Incomplete Distributed Constraint Optimization Problems

Atena M. Tabakhi and William Yeoh

Department of Computer Science and Engineering, Washington University in St. Louis
{amtabakhi,wyeoh}@wustl.edu

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 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 smaller number of elicitations. 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: DCOPs · Preference Elicitation · Multi-Agent Systems.

1 Introduction

The *Distributed Constraint Optimization Problem* (DCOP) [13, 17] 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 [11], sensor and wireless networks [1, 25], multi-robot teams coordination [26], smart grids [7, 12, 3, 20], coalition structure generation [22], and smart homes [18, 2, 21].

The field of DCOP has matured significantly over the past decade since its inception [13]. DCOP researchers have proposed a wide variety of solution approaches, from complete approaches that use distributed search-based techniques [13, 24, 14] to distributed inference-based techniques [17, 23]. There is also a significant body of work on incomplete methods that can be similarly categorized into local search based methods [10, 1], GDL-based techniques [23], and sampling-based methods [16, 15].Researchers have also proposed the use of other off-the-shelf solvers such as logic programming solvers [9, 8] and mixed-integer programming solvers [5].

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* [19]. 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, in this paper, we propose the *Incomplete DCOP* (I-DCOP) model, which *integrates* both the elicitation and solving problem into a single integrated optimization problem. 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, we introduce a number of heuristics that can be used in conjunction with commonly-used synchronous DCOP search algorithms such as Synchronous Branch-and-Bounds (SyncBB) [6]. These heuristics are also parameterized in such a way that they allow users to trade off solution quality for faster runtimes and 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.

2 Background

We now describe *Distributed Constraint Optimization Problems* (DCOPs) [13, 17], which we will later extend to Incomplete DCOPs, as well as the *Synchronous Branch-and-Bound* (SyncBB) algorithm [6], which we will use as the underlying DCOP search algorithm that uses our proposed heuristics.

2.1 Distributed Constraint Optimization Problems

A *Distributed Constraint Optimization Problem* (DCOP) is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \alpha \rangle$, 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 $\mathbf{x}^* = \text{argmin}_{\mathbf{x}} \mathcal{F}(\mathbf{x})$.

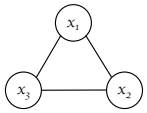
A *constraint graph* visualizes a DCOP, where nodes in the graph correspond to variables in the DCOP and edges connect pairs of variables appearing in the same constraint. A *pseudo-tree* arrangement has the same nodes as the constraint graph and includes all the edges of the constraint graph. The edges 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*.

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 Motivating 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 [11, 17, 24]. While there are a number of possible formulations, we use the *Private Events as Variables* (PEAV) formulation proposed by Maheswaran *et al.* [11] in this paper. In the PEAV formulation, the agents are meeting participants, their variables correspond to the different meetings that they must attend, and their values correspond to the different time slots of the meetings.[1] Equality constraints are imposed on variables of all agents involved in the same meeting – this enforces that all participants of a meeting agree on the time of that meeting – and inequality constraints are imposed on all variables of a single agent – this enforces that each participant cannot attend two meetings at the same time. Finally, unary constraints are imposed on each of the agent's variables where the costs correspond to the preferences of the participant on the different time slots.

To solve this problem, existing work has assumed that all the costs of such constraints are all known [11, 17, 24]. However, since these costs correspond to preferences of human users, it is unrealistic to assume that all the preferences are known a priori. These unknown preferences must thus be elicited if they are needed. Further, the elicitation of such preferences will incur elicitation costs that correspond to the degree at which a user is bothered by the elicitation process. As the existing canonical DCOP model is unable to capture these two features, we describe in the next section a DCOP extension that models unknown constraint costs that must be elicited as well as the cost of performing such elicitations.



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

Fig. 1: Example of Incomplete DCOP with Elicitation Costs

4 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 $\tilde{\mathbf{x}}$.

The *total cost* $\mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}}) = \mathcal{F}(\mathbf{x}) + \mathcal{E}(\tilde{\mathbf{x}})$ is the 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 $\tilde{\mathbf{x}}$. 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. 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 5.

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

5 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. We describe below how to adapt SyncBB to solve I-DCOPs as we will use this algorithm as the underlying search algorithm that uses our proposed heuristics later.

The operations of SyncBB can be visualized with search trees. 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 $g(n)$.[2] We refer to the sum of these values as f-values, denoted by $f(n, \tilde{\mathbf{x}}) = g(n) + \mathcal{E}(\tilde{\mathbf{x}})$.

6 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 $f(n, \tilde{\mathbf{x}}) = g(n) + h(n) + \mathcal{E}(\tilde{\mathbf{x}}) \geq \mathcal{F}(\mathbf{x}, \tilde{\mathbf{x}})$, 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 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 or can be set to 0 in the worst case since all costs are non-negative. The more informed the lower bound, the more effective the heuristics will be in pruning the search space.

Additionally, these heuristics are parameterized by two parameters – a relative weight $w \geq 1$ and an additive weight $\epsilon \geq 0$. 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.

6.1 Child's Ancestors' Constraints (CAC) Heuristic

Our 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)

[2] We use A* notations [4] here.

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,

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

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 (2):

$$\sum_{x_k \in Anc(x_l)} \min_{d_k \in D_k} \hat{f}(x_l = d_l, x_k = d_k) \quad (3)$$

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.

6.2 Agent's Descendants' Constraints (ADC) Heuristic

Our second heuristic is called *Agent's Descendants' Constraints* (ADC) heuristic. Like the CAC heuristic, it is also defined recursively from the leaf of the pseudo-chain used by SyncBB up to the root of the pseudo-chain. Agent x_i in the ordering computes a heuristic value $h(x_i = d_i)$ for each of its values $d_i \in D_i$ as follows: $h(x_i = d_i) = 0$ if x_i is the leaf of the pseudo-chain. Otherwise,

$$h(x_i = d_i) = \min_{d_c \in D_c} \Big[\hat{f}(x_i = d_i, x_c = d_c) + e(x_i = d_i, x_c = d_c) + h(x_c = d_c) \Big] + \sum_{x_j \in Des(x_i) \setminus \{x_c\}} \min_{d_j \in D_j} \Big[\hat{f}(x_i = d_i, x_j = d_j) + e(x_i = d_i, x_j = d_j) \Big] \quad (4)$$

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) 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.

Two other types of heuristics that can have a large impact on the efficiency of the search are the value- and variable-ordering heuristics:

Instead of choosing a random order to explore the different values of an agent, we order their values according to the best-available cost function $f(n, \tilde{\mathbf{x}}) = g(n) + h(n) + \mathcal{E}(\tilde{\mathbf{x}})$, where n is the node corresponding to the value of the agent and $\tilde{\mathbf{x}}$ is the current explored solution space. Instead of choosing a random ordering of variables for SyncBB, we order the variables based on the number of their constraints that has unknown costs – the variable with the fewest number of constraints with unknown costs as the root and the variable with the most number of constraints with unknown costs as the leaf.

7 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 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, and the runtimes of the algorithms (in sec). Data points are averaged of over 50 instances.

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. We vary the number of participants of the meetings (= variables) $|\mathcal{X}|$ from 9 to 18. All time preferences (constraint costs) and elicitation costs are randomly sampled from $[0, 20]$.

Tables 1 tabulates our empirical results, where we vary the number of variables $|\mathcal{X}|$. We make the following observations:
As expected, the runtimes and number of unknown costs elicited by all algorithms increase with increasing number of variables $|\mathcal{X}|$. The reason is that the size of the problem, in terms of the number of constraints in the problem, increases with increasing $|\mathcal{X}|$. And all algorithms need to elicit more unknown costs and evaluate the costs of more constraints before terminating. On problems with elicitation costs, SyncBB with heuristics is still faster than without heuristics. However, neither heuristic dominates the other. Overall, the use of heuristics reduces the number of unknown costs elicited and the runtime which highlights the strengths of using our proposed heuristics.

(a) Without Heuristics

$\|\mathcal{X}\|$	$\|\mathcal{F}\|$	Without Elicitation Costs			With Elicitation Costs				
		# of elic.	runtime	const. cost	# of elic.	runtime	total cost	const. cost	elic. cost
9	15	15.48	4.92E-01	72.66	14.72	5.83E-01	207.32	78.50	128.82
12	26	14.66	3.47E+00	107.36	14.34	3.59E+00	248.80	108.86	139.94
15	43	14.66	3.16E+01	136.24	14.64	3.15E+01	270.32	137.48	132.84
18	61	13.68	2.08E+02	157.28	13.56	1.91E+02	284.88	159.40	125.48

(b) With CAC Heuristic

9	15	13.62	4.14E-01	72.66	11.48	2.74E-01	164.20	83.20	81.00
12	26	12.98	2.62E+00	107.36	10.20	1.74E+00	192.82	115.22	77.60
15	43	12.60	2.58E+01	136.24	11.60	1.78E+01	226.30	141.78	84.52
18	61	11.32	1.79E+02	157.28	10.72	1.42E+02	242.56	163.28	79.28

(c) With ADC Heuristic

9	15	12.80	3.54E-01	72.66	10.58	1.73E-01	154.62	77.32	77.30
12	26	12.28	2.36E+00	107.36	8.60	5.39E-01	186.80	113.00	73.80
15	43	12.12	2.20E+01	136.24	10.42	1.23E+01	216.26	140.56	75.70
18	61	11.40	1.85E+02	157.28	10.64	1.15E+02	240.04	163.96	76.08

Table 1: Meeting Scheduling Problems Varying Number of Variables $|\mathcal{X}|$

8 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 smaller 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. 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)
2. Fioretto, F., Yeoh, W., Pontelli, E.: A multiagent system approach to scheduling devices in smart homes. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 981–989 (2017)
3. Fioretto, F., Yeoh, W., Pontelli, E., Ma, Y., Ranade, S.: A distributed constraint optimization (DCOP) approach to the economic dispatch with demand response. In: AAMAS. pp. 999–1007 (2017)
4. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
5. Hatano, D., Hirayama, K.: DeQED: An efficient divide-and-coordinate algorithm for DCOP. In: IJCAI. pp. 566–572 (2013)
6. Hirayama, K., Yokoo, M.: Distributed partial constraint satisfaction problem. In: Proceedings of CP. pp. 222–236 (1997)
7. Kumar, A., Faltings, B., Petcu, A.: Distributed constraint optimization with structured resource constraints. In: AAMAS. pp. 923–930 (2009)
8. Le, T., Fioretto, F., Yeoh, W., Son, T.C., Pontelli, E.: ER-DCOPs: A framework for distributed constraint optimization with uncertainty in constraint utilities. In: AAMAS (2016)
9. Le, T., Son, T.C., Pontelli, E., Yeoh, W.: Solving distributed constraint optimization problems with logic programming. In: Proceedings of AAAI (2015)
10. Maheswaran, R., Pearce, J., Tambe, M.: Distributed algorithms for DCOP: A graphical game-based approach. In: Proceedings of the International Conference on Parallel and Distributed Computing Systems (PDCS). pp. 432–439 (2004)
11. Maheswaran, R., Tambe, M., Bowring, E., Pearce, J., Varakantham, P.: Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In: Proceedings of AAMAS. pp. 310–317 (2004)
12. Miller, S., Ramchurn, S., Rogers, A.: Optimal decentralised dispatch of embedded generation in the smart grid. In: AAMAS. pp. 281–288 (2012)
13. 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)
14. Netzer, A., Grubshtein, A., Meisels, A.: Concurrent forward bounding for distributed constraint optimization problems. Artificial Intelligence **193**, 186–216 (2012)
15. Nguyen, D.T., Yeoh, W., Lau, H.C.: Distributed Gibbs: A memory-bounded sampling-based DCOP algorithm. In: Proceedings of AAMAS. pp. 167–174 (2013)
16. Ottens, B., Dimitrakakis, C., Faltings, B.: 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)
17. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: Proceedings of IJCAI. pp. 1413–1420 (2005)
18. Rust, P., Picard, G., Ramparany, F.: Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems. In: Proceedings of IJCAI. pp. 468–474 (2016)
19. Tabakhi, A.M., Le, T., Fioretto, F., Yeoh, W.: Preference elicitation for DCOPs. In: Proceedings of CP. pp. 278–296 (2017)
20. Tabakhi, A.M., Yeoh, W., Tourani, R., Natividad, F., Misra, S.: Communication-sensitive pseudo-tree heuristics for dcop algorithms. International Journal on Artificial Intelligence Tools **27**(07), 1860008 (2018)
21. Tabakhi, A.M., Yeoh, W., Yokoo, M.: Parameterized heuristics for Incomplete Weighted CSPs with elicitation costs. In: Proceedings of AAMAS (2019)

22. Ueda, S., Iwasaki, A., Yokoo, M., Silaghi, M., Hirayama, K., Matsui, T.: Coalition structure generation based on distributed constraint optimization. In: Proceedings of AAAI. pp. 197–203 (2010)
23. 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)
24. Yeoh, W., Felner, A., Koenig, S.: BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. Journal of Artificial Intelligence Research **38**, 85–133 (2010)
25. Yeoh, W., Yokoo, M.: Distributed problem solving. AI Magazine **33**(3), 53–65 (2012)
26. 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)