Incomplete Distributed Constraint Optimization Problems: Model, Algorithms, and Heuristics*

Atena M. Tabakhi[1], William Yeoh[1], and Roie Zivan[2]
{amtabakhi,wyeoh}@wustl.edu and zivanr@bgu.ac.il

[1] Washington University in St. Louis, USA
[2] Ben Gurion University of the Negev, Israel

Abstract. 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 paper, 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 extend SyncBB, a complete DCOP algorithm, and ALS-MGM, an incomplete DCOP algorithm, to solve I-DCOPs. We also propose parameterized heuristics that those algorithms can utilize to trade off solution quality for faster runtime and fewer elicitation. They also provide theoretical quality guarantees when used by SyncBB when 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.

Keywords: Multi-Agent Problems; Distributed Constraint Optimization Problems; Preference Elicitation; Distributed Problem Solving

1 Introduction

The *Distributed Constraint Optimization Problem* (DCOP) [22, 25, 6] 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 aggregated constraint costs. This model is widely employed for representing distributed problems such as meeting scheduling [19], sensor and wireless networks [37], multi-robot teams coordination [41], smart grids [21], and smart homes [7, 31].

The study and use of DCOPs have matured significantly over more than a decade since its inception [22]. DCOP researchers have proposed a wide variety of

* This research is partially supported by BSF grant #2018081 and NSF grants #1812619 and #1838364.

solution approaches, from complete approaches that use distributed search-based techniques [22, 37] to distributed inference-based techniques [25]. There is also a significant body of work on incomplete methods that can be similarly categorized into local search-based methods [39, 5], inference GDL-based techniques [35], and sampling-based methods [24].

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* [29]. 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 paper, we propose the *Incomplete DCOP* (I-DCOP) [30, 36] 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) [14] – and an incomplete algorithm – an *Anytime Local Search* (ALS) [40] variant of *Maximum Gain Message* (MGM) [18], 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 provide quality guarantees for I-DCOPs without elicitation costs when the underlying DCOP algorithm is correct and complete.

2 Background

Distributed Constraint Optimization Problems (DCOPs): A DCOP [22, 25, 6] is defined by $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \alpha \rangle$, where $\mathcal{A} = \{a_i\}_{i=1}^{p}$ 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

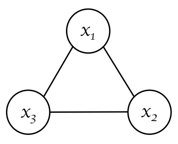
f_i; and $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ is a *mapping function* that associates each decision variable to one agent.

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.

Synchronous Branch-and-Bound (SyncBB): SyncBB [14] 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).

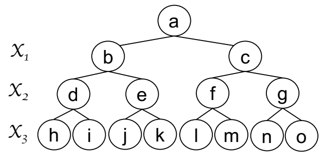
Anytime Local Search Algorithms: *Local search* DCOP algorithms (e.g., MGM, DSA) [18, 39] are synchronous iterative processes, where, in each step of the algorithm, each agent sends its value assignment to all its neighbors in the constraint graph and waits to receive the value assignments of all its 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 [40] 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 propa-



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



(a) Constraint Graph (b) Incomplete Constraint Costs and Elicitation Costs (c) Labels

Fig. 1: Incomplete DCOP with Elicitation Costs and Search Tree Nodes

gates 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.

3 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 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 $(\mathbf{x}^*, \tilde{\mathbf{x}}^*) = \text{argmin}_{(\mathbf{x}, \tilde{\mathbf{x}})} \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}})$.

Figure 1(a) shows the constraint graph of an example I-DCOP that we will use as a running example in this paper. 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 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 paper. 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 Solving 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. We describe below how to adapt SyncBB, a complete search algorithm, and ALS-MGM, a variant of the MGM [18] local search algorithm using the ALS framework [40], to solve I-DCOPs. We will also employ SyncBB and ALS-MGM as the underlying algorithms that use our proposed heuristics later.

4.1 SyncBB

The operations of SyncBB can be visualized with search trees. Figure 1(c) shows the search tree for our example I-DCOP shown in Figures 1(a) and 1(b), where levels 1, 2, and 3 correspond to variable x_1, x_2, and x_3, respectively. Left branches correspond to the variables being assigned the value 0 and right branches correspond to the variables 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. 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, denoted by $g(n)$.[3] We refer to the weighted sum of these values as f-values, denoted by $f(n, \tilde{\mathbf{x}}) = \alpha_f \cdot g(n) + \alpha_e \cdot \mathcal{E}(\tilde{\mathbf{x}})$.

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.

4.2 ALS-MGM

Like for regular DCOPs, ALS-MGM for I-DCOPs 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

[3] We use A* notations [13] here.

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.

5 SyncBB Cost-Estimate Heuristic

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:

$$w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon \geq \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}}) \tag{1}$$

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), 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.

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, $h(x_i = d_i)$ is:

$$\min_{d_c \in D_c} \Big[\alpha_f \cdot \hat{f}(x_i = d_i, x_c = d_c) + \alpha_e \cdot e(x_i = d_i, x_c = d_c) + h(x_c = d_c) \Big]$$
$$+ \sum_{x_k \in Anc(x_c) \setminus \{x_i\}} \min_{d_k \in D_k} \Big[\alpha_f \cdot \hat{f}(x_c = d_c, x_k = d_k) + \alpha_e \cdot e(x_c = d_c, x_k = d_k) \Big] \tag{2}$$

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 is 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 (2):

$$\sum_{x_k \in Anc(x_l)} \min_{d_k \in D_k} \left[\alpha_f \cdot \hat{f}(x_l = d_l, x_k = d_k) + \alpha_f \cdot e(x_l = d_l, x_k = d_k) \right] \tag{3}$$

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 (2), computes the latter term similar to Equation (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.

6 ALS-MGM Cost-Estimate 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.

Neighbors' Constraints (NHC) Heuristic: Each agent x_i computes a heuristic value $h(x_i = d_i)$ for each of its values $d_i \in D_i$ as follows:

$$\sum_{x_c \in Nh(x_i)} \min_{d_c \in D_c} \left[\alpha_f \cdot \hat{f}(x_i = d_i, x_c = d_c) + \alpha_e \cdot e(x_i = d_i, x_c = d_c) \right] \tag{4}$$

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 $\mathbf{x}$.

7 Theoretical Results

Theorem 1. *The computation of the CAC heuristic requires $O(|\mathcal{A}|)$ number of messages. The computation of the NHC heuristic requires no messages.*

Proof. The CAC heuristics is 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 and NHC heuristics are admissible.*

Proof. We only prove the admissibility of the CAC heuristic since the same proof applies to NHC. 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:

$$\begin{aligned} &h(x_k = d_k) \\ &\quad = \min_{d_c \in D_c} \Big[\alpha_f \cdot \hat{f}(x_k = d_k, x_c = d_c) + \alpha_e \cdot e(x_k = d_k, x_c = d_c) + h(x_c = d_c)\Big] \\ &\quad + \sum_{x_m \in Anc(x_c) \setminus \{x_k\}} \min_{d_m \in D_m} \Big[\alpha_f \cdot \hat{f}(x_c = d_c, x_m = d_m) + \alpha_e \cdot e(x_c = d_c, x_m = d_m)\Big] \end{aligned} \tag{5}$$

  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:

$$h(n) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n), \tag{6}$$

  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:

$$\hat{f}(x_c = d_c, x_m = d_m) \leq f(x_c = d_c, x_m = d_m), \tag{7}$$

  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:

$$\begin{aligned}
&h(x_k\!=\!d_k) \\
&\quad= \min_{d_c \in D_c}\Big[\alpha_f \cdot \hat{f}(x_k\!=\!d_k, x_c\!=\!d_c) + \alpha_e \cdot e(x_k\!=\!d_k, x_c\!=\!d_c) + h(x_c\!=\!d_c)\Big] \\
&\quad+ \sum_{x_m \in Anc(x_c)\setminus\{x_k\}} \min_{d_m \in D_m}\Big[\alpha_f \cdot \hat{f}(x_c\!=\!d_c, x_m\!=\!d_m) + \alpha_e \cdot e(x_c\!=\!d_c, x_m\!=\!d_m)\Big] \qquad (8) \\
&\quad\leq \min_{d_c \in D_c}\Big[\alpha_f \cdot f(x_k\!=\!d_k, x_c\!=\!d_c) + h(x_c\!=\!d_c)\Big] \\
&\quad+ \sum_{x_m \in Anc(x_c)\setminus\{x_k\}} \min_{d_m \in D_m} \alpha_f \cdot f(x_c\!=\!d_c, x_m\!=\!d_m) \leq \mathcal{F}(\mathbf{x}_n) - \alpha_f \cdot g(n) \qquad (9)
\end{aligned}$$

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

Theorem 2. *When all elicitation costs are zero, SyncBB with the CAC heuristic 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 [37] 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 heuristic is admissible according to Lemma 1. ■

8 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 the work proposed by Tabakhi *et al.* [29] discussed in Section 1, the body of work that is most related to ours is the work on *Incomplete Weighted CSPs* (IWCSPs) [9, 32, 10]. 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.*

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* [38], where the outcomes of constraints are parameterized; *Open CSPs* [4], where the domains of variables and constraints are incrementally discovered; *Dynamic CSPs* [3], where the CSP can change over time; as well as distributed variants of these models [23, 17].

(a) SyncBB Without Heuristic

$\|\mathcal{A}\|$	# unk.	Without Elicitation Costs				With Elicitation Costs					
	costs	#elic.	runtime	const. cost	#exp. nodes	#elic.	runtime	total cost	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

Table 1: Varying Number of Agents $|\mathcal{A}|$

In the context of the broader preference elicitation area, there is a very large body of work [11], and we focus on techniques that are most closely related to our approach. They include techniques that ask users a number of preset questions [33, 29, 28] as well as send alerts and notification messages to interact with users [2], techniques that ask users to rank alternative options or user-provided option improvements to learn a (possibly approximately) user preference function [1],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 [34, 16]. 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.

Finally, in the context of the broader heuristic search area, starting with Weighted A* [26], 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 [27]; using weighted heuristic with other heuristic search algorithms like DFBnB [8], RBFS [15], and AND/OR search [20]; as well as extending them to provide anytime characteristics [12].

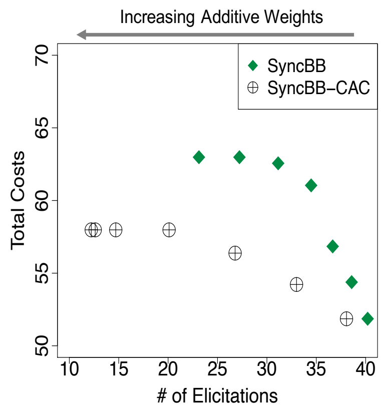
9 Empirical Evaluations

We evaluate SyncBB using the CAC heuristic and ALS-MGM using the NHC heuristic against their baselines without heuristics on I-DCOPs with and without elicitation costs. We evaluate them on random graphs where we measure the various costs of the solutions found – the cumulative constraint costs (i.e., const. cost), cumulative elicitation costs (i.e., elic. cost), and their aggregated total costs (i.e., total cost) – the number of unknown costs elicited (i.e., # elic.), the number of nodes expanded after SyncBB terminates (i.e., # exp. nodes), and the runtimes of the algorithms (in sec). In all experiments we set $\alpha_f = \alpha_e = 0.5$. Data points are averaged over 25 instances.We generate 25 random (binary) graphs, 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 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: $|\mathcal{A}| = 10$, $|D_i| = 2$, $w = 1$, 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 is the height of the BFS tree. Since the ALS framework requires that $m \geq H$, we vary m from H to $H + 240$.

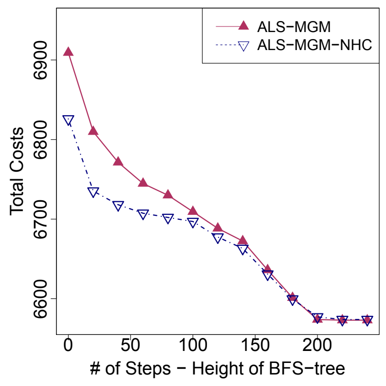
Table 1 tabulates our empirical results, where we vary the number of agents $|\mathcal{A}|$. Figure 2(b) plots the convergence rate of ALS-MGM when elicitation is free. We make the following observations:

- 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 $|\mathcal{A}|$.
- 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 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.
- 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.
- 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. Therefore, these results highlight the strengths of using our proposed heuristics for solving I-DCOPs.

Figure 2(a) 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 increase from right to left on the top axis of the Figure. Each data point in the figures thus shows the result 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 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



(a) Varying Additive Weights, with $|\mathcal{A}| = 10$



(b) Varying Number of Steps, with $|\mathcal{A}| = 100$

Fig. 2: Evaluation of ALS-MGM and SyncBB on Random Graphs

elicitation costs as their trends are similar to those without elicitation costs for both additive and relative weights.

Figure 2(b) 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.

10 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 incurs elicitation costs. To solve I-DCOPs, we adapted SyncBB and ALS-MGM as well as proposed new heuristics that can be used in conjunction with those algorithms to improve their runtimes or quality of solutions found as well as trade off solution quality for faster runtimes and fewer elicitations. They also provide theoretical quality guarantees when used by SyncBB when elicitations are free. 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.

References

1. Boutilier, C., Patrascu, R., Poupart, P., Schuurmans, D.: Constraint-based optimization and utility elicitation using the minimax decision criterion. Artificial Intelligence **170**(8-9), 686–713 (2006)
2. Costanza, E., Fischer, J.E., Colley, J.A., Rodden, T., Ramchurn, S.D., Jennings, N.R.: Doing the laundry with agents: a field trial of a future smart energy system in the home. In: CHI. pp. 813–822 (2014)
3. Dechter, R., Dechter, A.: Belief maintenance in dynamic constraint networks. In: AAAI. pp. 37–42 (1988)
4. Faltings, B., Macho-Gonzalez, S.: Open constraint programming. Artificial Intelligence **161**(1-2), 181–208 (2005)
5. Farinelli, A., Rogers, A., Petcu, A., Jennings, N.: Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In: AAMAS. pp. 639–646 (2008)
6. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research **61**, 623–698 (2018)
7. Fioretto, F., Yeoh, W., Pontelli, E.: A multiagent system approach to scheduling devices in smart homes. In: AAMAS. pp. 981–989 (2017)
8. Flerova, N., Marinescu, R., Dechter, R.: Weighted heuristic anytime search: new schemes for optimization over graphical models. Annals of Mathematics and Artificial Intelligence (2017)
9. Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. Artificial Intelligence **174**(3-4), 270–294 (2010)
10. Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: A local search approach to solve incomplete fuzzy csps. In: ICAART. pp. 582–585 (2011)
11. Goldsmith, J., Junker, U.: Preference handling for artificial intelligence. AI Magazine **29**(4), 9–12 (2008)
12. Hansen, E.A., Zhou, R.: Anytime heuristic search. Journal of Artificial Intelligence Research **28**, 267–297 (2007)
13. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **SSC4**(2), 100–107 (1968)
14. Hirayama, K., Yokoo, M.: Distributed partial constraint satisfaction problem. In: CP. pp. 222–236 (1997)
15. Korf, R.: Linear-space best-first search. Artificial Intelligence **62**(1), 41–78 (1993)
16. Le, T., Tabakhi, A.M., Tran-Thanh, L., Yeoh, W., Son, T.C.: Preference elicitation with interdependency and user bother cost. In: AAMAS. pp. 1459–1467 (2018)
17. Léauté, T., Faltings, B.: Distributed constraint optimization under stochastic uncertainty. In: AAAI. pp. 68–73 (2011)
18. Maheswaran, R., Pearce, J., Tambe, M.: Distributed algorithms for DCOP: A graphical game-based approach. In: PDCS. pp. 432–439 (2004)
19. Maheswaran, R., Tambe, M., Bowring, E., Pearce, J., Varakantham, P.: Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In: AAMAS. pp. 310–317 (2004)
20. Marinescu, R., Dechter, R.: AND/OR branch-and-bound search for combinatorial optimization in graphical models. Artificial Intelligence **173**(16-17), 1457–1491 (2009)

21. Miller, S., Ramchurn, S., Rogers, A.: Optimal decentralised dispatch of embedded generation in the smart grid. In: AAMAS. pp. 281–288 (2012)
22. Modi, P., Shen, W.M., Tambe, M., Yokoo, M.: ADOPT: Asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence **161**(1–2), 149–180 (2005)
23. Nguyen, D.T., Yeoh, W., Lau, H.C.: Stochastic dominance in stochastic DCOPs for risk-sensitive applications. In: AAMAS. pp. 257–264 (2012)
24. Nguyen, D.T., Yeoh, W., Lau, H.C., Zivan, R.: Distributed Gibbs: A linear-space sampling-based DCOP algorithm. Journal of Artificial Intelligence Research **64**, 705–748 (2019)
25. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: IJCAI. pp. 1413–1420 (2005)
26. Pohl, I.: Heuristic search viewed as path finding in a graph. Artificial Intelligence **1**(3-4), 193–204 (1970)
27. Sun, X., Druzdzel, M., Yuan, C.: Dynamic weighting A* search-based MAP algorithm for Bayesian networks. In: IJCAI. pp. 2385–2390 (2007)
28. Tabakhi, A.M.: Preference elicitation in DCOPs for scheduling devices in smart buildings. In: AAAI. pp. 4989–4990 (2017)
29. Tabakhi, A.M., Le, T., Fioretto, F., Yeoh, W.: Preference elicitation for DCOPs. In: CP. pp. 278–296 (2017)
30. Tabakhi, A.M., Xiao, Y., Yeoh, W., Zivan, R.: Branch-and-bound heuristics for incomplete DCOPs. In: AAMAS. pp. 1677–1679 (2021)
31. Tabakhi, A.M., Yeoh, W., Fioretto, F.: The smart appliance scheduling problem: A bayesian optimization approach. In: (PRIMA). pp. 100–115 (2020)
32. Tabakhi, A.M., Yeoh, W., Yokoo, M.: Parameterized heuristics for Incomplete Weighted CSPs with elicitation costs. In: AAMAS. pp. 476–484 (2019)
33. Trabelsi, W., Brown, K.N., O'Sullivan, B.: Preference elicitation and reasoning while smart shifting of home appliances. Energy Procedia **83**, 389 – 398 (2015)
34. Truong, N.C., Baarslag, T., Ramchurn, S.D., Tran-Thanh, L.: Interactive scheduling of appliance usage in the home. In: IJCAI. pp. 869–877 (2016)
35. Vinyals, M., Rodríguez-Aguilar, J., Cerquides, J.: 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)
36. Xiao, Y., Tabakhi, A.M., Yeoh, W.: Embedding preference elicitation within the search for DCOP solutions. In: AAMAS. pp. 2044–2046 (2020)
37. Yeoh, W., Felner, A., Koenig, S.: BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. Journal of Artificial Intelligence Research **38**, 85–133 (2010)
38. Yorke-Smith, N., Gervet, C.: Certainty closure: A framework for reliable constraint reasoning with uncertainty. In: CP. pp. 769–783 (2003)
39. Zhang, W., Wang, G., Xing, Z., Wittenburg, L.: Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. Artificial Intelligence **161**(1-2), 55–87 (2005)
40. Zivan, R., Okamoto, S., Peled, H.: Explorative anytime local search for distributed constraint optimization. Artificial Intelligence **212**, 1–26 (2014)
41. Zivan, R., Yedidsion, H., Okamoto, S., Glinton, R., Sycara, K.: Distributed constraint optimization for teams of mobile sensing agents. Journal of Autonomous Agents and Multi-Agent Systems **29**(3), 495–536 (2015)