Dynamic Continuous Distributed Constraint Optimization Problems

Khoi D. Hoang and William Yeoh

Washington University in St. Louis
{khoi.hoang,wyeoh}@wustl.edu

Abstract. The Distributed Constraint Optimization Problem (DCOP) formulation is a powerful tool to model multi-agent coordination problems that are distributed by nature. While DCOPs assume that variables are discrete and the environment does not change over time, agents often interact in a more dynamic and complex environment. To address these limiting assumptions, researchers have proposed Dynamic DCOPs (D-DCOPs) to model how DCOPs dynamically change over time and Continuous DCOPs (C-DCOPs) to model DCOPs with continuous variables and constraints in functional form. However, these models address each limiting assumption of DCOPs in isolation, and it remains a challenge to model problems that *both* have continuous variables and are in dynamic environment. Therefore, in this paper, we propose *Dynamic Continuous DCOPs (DC-DCOPs)*, a novel formulation that models both dynamic nature of the environment and continuous nature of the variables, which are inherent in many multi-agent problems. In addition, we introduce several greedy algorithms to solve DC-DCOPs and discuss their theoretical properties. Finally, we empirically evaluate the algorithms in random networks and in distributed sensor network application.

Keywords: Multiagent Systems · Distributed Constraint Optimization Problems · Continuous DCOPs · Dynamic DCOPs

1 Introduction

Distributed Constraint Optimization Problems (DCOPs) [22, 24, 37, 7] are problems where agents coordinate their value assignments to maximize the sum of the utility functions. The model has been applied to solve a wide range of multi-agent coordination problems including distributed meeting scheduling [20, 35], sensor and wireless network coordination [6, 37], multi-robot coordination [40], smart grid optimization [18, 21, 10], smart home automation [27, 9], and cloud computing applications [23, 15].

Typically, DCOPs assume that the domains of variables are discrete and the environment does not change over time. However, in many distributed multi-agent problems, agents often interact in a more dynamic and complex environment. For example, in distributed sensor networks, targets usually move from one location to another location over time, and to adapt to such a dynamic environment, the sensors should be augmented with the capability to change their sensing directions accordingly. To address this concern, researchers have proposed *Dynamic DCOPs (D-DCOPs)* [25, 26, 19], which model how the problem evolves during the solving process. Additionally, to better sense the targets of interest, whose locations correspond to a wide range of possibilities (i.e., the set of all possible locations in a two-dimensional plane or three-dimensional space of the network), the sensors should be equipped with a continuous range of sensing directions. Therefore, researchers have introduced *Continuous DCOPs* (C-DCOPs) [29, 32], which model continuous variables with a bounded domain and represent the constraints in functional form.

While D-DCOPs and C-DCOPs have been proposed to address the two limiting assumptions of DCOPs, the two models only address these assumptions in isolation. Thus, it remains a challenge to model and solve the problems that are both dynamically changing over time and have continuous variables. Therefore, in this paper, we propose *Dynamic Continuous DCOPs (DC-DCOPs)*, a novel formulation that models both the dynamic environment and continuous variables, which are present in many multi-agent problems. In addition, we introduce several greedy algorithms to solve DC-DCOPs and discuss their theoretical properties. Finally, we empirically evaluate the algorithms in random networks and in a distributed sensor network application.

2 Background

In this section, we provide a brief overview of DCOPs, Dynamic DCOPs, and Continuous DCOPs.

2.1 Distributed Constraint Optimization Problems (DCOPs)

A *Distributed Constraint Optimization Problem (DCOP)* [22, 24, 7] is a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F}, \alpha \rangle$, where:

- $\mathbf{A} = \{a_i\}_{i=1}^{p}$ is a set of *agents*.
- $\mathbf{X} = \{x_i\}_{i=1}^{n}$ is a set of *decision variables*.
- $\mathbf{D} = \{D_x\}_{x \in \mathbf{X}}$ is a set of finite *domains*, where each variable $x \in \mathbf{X}$ takes values from the set $D_x \in \mathbf{D}$.
- $\mathbf{F} = \{f_i\}_{i=1}^{m}$ is a set of *utility functions*, each defined over a set of decision variables: $f_i : \prod_{x \in \mathbf{x}^{f_i}} D_x \to \mathbb{R}_0^+ \cup \{-\infty\}$, where infeasible configurations have $-\infty$ utilities and $\mathbf{x}^{f_i} \subseteq \mathbf{X}$ is the *scope* of f_i.[1]
- $\alpha : \mathbf{X} \to \mathbf{A}$ is a function that associates each decision variable to one agent.

A *solution* σ is a value assignment to a set $\mathbf{x}_\sigma \subseteq \mathbf{X}$ of decision variables that is consistent with their respective domains. The utility $\mathbf{F}(\sigma) = \sum_{f \in \mathbf{F}, \mathbf{x}^f \subseteq \mathbf{x}_\sigma} f(\sigma)$ is the sum of the utilities across all applicable utility functions in σ. A solution σ is *complete* if $\mathbf{x}_\sigma = \mathbf{X}$. The goal of a DCOP is to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} \mathbf{F}(\mathbf{x})$.

[1] The scope of a function is the set of variables that are associated with the function.

2.2 Dynamic DCOPs

A *Dynamic DCOP (D-DCOP)* [25, 26, 19, 36] is defined as a sequence of DCOPs with changes between them. Changes between DCOPs occur over time due to addition or removal of variables, addition or removal of values in the variable's domain, addition or removal of utility functions, and increase or decrease in the utility values. Solving a D-DCOP optimally means finding a utility-maximal solution for each DCOP in the sequence. Therefore, this approach is *reactive* since solving each DCOP in the sequence does not consider future changes. Its advantage is that solving a D-DCOP is no harder than solving h DCOPs, where h is the horizon of the problem. Researchers have used this approach to solve D-DCOPs, where they introduce search- and inference-based approaches that are able to reuse information from previous DCOPs to speed up the search for the solution for the current DCOP [25, 36]. Alternatively, a *proactive* approach predicts future changes in the D-DCOP and finds robust solutions that require little or no changes in the sequence of DCOP solutions despite future changes to the DCOP [13, 14, 12]

2.3 Continuous DCOPs

A *Continuous DCOP (C-DCOP)* [29, 32, 15] is defined as a DCOP where the variables take values from a continuous domain. In a typical (discrete) DCOP, constraints are represented in tabular form, which enumerates all possible values of the discrete variables involved in the constraint. Since variables in C-DCOPs are continuous, the model represents the constraints in functional forms such as linear piecewise function, quadratic function, or a more general differentiable function. Recently, researchers have proposed several algorithms to solve C-DCOPs including approximate approaches [15, 28, 3] and exact approach that solves C-DCOP under a specific setting [15].

3 Motivating Application: Distributed Radar Coordination and Scheduling Problem

We motivate our work using the *Distributed Radar Coordination and Scheduling Problem (DRCSP)* [12], which is based on NetRad, a real-time weather radar sensor system [4, 17, 39]. The NetRad system is consisted meteorological command and controls (MCCs), each controls a set of radars with a limited sensing range. The radars in NetRad are tasked by the MCCs to scan a specific area of interest in a coordinated fashion, where each radar takes 360-degree volume scan. For example, in Figure 1, the NetRad system has five radars scanning the area with two weather phenomena, represented as a yellow star and a red star. The goal of a DRCSP is to find a coordination strategy that maximizes the aggregated utility by scanning the highest-utility phenomena in the area. Since each sensor is able to take a continuous 360-degree scan, we model the sensing direction of the sensors with the continuous variables. In addition, since the

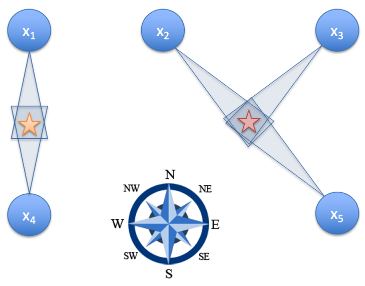

Fig. 1. Distributed Radar Coordination and Scheduling Problem

weather phenomena may move continuously over time, we model this dynamism by incorporating random variables representing the weather phenomena.

4 Dynamic Continuous DCOP Model

A *Dynamic Continuous DCOP (DC-DCOP)* is a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{Y}, \mathbf{D_x}, \mathbf{D_y}, \mathbf{F}, p^0_{\mathbf{Y}}, \mathbf{T}, \gamma, h, \mathbf{C}, \alpha \rangle$, where:

- $\mathbf{A} = \{a_i\}_{i=1}^p$ is a set of *agents*.
- $\mathbf{X} = \{x_i\}_{i=1}^n$ is a set of *decision variables*, which are variables controlled by the agents.
- $\mathbf{Y} = \{y_i\}_{i=1}^m$ is a set of *random variables*, which are variables that are uncontrollable and model stochastic events (e.g., weather phenomena location or intensity)
- $\mathbf{D_x} = \{D_x\}_{x \in \mathbf{X}}$ is a set of continuous *domains* of the decision variables. Each variable $x \in \mathbf{X}$ takes values from the interval $D_x = [LB_x, UB_x]$.
- $\mathbf{D_y} = \{D_y\}_{y \in \mathbf{Y}}$ is a set of continuous state space of the random variables. Each variable $y \in \mathbf{Y}$ has state space $D_y \in \mathbf{D_y}$.
- $\mathbf{F} = \{f_i\}_{i=1}^k$ is a set of *utility functions*, each defined over a mixed set of decision and random variables: $f_i : \prod_{x \in \mathbf{X} \cap \mathbf{x}^{f_i}} D_x \times \prod_{y \in \mathbf{Y} \cap \mathbf{x}^{f_i}} D_y \rightarrow \mathbb{R}_0^+ \cup \{-\infty\}$, where infeasible configurations have $-\infty$ rewards and $\mathbf{x}^{f_i} \subseteq \mathbf{X} \cup \mathbf{Y}$ is the scope of f_i. We divide the set of utility functions into two sets: $\mathbf{F_X} = \{f_x\}$, where $\mathbf{x}^{f_x} \cap \mathbf{Y} = \emptyset$, and $\mathbf{F_Y} = \{f_y\}$, where $\mathbf{x}^{f_y} \cap \mathbf{Y} \neq \emptyset$. Note that $\mathbf{F_X} \cup \mathbf{F_Y} = \mathbf{F}$ and $\mathbf{F_X} \cap \mathbf{F_Y} = \emptyset$.
- $p^0_{\mathbf{Y}} = \{p^0_y\}_{y \in \mathbf{Y}}$ is a set of initial *probability density functions* of the random variable $y \in \mathbf{Y}$.
- $\mathbf{T} = \{T_y\}_{y \in \mathbf{Y}}$ is a set of *transition functions*, where each transition function is a conditional density function $T_y : D_y \times \mathcal{P}(D_y) \rightarrow [0, 1]$ that specifies the transition from a value $d_y \in D_y$ to a subset of D_y.
- $\gamma \in [0, 1]$ is a *discount factor*, which represents the decrease in the importance of future rewards.
- $h \in \mathbb{N}$ is a finite *horizon*.

- $\mathbf{C} = \{c_x\}_{x \in \mathbf{X}}$ is a set of *switching cost functions*, each defined over a set of decision variables: $c_x : D_x \times D_x \to \mathbb{R}_0^+$. Each switching cost function c_x models the cost associated with the change in the value of the decision variable x from one time step to the next.
- $\alpha : \mathbf{X} \to \mathbf{A}$ is a function that associates each decision variable to one agent.

Throughout this article, we assume that each agent controls exactly one decision variable and that each utility function is associated with at most one random variable.[2] In the case where one agent controls more than one variable, one can use standard DCOP reformulation techniques [2, 38], such as *compilation*, where each agent creates a new *pseudo-variable*, whose domain is the Cartesian product of the domains of all variables of the agent; and *decomposition*, where each agent creates a *pseudo-agent* for each of its variables. More recently, researchers have also proposed a multi-variable decomposition method that exploits the co-locality of variables to more efficiently solve the problem [8].

The goal of a DC-DCOP is to find a sequence of $h+1$ assignments $\mathbf{x}^*$ for all the decision variables in $\mathbf{X}$:

$$\mathbf{x}^* = \underset{\mathbf{x}=\langle \mathbf{x}^0,\ldots,\mathbf{x}^h\rangle \in \Sigma^{h+1}}{\operatorname{argmax}} \mathcal{F}^h(\mathbf{x})$$

$$\mathcal{F}^h(\mathbf{x}) = \underbrace{\sum_{t=0}^{h} \gamma^t \left[\mathcal{F}_x^t(\mathbf{x}^t) + \mathcal{F}_y^t(\mathbf{x}^t)\right]}_{\mathbf{P}} - \underbrace{\sum_{t=0}^{h-1} \gamma^t \left[C_{\mathbf{x}}(\mathbf{x}^t, \mathbf{x}^{t+1})\right]}_{\mathbf{Q}}$$

where Σ is the assignment space for the decision variables of the DC-DCOP. The first term $\mathbf{P}$ refers to the optimization over $h+1$ time steps, with:

$$\mathcal{F}_x^t(\mathbf{x}) = \sum_{f_i \in \mathbf{F_X}} f_i(\mathbf{x}_i)$$

$$\mathcal{F}_y^t(\mathbf{x}) = \sum_{f_i \in \mathbf{F_Y}} \int_{D_{y_i}} f_i(\mathbf{x}_i, y_i) \cdot p_{y_i}^t(y_i) dy_i$$

where $\mathbf{x}_i$ is an assignment for all the variables in the scope $\mathbf{x}^{f_i}$ of the function f_i; $p_{y_i}^t$ is the probability density function of the random variable y_i at time step t, and defined as:

$$p_{y_i}^t(y_i) = \int_{D_{y_i}} p_{y_i}^{t-1}(y_i) \cdot T(y_i, D_{y_i}) dy_i$$

The second term $\mathbf{Q}$ considers the penalty due to changes in decision variables' values during the optimization process:

$$C_{\mathbf{x}}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \sum_{x \in \mathbf{X}} c_x(x^t, x^{t+1})$$

is a penalty function that takes into account the difference in the decision variable assignments between two time steps.

[2] If multiple random variables are associated with a utility function, w.l.o.g., they can be merged into a single variable.

5 DC-DCOP Algorithms

We now introduce our DC-DCOP algorithms, which are built upon two sequential greedy Dynamic DCOP algorithms: FORWARD and BACKWARD [14]. The two algorithms have been applied to solve the Dynamic DCOPs where each subproblem is a *discrete* DCOP. However, in DC-DCOPs, the subproblem at every time step is a *Continuous* DCOP. Thus the original version of FORWARD and BACKWARD cannot be applied to solve DC-DCOPs. In this section, we propose a new version of the two algorithms that can address and solve the C-DCOP at every time step.

5.1 FORWARD

In general, FORWARD greedily solves each subproblem in DC-DCOPs one time step at a time starting from the first time step. In other words, it successively solves the C-DCOP at each time step starting from $t = 0$ to $t = h$. When solving each C-DCOP, it takes into account the switching cost incurred by changing the solution from time step $t - 1$ to the optimal solution at time step t. Specifically, before solving the C-DCOP at each time step, the agents run a pre-processing step, where they (1) reformulate the constraint between decision and random variables, and (2) capture the cost of switching values between time steps in new unary constraints of decision variables. For each constraint $f_i \in \mathbf{F}_Y$ between decision variables $\mathbf{x}_i$ and a random variable y_i, the following new constraint is created for each time step $0 \leq t \leq h$:

$$F_i^t(\mathbf{x}_i) = \int_{D_{y_i}} f_i(\mathbf{x}_i, y_i) \cdot p_{y_i}^t(y_i) dy_i \tag{1}$$

where $p_{y_i}^t(\cdot)$ is the probability density function of random variable y_i at time step t.

After reformulating the constraints between decision variables and random variable, the agents create a new constraint to capture the cost of switching values across time steps. Specifically, for each decision variable $x \in \mathbf{X}$, the following new unary constraint is created for each time step $0 < t \leq h$:

$$C_x^t(x^t) = -c_x(x^{t-1}, x^t) \tag{2}$$

After adding the switching cost constraints, the agents successively solve each C-DCOP from time step $t = 0$ onwards using any off-the-shelf C-DCOP algorithm.

In this paper, we use the following off-the-shelf C-DCOP algorithms to solve the problem at each time step: *AC-DPOP*, *CAC-DPOP*, *HCMS*, and *C-DSA* [15]. AC-DPOP, CAC-DPOP, and HCMS are *inference*-based algorithms, while C-DSA is a *local search* algorithm. AC-DPOP solves C-DCOPs by first discretizing the domains of the variables into initial discrete values and then using gradient methods to move the values of the parent and psedo-parent variables in order to better approximate the constraint utilities. CAC-DPOP is a variant of

AC-DPOP that reduces the memory and time consumption of AC-DPOP by clustering the values of the agents before sending them up the pseudo-tree. Instead of using pseudo-tree, HCMS uses a factor graph to represent C-DCOPs and gradually adjusts agents' values over a number of iterations. Finally, C-DSA is a continuous *stochastic* algorithm, where each agent communicates their assignment with neighboring agents and stochastically determines to keep the current assignment or change to a better one.

5.2 Backward

Instead of solving the DC-DCOP one time step at a time forward starting from $t = 0$ towards h, one can also greedily solve the problem backwards from $t = h$ towards the first time step. Similar to Forward, before solving the C-DCOP at each time step, agents in Backward run a pre-processing step to reformulate the constraint between decision and random variables, and to capture the switching cost between two time steps. To reformulate the constraints between decision variables and a random variable, the agents calls Equation (1) and create a new constraints for each time steps $0 \leq t \leq h$. However, the key difference between Backward and Forward is how the agents compute the new switching cost constraint at each time step t. Specifically, when solving the C-DCOP at time step t, instead of taking into account the switching cost between time step t and time step $t-1$, agents in Backward takes into account the switching cost between time step t and time step $t+1$.

Specifically, before solving each subproblem, Backward creates a unary constraint for each time step $0 \leq t < h$:

$$C_x^t(x^t) = -c_x(x^t, x^{t+1}) \tag{3}$$

After adding the switching cost constraints and the reformulated constraints between decision variables and a random variable, the agents successively solve each C-DCOP from time step $t = h$ backward using any off-the-shelf C-DCOP algorithm. Similar to Forward, we use AC-DPOP, CAC-DPOP, HCMS, and C-DSA to solve the C-DCOP at each time step. We will empirically evaluate both greedy versions of these C-DCOP algorithms in the experimental result section and will also discuss their theoretical properties in Section 6.

6 Theoretical Properties

We now describe below some theoretical properties on the error bounds and communication complexities for some of our algorithms.

We denote U^∞ as the optimal solution quality of a DC-DCOP with an infinite horizon and U^h as the optimal solution quality when the horizon h is finite. Let $F_{\mathbf{y}}(\mathbf{x})$ be the utility of a regular C-DCOP where the decision variables are assigned $\mathbf{x}$ given values $\mathbf{y}$ of the random variables. We define $F_{\mathbf{y}}^{\Delta} = \max_{\mathbf{x} \in \Sigma} F_{\mathbf{y}}(\mathbf{x}) - \min_{\mathbf{x} \in \Sigma} F_{\mathbf{y}}(\mathbf{x})$ as the maximum loss in solution quality of a regular DCOP for a given random variable assignment $\mathbf{y}$ and $F^{\Delta} = \max_{\mathbf{y} \in \Sigma_{\mathbf{Y}}} F_{\mathbf{y}}^{\Delta}$ where $\Sigma_{\mathbf{Y}} = \prod_{y \in \mathbf{Y}} \Omega_y$ is the assignment space for all random variables.

Theorem 1. *When $\gamma < 1$, the error $U^\infty - U^h$ of the optimal solution from solving DC-DCOPs with a finite horizon h instead of an infinite horizon is bounded from above by $\frac{\gamma^h}{1-\gamma} F^\Delta$.*

Proof. Let $\hat{\mathbf{x}}^* = \langle \hat{\mathbf{x}}^*_0, \ldots, \hat{\mathbf{x}}^*_h, \hat{\mathbf{x}}^*_{h+1}, \ldots \rangle$ be the optimal solution of DC-DCOPs with infinite horizon ∞:

$$U^\infty = \sum_{t=0}^{\infty} \gamma^t \left[\mathcal{F}^t_x(\hat{\mathbf{x}}^*_t) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_t) - C_{\mathbf{x}}(\hat{\mathbf{x}}^*_t, \hat{\mathbf{x}}^*_{t+1}) \right]$$

Ignoring switching costs after time step h, an upper bound U^∞_+ of U^∞ is defined as:

$$\begin{aligned} U^\infty_+ = &\sum_{t=0}^{h-1} \gamma^t \left[\mathcal{F}^t_x(\hat{\mathbf{x}}^*_t) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_t) - C_{\mathbf{x}}(\hat{\mathbf{x}}^*_t, \hat{\mathbf{x}}^*_{t+1}) \right] \\ &+ \sum_{t=h}^{\infty} \gamma^t \left[\mathcal{F}^t_x(\hat{\mathbf{x}}^*_t) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_t) \right] \end{aligned}$$

Let $\mathbf{x}^* = \langle \mathbf{x}^*_0, \ldots, \mathbf{x}^*_h \rangle$ be the optimal solution of the DC-DCOPs with a finite horizon h:

$$\begin{aligned} U^h = &\sum_{t=0}^{h-1} \gamma^t \left[\mathcal{F}^t_x(\mathbf{x}^*_t) + \mathcal{F}^t_y(\mathbf{x}^*_t) - C_{\mathbf{x}}(\mathbf{x}^*_t, \mathbf{x}^*_{t+1}) \right] \\ &+ \sum_{t=h}^{\infty} \gamma^t \left[\mathcal{F}^t_x(\mathbf{x}^*_h) + \mathcal{F}^t_y(\mathbf{x}^*_h) \right] \end{aligned}$$

For $\hat{\mathbf{x}}^*$, if we change the solution for every C-DCOP after time step h to $\hat{\mathbf{x}}^*_h$, as $\langle \hat{\mathbf{x}}^*_0, \ldots, \hat{\mathbf{x}}^*_h, \hat{\mathbf{x}}^*_h, \ldots \rangle$, we get a lower bound U^∞_- of U^h:

$$\begin{aligned} U^\infty_- = &\sum_{t=0}^{h-1} \gamma^t \left[\mathcal{F}^t_x(\hat{\mathbf{x}}^*_t) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_t) - C_{\mathbf{x}}(\hat{\mathbf{x}}^*_t, \hat{\mathbf{x}}^*_{t+1}) \right] \\ &+ \sum_{t=h}^{\infty} \gamma^t \left[\mathcal{F}^t_x(\hat{\mathbf{x}}^*_h) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_h) \right] \end{aligned}$$

Therefore, we get $U^\infty_- \leq U^h \leq U^\infty \leq U^\infty_+$.

Next, we compute the difference between the two bounds:

$$\begin{aligned} U^\infty - U^h &\leq U^\infty_+ - U^\infty_- \\ &= \sum_{t=h}^{\infty} \gamma^t \left[(\mathcal{F}^t_x(\hat{\mathbf{x}}^*_t) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_t)) - (\mathcal{F}^t_x(\hat{\mathbf{x}}^*_h) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_h)) \right] \end{aligned}$$

Notice that the quantity $(\mathcal{F}^t_x(\hat{\mathbf{x}}^*_t) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_t)) - (\mathcal{F}^t_x(\hat{\mathbf{x}}^*_h) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_h))$ is the utility difference between the value assignment $\hat{\mathbf{x}}^*_t$ and $\hat{\mathbf{x}}^*_h$ for a subproblem in time step t, and thus is bounded by the maximum loss of a regular C-DCOP:

$$(\mathcal{F}^t_x(\hat{\mathbf{x}}^*_t) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_t)) - (\mathcal{F}^t_x(\hat{\mathbf{x}}^*_h) + \mathcal{F}^t_y(\hat{\mathbf{x}}^*_h)) \leq F^\Delta$$

Thus,

$$\begin{aligned}
U^{\infty} - U^{h} &\leq U_{+}^{\infty} - U_{-}^{\infty} \\
&\leq \sum_{t=h}^{\infty} \gamma^{t} \left[\mathcal{F}_{x}^{t}(\hat{\mathbf{x}}_{t}^{*}) + \mathcal{F}_{y}^{t}(\hat{\mathbf{x}}_{t}^{*}) - \mathcal{F}_{x}^{t}(\hat{\mathbf{x}}_{h}^{*}) - \mathcal{F}_{y}^{t}(\hat{\mathbf{x}}_{h}^{*})\right] \\
&\leq \sum_{t=h}^{\infty} \gamma^{t} F^{\Delta} \\
&\leq \frac{\gamma^{h}}{1-\gamma} F^{\Delta}
\end{aligned}$$

which concludes the proof. □

Error Bounds from C-DCOP Algorithms: For each reward function $f(x_i, x_{i_1}, \ldots, x_{i_k})$ of an agent x_i and its separator agents $x_{i_1}, \ldots, x_{i_k}$, assume that agent x_i discretizes the domains of the reward function into hypercubes of size m (i.e., the distance between two neighboring discrete points for the same agent x_{i_j} is m). Let $\nabla f(v)$ denote the gradient of the function $f(x_i, x_{i_1}, \ldots, x_{i_k})$ at $v = (v_i, v_{i_1}, \ldots, v_{i_k})$:

$$\nabla f(v) = (\frac{\partial f}{\partial x_i}(v_i), \frac{\partial f}{\partial x_{i_1}}(v_{i_1}), \ldots, \frac{\partial f}{\partial x_{i_k}}(v_{i_k}))$$

Furthermore, let $|\nabla f(v)|$ denote the sum of magnitude:

$$|\nabla f(v)| = |\frac{\partial f}{\partial x_i}(v_i)| + |\frac{\partial f}{\partial x_{i_1}}(v_{i_1})| + \ldots + |\frac{\partial f}{\partial x_{i_k}}(v_{i_k})|$$

Assume that $|\nabla f(v)| \leq \delta$ holds for all utility functions in the DCOP and for all v.

Theorem 2. *The error of* AC-DPOP*-based algorithms is bounded above by* $h \cdot |\mathbf{F}|(m + |\mathbf{A}|k\alpha\delta)\delta + (h-1) \cdot \Theta|\mathbf{A}|$*, where* k *is the number of times each agent "moves" values of its separator, and* $\Theta = \max_{x \in \mathbf{X}} c_x(v, v')$ *is the maximum of the bounded switching cost functions.*

Proof. According to Theorem 5.2 by Hoang *et al.* [15], the error bound of solving a C-DCOP using AC-DPOP algorithm is $|\mathbf{F}|(m + |\mathbf{A}|k\alpha\delta)\delta$. In DC-DCOPs, there are h C-DCOPs, each is a subproblem at every time step. Without taking into account the switching cost, the error bound of AC-DPOP-based algorithms (e.g., Forward-AC-DPOP and Backward-AC-DPOP) is $h \cdot |\mathbf{F}|(m + |\mathbf{A}|k\alpha\delta)\delta$. Given $\Theta = \max_{x,x'} c(x, x')$ as the maximum value of switching cost between two time steps, and considering there are at most $h-1$ switching times between h time steps from $|\mathbf{A}|$ agents, the upper bound is thus $h \cdot |\mathbf{F}|(m + |\mathbf{A}|k\alpha\delta)\delta + (h-1) \cdot \Theta|\mathbf{A}|$. □

Theorem 3. *In a binary constraint graph* $G = (\mathbf{X}, E)$*, the number of messages of* HCMS*-based algorithms and* C-DSA*-based algorithms with* k *iterations is* $h \cdot 4k|E|$ *and* $h \cdot 2k|E|$*, respectively. The number of messages of* AC-DPOP-*, and* CAC-DPOP*-based algorithms is* $h \cdot 2|\mathbf{X}|$.

Proof. According to Theorem 5.3 by Hoang *et al.* [15], the number of messages of *HCMS*-based algorithms and *C-DSA*-based algorithms with k iterations is $4k|E|$ and $2k|E|$, respectively. The number of messages of *AC-DPOP*-, and *CAC-DPOP*-based algorithms is $2|\mathbf{X}|$. Since solving a DC-DCOP is equivalent to solving h C-DCOPs, each at a time step, the number of messages is thus h times the number of messages needed to solve a single C-DCOP. □

7 Related Work

Aside from the D-DCOP model described in the introduction and background, several approaches have been proposed to solve related constraint models with discrete variables including *Dynamic CSPs*, where value assignments of variables or utilities of constraints may change according to some probabilistic model [33, 16]. The goal is typically to find a solution that is robust to possible changes. Other related models include *Mixed CSPs* [5], which model decision problems under uncertainty by introducing state variables, which are not under control of the solver, and seek assignments that are consistent to any state of the world; and *Stochastic CSPs* [34, 31], which introduce probability distributions that are associated to outcomes of state variables, and seek solutions that maximize the probability of constraint consistencies. While these approaches have been used to solve CSP variants, they have not been used to solve D-DCOPs with continuous variables to the best of our knowledge.

Researchers have proposed several algorithms to solve C-DCOPs. One of such algorithms is Continuous Max-Sum (CMS) [29], which is based on Max-Sum [6], a belief propagation algorithm. To represent the constraints, CMS uses multivariate continuous piecewise linear functions (CPLFs) and later encodes the n-ary CPLFs as n-simplexes. To add two CPLFs, CMS partitions the domains of the two functions and then finds the simplexes that make up the resulting summation function. To project a CPLF, the function is projected onto the corresponding plane and the result is the upper envelope of the simplexes. However, CMS is not suitable for the problems where constraint functions are smooth, and it does not provide quality guarantee for the solution. Later, Voice *et al.* [32] proposed Hybrid Continuous Max-Sum (HCMS) to solve C-DCOPs with differentiable functions. Instead of working directly on continuous domains, HCMS first discretizes the domain into a number of initial discrete points and then uses continuous non-linear optimization techniques such as gradient method and Newton method to optimize the marginal function at each variable. However, similar to CMS, HCMS does not provide solution quality guarantee. Recently, Choudhury *et al.* [3] proposed *Particle Swarm Optimization Based Functional DCOP (PFD)* which is based on the *Particle Swarm Optimization (PSO)* technique. While being an iterative and a heuristic algorithm, PFD shares the same limitation with CMS and HCMS that they do not provide guarantee for their solutions. Finally, Fransman *et al.* [11] proposed *Bayesian DPOP (B-DPOP)*, a Bayesian optimization based algorithm, to solve C-DCOPs. While B-DPOP guarantees that it will eventually converge to the global optimum for Lipschitz-

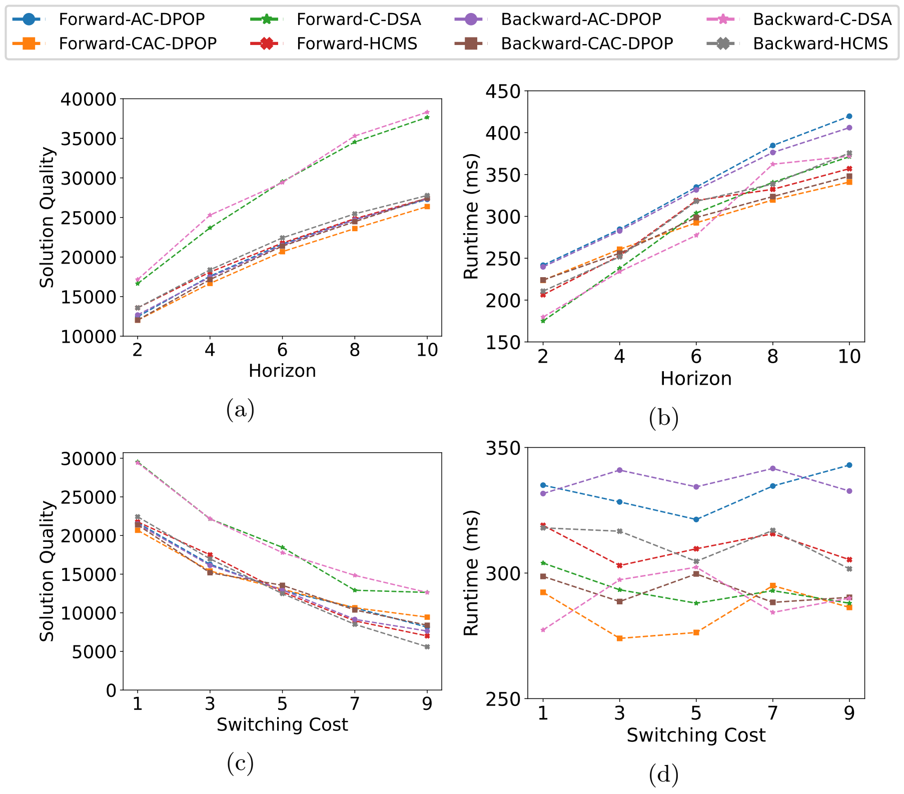

Fig. 2. Experimental Results Varying Horizon and Switching Cost on Sparse Random Networks

continuous objective functions, it does not provide guarantees on intermediate solutions prior to convergence.

8 Experimental Evaluations

We empirically evaluate the following DC-DCOP algorithms: FORWARD- (labeled 'F-' in the tables) and BACKWARD- (labeled 'B-' in the tables) versions of AC-DPOP, CAC-DPOP, C-DSA, and HCMS [15] on random networks and distributed sensor network problems. Our experiments are performed on a 2.1GHz machine with 16GB of RAM using JADE framework [1]. We report solution quality and simulated runtime [30] averaged over 30 independent runs, each with a timeout of 30 minutes.

8.1 Random Networks

We use the following default configuration: Number of agents and random variables $|\mathbf{A}| = |\mathbf{X}| = |\mathbf{Y}| = 12$; domains of decision and random variables

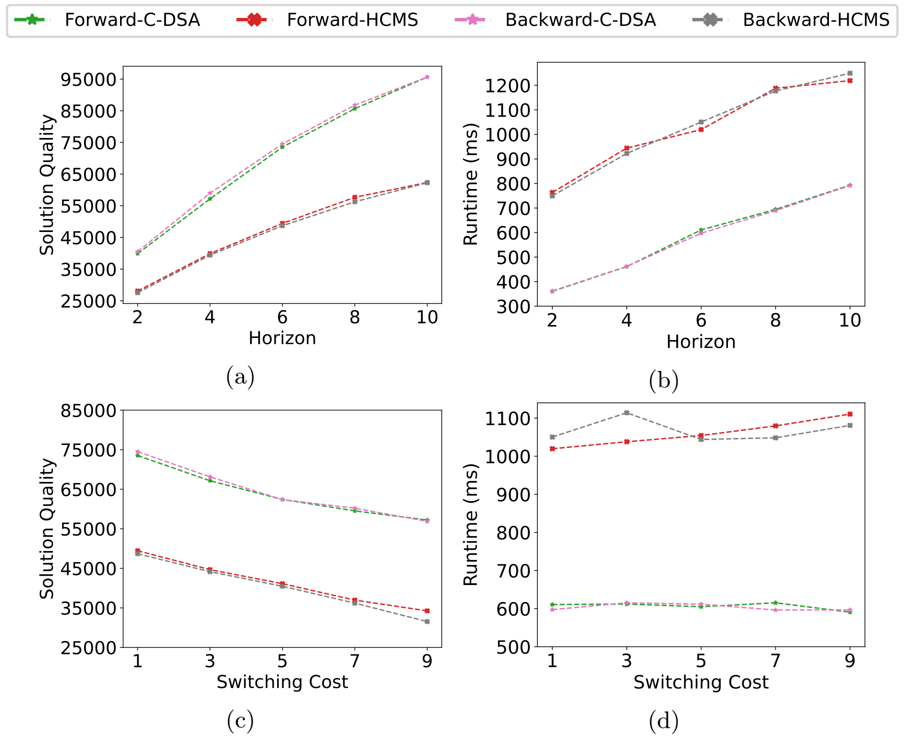

Fig. 3. Experimental Results Varying Horizon and Switching Cost on Dense Random Networks

$D_x = D_y = [-10, 10]$; discount factor $\gamma = 0.9$; horizon $h = 6$; switching cost function $c(x, x') = c \cdot (x - x')^2$ with the default cost $c = 1$. We set the number of discrete points to 3 for AC-DPOP-, CAC-DPOP-, and HCMS-based algorithms. For all algorithms, we set the number of iterations as 20.[3]

We first vary the horizon h to evaluate the performance of the algorithms with different horizon length. Figures 2(a) and 2(b) show the solution quality and runtime with horizon varying from 2 to 10 on sparse networks with $p_1 = 0.2$. When the horizon increases, both Forward-C-DSA and Backward-C-DSA produce the highest solution quality and outperform all other algorithms. The reason is that C-DSA-based algorithms do not depend on a number of initial discrete points and thus they are free to explore the search space. Interestingly, when the horizon becomes longer, their runtime is as small as other algorithms. This result shows that while C-DSA-based algorithms have the best solution quality, they do not come with the cost of higher runtime. Since AC-DPOP

[3] For AC-DPOP- and CAC-DPOP-based algorithms, that is the number of iterations to move the values of parent and pseudo-parent variables. For HCMS- and C-DSA-based algorithms, it is the number of iterations to perform the local search.

$\lvert\mathbf{A}\rvert$	F-AC-DPOP		B-AC-DPOP		F-CAC-DPOP		B-CAC-DPOP		F-C-DSA		B-C-DSA		F-HCMS		B-HCMS	
	q	t	q	t	q	t	q	t	q	t	q	t	q	t	q	t
8	20789	285	20467	280	20778	284	20656	281	25824	280	26354	272	18668	285	18792	285
12	27269	420	27356	406	26378	341	27445	348	37653	371	38306	372	27423	357	27780	376
16	42473	111276	43327	119863	37708	874	39005	870	59390	542	59296	545	43511	611	44373	591
20	–	–	–	–	65345	3697	65075	3690	100725	690	102904	709	73269	751	74140	760
24	–	–	–	–	70058	49275	68965	43269	134213	774	134964	782	93427	958	94610	956
28	–	–	–	–	88135	493444	88156	524903	176429	884	175571	887	120822	1247	120917	1259
32	–	–	–	–	–	–	–	–	224743	1014	227258	1015	162918	1498	163627	1623

Table 1. Varying the Number of Agents on Sparse Random Networks with $p_1 = 0.2$

$\lvert\mathbf{A}\rvert$	F-CAC-DPOP		B-CAC-DPOP		F-C-DSA		B-C-DSA		F-HCMS		B-HCMS	
	q	t	q	t	q	t	q	t	q	t	q	t
8	30359	6604	31158	6229	47747	436	47594	418	33166	684	32958	674
12	–	–	–	–	95133	782	96722	791	62861	1289	62767	1273
16	–	–	–	–	160255	1273	160713	1289	109128	2222	110232	2227
20	–	–	–	–	217548	1603	215767	1601	148709	3243	149640	3284
24	–	–	–	–	281505	1842	283987	1836	196808	4508	196467	4453
28	–	–	–	–	375106	1972	374697	1986	255482	5565	252306	5643
32	–	–	–	–	466945	2138	463594	2121	313125	6891	318854	6860

Table 2. Varying the Number of Agents on Dense Random Networks with $p_1 = 0.7$

$\lvert\mathbf{A}\rvert$	F-AC-DPOP		B-AC-DPOP		F-CAC-DPOP		B-CAC-DPOP		F-C-DSA		B-C-DSA		F-HCMS		B-HCMS	
	q	t	q	t	q	t	q	t	q	t	q	t	q	t	q	t
4	4750	108	5053	104	4750	171	5052	168	7302	131	7432	136	4344	127	4406	125
8	19166	275	21155	263	19287	296	19208	298	24958	252	25045	253	17415	282	17969	282
12	31809	387	31916	387	31116	411	30773	418	40014	392	40912	404	28357	373	28892	358
16	37842	507	38427	493	36007	450	36615	469	50069	459	51274	458	36454	412	36265	411
20	58987	696	60088	695	51418	570	51265	567	70953	513	71888	514	52402	444	52201	465

Table 3. Varying the Number of Agents on Sensor Networks

takes the longest time to solve each single C-DCOP [15], Forward-AC-DPOP and Backward-AC-DPOP are the slowest algorithms across different horizon length. Similarly, on dense networks with $p_1 = 0.7$, Figures 3(a) and 3(b) show that both versions of C-DSA again outperform the HCMS-based algorithms in terms of solution quality with smaller runtime. We do not include AC-DPOP- and CAC-DPOP-based algorithms in Figure 3 since they time out on the dense networks.

Figures 2(c) and 2(d) show the result of varying the switching cost c in the switching cost function $c \cdot (x - x')^2$. The result shows that the solution quality of all algorithms decreases when the switching cost increases. If there is no switching cost (i.e., $c = 0$), the optimal solution of DC-DCOP consists of the optimal solution of the C-DCOP at each time step. However, with higher switching cost, the solution quality found by algorithms is likely to decreases due to the higher penalty incurred by different solutions across time steps. We also observe that C-DSA-based algorithms have the best solution quality, which is consistent with the result on dense graph reported in Figures 3(c) and 3(d).

Finally, we vary the number of agents $|\mathbf{A}|$ (and thus the number of decision $|\mathbf{X}|$ and random variables $|\mathbf{Y}|$) of the problems from 8 to 32 with horizon $h = 10$. Table 1 tabulates the solution quality (denoted by q) and simulated runtime (denoted by t in ms) of the algorithms on sparse networks with $p_1 = 0.2$. Since AC-DPOP takes the longest time to solve the C-DCOP at each time step, both Forward-AC-DPOP and Backward-AC-DPOP can only solve small instances with 8, 12 and 16 agents and time out with larger instances. CAC-DPOP, which is the clustering version of AC-DPOP, reduces the memory used in the UTIL phrase and is more scalable to solve C-DCOPs with more number of agents [15]. Thus both Forward- and Backward-CAC-DPOP are able to solve instances with more number of agents than AC-DPOP-based algorithms and only time out with 32 agents. On the other hand, since C-DSA and HCMS are more scalable, it takes less time for them to solve each individual C-DCOP, and their DC-DCOP algorithms are able to solve all instances with much smaller runtime than AC-DPOP and CAC-DPOP. Interestingly, HCMS-based algorithms report a slightly larger runtime than C-DSA-based algorithms. Similar to the results from Figure 2, C-DSA-based algorithms report the highest solution quality than those from AC-DPOP-, CAC-DPOP-, and HCMS-based algorithms on different numbers of agents.

Table 2 shows the result varying agents on a dense random networks with $p_1 = 0.7$. Since both Forward- and Backward-AC-DPOP time out with 8 agents, we do not include these algorithms in the table. While CAC-DPOP-based algorithms are able to solve the instances with 8 agents, they time out on larger instances. Both C-DSA- and HCMS- based algorithms are able to scale to solve larger instances with incremental runtime. While C-DSA-based algorithms outperform HCMS-based algorithms on all instances, it also takes them less time than the counterpart algorithms.

8.2 Distributed Sensor Network Problems

We evaluate our DC-DCOP algorithms on distributed sensor network problems, which is our motivating application described in Section 3. We use grid networks to represent the sensor networks where sensors are arranged in a rectangular grid. Each sensor is connected to its four neighboring sensors in the cardinal direction. Those sensors on the edges are connected to three neighboring sensors, and corner sensors are connected to two neighbors. The random variables, which represent the possible location of the targets, are randomly placed on the network.

Table 3 shows the quality solution and runtime (in ms) of DC-DCOP algorithms on sensor network problems with agents varying from 4 to 20. Both AC-DPOP- and CAC-DPOP-based algorithms run from smaller (4 sensors) to larger instances (20 sensors) without timeout and have slightly higher solution quality than HCMS-based algorithms. However, both versions of C-DSA outperform all other algorithms by providing the best solution quality from small to large instances. In addition, C-DSA-based algorithms have smaller runtime than AC-DPOP- and CAC-DPOP-based since C-DSA is a local search algorithm on C-DCOP. However, on grid network problems, HCMS-based algorithms execute faster than C-DSA-based algorithms.

9 Conclusions

In many real-world applications, agents often act in a complex and dynamic environment. While DCOPs have been widely used to solve several multi-agent problems, the formulation lacks the capability to model the dynamic and continuous nature in complex environments. Consequently, researchers have proposed D-DCOPs to model how the environment changes over time and C-DCOPs to model the continuous domain of decision variables. However, they can only address the DCOP limitations in isolation. In this paper, we introduced Dynamic Continuous DCOPs (DC-DCOPs), which model *both* the dynamic environment *and* decision variables with continuous domain. To solve DC-DCOPs, we proposed several sequential greedy algorithms that can use any off-the-shelf C-DCOP algorithms to solve DC-DCOPs and we discussed their theoretical properties. Finally, we evaluated our algorithms on random networks and on distributed sensor network problems, which are our motivating application for this line of work.

Acknowledgments

This work is partially supported by NSF grants 1812619 and 1838364. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

1. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE–a Java agent development framework. In: Multi-agent programming, pp. 125–147 (2005)
2. Burke, D., Brown, K.: Efficiently handling complex local problems in distributed constraint optimisation. In: Proceedings of ECAI. pp. 701–702 (2006)
3. Choudhury, M., Mahmud, S., Khan, M.M.: A particle swarm based algorithm for functional distributed constraint optimization problems. In: Proceedings of AAAI (2020)
4. Deng, Y., An, B.: Speeding up incomplete GDL-based algorithms for multi-agent optimization with dense local utilities. In: Proceedings of IJCAI. pp. 31–38 (2020)
5. Fargier, H., Lang, J., Schiex, T.: Mixed constraint satisfaction: A framework for decision problems under incomplete knowledge. In: Proceedings of AAAI. pp. 175–180 (1996)
6. Farinelli, A., Rogers, A., Petcu, A., Jennings, N.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: Proceedings of AAMAS. pp. 639–646 (2008)

7. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research **61**, 623–698 (2018)
8. Fioretto, F., Yeoh, W., Pontelli, E.: Multi-variable agents decomposition for DCOPs. In: Proceedings of AAAI. pp. 2480–2486 (2016)
9. Fioretto, F., Yeoh, W., Pontelli, E.: A multiagent system approach to scheduling devices in smart homes. In: Proceedings of AAMAS. pp. 981–989 (2017)
10. Fioretto, F., Yeoh, W., Pontelli, E., Ma, Y., Ranade, S.: A DCOP approach to the economic dispatch with demand response. In: Proceedings of AAMAS. pp. 999–1007 (2017)
11. Fransman, J., Sijs, J., Dol, H., Theunissen, E., Schutter, B.D.: Bayesian-DPOP for continuous distributed constraint optimization problems. In: Proceedings of AAMAS. pp. 1961–1963 (2019)
12. Hoang, K.D., Fioretto, F., Hou, P., Yeoh, W., Yokoo, M., Zivan, R.: Proactive dynamic distributed constraint optimization problems. Journal of Artificial Intelligence Research **74**, 179–225 (2022)
13. Hoang, K.D., Fioretto, F., Hou, P., Yokoo, M., Yeoh, W., Zivan, R.: Proactive dynamic distributed constraint optimization. In: Proceedings of AAMAS. pp. 597–605 (2016)
14. Hoang, K.D., Hou, P., Fioretto, F., Yeoh, W., Zivan, R., Yokoo, M.: Infinite-horizon proactive dynamic DCOPs. In: Proceedings of AAMAS. pp. 212–220 (2017)
15. Hoang, K.D., Yeoh, W., Yokoo, M., Rabinovich, Z.: New algorithms for continuous distributed constraint optimization problems. In: Proceedings of AAMAS. p. 502–510 (2020)
16. Holland, A., O'Sullivan, B.: Weighted super solutions for constraint programs. In: Proceedings of AAAI. pp. 378–383 (2005)
17. Kim, Y., Krainin, M., Lesser, V.: Effective variants of the max-sum algorithm for radar coordination and scheduling. In: Proceedings of WI-IAT. pp. 357–364 (2011)
18. Kumar, A., Faltings, B., Petcu, A.: Distributed constraint optimization with structured resource constraints. In: Proceedings of AAMAS. pp. 923–930 (2009)
19. Lass, R., Sultanik, E., Regli, W.: Dynamic distributed constraint reasoning. In: Proceedings of AAAI. pp. 1466–1469 (2008)
20. 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)
21. Miller, S., Ramchurn, S., Rogers, A.: Optimal decentralised dispatch of embedded generation in the smart grid. In: Proceedings of 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. Paulos, A., Dasgupta, S., Beal, J., Mo, Y., Hoang, K.D., Lyles, J.B., Pal, P., Schantz, R., Schewe, J., Sitaraman, R., Wald, A., Wayllace, C., Yeoh, W.: A framework for self-adaptive dispersal of computing services. In: IEEE Self-Adaptive and Self-Organizing Systems Workshops (June 2019)
24. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: Proceedings of IJCAI. pp. 1413–1420 (2005)
25. Petcu, A., Faltings, B.: Superstabilizing, fault-containing multiagent combinatorial optimization. In: Proceedings of AAAI. pp. 449–454 (2005)
26. Petcu, A., Faltings, B.: Optimal solution stability in dynamic, distributed constraint optimization. In: Proceedings of IAT. pp. 321–327 (2007)

27. 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)
28. Sarker, A., Choudhury, M., Khan, M.M.: A local search based approach to solve Continuous DCOPs. In: Proceedings of AAMAS. pp. 1127–1135 (2021)
29. Stranders, R., Farinelli, A., Rogers, A., Jennings, N.: Decentralised coordination of continuously valued control parameters using the Max-Sum algorithm. In: Proceedings of AAMAS. pp. 601–608 (2009)
30. Sultanik, E., Lass, R., Regli, W.: DCOPolis: A framework for simulating and deploying distributed constraint reasoning algorithms. In: Proceedings of AAMAS. pp. 1667–1668 (2008)
31. Tarim, S.A., Manandhar, S., Walsh, T.: Stochastic constraint programming: A scenario-based approach. Constraints **11**(1), 53–80 (2006)
32. Voice, T., Stranders, R., Rogers, A., Jennings, N.: A hybrid continuous max-sum algorithm for decentralised coordination. In: Proceedings of ECAI. pp. 61–66 (2010)
33. Wallace, R., Freuder, E.: Stable solutions for dynamic constraint satisfaction problems. In: Proceedings of CP. pp. 447–461 (1998)
34. Walsh, T.: Stochastic constraint programming. In: Proceedings of ECAI. pp. 111–115 (2002)
35. Yeoh, W., Felner, A., Koenig, S.: BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. Journal of Artificial Intelligence Research **38**, 85–133 (2010)
36. Yeoh, W., Varakantham, P., Sun, X., Koenig, S.: Incremental DCOP search algorithms for solving dynamic DCOPs. In: Proceedings of IAT. pp. 257–264 (2015)
37. Yeoh, W., Yokoo, M.: Distributed problem solving. AI Magazine **33**(3), 53–65 (2012)
38. Yokoo, M. (ed.): Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems. Springer (2001)
39. Zink, M., Westbrook, D., Abdallah, S., Horling, B., Lakamraju, V., Lyons, E., Manfredi, V., Kurose, J., Hondl, K.: Meteorological command and control: An end-to-end architecture for a hazardous weather detection sensor network. In: Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services. USENIX Association (2005)
40. 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)