alpha_f \cdot \hat{f}(x_c{=}d_c, x_m{=}d_m) + \alpha_e \cdot e(x_c{=}d_c, x_m{=}d_m)\Big] \end{aligned} \tag{5.6}$$

where $x_c$ is the next agent in the ordering (i.e., the $(k-1)$-th agent), $Anc(x_c)$ is the set of variables higher up in the ordering that $x_c$ is constrained with. Based on our induction assumption the lemma holds for all agents up to the $(k-1)$-th agent up the pseudo-chain, hence, we have:

$$h(n) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n), \tag{5.7}$$

where node $n$ corresponds to the $(k-1)$-th agent in the pseudo-chain, which we denote it as agent $x_c$. For each estimated cost function $\hat{f}$ in the CAC heuristic, it is easy to see:

$$\hat{f}(x_c{=}d_c, x_m{=}d_m) \leq f(x_c{=}d_c, x_m{=}d_m), \tag{5.8}$$

for any pair of agents $x_c$ and $x_m$ with any of their value combinations since all unknown costs ? are replaced with the lower bound $\mathcal{L}$ on all constraint costs. Thus, combined with the premise that elicitation costs are all zero and the induction assumption, we get:

$$\begin{aligned} h(x_k{=}d_k) &= \\ &\min_{d_c \in D_c} \Big[\alpha_f \cdot \hat{f}(x_k{=}d_k, x_c{=}d_c) + \alpha_e \cdot e(x_k{=}d_k, x_c{=}d_c) + h(x_c{=}d_c)\Big] \\ &+ \sum_{x_m \in Anc(x_c) \setminus \{x_k\}} \min_{d_m \in D_m} \Big[\alpha_f \cdot \hat{f}(x_c{=}d_c, x_m{=}d_m) + \alpha_e \cdot e(x_c{=}d_c, x_m{=}d_m)\Big] \\ &\leq \min_{d_c \in D_c} \Big[\alpha_f \cdot f(x_k{=}d_k, x_c{=}d_c) + h(x_c{=}d_c)\Big] \end{aligned} \tag{5.9}$$

$$+ \sum_{x_m \in Anc(x_c) \setminus \{x_k\}} \min_{d_m \in D_m} \alpha_f \cdot f(x_c = d_c, x_m = d_m) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n) \tag{5.10}$$

where node $n$ corresponds to the agent $x_k$ taking on its value $d_k \in D_k$. ■

To prove the admissibility of ADC which is very similar to that of CAC, we need to prove that $h(n) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n)$, where $\mathbf{x}_n$ is the best complete solution in the subtree rooted at node $n$, for all nodes $n$ in the search tree. We prove this by induction from the leaf agent up the pseudo-chain:

- **Leaf Agent:** For a leaf agent $x_i$, $h(x_i = d_i) = 0$ for each of its values $d_i \in D_i$. Therefore, the inequality $h(n) = 0 \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n)$ trivially applies for all nodes $n$ corresponding the agent $x_i$ taking on its values $d_i \in D_i$.
- **Induction Assumption:** Assume that the lemma holds for all agents up to the $(k-1)$-th agent up the pseudo-chain.
- **The $k$-th Agent:** For the $k$-th agent $x_k$ from the leaf:

$$\begin{aligned} h(x_k = d_k) = & \\ & \min_{d_c \in D_c} \left[ \alpha_f \cdot \hat{f}(x_k = d_k, x_c = d_c) + \alpha_e \cdot e(x_k = d_k, x_c = d_c) + h(x_c = d_c) \right] \\ & + \sum_{x_m \in Des(x_k) \setminus \{x_c\}} \min_{d_m \in D_m} \left[ \alpha_f \cdot \hat{f}(x_k = d_k, x_m = d_m) + \alpha_e \cdot e(x_k = d_k, x_m = d_m) \right] \end{aligned} \tag{5.11}$$

where $x_c$ is the next agent in the ordering (i.e., the $(k-1)$-th agent), $Des(x_k)$ is the set of variables lower down in the ordering that $x_k$ is constrained with (which is the set of descendants of variable $x_k$). Based on our induction assumption the lemma holds for all agents up to the $(k-1)$-th agent up the pseudo-chain, hence, we have:

$$h(n) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n), \tag{5.12}$$

where node $n$ corresponds to the $(k-1)$-th agent in the pseudo-chain, which we denote it as agent $x_c$. For each estimated cost function $\hat{f}$ in the ADC heuristic, it is easy to see:

$$\hat{f}(x_k\!=\!d_k, x_m\!=\!d_m) \leq f(x_k\!=\!d_k, x_m\!=\!d_m), \tag{5.13}$$

for any pair of agents $x_k$ and $x_m$ with any of their value combinations since all unknown costs ? are replaced with the lower bound $\mathcal{L}$ on all constraint costs. Thus, combined with the premise that elicitation costs are all zero and the induction assumption, we get:

$$\begin{aligned}
&h(x_k\!=\!d_k) = \\
&\quad \min_{d_c \in D_c} \Big[\alpha_f \cdot \hat{f}(x_k\!=\!d_k, x_c\!=\!d_c) + \alpha_e \cdot e(x_k\!=\!d_k, x_c\!=\!d_c) + h(x_c\!=\!d_c)\Big] \\
&\quad + \sum_{x_m \in Des(x_k)\setminus\{x_c\}} \min_{d_m \in D_m} \Big[\alpha_f \cdot \hat{f}(x_k\!=\!d_k, x_m\!=\!d_m) + \alpha_e \cdot e(x_k\!=\!d_k, x_m\!=\!d_m)\Big] \qquad (5.14) \\
&\quad \leq \min_{d_c \in D_c} \Big[\alpha_f \cdot f(x_k\!=\!d_k, x_c\!=\!d_c) + h(x_c\!=\!d_c)\Big] \\
&\quad + \sum_{x_m \in Des(x_k)\setminus\{x_c\}} \min_{d_m \in D_m} \alpha_f \cdot f(x_k\!=\!d_k, x_m\!=\!d_m) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n) \qquad (5.15)
\end{aligned}$$

where node $n$ corresponds to the agent $x_k$ taking on its value $d_k \in D_k$. ■

**Theorem 5** *When all elicitation costs are zero, SyncBB with the CAC and ADC heuristics parameterized by a user-defined relative weight $w \geq 1$ and a user-defined additive weight $\epsilon \geq 0$ will return an I-DCOP solution whose cost is bounded from above by $w \cdot OPT + \epsilon$, where $OPT$ is the optimal solution cost.*

*Proof*: The proof is similar to the proofs of similar properties [136] for other DCOP search algorithms that also use heuristics. The key assumption in the proofs is that the heuristics employed are admissible heuristics – and the CAC and ADC are admissible according to Lemma 1. ■

## 5.5 Experimental Evaluations

We evaluate our algorithms, SyncBB using the CAC and ADC heuristics and ALS-MGM using the NHC heuristic, against their baselines without heuristics on I-DCOPs with and without elicitation costs. In all experiments we set $\alpha_f = \alpha_e = 0.5$. Data points are averaged over 25 instances.

### 5.5.1 Metrics

We evaluate our algorithms on two benchmarks random graphs, and distributed meeting scheduling, where we measure the various costs of the solutions found – the cumulative constraint costs, cumulative elicitation costs, and their aggregated total costs – the number of unknown costs elicited, the number of nodes expanded after SyncBB terminates, and we measure the simulated runtimes of our algorithms (in sec).

### 5.5.2 Random Graphs

We generate 25 random (binary) graphs [24], where we vary the number of agents/variables $|\mathcal{A}|$ from 10 to 180; the user-defined relative weight $w$ from 1 to 10; and the user-defined additive weight $\epsilon$ from 0 to 50. The constraint density $p_1$ is set to 0.4, the tightness $p_2$ is set to 0; the fraction of unknown costs in the problem is set to 0.6. In our experiments below, we only vary one parameter at a time, setting the rest at their default values: $|\mathcal{A}| = 10$, $|D_i| = 2$, $w = 1$, and $\epsilon = 0$. All constraint costs are randomly sampled from $[2, 5]$ and all elicitation costs are randomly sampled from $[0, 20]$. As mentioned earlier, in the ALS-MGM algorithm, the number of steps that the algorithm needs to run before termination is equal to $m + H$, where $m$ is the number of steps that a regular MGM algorithm would run and $H$

(a) SyncBB Without Heuristics

| $\|\mathcal{A}\|$ | # unk. costs | Without Elicitation Costs | | | | With Elicitation Costs | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | # of elic. | runtime | const. cost | # exp. nodes | # of elic. | runtime | total costs | const. cost | elic. cost | # exp. nodes |
| 10 | 43 | 39.92 | 7.55E-01 | 51.88 | 1.65E+03 | 18.08 | 5.60E-02 | 189.28 | 60.20 | 129.08 | 6.70E+01 |
| 12 | 62 | 58.80 | 2.59E+00 | 75.48 | 6.76E+03 | 25.52 | 9.28E-02 | 264.88 | 87.16 | 177.72 | 1.25E+02 |
| 14 | 86 | 81.88 | 9.18E+00 | 107.40 | 2.40E+04 | 35.16 | 7.68E-02 | 363.80 | 123.36 | 240.44 | 9.24E+01 |
| 16 | 115 | 110.92 | 3.58E+01 | 145.40 | 9.51E+04 | 48.56 | 1.13E-01 | 485.40 | 163.80 | 321.60 | 1.60E+02 |
| 18 | 146 | 139.80 | 1.24E+02 | 184.44 | 3.54E+05 | 60.76 | 1.29E-01 | 621.04 | 206.68 | 414.36 | 3.08E+02 |
| 20 | 182 | 175.56 | 5.52E+02 | 231.64 | 1.36E+06 | 72.84 | 1.63E-01 | 741.76 | 252.88 | 488.88 | 2.79E+02 |

(b) SyncBB with CAC Heuristic

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 43 | 37.96 | 3.63E-01 | 51.88 | 7.73E+02 | 12.96 | 2.15E-02 | 173.88 | 61.64 | 112.24 | 2.21E+01 |
| 12 | 62 | 57.32 | 1.22E+00 | 75.48 | 2.97E+03 | 18.32 | 1.82E-02 | 242.12 | 88.72 | 153.40 | 2.99E+01 |
| 14 | 86 | 80.32 | 3.58E+00 | 107.40 | 9.50E+03 | 26.08 | 3.69E-02 | 350.48 | 125.48 | 225.00 | 3.90E+01 |
| 16 | 115 | 110.80 | 1.61E+01 | 145.40 | 4.26E+04 | 35.56 | 3.03E-02 | 464.16 | 165.24 | 298.92 | 4.83E+01 |
| 18 | 146 | 139.12 | 4.16E+01 | 184.44 | 1.21E+05 | 44.84 | 4.75E-02 | 590.20 | 206.64 | 383.56 | 6.11E+01 |
| 20 | 182 | 164.76 | 3.67E+02 | 231.64 | 4.09E+05 | 54.52 | 6.29E-02 | 722.68 | 258.28 | 464.40 | 5.06E+01 |

(c) SyncBB with ADC Heuristic

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 43 | 39.00 | 4.89E-01 | 51.86 | 1.56E+03 | 14.10 | 2.31E-02 | 190.74 | 60.96 | 129.78 | 2.33E+01 |
| 12 | 62 | 58.52 | 1.93E+00 | 75.48 | 6.25E+03 | 19.80 | 1.77E-02 | 267.36 | 88.60 | 178.76 | 3.06E+01 |
| 14 | 86 | 81.28 | 7.81E+00 | 107.40 | 2.23E+04 | 26.72 | 3.20E-02 | 375.30 | 122.22 | 253.08 | 3.25E+01 |
| 16 | 115 | 110.18 | 3.22E+01 | 145.40 | 8.89E+04 | 35.98 | 2.63E-02 | 496.68 | 164.80 | 331.88 | 3.96E+01 |
| 18 | 146 | 139.68 | 1.23E+02 | 184.44 | 3.39E+05 | 44.40 | 3.60E-02 | 621.42 | 208.14 | 413.28 | 3.93E+01 |
| 20 | 182 | 173.88 | 1.17E+03 | 231.64 | 1.29E+06 | 54.80 | 5.41E-02 | 771.60 | 259.24 | 512.36 | 4.08E+01 |

Table 5.1: Random Graphs: I-DCOP Empirical Results with SyncBB and its Heuristics

is the height of the BFS tree. Since the ALS framework requires that $m \geq H$, we vary $m$ from $H$ to $H + 240$.

Tables 5.1 and 5.2 tabulate our empirical results, where we vary the number of agents $|\mathcal{A}|$. Figure 5.5 plots the convergence rate of ALS-MGM when elicitation is free. We make the following observations:

- As expected, the runtimes and number of unknown costs elicited by all algorithms increase with increasing the number of agents $|\mathcal{A}|$. The reason is that the size of the problem, in terms of the number of constraints in the problem, increases with increasing $|\mathcal{A}|$. And all algorithms need to elicit more unknown costs and evaluate the costs of more constraints before terminating.
- On problems without elicitation costs, SyncBB with CAC is faster than without CAC. The reason is the following: The CAC heuristic value includes estimates of not only

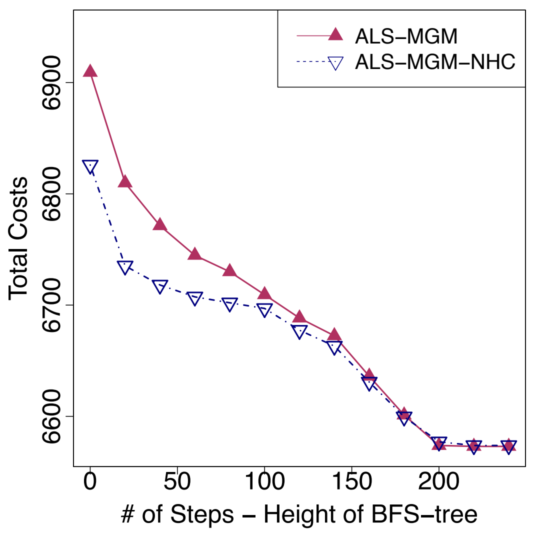


Figure 5.5: Random Graphs: Varying Number of Steps with $|\mathcal{A}| = 100$

all constraints between its descendant agents, but also constraints between any of its descendant agents with any of its ancestor agents. The CAC heuristic is thus likely to be more informed and provide better estimates. For ALS-MGM, its runtimes with and without NHC are about the same as the runtimes are dependent on the the number of steps the algorithms run, and they both run for the same number of $m + H$ steps. However, ALS-MGM finds better solutions (i.e., solutions with smaller costs) with NHC than without NHC because it starts with a better initial solution with the heuristic. Figure 5.5 also clearly shows that the difference in the quality of solutions is largest at the start of the algorithm and decreases as the algorithm runs more steps. Therefore, the heuristic is ideally suited for time-sensitive applications with short deadlines, where there is not enough time to let ALS-MGM run for a long time until convergence.

- On problems with elicitation costs, SyncBB with CAC is still faster than without CAC. The reason is that the number of nodes expanded is significantly smaller with CAC than without CAC. For ALS-MGM, the same trend as above applies here as well and for the same reasons.

(a) ALS-MGM Without Heuristic

| $\vert\mathcal{A}\vert$ | # of unk. costs | Without Elicitation Costs | | | With Elicitation Costs | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | # of elic. | runtime | const. cost | # of elic. | runtime | total cost | const. cost | elic. cost |
| 20 | 182 | 68.12 | 1.27E-01 | 253.04 | 69.44 | 1.26E-01 | 951.48 | 256.76 | 694.72 |
| 60 | 1699 | 492.00 | 2.47E+00 | 2459.04 | 496.00 | 2.47E+00 | 7429.16 | 2464.24 | 4964.92 |
| 100 | 4752 | 1302.44 | 2.58E+00 | 6909.20 | 1313.28 | 2.56E+00 | 20112.36 | 6926.20 | 13186.16 |
| 140 | 9341 | 2494.48 | 7.62E+00 | 13594.44 | 2502.52 | 7.62E+00 | 38725.12 | 13612.04 | 25113.08 |
| 180 | 15466 | 4080.32 | 9.79E+00 | 22521.76 | 4077.44 | 9.90E+00 | 63223.6 | 22530.52 | 40693.08 |

(b) ALS-MGM with NHC Heuristic

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 182 | 72.04 | 1.25E-01 | 246.92 | 65.96 | 1.22E-01 | 865.96 | 249.96 | 616.00 |
| 60 | 1699 | 521.60 | 2.46E+00 | 2417.20 | 476.56 | 2.44E+00 | 6913.00 | 2429.84 | 4483.16 |
| 100 | 4752 | 1364.92 | 2.58E+00 | 6825.92 | 1262.44 | 2.57E+00 | 18844.00 | 6870.28 | 11973.72 |
| 140 | 9341 | 2591.76 | 7.68E+00 | 13458.80 | 2416.52 | 7.66E+00 | 36647.04 | 13519.4 | 23127.64 |
| 180 | 15466 | 4210.04 | 9.93E+00 | 22356.04 | 3956.84 | 1.01E+01 | 60443.8 | 22407.04 | 38036.76 |

Table 5.2: Random Graphs: I-DCOPs Empirical Results with ALS-MGM and its Heuristic

- Overall, the use of heuristics in conjunction with SyncBB reduces the number of unknown costs elicited by up to 22% and the runtime by up to 57% when elicitation is not free; and the use of heuristics in conjunction with ALS-MGM improves the quality of solutions found. Therefore, these results highlight the strengths of using our proposed heuristics for solving I-DCOPs.
- We observe that ADC follows a similar trend to that of CAC. On problems without elicitation costs, SyncBB with CAC is faster than with ADC, which is faster than without heuristics. The reason is the following: The ADC heuristic value of an agent includes estimates of all constraints between all its descendant agents. The CAC heuristic value includes estimates of not only these constraints, but also constraints between any of its descendant agents with any of its ancestor agents. The CAC heuristic is thus likely to be more informed and provide better estimates. On problems with elicitation costs, SyncBB with heuristics is still faster than without heuristics. The reason is that the number of nodes expanded is significantly smaller than without heuristics. However, neither heuristic dominates the other.

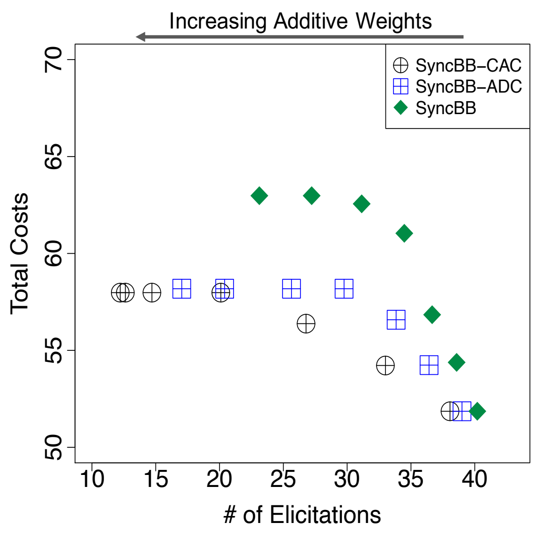


Figure 5.6: Random Graphs: Varying Additive Weights with $|\mathcal{A}| = 10$

Figure 5.6 plots our empirical results, where we vary the user-defined additive bound (weight) $\epsilon$ for the problems when elicitation is free (i.e., all elicitation costs are zero). Additive weights increases from right to left on the top axis of the Figure. Each data point in the figures thus show the result for one of the algorithms with one of the values of $\epsilon$. Data points for smaller values of $\epsilon$ are in the bottom right of the figures and data points for larger values are in the top left of the figures. We plot the tradeoffs between total cost (= cumulative constraint and elicitation costs) and number of elicited costs. As expected, as the additive bound $\epsilon$ increases, the number of elicitations decreases. However, this comes at the cost of larger total costs. Between the two algorithms, SyncBB with CAC is the best.

We omit plots of results where we vary the relative weight $w$ as their trends are similar to those shown here, and we also omit plots of results with elicitation costs as their trends are similar to those without elicitation costs for both additive and relative weights.

Figure 5.7 plots the convergence rate of ALS-MGM for problems with $|\mathcal{A}| = 20$, when elicitation is free. As expected, the total cost decreases with increasing the number of steps in

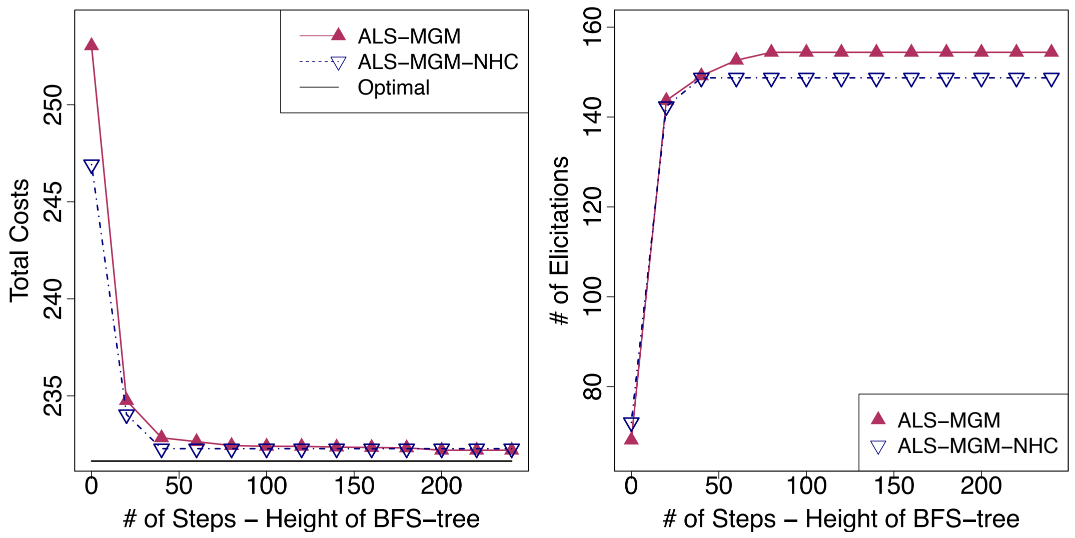


Figure 5.7: Random Graphs: Varying Number of Steps with $|\mathcal{A}| = 20$

ALS-MGM with NHC and without NHC in the left figure. Both ALS-MGM and ALS-MGM-NHC can find very close to optimal solutions as the number of steps increases. However, the number of elicited constraints costs increases with increasing the number of steps in both ALS-MGM and ALS-MGM-NHC, in the right figure, which is still smaller than the number of elicited constraint costs by SyncBB with CAC when finding the optimal solution (i.e., 164.76). As we increase the number of steps to find better solutions, ALS-MGM with NHC heuristic requires fewer number of elicitations than ALS-MGM without heuristic. The reason is ALS-MGM with the NHC heuristic starts with better initial solutions and converge to better solutions faster than ALS-MGM without heuristic, therefore requires fewer number of elicitations. Figure 5.5 clearly shows that the difference in the quality of solutions is largest at the start of the algorithm and decreases as the algorithm runs more steps. Therefore, the heuristic is ideally suited for time-sensitive applications with short deadlines, where there is not enough time to let ALS-MGM run for a long time until convergence. Moreover, Figure 5.5 shows the significant scalability of ALS-MGM compare to the SyncBB algorithm with heuristics.

(a) SyncBB Without Heuristics

| $\lvert\mathcal{X}\rvert$ | $\lvert\mathcal{F}\rvert$ | Without Elicitation Costs | | | | With Elicitation Costs | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | # of elic. | runtime | const. cost | # of nodes expanded | # of elic. | runtime | total cost | const. cost | elic. cost | #of nodes expanded |
| 9 | 16 | 14.20 | 1.36E+00 | 72.66 | 2.14E+03 | 12.00 | 3.60E-01 | 199.76 | 82.16 | 117.6 | 1.69E+03 |
| 12 | 26 | 13.12 | 8.73E+00 | 107.36 | 1.47E+04 | 11.54 | 2.70E+00 | 236.48 | 115.82 | 120.66 | 1.24E+04 |
| 15 | 43 | 12.82 | 8.12E+01 | 136.24 | 1.25E+05 | 13.32 | 3.15E+01 | 272.58 | 146.05 | 126.51 | 1.19E+05 |
| 18 | 61 | 13.80 | 5.69E+02 | 152.50 | 9.06E+05 | 13.56 | 1.83E+02 | 291.70 | 158.40 | 133.30 | 7.79E+05 |

(b) SyncBB With CAC Heuristic

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 16 | 13.62 | 1.24E+00 | 72.66 | 1.94E+03 | 11.00 | 2.73E-01 | 195.04 | 83.20 | 111.84 | 1.32E+03 |
| 12 | 26 | 12.98 | 7.00E+00 | 107.36 | 1.18E+04 | 10.20 | 1.72E+00 | 220.58 | 115.22 | 105.36 | 8.03E+03 |
| 15 | 43 | 12.60 | 6.96E+01 | 136.24 | 1.07E+05 | 12.00 | 1.92E+01 | 263.83 | 146.46 | 117.37 | 8.43E+04 |
| 18 | 61 | 11.32 | 4.51E+02 | 152.50 | 7.26E+05 | 10.72 | 1.27E+02 | 290.7 | 158.40 | 132.30 | 5.49E+05 |

(c) SyncBB With ADC Heuristic

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 16 | 12.80 | 1.06E+00 | 72.66 | 1.64E+03 | 10.58 | 1.73E-01 | 179.08 | 77.32 | 101.76 | 8.10E+02 |
| 12 | 26 | 12.28 | 6.26E+00 | 107.36 | 1.05E+04 | 8.60 | 1.27E+00 | 218.10 | 113.88 | 104.22 | 5.85E+03 |
| 15 | 43 | 12.12 | 5.94E+01 | 136.24 | 9.07E+04 | 11.00 | 1.13E+01 | 248.14 | 144.60 | 103.54 | 4.98E+04 |
| 18 | 61 | 11.40 | 4.69E+02 | 152.50 | 7.48E+05 | 10.64 | 1.15E+02 | 291.30 | 157.80 | 133.50 | 4.88E+05 |

Table 5.3: Distributed Meeting Scheduling: I-DCOP Empirical Results with SyncBB and its Heuristics

### 5.5.3 Distributed Meeting Scheduling

We generate 50 random problems, where we set the number of meeting participants (= agents) $|\mathcal{A}| = 10$, meeting time slots (= domain size) $|D_i| = 3$, density $p_1$ to 0.4, and tightness $p_2$ to 0.6 and also the number of unknown costs to 20. We vary the number of meetings (= variables) $|\mathcal{X}|$ from 9 to 27. All time preferences (constraint costs) and elicitation costs are randomly sampled from $[0, 20]$.

We evaluate our SyncBB algorithms on these problems. Table 5.3 tabulates our empirical results. As expected, the trends are similar to those in random graphs. In addition, we can observe that, the number of elicited costs are between $50\% - 70\%$ of the total number of unknown costs in order to find optimal and sub-optimal solutions.

(a) ALS-MGM Without Heuristics

| $\lvert\mathcal{X}\rvert$ | $\lvert\mathcal{F}\rvert$ | Without Elicitation Costs | | | With Elicitation Costs | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | # of elic. | runtime | const. cost | # of elic. | runtime | total cost | const. cost | elic. cost |
| 9 | 16 | 5.04 | 4.31E-01 | 89.84 | 4.92 | 4.32E-01 | 138.12 | 89.66 | 48.46 |
| 15 | 43 | 3.92 | 1.06E+00 | 155.71 | 4.06 | 1.06E+00 | 187.76 | 155.35 | 32.41 |
| 21 | 80 | 3.91 | 2.12E+00 | 213.00 | 3.67 | 2.12E+00 | 252.61 | 216.24 | 36.36 |
| 27 | 136 | 4.00 | 3.06E+00 | 265.29 | 3.90 | 3.05E+00 | 298.43 | 266.29 | 32.14 |

(b) ALS-MGM With NHC Heuristic

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 16 | 4.72 | 4.31E-01 | 81.76 | 4.84 | 5.70E-02 | 130.28 | 81.56 | 48.72 |
| 15 | 43 | 3.48 | 1.06E+00 | 147.04 | 3.42 | 1.07E+00 | 183.22 | 147.28 | 36.12 |
| 21 | 80 | 3.51 | 2.13E+00 | 203.33 | 3.51 | 2.14E+00 | 240.57 | 203.33 | 37.54 |
| 27 | 136 | 3.76 | 3.00E+00 | 264.66 | 3.47 | 3.02E+00 | 298.22 | 264.66 | 34.23 |

Table 5.4: Distributed Meeting Scheduling: I-DCOP Empirical Results with ALS-MGM and its Heuristic

Table 5.4 tabulates our empirical results as we evaluate our ALS-MGM algorithms. We can observe that the number of elicited costs by ALS-MGM and its heuristic is significantly less than those elicited by SyncBB with heuristics. The reason is: ALS-MGM is a local search algorithm that does not guarantee optimality. However, because of its any-time property, it improves its solution quality at every iteration. Consequently, it finds a solution that is close to the optimal solution found by SyncBB. Moreover, the SyncBB algorithm runs out of time and space when solving problems with larger number of variables, while ALS-MGM can easily solve larger class of time-sensitive problems.

## 5.6 Related Work

As our work lies in the intersection of constraint-based models, preference elicitation, and heuristic search, we will first focus on related work in this intersection before covering the three broader areas. Aside from our work [109] discussed in Chapter 3 discussed, the body of work that is most related to ours is the work on *Incomplete Weighted CSPs* (IWCSPs) [35, 36, 114]. IWCSPs can be seen as centralized versions of I-DCOPs. Researchers have proposed a family of algorithms based on depth-first branch-and-bound and local search to solve IWCSPs including heuristics that can be parameterized like ours. Aside from IWCSPs, similar centralized constraint-based models include *Incomplete Fuzzy CSPs* and *Incomplete Soft Constraint Satisfaction Problems*.

Another body of work in this intersection is the use of weighted heuristics in canonical DCOPs. For example, additive weights were introduced for search-based algorithms [76, 136], which provide additive quality guarantees as those in Theorem 5. Similarly, relative weights were also introduced for ADOPT and other search-based algorithms [137], which provide relative quality guarantees as those in Theorem 5.

*Conditional-Preference Networks* (CP-nets) [7, 97] also lie in this intersection. CP-nets are a graphical representation model for qualitative preferences and reflects conditional dependencies between sets of preference statements. In contrast, I-DCOPs focus more on the notion of *conditional additive independence* [4], which requires that the cost of an outcome to be the sum of the "costs" of the different variable values of the outcome. Additionally, CP-nets are centralized problems while I-DCOPs are decentralized problems.

Finally, in the context of the broader preference elicitation area, there is a very large body of work [41], and we focus on techniques that are most closely related to our approach. They include techniques that ask users a number of preset questions [109, 121] as well as send alerts and notification messages to interact with users [18], techniques that ask users to rank alternative options or user-provided option improvements to learn a (possibly approximately) user preference function [8, 13, 126], and techniques that associate costs to eliciting preferences and takes these costs into account when identifying which preference to elicit as well as when to stop eliciting preferences [63, 122]. The key difference between all these approaches and ours is that they identify preferences to elicit a priori before the search while we embed the preference elicitation in the underlying DCOP search algorithm.

## 5.7 Conclusions

*Distributed Constraint Optimization Problems* (DCOPs) have been used to model a variety of cooperative multi-agent problems. However, they assume that all constraints are fully specified, which may not hold in applications where constraints encode preferences of human users. To overcome this limitation, we proposed *Incomplete DCOPs* (I-DCOPs), which extends DCOPs by allowing some constraints to be partially specified and the elicitation of unknown costs in such constraints incur elicitation costs.

We propose two parameterized heuristics – CAC and ADC – that can be used in conjunction with Synchronous Branch-and-Bound (SyncBB) to solve I-DCOPs. These heuristics allow users to trade off solution quality for faster runtimes and smaller number of elicitations. In addition, we propose NHC heuristic that can be used in conjunction with a local search algorithm ALS-MGM with any-time property to solve a larger class of I-DCOPs. Further, in problems where elicitations are free, SyncBB and its heuristics provide theoretical quality guarantees on the solutions found. In conclusion, our new model, adapted algorithms, and new heuristics improve the practical applicability of DCOPs as they are now better suited to model multi-agent applications with user preferences.

# Chapter 6

# Discussion and Future Directions

*Constraint Satisfaction Problems* (CSPs) [3, 44] and its variant *Weighted Constraint Satisfaction Problems* (WCSPs) [59, 102], as well as its decentralized variant, *Distributed Constraint Optimization Problems* (DCOPs) [86, 137] are powerful paradigms for formulating many combinatorial and optimization problems. We refer to these paradigms as "*Constraint-Based Models*." Over the past decades, researchers have developed many resolution algorithms to solve problems formulated by constraint-based models using centralized and distributed approaches.

The importance of constraint-based models is outlined by the impact of their applications in a wide range of agent-based systems. Example of such applications are supply-chain management [34, 95], roster scheduling [1, 11], meeting scheduling [69], combinatorial auctions [99], bioinformatics [2, 12, 28], and smart home automation [31, 98, 112].

One of the key drawbacks of these constraint-based models is the assumption that *all* the constraint costs are specified or known a priori. For instance, in several applications (e.g., a scheduling problem), some constraints encode the preferences of human users. Such constraints may not be fully specified because it is unrealistic to accurately know the preferences of users for all possible scenarios in an application. These constraint costs are only known after they are queried or elicited from domain experts or human users. To have a complete knowledge of all these preferences, there is a need for preference elicitation technique. Preference elicitation is a process of asking questions about the users' preferences. This process allows users to intelligently interact with the constraint-based solver (i.e., resolution algorithms) without being forced to state all their constraints, or preferences, at the beginning of the interaction. More specifically, preference elicitation is more pronounced in scenarios where the users want to avoid revealing all of their preferences at the beginning of the interaction due to privacy reasons.

Such an assumption in constraint-based models restrains their capabilities to model and solve many human-in-the-loop optimization problems in a centralized or decentralized manner. To address this limitation, this dissertation makes the following contributions:

- In Chapter 3, we investigated how constraint-based models can be extended to allow uncertainty in constraints. We introduced uncertain constraint-based models (uncertain COPs/DCOPs) and the pre-execution elicitation approach. In uncertain constraint-based models, constraint costs are represented as random variables that follow Normal (i.e., Gaussian) distributions. As existing resolution algorithms are not able to solve the new constraint-based models, we introduce Minimax Regret and Maximum Standard Deviation heuristics which are probabilistic strategies to elicit a set of constraints before the execution of any resolution algorithm. In this approach, we decoupled the elicitation process from the search for an optimal solution. Due to this independency between elicitation process and resolution algorithm, we employed an off-the-shelf constraint-based solver to find a solution for the underlying problem. Prior to running the solver, our probabilistic strategies elicit the necessary constraints and prepare them for the solver by realizing the uncertain cost tables from their corresponding distributions.

The objective of our proposed framework is to minimize the error between the solution quality of the oracle DCOP and the realized DCOP. We evaluated our framework on two main benchmarks, random graph problems as well as our real-world motivating application SHDS. Our empirical results demonstrated that our probabilistic elicitation methods outperformed their baseline random strategy in minimizing the error in both benchmarks and showed the significance of our contributions in real-world scenarios.

By introducing the uncertain constraint-based models, we made the foundational contributions necessary in deploying COP/DCOP algorithms on practical applications, where preferences or constraint costs must be elicited or estimated.

- In Chapter 4, we investigated how to extend IWCSPs so that constraint costs are allowed to be partially unknown where elicitation of these unknown costs incur penalties. We introduced the Incomplete WCSPs with Elicitation Costs (IWCSPs+EC) framework, where only a set of constraint costs are specified a priori and the rest remain unknown. The proposed framework associated pre-defined costs for eliciting the unknown constraints to represent how much a user is annoyed by multiple queries. Consequently, the objective of IWCSPs+EC is to find a solution that minimizes both constraint and elicitation costs. To solve IWCSPs+EC in a centralized manner, we extended a depth first complete search algorithm (DFBnB) in which, we interleaved elicitation strategies with the search algorithm to optimize both constraint and elicitation costs. Moreover, we introduced parameterized heuristics – Least Unknown Cost (LUC), Least Known Cost (LKC) and their combination heuristic (COM) – to allow users to trade off solution quality for fewer elicited preferences and faster computation times.

  We evaluated our model and algorithms on two main benchmarks, random graph problems as well as our real-world motivating application SHDS. Our empirical results demonstrated that DFBnB with all three heuristics – LUC, LKC, and COM – outperform the baseline random heuristic in terms of the number of elicited costs, solution quality, and runtime. COM finds solutions with larger constraint costs than LKC and LUC, but finds them faster and with fewer elicitations than LKC and LUC. Therefore, COM is the preferred heuristic in critical time-sensitive domains. COM also does a better job at trading off solution quality for smaller runtimes, especially when runtimes are large, through the use of user-defined weights.

By introducing IWCSP+EC and its elicitation heuristics we took a step forward in improving the practical applicability of WCSPs and IWCSPs as they now take into account unknown costs with their elicitation costs and provide control knobs, in the form of user-defined weights, to perform tradeoffs along three key dimensions – solution quality, runtime, and number of elicited preferences.

- In Chapter 5, we investigated how to extend DCOPs so that constraint costs are allowed to be partially unknown where elicitation of these unknown costs incur penalties. Similar to IWCSPs+EC, we introduced the Incomplete DCOPs (I-DCOPs) framework, which is the distributed version, to formulate problems that are distributed. The objective of I-DCOPs is to find a solution that minimizes both constraint and elicitation costs. To solve such distributed framework, we developed several distributed heuristics and interleaved elicitation strategies with search algorithms. The first search algorithm is the synchronous branch-and-bound (SyncBB) algorithm with two parametrized heuristics – CAC and ADC – which aims at finding an optimal solution when elicitation costs are zero and the best possible solution otherwise. Our parameterized heuristics allow users to trade off solution quality for fewer elicited preferences and faster computation times. The second algorithm, is a local search with any-time property ALS-MGM with its heuristic – NHC – which aims at solving a larger class of incomplete DCOPs and finding sub-optimal solutions with a fewer number of elicited costs.

We evaluated our model and algorithms on two main benchmarks, random graph problems as well as our real-world motivating application distributed meeting scheduling. Our empirical results demonstrated that our parameterized heuristics improve both complete and local search algorithms (i.e., SyncBB and ALS-MGM algorithms) in solving I-DCOPs. Further, in problems where elicitations are free, SyncBB and its heuristics provide theoretical quality guarantees on the solutions found. ALS-MGM with its heuristic, can solve significantly larger I-DCOPs and provide sub-optimal solutions.

By introducing our new model, adapted algorithms, and new heuristics, we took a step forward in improving the practical applicability of DCOPs as they are now better suited to model multi-agent applications with user preferences.

This dissertation demonstrates that one can improve the applicability of constraint-based models by developing new formulations, where constraint costs can be uncertain or unspecified (i.e., unknown) and applying elicitation strategies to constraint-based algorithms to solve such models. In the following, we list possible approaches that may help to improve preference elicitation of constraint-based models and consequently improve the applicability of constraint-based models.

- The proposed frameworks in both centralized and distributed manners associate random costs or penalties for eliciting the unknown constraints. These frameworks assume that one can have unlimited resources for eliciting the unknown constraints when elicitation costs are taken into account. This assumption can become unrealistic in scenarios with limited resources. Hence, to address this matter, one can associate a specific budget for elicitation of a limited set of unknown constraints and use a modified version of a knapsack problem [33] to approach it [63, 100].
- This dissertation only focused on explicitly eliciting unknown preferences, which requires several interactions between systems and human users. Despite the fact that our proposed

models have taken into account elicitation costs of those unknown preferences, our models rely on users with the assumption that the data elicited from the users are noise-free and trustworthy. This assumption can become unrealistic in scenarios when an annoyed user refuses to provide feedback to the system or provides wrong input or due to many other personal reasons. Therefore, to address this assumption, instead of explicitly eliciting users' preferences one can develop a model along with a learning algorithm that is able to perceive and learn the users behavior and account for noisy data in the system [112, 124, 125].

- The proposed frameworks in both centralized and distributed manners associate random costs or penalties for eliciting the unknown constraints, which we refer to as elicitation costs. Such elicitation costs can be modeled more accurately. For instance, the cost of repeated elicitation is higher than the first one [63]. Furthermore, such elicitation costs can also depend on the complexity of the queries [39]. People tend to get annoyed more quickly if they have to answer more complex questions. Therefore, we believe that one can conduct user studies to empirically model such elicitation costs [93].

Towards improving the practical applicability of constraint-based models further, we envision that these approaches can be interesting avenues to investigate in the future.

# References

[1] Slim Abdennadher and Hans Schlenker. “Nurse Scheduling using Constraint Logic Programming.” In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI).* 1999, pp. 838–843.

[2] David Allouche, Isabelle André, Sophie Barbe, Jessica Davies, Simon de Givry, George Katsirelos, Barry O’Sullivan, Steven David Prestwich, Thomas Schiex, and Seydou Traoré. “Computational protein design as an optimization problem.” In: *Artificial Intelligence* 212 (2014), pp. 59–79.

[3] Krzysztof R. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.

[4] Fahiem Bacchus and Adam J. Grove. “Utility Independence in a Qualitative Decision Theory.” In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR).* 1996, pp. 542–552.

[5] Fahiem Bacchus and Adam J. Grove. “Utility Independence in a Qualitative Decision Theory.” In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR).* 1996, pp. 542–552.

[6] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. “Semiring-based constraint satisfaction and optimization.” In: *Journal of the ACM,* 44.2 (1997), pp. 201–236.

[7] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. “CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements.” In: *Journal of Artificial Intelligence Research* 21 (2004), pp. 135–191.

[8] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. “Constraint-based optimization and utility elicitation using the minimax decision criterion.” In: *Artificial Intelligence* 170.8-9 (2006), pp. 686–713.

[9] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. “Regret-based Utility Elicitation in Constraint-based Decision Problems.” In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).* 2005, pp. 929–934.

[10] Darius Braziunas and Craig Boutilier. "Local Utility Elicitation in GAI Models." In: *Proceedings of the Conference in Uncertainty in Artificial Intelligence (UAI)*, pp. 42–49.

[11] Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. "The state of the art of nurse rostering." In: *Journal of Scheduling* 7.6 (2004), pp. 441–499.

[12] Federico Campeotto, Alessandro Dal Palù, Agostino Dovier, Ferdinando Fioretto, and Enrico Pontelli. "A Constraint Solver for Flexible Protein Model." In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 953–1000.

[13] Urszula Chajewska, Daphne Koller, and Ronald Parr. "Making Rational Decisions Using Adaptive Utility Elicitation." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2000, pp. 363–369.

[14] Dingding Chen, Yanchen Deng, Ziyu Chen, Wenxing Zhang, and Zhongshi He. "HS-CAI: A hybrid DCOP algorithm via combining search with context-based inference." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. Vol. 34. 05. 2020, pp. 7087–7094.

[15] Li Chen and Pearl Pu. "Survey of Preference Elicitation Methods." In: (2004).

[16] Edward H Clarke. "Multipart pricing of public goods." In: *Public choice* (1971), pp. 17–33.

[17] Wolfram Conen and Tuomas Sandholm. "Partial-revelation VCG mechanism for combinatorial auctions." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2002, pp. 367–372.

[18] Enrico Costanza, Joel E. Fischer, James A. Colley, Tom Rodden, Sarvapali D. Ramchurn, and Nicholas R. Jennings. "Doing the laundry with agents: a field trial of a future smart energy system in the home." In: *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. 2014, pp. 813–822.

[19] Konrad K Dabrowski, Peter Jonsson, Sebastian Ordyniak, and George Osipov. "Solving Infinite-Domain CSPs Using the Patchwork Property." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI), to appear*. 2021.

[20] Rina Dechter. "Bucket Elimination: A Unifying Framework for Reasoning." In: *Artificial Intelligent* 113.1-2 (1999), pp. 41–85.

[21] Rina Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003.

[22] Rina Dechter and Avi Dechter. "Belief Maintenance in Dynamic Constraint Networks." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 1988, pp. 37–42.

[23] Joanna Drummond and Craig Boutilier. "Preference Elicitation and Interview Minimization in Stable Matchings." In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. Ed. by Carla E. Brodley and Peter Stone. 2014, pp. 645–653.

[24] Paul Erdös and Alfréd Rényi. "On random graphs, I." In: *Publicationes Mathematicae (Debrecen)* 6 (1959), pp. 290–297.

[25] Boi Faltings and Santiago Macho-Gonzalez. "Open constraint programming." In: *Artificial Intelligence* 161.1-2 (2005), pp. 181–208.

[26] Boi Faltings and Santiago Macho-Gonzalez. "Open Constraint Satisfaction." In: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*. Vol. 2470. 2002, pp. 356–370.

[27] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas Jennings. "Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm." In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2008, pp. 639–646.

[28] Ferdinando Fioretto, Agostino Dovier, and Enrico Pontelli. "Constrained Community-Based Gene Regulatory Network Inference." In: *ACM Transactions on Modeling and Computer Simulation* 25.2 (2015), 11:1–11:26.

[29] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. "Distributed Constraint Optimization Problems and Applications: A Survey." In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 623–698.

[30] Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. "A Dynamic Programming-based MCMC Framework for Solving DCOPs with GPUs." In: *Proceedings of Principles and Practice of Constraint Programming (CP)*. 2016, pp. 813–831.

[31] Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. "A Multiagent System Approach to Scheduling Devices in Smart Homes." In: *Proceedings of the Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2017, pp. 981–989.

[32] Natalia Flerova, Radu Marinescu, and Rina Dechter. "Weighted heuristic anytime search: new schemes for optimization over graphical models." In: *Annals of Mathematics and Artificial Intelligence* (2017).

[33] B Fortz, M Labbé, F Louveaux, and M Poss. *The Knapsack Problem with Gaussian Weights*. Tech. rep. Technical Report 592, GOM, Université Libre de Bruxelles, 2008.

[34] Jonathan Gaudreault, Jean-Marc Frayret, and Gilles Pesant. "Distributed search for supply chain coordination." In: *Computers in Industry* 60.6 (2009), pp. 441–451.

[35] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, K Brent Venable, and Toby Walsh. "Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies." In: *Artificial Intelligence* 174.3-4 (2010), pp. 270–294.

[36] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. "A Local Search Approach for Incomplete Soft Constraint Problems: Experimental Results on Meeting Scheduling Problems." In: *Proceedings of the the International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*. 2017, pp. 403–418.

[37] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. "Elicitation Strategies for Fuzzy Constraint Problems with Missing Preferences: Algorithms and Experimental Studies." In: *Proceedings of Principles and Practice of Constraint Programming (CP)*. 2008, pp. 402–417.

[38] Amir Gershman, Amnon Meisels, and Roie Zivan. "Asynchronous Forward Bounding for Distributed COPs." In: *Journal of Artificial Intelligence Research* 34 (2009), pp. 61–88.

[39] Paul W Goldberg, Edwin Lock, and Francisco Marmolejo-Cossío. "Learning Strong Substitutes Demand via Queries." In: *International Conference on Web and Internet Economics*. Springer. 2020, pp. 401–415.

[40] Judy Goldsmith and Ulrich Junker. "Preference Handling for Artificial Intelligence." In: *AI Magazine* 29.4 (2008), pp. 9–12.

[41] Judy Goldsmith and Ulrich Junker. "Preference Handling for Artificial Intelligence." In: *AI Magazine* 29.4 (2008), pp. 9–12.

[42] Theodore Groves. "Incentives in teams." In: *Econometrica: Journal of the Econometric Society* (1973), pp. 617–631.

[43] Eric A Hansen and Rong Zhou. "Anytime heuristic search." In: *Journal of Artificial Intelligence Research* 28 (2007), pp. 267–297.

[44] Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter. "Handbook of Knowledge Representation." In: *Constraint Programming*. Vol. 3. Elsevier, 2008, pp. 181–211.

[45] Paul Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." In: *IEEE Transactions on Systems Science and Cybernetics* SSC4.2 (1968), pp. 100–107.

[46] Daisuke Hatano and Katsutoshi Hirayama. "DeQED: An Efficient Divide-and-Coordinate Algorithm for DCOP." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2013, pp. 566–572.

[47] David Hemmi, Guido Tack, and Mark Wallace. "A recursive scenario decomposition algorithm for combinatorial multistage stochastic optimisation problems." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. Vol. 32. 1. 2018.

[48] Katsutoshi Hirayama and Makoto Yokoo. "Distributed partial constraint satisfaction problem." In: *Proceedings of Principles and Practice of Constraint Programming (CP)*. 1997, pp. 222–236.

[49] Khoi D. Hoang, Ferdinando Fioretto, Ping Hou, Makoto Yokoo, William Yeoh, and Roie Zivan. "Proactive Dynamic Distributed Constraint Optimization." In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2016, pp. 597–605.

[50] Khoi D. Hoang, Ping Hou, Ferdinando Fioretto, William Yeoh, Roie Zivan, and Makoto Yokoo. “infinite-Horizon Proactive Dynamic DCOPs.” In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS).* 2017, pp. 212–220.

[51] Andrea Iovine, Pasquale Lops, Fedelucio Narducci, Marco de Gemmis, and Giovanni Semeraro. “Improving preference elicitation in a conversational recommender system with active learning strategies.” In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing.* 2021, pp. 1375–1382.

[52] Anju K. James, George Torres, Sharad Shrestha, Reza Tourani, and Satyajayant Misra. “iCAAP: information-Centric network Architecture for Application-specific Prioritization in Smart Grid.” In: *Proceedings of the IEEE Conference on Power & Energy Society Innovative Smart Grid Technologies.* IEEE, 2021, pp. 1–5.

[53] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. “Partially Observable Markov Decision Processes for Artificial Intelligence.” In: *Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics (RUR).* Vol. 981. Lecture Notes in Computer Science. Springer, 1995, pp. 1–17.

[54] Richard Korf. “Linear-Space Best-First Search.” In: *Artificial Intelligence* 62.1 (1993), pp. 41–78.

[55] Frédéric Koriche and Bruno Zanuttini. “Learning Conditional Preference Networks with Queries.” In: *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009.* Ed. by Craig Boutilier. 2009, pp. 1930–1935.

[56] Mohit Kumar, Samuel Kolb, Stefano Teso, and Luc De Raedt. “Learning MAX-SAT from Contextual Examples for Combinatorial Optimisation.” In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI).* 2020, pp. 4493–4500.

[57] Evelina Lamma, Paola Mello, Michela Milano, Rita Cucchiara, Marco Gavanelli, and Massimo Piccardi. “Constraint Propagation and Value Acquisition: Why we should do it Interactively.” In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).* 1999, pp. 468–477.

[58] Jérôme Lang, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. “Winner Determination in Sequential Majority Voting.” In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).* 2007, pp. 1372–1377.

[59] Javier Larrosa. “Node and arc consistency in weighted CSP.” In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI).* 2002, pp. 48–53.

[60] Javier Larrosa and Thomas Schiex. “Solving weighted CSP by maintaining arc consistency.” In: *Artificial Intelligence* 159.1-2 (2004), pp. 1–26.

[61] Tiep Le, Ferdinando Fioretto, William Yeoh, Tran Cao Son, and Enrico Pontelli. "ER-DCOPs: A Framework for Distributed Constraint Optimization with Uncertainty in Constraint Utilities." In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2016, pp. 606–614.

[62] Tiep Le, Tran Cao Son, Enrico Pontelli, and William Yeoh. "Solving Distributed Constraint Optimization Problems with Logic Programming." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2015, pp. 1174–1181.

[63] Tiep Le, Atena M Tabakhi, Long Tran-Thanh, William Yeoh, and Tran Cao Son. "Preference Elicitation with Interdependency and User Bother Cost." In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2018, pp. 1459–1467.

[64] Thomas Léauté and Boi Faltings. "Distributed Constraint Optimization under Stochastic Uncertainty." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2011, pp. 68–73.

[65] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. "Truth revelation in approximately efficient combinatorial auctions." In: *Journal of the ACM (JACM)* 49.5 (2002), pp. 577–602.

[66] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. "ARA*: Anytime A* with provable bounds on sub-optimality." In: *Proceedings of the Annual Conference in Advances in Neural Information Processing Systems (NIPS)*. 2004, pp. 767–774.

[67] Santiago Macho-Gonzalez and Pedro Meseguer. "On the relation among open, interactive and dynamic CSP." In: *Modelling and Solving Problems with Constraints* (2005), p. 76.

[68] Rajiv Maheswaran, Jonathan Pearce, and Milind Tambe. "Distributed Algorithms for DCOP: A Graphical Game-Based Approach." In: *Proceedings of the International Conference on Parallel and Distributed Computing Systems (PDCS)*. 2004, pp. 432–439.

[69] Rajiv Maheswaran, Milind Tambe, Emma Bowring, Jonathan Pearce, and Pradeep Varakantham. "Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Event Scheduling." In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2004, pp. 310–317.

[70] Radu Marinescu and Rina Dechter. "AND/OR Branch-and-Bound for Graphical Models." In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*. 2005, pp. 224–229.

[71] Radu Marinescu and Rina Dechter. "AND/OR Branch-and-Bound search for combinatorial optimization in graphical models." In: *Artificial Intelligent* 173.16-17 (2009), pp. 1457–1491.

[72] Travis Mick, Reza Tourani, and Satyajayant Misra. “LASeR: Lightweight Authentication and Secured Routing for NDN IoT in Smart Cities.” In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 755–764.

[73] Sam Miller, Sarvapali Ramchurn, and Alex Rogers. “Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid.” In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2012, pp. 281–288.

[74] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. “Internet of things: Vision, applications and research challenges.” In: *Ad Hoc Networks* 10.7 (2012), pp. 1497–1516.

[75] Pragnesh Modi. “Distributed Constraint Optimization for Multiagent Systems.” PhD thesis. Los Angeles (United States): University of Southern California, 2003.

[76] Pragnesh Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. “ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees.” In: *Artificial Intelligence* 161.1–2 (2005), pp. 149–180.

[77] Paul Morris. “The Breakout Method for Escaping from Local Minima.” In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Ed. by Richard Fikes and Wendy G. Lehnert. 1993, pp. 40–45.

[78] Arnon Netzer, Alon Grubshtein, and Amnon Meisels. “Concurrent forward bounding for distributed constraint optimization problems.” In: *Artificial Intelligence* 193 (2012), pp. 186–216.

[79] D. T. Nguyen, W. Yeoh, H. C. Lau, S. Zilberstein, and C. Zhang. “Decentralized Multi-Agent Reinforcement Learning in Average-Reward Dynamic DCOPs.” In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2014, pp. 1447–1455.

[80] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. “Distributed Gibbs: A Memory-Bounded Sampling-Based DCOP Algorithm.” In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2013, pp. 167–174.

[81] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. “Stochastic Dominance in Stochastic DCOPs for Risk-Sensitive Applications.” In: *Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2012, pp. 257–264.

[82] Noam Nisan and Amir Ronen. “Computationally feasible VCG mechanisms.” In: *Journal of Artificial Intelligence Research* 29 (2007), pp. 19–47.

[83] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. “DUCT: An Upper Confidence Bound Approach to Distributed Constraint Optimization Problems.” In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2012, pp. 528–534.

[84] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. "DUCT: An Upper Confidence Bound Approach to Distributed Constraint Optimization Problems." In: *ACM Transactions on Intelligent Systems and Technology* 8.5 (2017), 69:1–69:27.

[85] Javier Parapar and Filip Radlinski. "Diverse User Preference Elicitation with Multi-Armed Bandits." In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 130–138.

[86] Adrian Petcu and Boi Faltings. "A Scalable Method for Multiagent Constraint Optimization." In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*. 2005, pp. 1413–1420.

[87] Adrian Petcu and Boi Faltings. "ODPOP: An Algorithm for Open/Distributed Constraint Optimization." In: *Proceedings of The Conference on Advances in Artificial Intelligence (AAAI)*. 2006, pp. 703–708.

[88] Adrian Petcu and Boi Faltings. "Superstabilizing, Fault-Containing Multiagent Combinatorial Optimization." In: *Proceedings of the Conference on Advance in Artificial Intelligence (AAAI)*. 2005, pp. 449–454.

[89] Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. "Computing Possible and Necessary Winners from Incomplete Partially-Ordered Preferences." In: *Proceedings of the European Conference on Artificial Intelligence, (ECAI)*. Vol. 141, pp. 767–768.

[90] Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. "Incompleteness and incomparability in preference aggregation: Complexity results." In: *Artificial Intelligence* 175.7-8 (2011), pp. 1272–1289.

[91] Ira Pohl. "Heuristic search viewed as path finding in a graph." In: *Artificial Intelligence* 1.3-4 (1970), pp. 193–204.

[92] Ira Pohl. "The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1973, pp. 12–17.

[93] Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. "Coached Conversational Preference Elicitation: A Case Study in Understanding Movie Preferences." In: (2019).

[94] Gelli Ravikumar, Dan Ameme, Satyajayant Misra, Sukumar Brahma, and Reza Tourani. "iCASM: An Information-Centric Network Architecture for Wide Area Measurement Systems." In: *IEEE Transactions on Smart Grid* 11.4 (2020), pp. 3418–3427.

[95] L.C.A. Rodrigues and L. Magatao. "Enhancing Supply Chain Decisions Using Constraint Programming: A Case Study." In: *Proceedings of the Mexican International Conference on Artificial Intelligence*. 2007, pp. 1110–1121.

[96] Francesca Rossi, Charles J. Petrie, and Vasant Dhar. "On the Equivalence of Constraint Satisfaction Problems." In: *Proceedings of the European Conference on Artificial Intelligence (ECAI)*. 1990, pp. 550–556.

[97] Francesca Rossi, Kristen Brent Venable, and Toby Walsh. "Preferences in Constraint Satisfaction and Optimization." In: *AI Magazine* 29.4 (2008), pp. 58–68.

[98] Pierre Rust, Gauthier Picard, and Fano Ramparany. "Using Message-Passing DCOP Algorithms to Solve Energy-Efficient Smart Environment Configuration Problems." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2016, pp. 468–474.

[99] Tuomas Sandholm. "Algorithm for Optimal Winner Determination in Combinatorial Auctions." In: *Artificial Intelligence* 135.1–2 (2002), pp. 1–54.

[100] Tugba Saraç and Aydin Sipahioglu. "Generalized quadratic multiple knapsack problem and two solution approaches." In: *Computers & Operations Research* 43 (2014), pp. 78–89.

[101] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. "Valued Constraint Satisfaction Problems: Hard and Easy Problems." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1995, pp. 631–637.

[102] Linda G. Shapiro and Robert M. Haralick. "Structural descriptions and inexact matching." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 3.5 (1981), pp. 504–519.

[103] R. Stranders, F. Delle Fave, A. Rogers, and N. Jennings. "DCOPs and Bandits: Exploration and Exploitation in Decentralised Coordination." In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2012, pp. 289–297.

[104] Peter J Stuckey, Ralph Becket, Sebastian Brand, Mark Brown, Thibaut Feydy, Julien Fischer, Maria Garcia de la Banda, Kim Marriott, and Mark Wallace. "The evolving world of MiniZinc." In: *Constraint Modelling and Reformulation* (2007), pp. 156–170.

[105] Xiaoxun Sun, Marek Druzdzel, and Changhe Yuan. "Dynamic Weighting A* Search-based MAP Algorithm for Bayesian Networks." In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. 2007, pp. 2385–2390.

[106] Atena M. Tabakhi. "Parameterized Heuristics for Incomplete Weighted CSPs." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2019, pp. 10045–10046.

[107] Atena M. Tabakhi. "Preference elicitation in DCOPs for scheduling devices in smart buildings." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2017, pp. 4989–4990.

[108] Atena M. Tabakhi. "Pseudo-Tree Construction Heuristics for DCOPs with Variable Communication Times." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. Ed. by Dale Schuurmans and Michael P. Wellman. 2016, pp. 4238–4239.

[109] Atena M Tabakhi, Tiep Le, Ferdinando Fioretto, and William Yeoh. "Preference elicitation for DCOPs." In: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*. 2017, pp. 278–296.

[110] Atena M. Tabakhi, Reza Tourani, Francisco Natividad, William Yeoh, and Satyajayant Misra. "Pseudo-Tree Construction Heuristics for DCOPs and Evaluations on the ns-2 Network Simulator." In: *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. 2017, pp. 1105–1112.

[111] Atena M. Tabakhi, Yuanming Xiao, William Yeoh, and Roie Zivan. "Branch-and-Bound Heuristics for Incomplete DCOPs." In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2021, pp. 1677–1679.

[112] Atena M. Tabakhi, William Yeoh, and Ferdinando Fioretto. "The Smart Appliance Scheduling Problem: A Bayesian Optimization Approach." In: *Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA)*. 2020, pp. 100–115.

[113] Atena M. Tabakhi, William Yeoh, Reza Tourani, Francisco Natividad, and Satyajayant Misra. "Communication-Sensitive Pseudo-Tree Heuristics for DCOP Algorithms." In: *International Journal of Artificial Intelligence Tools (IJAIT)* 27.7 (2018), 1860008:1–1860008:24.

[114] Atena M. Tabakhi, William Yeoh, and Makoto Yokoo. "Parameterized Heuristics for Incomplete Weighted CSPs with Elicitation Costs." In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2019, pp. 476–484.

[115] M. Taylor, M. Jain, P. Tandon, M. Yokoo, and M. Tambe. "Distributed on-line multi-agent optimization under uncertainty: Balancing exploration and exploitation." In: *Advances in Complex Systems* 14.03 (2011), pp. 471–528.

[116] Stefano Teso, Paolo Dragone, and Andrea Passerini. "Coactive Critiquing: Elicitation of Preferences and Features." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2017, pp. 2639–2645.

[117] Reza Tourani, Austin Bos, Satyajayant Misra, and Flavio Esposito. "Towards security-as-a-service in multi-access edge." In: *Proceedings of the ACM/IEEE Symposium on Edge Computing (SEC)*. Ed. by Songqing Chen, Ryokichi Onishi, Ganesh Ananthanarayanan, and Qun Li. ACM, 2019, pp. 358–363.

[118] Reza Tourani, Satyajayant Misra, Travis Mick, Sukumar Brahma, Milan Biswal, and Dan Ameme. "iCenS: An information-centric smart grid network architecture." In: *The IEEE International Conference on Smart Grid Communications (Smart Grid Comm)*. 2016, pp. 417–422.

[119] Reza Tourani, Satyajayant Misra, Travis Mick, and Gaurav Panwar. "Security, Privacy, and Access Control in Information-Centric Networking: A Survey." In: *IEEE Communications Surveys and Tutorials* 20.1 (2018), pp. 566–600.

[120] Reza Tourani, Abderrahmen Mtibaa, and Satyajayant Misra. "Distributed Data-Gathering and -Processing in Smart Cities: An Information-Centric Approach." In: *Open Journal of Internet of Things* 5.1 (2019), pp. 93–104.

[121] Walid Trabelsi, Kenneth N. Brown, and Barry O'Sullivan. "Preference Elicitation and Reasoning While Smart Shifting of Home Appliances." In: *Energy Procedia* 83 (2015), pp. 389 –398.

[122] Ngoc Cuong Truong, Tim Baarslag, Sarvapali D. Ramchurn, and Long Tran-Thanh. "Interactive Scheduling of Appliance Usage in the Home." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2016, pp. 869–877.

[123] Suguru Ueda, Atsushi Iwasaki, Makoto Yokoo, Marius Silaghi, Katsutoshi Hirayama, and Toshihiro Matsui. "Coalition Structure Generation Based on Distributed Constraint Optimization." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2010, pp. 197–203.

[124] Phebe Vayanos, Duncan McElfresh, Yingxiao Ye, John Dickerson, and Eric Rice. "Active Preference Elicitation via Adjustable Robust Optimization." In: *arXiv preprint arXiv:2003.01899* (2020).

[125] Ivan Vendrov, Tyler Lu, Qingqing Huang, and Craig Boutilier. "Gradient-Based Optimization for Bayesian Preference Elicitation." In: *The 34th Conference on Advances in Artificial Intelligence (AAAI)*. 2020, pp. 10292–10301.

[126] Paolo Viappiani and Craig Boutilier. "Optimal Bayesian Recommendation Sets and Myopically Optimal Choice Query Sets." In: *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 2010, pp. 2352–2360.

[127] William Vickrey. "Counterspeculation, auctions, and competitive sealed tenders." In: *The Journal of finance* 16.1 (1961), pp. 8–37.

[128] Meritxell Vinyals, Juan Rodríguez-Aguilar, and Jesús Cerquides. "Constructing a Unifying Theory of Dynamic Programming DCOP Algorithms via the Generalized Distributive Law." In: *Journal of Autonomous Agents and Multi-Agent Systems* 22.3 (2011), pp. 439–464.

[129] Richard Wallace and Eugene Freuder. "Stable Solutions for Dynamic Constraint Satisfaction Problems." In: *CP*. 1998, pp. 447–461.

[130] Toby Walsh. "Complexity of terminating preference elicitation." In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Ed. by Lin Padgham, David C. Parkes, Jörg P. Müller, and Simon Parsons. 2008, pp. 967–974.

[131] Toby Walsh. "Uncertainty in preference elicitation and aggregation." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. 2007, pp. 3–8.

[132] Tianhan Wang and Craig Boutilier. "Incremental Utility Elicitation with the Minimax Regret Decision Criterion." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2003, pp. 309–318.

[133] F. Wu and N. Jennings. "Regret-based Multi-agent Coordination with Uncertain Task Rewards." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2014, pp. 1492–1499.

[134] Yuanming Xiao, Atena M. Tabakhi, and William Yeoh. "Embedding Preference Elicitation Within the Search for DCOP Solutions." In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2020, pp. 2044–2046.

[135] Roland HC Yap, Wei Xia, and Ruiwei Wang. "Generalized Arc Consistency Algorithms for Table Constraints: A Summary of Algorithmic Ideas." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. Vol. 34. 09. 2020, pp. 13590–13597.

[136] William Yeoh, Ariel Felner, and Sven Koenig. "BnB-ADOPT: An Asynchronous Branch-and-Bound DCOP Algorithm." In: *Journal of Artificial Intelligence Research* 38 (2010), pp. 85–133.

[137] William Yeoh, Xiaoxun Sun, and Sven Koenig. "Trading Off Solution Quality for Faster Computation in DCOP Search Algorithms." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2009, pp. 354–360.

[138] William Yeoh, Pradeep Varakantham, Xiaoxun Sun, and Sven Koenig. "Incremental DCOP Search Algorithms for Solving Dynamic DCOPs." In: *Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. 2015, pp. 257–264.

[139] William Yeoh and Makoto Yokoo. "Distributed Problem Solving." In: *AI Magazine* 33.3 (2012), pp. 53–65.

[140] Makoto Yokoo. *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Springer Science & Business Media, 2012.

[141] Makoto Yokoo, Edmund H Durfee, Toru Ishida, and Kazuhiro Kuwabara. "The distributed constraint satisfaction problem: Formalization and algorithms." In: *IEEE Transactions on Knowledge and Data Engineering Journal* 10.5 (1998), pp. 673–685.

[142] Neil Yorke-Smith and Carmen Gervet. "Certainty Closure: A Framework for Reliable Constraint Reasoning with Uncertainty." In: *Proceedings of Principles and Practice of Constraint Programming (CP)*. 2003, pp. 769–783.

[143] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. "Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks." In: *Artificial Intelligence Journal* 161.1-2 (2005), pp. 55–87.

[144] Zhen Zhang, Xinyue Kou, Iván Palomares, Wenyu Yu, and Junliang Gao. "Stable two-sided matching decision making with incomplete fuzzy preference relations: A disappointment theory based approach." In: *Journal of Applied Soft Computing* 84 (2019), p. 105730.

[145] Roie Zivan, Omer Lev, and Rotem Galiki. "Beyond trees: Analysis and convergence of belief propagation in graphs with multiple cycles." In: *Proceedings of the Conference on Advances in Artificial Intelligence (AAAI)*. Vol. 34. 05. 2020, pp. 7333–7340.

[146] Roie Zivan, Steven Okamoto, and Hilla Peled. "Explorative anytime local search for distributed constraint optimization." In: *Artificial Intelligence* 212 (2014), pp. 1–26.

[147] Roie Zivan, Harel Yedidsion, Steven Okamoto, Robin Glinton, and Katia Sycara. "Distributed Constraint Optimization for Teams of Mobile Sensing Agents." In: *Journal of Autonomous Agents and Multiagent Systems* 29.3 (2015), pp. 495–536.