Spring 5-2020

Embedding Preference Elicitation Within the Search for DCOP Solutions

Yuanming Xiao
Washington University in St. Louis

Washington University in St. Louis

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
William Yeoh, Chair
Ayan Chakrabarti
Chien-Ju Ho

Embedding Preference Elicitation Within the Search for DCOP Solutions

by

Yuanming Xiao

A thesis presented to the Graduate School of Arts and Sciences
of Washington University in partial fulfillment of the
requirements for the degree of

Master of Science

May 2020
Saint Louis, Missouri

Contents

List of Tables . . . . **iii**

List of Figures . . . . **iv**

Acknowledgments . . . . **v**

Abstract . . . . **vi**

1 Introduction **1**
- 1.1 Motivation Domain: Distributed Meeting Scheduling Problem 2

2 Related Work **4**

3 Background **6**
- 3.1 Distributed Constraint Optimization Problems 6
- 3.2 Synchronous Branch-and-Bound 7

4 Solving Incomplete Distributed Constraint Optimization Problems **8**
- 4.1 Incomplete DCOPs 8
- 4.2 Using SyncBB to Solve I-DCOPs 9
- 4.3 Cost-Estimate Heuristics 13
  - 4.3.1 Child's Ancestors' Constraints (CAC) Heuristic 13
  - 4.3.2 Agent's Descendants' Constraints (ADC) Heuristic 15
- 4.4 Value- and Variable-Ordering Heuristics 16

5 Theoretical Results **17**

6 Empirical Evaluations **19**
- 6.1 Random Graphs 19
- 6.2 Distributed Meeting Scheduling Problems 22

7 Conclusions **24**

References **25**

List of Tables

6.1 Varying Number of Agents $|\mathcal{A}|$ 20
6.2 Random Graphs Varying Domain Size $|D_i|$ on Problems with Elicitation Costs . . 21
6.3 Meeting Scheduling Problems Varying Number of Variables $|\mathcal{X}|$ 22

List of Figures

4.1 Example of Incomplete DCOP with Elicitation Costs 9
4.2 Labels of Search Tree Nodes 10
4.3 Simplified Execution Trace of SyncBB Without Heuristics 11
4.4 Simplified Execution Trace of SyncBB with CAC Heuristic 15

6.1 Random Graphs Varying Additive Weights 21

Acknowledgments

First, I would like to thank Professor William Yeoh and Ms. Atena M. Tabakhi for providing me such an opportunity and advising me through this project.

Also, I would like to thank many graduate students and distinguished faculty within my department who have reviewed this thesis and helped support the related research.

Last but not least, a special thanks to all my *Xiao-Di*s and Yizhi Jiang.

Yuanming Xiao

Washington University in Saint Louis
May 2020

ABSTRACT OF THE THESIS

Embedding Preference Elicitation Within the Search for DCOP Solutions

by

Yuanming Xiao

Master of Science in Computer Science

Washington University in St. Louis, May 2020

Research Advisor: Professor William Yeoh

The *Distributed Constraint Optimization Problem* (DCOP) formulation is a powerful tool to model cooperative multi-agent problems, especially when they are sparsely constrained with one another. A key assumption in this model is that all constraints are fully specified or known a priori, which may not hold in applications where constraints encode preferences of human users. In this thesis, we extend the 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 propose two parameterized heuristics that can be used in conjunction with Synchronous Branch-and-Bound to solve I-DCOPs. These heuristics allow users to trade off solution quality for faster runtimes and a smaller number of elicitations. They also provide theoretical quality guarantees for problems where elicitations are free. Our model and heuristics thus extend the state of the art in distributed constraint reasoning to better model and solve distributed agent-based applications with user preferences.

Chapter 1

Introduction

The *Distributed Constraint Optimization Problem* (DCOP) [43, 48] formulation is a powerful tool to model cooperative multi-agent problems. DCOPs are well-suited to model many problems that are distributed by nature and where agents need to coordinate their value assignments to minimize the aggregate constraint costs. This model is widely employed to model distributed problems such as meeting scheduling problems [37], sensor and wireless networks [16, 74], multi-robot teams coordination [76], smart grids [41], and smart homes [56, 18].

The field of DCOP has matured significantly over the past decade since its inception [43]. DCOP researchers have proposed a wide variety of solution approaches, from complete approaches that use distributed search-based techniques [43, 71] to distributed inference-based techniques [48]. There is also a significant body of work on incomplete methods that can be similarly categorized into local search based methods [16], GDL-based techniques [68], and sampling-based methods [47]. Researchers have also proposed the use of other off-the-shelf solvers such as logic programming solvers [32, 31] and mixed-integer programming solvers [26].

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 application, 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* [61]. 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. This approach suffers from two limitations: First, it assumes that the cost of eliciting constraints is uniform across all constraints. This 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, this thesis proposes the *Incomplete DCOP* (I-DCOP) model, which *integrates* both the elicitation and solving problem into a single integrated optimization problem.[1] In an I-DCOP, some constraint costs are unknown and can be elicited. Elicitation of unknown 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, this thesis introduces a number of heuristics that can be used in conjunction with commonly-used synchronous DCOP search algorithms such as Synchronous Branch-and-Bounds (SyncBB) [27]. These heuristics are also parameterized in such a way that they allow users to trade off solution quality for faster runtimes and a smaller number of elicitations. They also provide quality guarantees when solving problems without elicitation costs when the underlying DCOP search algorithm is correct and complete.

1.1 Motivation Domain: Distributed Meeting Scheduling Problem

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 [37, 48, 71]. While there are a number of possible formulations, we use the *Private Events as Variables* (PEAV) formulation proposed by Maheswaran *et al.* [37] in this thesis. 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.[2] Equality constraints are

[1]An extended abstract of this thesis was published at AAMAS 2020 [70].

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

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; 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 [37, 48, 71]. 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 necessary. 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, this thesis describes 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.

Chapter 2

Related Work

As this work lies in the intersection of constraint-based models, preference elicitation, and heuristic search, this thesis will first focus on related work in this intersection before covering the three broader areas. Aside from the work proposed by Tabakhi *et al.* [61] discussed in Chapter 1, the body of work that is most related to ours is the work on *Incomplete Weighted CSPs* (IWCSPs) [22, 62]. IWCSPs can be seen as centralized versions of I-DCOPs. Researchers have proposed a family of algorithms based on depth-first branch-and-bound 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*.

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* [75], where the outcomes of constraints are parameterized; *Open CSPs* [15], where the domains of variables and constraints are incrementally discovered; *Dynamic CSPs* [13], where the CSP can change over time; as well as distributed variants of these models [45, 34, 49].

In the context of the broader preference elicitation area, there is a very large body of work [**?**], and this thesis focuses on techniques that are most closely related to our approach. They include techniques that ask users a number of preset questions [64, 61] as well as send alerts and notification messages to interact with users [12], techniques that ask users to rank alternative options or user-provided option improvements to learn a (possibly approximately) user preference function [11, 8, 67], 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 [65, 33]. The key difference between all these approaches and ours is that they identify preferences to elicit a priori before the search while this thesis embeds the preference elicitation in the underlying DCOP search algorithm.

Finally, in the context of the broader heuristic search area, starting with Weighted A* [51], researchers have long used weighted heuristics to speed up the search process in general search problems. Researchers have also investigated the use of dynamically-changing weights [60, 52]; using weighted heuristic with other heuristic search algorithms like DFBnB [20], RBFS [28], and AND/OR search [40, 38]; as well as extending them to provide anytime characteristics [35, 24].

Chapter 3

Background

We now describe *Distributed Constraint Optimization Problems* (DCOPs) [43, 48], which we will later extend to Incomplete DCOPs, as well as the *Synchronous Branch-and-Bound* (SyncBB) algorithm [27], which we will use as the underlying DCOP search algorithm that uses our proposed heuristics.

3.1 Distributed Constraint Optimization Problems

A *Distributed Constraint Optimization Problem* (DCOP) is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \alpha \rangle$:

- $\mathcal{A} = \{a_i\}_{i=1}^{p}$ is a set of *agents*;
- $\mathcal{X} = \{x_i\}_{i=1}^{n}$ is a set of decision *variables*;
- $\mathcal{D} = \{D_x\}_{x \in \mathcal{X}}$ is a set of finite *domains* and each variable $x \in \mathcal{X}$ takes values from the set D_x;
- $\mathcal{F} = \{f_i\}_{i=1}^{m}$ is a set of *constraints*, each defined over a set of decision variables: $f_i : \prod_{x \in \mathbf{x}^{f_i}} D_x \rightarrow \mathbb{R} \cup \{\infty\}$, where infeasible configurations have ∞ costs, $\mathbf{x}^{f_i} \subseteq \mathcal{X}$ is the *scope* of f_i; and
- $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ is a *mapping function* that associates each decision variable to one agent.

A *solution* σ 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 σ 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 $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \mathcal{F}(\mathbf{x})$.

A *constraint graph* 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. 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 SyncBB and in our descriptions later on.

Finally, unless otherwise specified, we assume that each agent controls exactly one decision variable and thus use the terms "agent" and "variable" interchangeably.

3.2 Synchronous Branch-and-Bound

Synchronous Branch-and-Bound (SyncBB) [27] 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).

Chapter 4

Solving Incomplete Distributed Constraint Optimization Problems

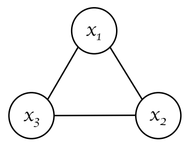
We now describe how we extend DCOPs to Incomplete DCOPs, how to solve them using SyncBB, and several novel heuristics that can be used to speed up the search.

4.1 Incomplete DCOPs

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 α are exactly the same as in a DCOP. There are two key differences:

- The set of constraints $\mathcal{F}$ are not known to an I-DCOP algorithm. 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 $\tilde{f}_i : \prod_{x \in \mathbf{x}^{f_i}} D_x \to \mathbb{R} \cup \{\infty, ?\}$, 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 $e_i : \prod_{x \in \mathbf{x}^{f_i}} D_x \to \mathbb{R}$ 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 $\tilde{\mathbf{x}}$.



(a) Constraint Graph

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

(b) Incomplete Constraint Costs and Elicitation Costs

Figure 4.1: Example of Incomplete DCOP with Elicitation Costs

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 4.1(a) shows the constraint graph of an example I-DCOP that we will use as a running example in this thesis. 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 4.1(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 thesis. 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 $\alpha_f \cdot 3 + \alpha_e \cdot 2 = 0.5 \cdot 3 + 0.5 \cdot 2 = 2.5$.

4.2 Using SyncBB to Solve I-DCOPs

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. This thesis describes below how to adapt SyncBB to solve I-DCOPs as well as how to use this algorithm as the underlying search algorithm that uses the proposed heuristics later.

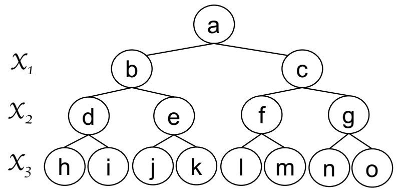


Figure 4.2: Labels of Search Tree Nodes

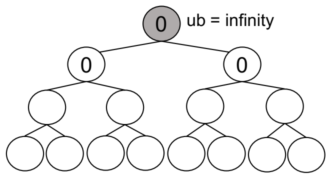
The operations of SyncBB can be visualized with search trees. Figure 4.2 shows the search tree for our example I-DCOP shown in Figures 4.1(a) and 4.1(b), where levels 1, 2, and 3 correspond to variable x_1, x_2, and x_3, respectively. Left branches correspond to the variable being assigned the value 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 node corresponds to a complete solution. These nodes also correspond to unique CPAs of agents when they run SyncBB. Labeled with an identifier so that each node of the search tree can be referred 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, denoted by $g(n)$.[3] This thesis refers to the weighted sum of these values as f-values, denoted by $f(n, \tilde{\mathbf{x}}) = \alpha_f \cdot g(n) + \alpha_e \cdot \mathcal{E}(\tilde{\mathbf{x}})$.

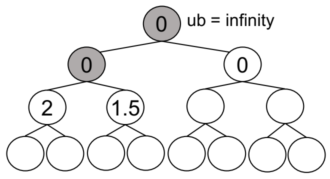
Figure 4.3 shows a simplified execution trace of SyncBB, where the CPA at each step of the algorithm corresponds to the shaded branch of search tree. For example, the CPA is $\langle x_1 = 1, x_2 = 0 \rangle$ in Step 7. The numbers in the shaded nodes correspond to their f-values when those nodes were 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 also shown in the figure beside the root node.

Here is the more detailed tracing process. 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.

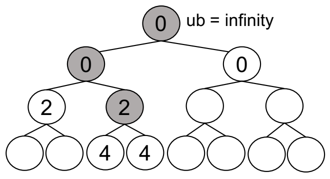
[3]A* notations [25] are used here.



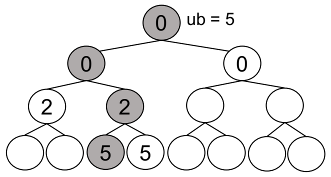
(a) Step 1



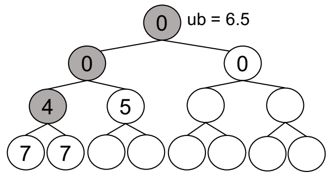
(b) Step 2



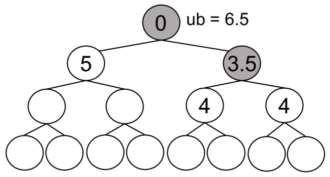
(c) Step 3



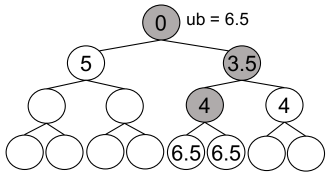
(d) Step 4



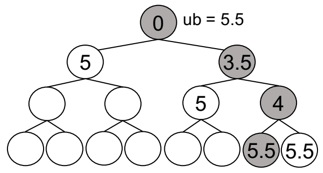
(e) Step 5



(f) Step 6



(g) Step 7



(h) Step 8

Figure 4.3: Simplified Execution Trace of SyncBB Without Heuristics

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 $\tilde{f}_1(x_1 = 0, x_2 = 0)$ and $\tilde{f}_1(x_1 = 0, x_2 = 1)$. Therefore, it estimates the cost of both nodes and expand the node with the smallest estimated cost.

Let's assume that all the agents know that a lower bound $\mathcal{L}$ on all the constraint costs is 1. In other words, all constraint costs are no smaller than 1. 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 (= $e_1(x_1 = 0, x_2 = 0) = \alpha_e \cdot 3$). 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 $f_1(x_1 = 0, x_2 = 1) = 2$, 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 $f_2(x_1 = 0, x_3 = 0) = 3$ and $f_3(x_2 = 1, x_3 = 0) = 1$, 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 $e_1(x_1 = 0, x_2 = 0)$ + 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.

4.3 Cost-Estimate Heuristics

To speed up the SyncBB algorithm, 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.

This thesis describes below two cost-estimate heuristics that can be used in conjunction with SyncBB to solve I-DCOPs. 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. In the worst case since all costs are non-negative, for the running example, the lower bound ($\mathcal{L}$) is set to 1. 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 $\epsilon \geq 0$. When using these parameters, SyncBB will prune a node n if:

$$w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon \geq \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}}) \tag{4.1}$$

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 ϵ 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), this thesis theoretically shows that the cost of solutions found are guaranteed to be at most $w \cdot OPT + \epsilon$, where OPT is the optimal solution cost.

4.3.1 Child's Ancestors' Constraints (CAC) Heuristic

First heuristic is called *Child's Ancestors Constraints* (CAC) heuristic. It 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,

$$\begin{aligned} h(x_i = d_i) = & \min_{d_c \in D_c} \left[\alpha_f \cdot \hat{f}(x_i = d_i, x_c = d_c) + \alpha_e \cdot e(x_i = d_i, x_c = d_c) + h(x_c = d_c) \right] \\ & + \sum_{x_k \in Anc(x_c) \setminus \{x_i\}} \min_{d_k \in D_k} \left[\alpha_f \cdot \hat{f}(x_c = d_c, x_k = d_k) + \alpha_e \cdot e(x_c = d_c, x_k = d_k) \right] \end{aligned} \tag{4.2}$$

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 $\mathbf{x}$.

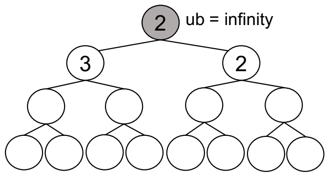
For a parent x_p of a leaf agent x_l, the heuristic value $h(x_p = d_p)$ is then the minimal constraint and elicitation cost between the two agents, under the assumption that the parent takes on value d_p, and the sum of the minimal constraint cost of the leaf agent with its ancestors. As the heuristic of a child agent is included in the heuristic of the parent agent, this summation of costs are recursively aggregated up the pseudo-chain.

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 computes the latter term in Equation (4.2):

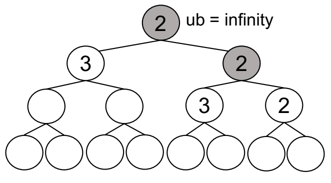
$$\sum_{x_k \in Anc(x_l)} \min_{d_k \in D_k} \left[\alpha_f \cdot \hat{f}(x_l = d_l, x_k = d_k) + \alpha_f \cdot e(x_l = d_l, x_k = d_k) \right] \tag{4.3}$$

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 (4.2), computes the latter term similar to Equation (4.3) 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.

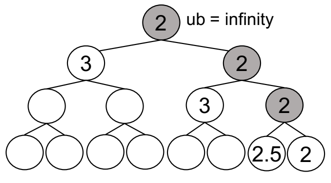
Figure 4.4 shows the order of node expansions conducted by SyncBB using the CAC heuristic on our example I-DCOP. Note that the algorithm needs to only expand 4 nodes and elicit 1 unknown constraint cost before returning a solution of weighted cost 2.5. In contrast, SyncBB without



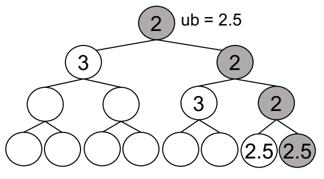
(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Figure 4.4: Simplified Execution Trace of SyncBB with CAC Heuristic

heuristics expanded 9 nodes elicited 4 unknown constraint costs before returning a solution of weighted cost 5.5 (see Figure 4.3).

4.3.2 Agent's Descendants' Constraints (ADC) Heuristic

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,

$$\begin{aligned} h(x_i = d_i) = \min_{d_c \in D_c} \Big[\alpha_f \cdot \hat{f}(x_i = d_i, x_c = d_c) + \alpha_e \cdot e(x_i = d_i, x_c = d_c) + h(x_c = d_c) \Big] \\ + \sum_{x_j \in Des(x_i) \setminus \{x_c\}} \min_{d_j \in D_j} \Big[\alpha_f \cdot \hat{f}(x_i = d_i, x_j = d_j) + \alpha_e \cdot e(x_i = d_i, x_j = d_j) \Big] \end{aligned} \tag{4.4}$$

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.

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 (4.4) 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.

4.4 Value- and Variable-Ordering Heuristics

Instead of choosing a random order to explore the different values of an agent, this thesis orders 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, this thesis orders 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.

Chapter 5

Theoretical Results

Theorem 1 *The computation of both CAC and ADC heuristics require $O(|\mathcal{A}|)$ number of messages.*

Proof : Both heuristics are recursively computed starting from the leaf to the root and will therefore take exactly $|\mathcal{A}| - 1$ number of messages. $\square$

Lemma 1 *When all elicitation costs are zero, the CAC heuristic is admissible.*

Proof : To prove admissibility, 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_i from the leaf:

$$h(x_i = d_i) = \min_{d_c \in D_c} \left[\alpha_f \cdot \hat{f}(x_i = d_i, x_c = d_c) + \alpha_e \cdot e(x_i = d_i, x_c = d_c) + h(x_c = d_c) \right] + \sum_{x_k \in Anc(x_c) \setminus \{x_i\}} \min_{d_k \in D_k} \left[\alpha_f \cdot \hat{f}(x_c = d_c, x_k = d_k) + \alpha_e \cdot e(x_c = d_c, x_k = d_k) \right]$$

where x_c is the next agent in the ordering (i.e., the $(k-1)$-th agent), $Anc(x_i)$ is the set of variables higher up in the ordering that x_i is constrained with, and each estimated cost function $\hat{f}$ is as defined by the CAC heuristic. First, it is easy to see:

$$\hat{f}(x_j = d_j, x_k = d_k) \leq f(x_j = d_j, x_k = d_k)$$

for any pair of agents x_j and x_k 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:

$$\begin{aligned}
h(x_i = d_i) = & \min_{d_c \in D_c} \left[\alpha_f \cdot \hat{f}(x_i = d_i, x_c = d_c) + \alpha_e \cdot e(x_i = d_i, x_c = d_c) \right] + h(x_c = d_c) \\
& + \sum_{x_k \in Anc(x_c) \setminus \{x_i\}} \min_{d_k \in D_k} \left[\alpha_f \cdot \hat{f}(x_c = d_c, x_k = d_k) + \alpha_e \cdot e(x_c = d_c, x_k = d_k) \right] \\
\leq & \min_{d_c \in D_c} \left[\alpha_f \cdot f(x_i = d_i, x_c = d_c) + h(x_c = d_c) \right] \\
& + \sum_{x_k \in Anc(x_c) \setminus \{x_i\}} \min_{d_k \in D_k} \alpha_f \cdot f(x_c = d_c, x_k = d_k) \\
\leq & \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n)
\end{aligned}$$

where node n corresponds to the agent x_i taking on its value $d_i \in D_i$. □

Lemma 2 *When all elicitation costs are zero, the ADC heuristic is admissible.*

Proof: The proof is similar to the proof for Lemma 1. □

Theorem 2 *When all elicitation costs are zero, SyncBB with either the CAC or 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 [71] 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 Lemmas 1 and 2. □

Chapter 6

Empirical Evaluations

We evaluate SyncBB using our two heuristics – CAC and ADC – against a baseline without heuristics on I-DCOPs with and without elicitation costs. We evaluate them on random graphs and distributed meeting scheduling problems, 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 the algorithm terminates, and the runtimes of the algorithms (in sec). In all experiments we set $\alpha_f = \alpha_e = 0.5$. Data points are averaged of over 50 instances.

6.1 Random Graphs

We generate 50 random (binary) graphs [14], where we vary the number of agents/variables $|\mathcal{A}|$ from 10 to 20; the domain size $|D_i|$ for all variables $x_i \in \mathcal{X}$ from 2 to 10 ; the user-defined relative weight w from 1 to 10; and the user-defined additive weight ϵ 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: $|\mathcal{X}| = 10$, $|D_i| = 2$, $w = 1$, and $\epsilon = 0$. All constraint costs are randomly sampled from $[2, 5]$ and all elicitation costs are randomly sampled from $[0, 20]$.

Tables 6.1 and 6.2, tabulates our empirical results, where we vary the number of agents $|\mathcal{A}|$ and the domain size $|D_i|$ respectively. We make the following observations:

(a) SyncBB Without Heuristics

$\|\mathcal{A}\|$	# unk. costs	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
10	43	40.62	5.09E-01	51.86	1.65E+03	17.98	3.27E-02	237.90	59.64	178.26	5.92E+01
12	62	59.04	1.99E+00	76.30	6.76E+03	25.60	4.38E-02	349.54	87.96	261.58	1.14E+02
14	86	82.44	8.02E+00	107.14	2.36E+04	35.30	5.95E-02	484.64	122.22	362.42	1.17E+02
16	115	111.74	3.22E+01	145.32	9.35E+04	47.88	8.04E-02	636.10	163.20	472.90	1.58E+02
18	146	140.84	1.18E+02	185.06	3.48E+05	60.52	1.29E-01	803.50	205.50	598.00	2.67E+02
20	182	177.08	1.23E+03	231.64	1.36E+06	70.64	1.63E-01	978.00	258.88	725.12	2.79E+02

(b) SyncBB with CAC Heuristic

10	43	38.06	2.38E-01	51.86	7.59E+02	12.90	2.15E-02	185.44	61.18	124.26	2.21E+01
12	62	57.68	9.19E-01	76.30	3.02E+03	18.72	1.74E-02	271.72	89.14	182.58	2.87E+01
14	86	80.28	3.43E+00	107.14	9.55E+03	25.90	3.60E-02	379.58	124.34	255.24	3.82E+01
16	115	110.22	1.35E+01	145.32	3.75E+04	35.16	2.84E-02	506.38	165.04	341.34	4.48E+01
18	146	139.30	4.41E+01	185.06	1.22E+05	44.80	4.78E-02	642.48	206.24	436.24	6.10E+01
20	182	174.92	3.67E+02	231.64	4.09E+05	54.88	6.29E-02	796.32	258.28	538.04	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	76.30	6.25E+03	19.80	1.77E-02	267.36	88.60	178.76	3.06E+01
14	86	81.28	7.81E+00	107.14	2.23E+04	26.72	3.20E-02	375.30	122.22	253.08	3.25E+01
16	115	110.18	3.22E+01	145.32	8.89E+04	35.98	2.63E-02	496.68	164.80	331.88	3.96E+01
18	146	139.68	1.23E+02	185.06	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 6.1: Varying Number of Agents $|\mathcal{A}|$

- As expected, the runtimes and number of unknown costs elicited by all algorithms increase with increasing number of agents $|\mathcal{A}|$ and domain size $|D_i|$. 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 $|D_i|$. 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 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.

$\|D_i\|$	# unk. costs	SyncBB Without Heuristics						SyncBB with CAC Heuristic					
		# of elic.	runtime	total cost	const. cost	elic. cost	# of nodes expanded	# of elic.	runtime	total cost	const. cost	elic. cost	# of nodes expanded
2	43	17.98	3.27E-02	237.90	59.64	178.26	5.92E+01	12.90	2.15E-02	185.44	61.18	124.26	2.21E+01
4	173	19.67	4.84E-02	254.10	58.85	195.25	1.96E+02	14.50	1.46E-02	202.12	58.73	143.39	4.68E+01
6	389	20.00	5.54E-02	254.66	58.72	195.94	3.97E+02	15.36	1.03E-02	212.30	58.44	153.86	7.01E+01
8	691	20.48	1.31E-01	253.66	58.46	195.20	1.43E+03	15.98	1.13E-02	213.90	58.22	155.68	9.89E+01
10	1080	20.70	2.19E-01	260.02	57.56	202.46	2.95E+03	16.10	1.07E-02	214.78	56.58	158.20	1.18E+02

Table 6.2: Random Graphs Varying Domain Size $|D_i|$ on Problems with Elicitation Costs

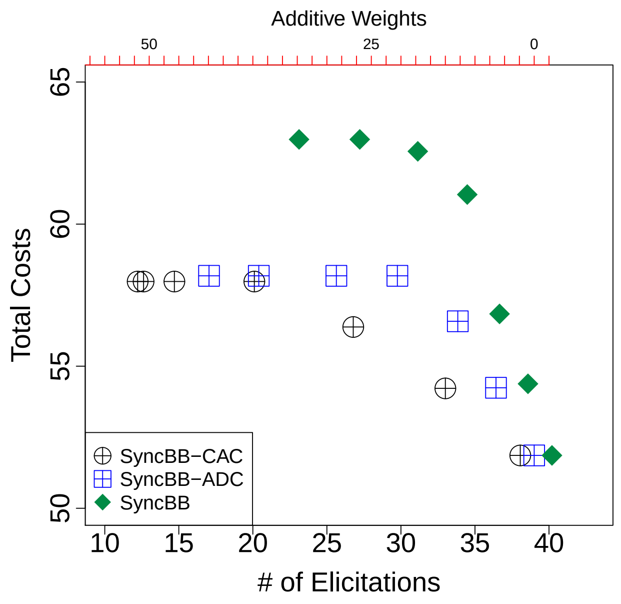


Figure 6.1: Random Graphs Varying Additive Weights

- 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. Therefore, for space reasons, we omit the results for SyncBB with ADC in Table 6.2.
- Overall, the use of heuristics reduces the number of unknown costs elicited by up to 22% and the runtime by up to 57%, when elicitation is not free, highlighting the strengths of using our proposed heuristics.

Figure 6.1 plots our empirical results, where we vary the user-defined additive bound (weight) ϵ 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

(a) 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	15	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	157.28	9.06E+05	13.56	1.83E+02	291.70	158.40	133.30	7.79E+05

(b) With CAC Heuristic

9	15	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	157.28	7.26E+05	10.72	1.27E+02	290.7	158.40	132.30	5.49E+05

(c) With ADC Heuristic

9	15	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	157.28	7.48E+05	10.64	1.15E+02	291.30	157.80	133.50	4.88E+05

Table 6.3: Meeting Scheduling Problems Varying Number of Variables $|\mathcal{X}|$

for one of the algorithms with one of the values of ϵ. Data points for smaller values of ϵ 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 ϵ increases, the number of elicitations decreases. However, this comes at the cost of larger total costs. Between the three algorithms, SyncBB with CAC is the best, followed by SyncBB with ADC, and SyncBB without heuristics.

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.

6.2 Distributed Meeting Scheduling Problems

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 18. All time preferences (constraint costs) and elicitation costs are randomly sampled from $[0, 20]$. Table 6.3 tabulates our empirical results, where the trends are similar to those in random graphs.

Chapter 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 propose *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. Additionally, 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 fewer number of elicitations. Further, in problems where elicitations are free, they provide theoretical quality guarantees on the solutions found. Our empirical results show that using our heuristics allow SyncBB to find solutions faster and with fewer elicitations. On problems without elicitation costs, CAC is also shown to dominate ADC. In conclusion, our new model and heuristics improve the practical applicability of DCOPs as they are now better suited to model multi-agent applications with user preferences.

References

[1] Slim Abdennadher and Hans Schlenker. Nurse Scheduling using Constraint Logic Programming. In *IAAI*, pages 838–843, 1999.

[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. *Artificial Intelligence*, 212:59–79, 2014.

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

[4] Fahiem Bacchus and Adam J. Grove. Utility independence in a qualitative decision theory. In *KR*, pages 542–552, 1996.

[5] E Balas and P Thoth. Branch and bound methods, the traveling salesman problem, el lawler, et al. Wiley, 1985.

[6] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM,*, 44(2):201–236, 1997.

[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. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[8] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8-9):686–713, 2006.

[9] Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.

[10] Federico Campeotto, Alessandro Dal Palù, Agostino Dovier, Ferdinando Fioretto, and Enrico Pontelli. A constraint solver for flexible protein model. *Journal of Artificial Intelligence Research*, 48:953–1000, 2013.

[11] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *AAAI*, pages 363–369, 2000.

[12] 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 *CHI*, pages 813–822, 2014.

[13] Rina Dechter and Avi Dechter. Belief maintenance in dynamic constraint networks. In *AAAI*, pages 37–42, 1988.

[14] Paul Erdös and Alfréd Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[15] Boi Faltings and Santiago Macho-Gonzalez. Open constraint programming. *Artificial Intelligence*, 161(1-2):181–208, 2005.

[16] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *AAMAS*, pages 639–646, 2008.

[17] Ferdinando Fioretto, Agostino Dovier, and Enrico Pontelli. Constrained community-based gene regulatory network inference. *ACM Transactions on Modeling and Computer Simulation*, 25(2):11:1–11:26, 2015.

[18] Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. A multiagent system approach to scheduling devices in smart homes. In *AAMAS* , pages 981–989, 2017.

[19] Ferdinando Fioretto, William Yeoh, Enrico Pontelli, Ye Ma, and Satishkumar Ranade. A distributed constraint optimization (DCOP) approach to the economic dispatch with demand response. In *AAMAS*, pages 999–1007, 2017.

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

[21] Jonathan Gaudreault, Jean-Marc Frayret, and Gilles Pesant. Distributed search for supply chain coordination. *Computers in Industry*, 60(6):441–451, 2009.

[22] 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. *Artificial Intelligence*, 174(3-4):270–294, 2010.

[23] Judy Goldsmith and Ulrich Junker. Preference handling for artificial intelligence. *AI Magazine*, 29(4):9–12, 2008.

[24] Eric A Hansen and Rong Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28:267–297, 2007.

[25] Paul Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC4(2):100–107, 1968.

[26] Daisuke Hatano and Katsutoshi Hirayama. DeQED: An efficient divide-and-coordinate algorithm for DCOP. In *IJCAI*, pages 566–572, 2013.

[27] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *CP*, pages 222–236, 1997.

[28] Richard Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, 1993.

[29] Akshat Kumar, Boi Faltings, and Adrian Petcu. Distributed constraint optimization with structured resource constraints. In *AAMAS*, pages 923–930, 2009.

[30] Javier Larrosa. Node and arc consistency in weighted csp. In *AAAI/IAAI*, pages 48–53, 2002.

[31] 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 *AAMAS*, pages 606–614, 2016.

[32] Tiep Le, Tran Cao Son, Enrico Pontelli, and William Yeoh. Solving distributed constraint optimization problems with logic programming. In *AAAI*, pages 1174–1181, 2015.

[33] Tiep Le, Atena M. Tabakhi, Long Tran-Thanh, William Yeoh, and Tran Cao Son. Preference elicitation with interdependency and user bother cost. In *AAMAS*, pages 1459–1467, 2018.

[34] Thomas Léauté and Boi Faltings. Distributed constraint optimization under stochastic uncertainty. In *AAAI*, pages 68–73, 2011.

[35] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *NIPS*, pages 767–774, 2004.

[36] Rajiv Maheswaran, Jonathan Pearce, and Milind Tambe. Distributed algorithms for DCOP: A graphical game-based approach. In *PDCS*, pages 432–439, 2004.

[37] 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 *AAMAS*, pages 310–317, 2004.

[38] Radu Marinescu and Rina Dechter. AND/OR branch-and-bound for graphical models. In *IJCAI*, 2005.

[39] Radu Marinescu and Rina Dechter. Best-first AND/OR search for graphical models. In *AAAI*, 2007.

[40] Radu Marinescu and Rina Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.

[41] Sam Miller, Sarvapali Ramchurn, and Alex Rogers. Optimal decentralised dispatch of embedded generation in the smart grid. In *AAMAS*, pages 281–288, 2012.

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

[43] Pragnesh Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1–2):149–180, 2005.

[44] Arnon Netzer, Alon Grubshtein, and Amnon Meisels. Concurrent forward bounding for distributed constraint optimization problems. *Artificial Intelligence*, 193:186–216, 2012.

[45] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. Stochastic dominance in stochastic DCOPs for risk-sensitive applications. In *AAMAS*, pages 257–264, 2012.

[46] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. Distributed Gibbs: A memory-bounded sampling-based DCOP algorithm. In *AAMAS*, pages 167–174, 2013.

[47] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. DUCT: An upper confidence bound approach to distributed constraint optimization problems. *ACM Transactions on Intelligent Systems and Technology*, 8(5):69:1–69:27, 2017.

[48] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 1413–1420, 2005.

[49] Adrian Petcu and Boi Faltings. Superstabilizing, fault-containing multiagent combinatorial optimization. In *AAAI*, pages 449–454, 2005.

[50] Adrian Petcu and Boi Faltings. ODPOP: An algorithm for open/distributed constraint optimization. In *AAAI*, pages 703–708, 2006.

[51] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4):193–204, 1970.

[52] Ira Pohl. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *IJCAI*, 1973.

[53] L.C.A. Rodrigues and L. Magatao. Enhancing Supply Chain Decisions Using Constraint Programming: A Case Study. In *MICAI*, pages 1110–1121, 2007.

[54] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[55] Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Preferences in constraint satisfaction and optimization. *AI Magazine*, 29(4):58–68, 2008.

[56] Pierre Rust, Gauthier Picard, and Fano Ramparany. Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems. In *IJCAI*, pages 468–474, 2016.

[57] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1–2):1–54, 2002.

[58] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *IJCAI*, pages 631–637, 1995.

[59] Linda G Shapiro and Robert M Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):504–519, 1981.

[60] Xiaoxun Sun, Marek Druzdzel, and Changhe Yuan. Dynamic weighting A* search-based MAP algorithm for Bayesian networks. In *IJCAI*, pages 2385–2390, 2007.

[61] Atena M. Tabakhi, Tiep Le, Ferdinando Fioretto, and William Yeoh. Preference elicitation for DCOPs. In *CP*, pages 278–296, 2017.

[62] Atena M. Tabakhi, William Yeoh, and Makoto Yokoo. Parameterized heuristics for Incomplete Weighted CSPs with elicitation costs. In *AAMAS*, pages 476–484, 2019.

[63] Stefano Teso, Paolo Dragone, and Andrea Passerini. Coactive critiquing: Elicitation of preferences and features. In *AAAI*, pages 2639–2645, 2017.

[64] Walid Trabelsi, Kenneth N. Brown, and Barry O'Sullivan. Preference elicitation and reasoning while smart shifting of home appliances. *Energy Procedia*, 83:389 – 398, 2015.

[65] Ngoc Cuong Truong, Tim Baarslag, Sarvapali D. Ramchurn, and Long Tran-Thanh. Interactive scheduling of appliance usage in the home. In *IJCAI*, pages 869–877, 2016.

[66] Suguru Ueda, Atsushi Iwasaki, Makoto Yokoo, Marius Silaghi, Katsutoshi Hirayama, and Toshihiro Matsui. Coalition structure generation based on distributed constraint optimization. In *AAAI*, pages 197–203, 2010.

[67] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *NIPS*, pages 2352–2360, 2010.

[68] Meritxell Vinyals, Juan Rodríguez-Aguilar, and Jesús Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *Journal of Autonomous Agents and Multi-Agent Systems*, 22(3):439–464, 2011.

[69] Richard Wallace and Eugene Freuder. Stable solutions for dynamic constraint satisfaction problems. In *CP*, pages 447–461, 1998.

[70] Yuanming Xiao, Atena M. Tabakhi, and William Yeoh. Embedding preference elicitation within the search for DCOP solutions. In *AAMAS*, 2020.

[71] William Yeoh, Ariel Felner, and Sven Koenig. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 38:85–133, 2010.

[72] William Yeoh, Xiaoxun Sun, and Sven Koenig. Trading off solution quality for faster computation in DCOP search algorithms. In *IJCAI*, 2009.

[73] William Yeoh, Pradeep Varakantham, Xiaoxun Sun, and Sven Koenig. Incremental DCOP search algorithms for solving dynamic DCOPs. In *IAT*, pages 257–264, 2015.

[74] William Yeoh and Makoto Yokoo. Distributed problem solving. *AI Magazine*, 33(3):53–65, 2012.

[75] Neil Yorke-Smith and Carmen Gervet. Certainty closure: A framework for reliable constraint reasoning with uncertainty. In *CP*, pages 769–783, 2003.

[76] Roie Zivan, Harel Yedidsion, Steven Okamoto, Robin Glinton, and Katia Sycara. Distributed constraint optimization for teams of mobile sensing agents. *Journal of Autonomous Agents and Multi-Agent Systems*, 29(3):495–536, 2015.

Embedding Preference Elicitation Within the Search for DCOP Solutions, Xiao, M.S. 2020