

Distributed Constraint Optimization Problems and Applications: A Survey

Ferdinando Fioretto^{a,b}, Enrico Pontelli^a, William Yeoh^a

ffiorett | epontell | wyeoh@cs.nmsu.edu

^aNew Mexico State University, Computer Science Department

^bUniversity of Udine, Mathematics and Computer Science Department

Abstract

The field of *Multi-Agent System (MAS)* is an active area of research within Artificial Intelligence, with an increasingly important impact in industrial and other real-world applications. Within a MAS, autonomous agents interact to pursue personal interests and/or to achieve common objectives. *Distributed Constraint Optimization Problems (DCOPs)* have emerged as one of the prominent agent architectures to govern the agents' autonomous behavior, where both algorithms and communication models are driven by the structure of the specific problem. During the last decade, several extensions to the DCOP model have enabled them to support MAS in complex, real-time, and uncertain environments.

This survey aims at providing an overview of the DCOP model, giving a classification of its multiple extensions and addressing both resolution methods and applications that find a natural mapping within each class of DCOPs. The proposed classification suggests several future perspectives for DCOP extensions, and identifies challenges in the design of efficient resolution algorithms, possibly through the adaptation of strategies from different areas.

Keywords: Distributed Constraint Optimization Problems

1. Introduction

An *agent* can be defined as an entity (or computer program) that behaves autonomously within an arbitrary system in the pursuit of some goals. A *multi-agent system (MAS)* is a system where multiple agents interact in the pursuit of goals. Within a MAS, agents may interact with each other *directly*, via communication acts, or *indirectly*, by acting on the shared environment. In addition, agents may decide to *cooperate*, to achieve a common goal, or to *compete*, to serve their own interests at the expense of other agents. In particular, agents may form cooperative *teams*, which can in turn compete against other teams of agents. Multi-agent systems play an important role in *distributed artificial intelligence*, thanks to their ability to model a wide variety of real-world scenarios, where information and control are decentralized and distributed among a set of agents.

Figure 2 illustrates a MAS, representing a sensor network scenario, where a group of agents, equipped with sensors, seeks to determine the position of some targets—identified in the figure as star-shaped objects. Agents may interact with each other—the dotted lines in the figure define the interaction graph. Agents may move away from the current position—the directional arrows illustrate such actions. In addition, various events that dynamically obstruct the sensors of the agents (e.g., the presence of an obstacle along the sensing range of an agent) may dynamically occur. Within a MAS, an agent is:

- *Autonomous*, as it operates without the direct intervention of humans or other entities and has full control over its own actions and internal state (e.g., in the example, an agent can decide to sense, to move, etc.);

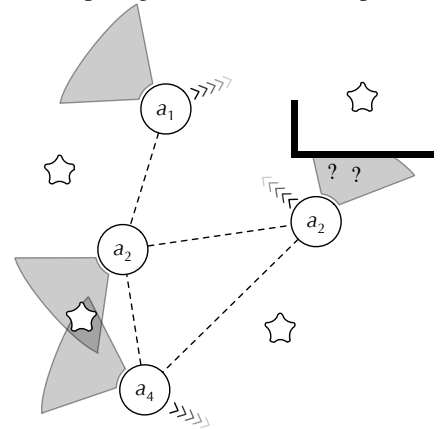


Figure 2: Illustration of a Multi-agent System: Sensors (agents) seek to determine the position of the targets.

- *Interactant*, in the sense that it interacts with other agents in order to achieve its objectives (e.g., in the example, agents may exchange information concerning results of sensing activities);
- *Reactive*, as it responds to changes that occur in the environment and/or to the requests from other agents (e.g., in the example, agents may react with a move to the sudden appearance of obstacles).
- *Proactive*, due to its goal-driven behavior, which allows the agent to take initiatives beyond the reactions in response to its environment.

Agent architectures are the fundamental mechanisms underlying the autonomous agent components, supporting their behavior in real-world, dynamic and uncertain environments. Currently, agent architectures based on *Decision Theory (DT)*, *Game Theory (GT)*, and *Constraint Programming (CP)* have successfully been developed and are popular in the *Autonomous Agents and Multi-Agent Systems (AAMAS)* community.

Decision Theory (DT) [1] assumes that the agent and the environment have inherent uncertainty, and DT models such uncertainty explicitly. Acting in complex and dynamic environments requires agents to deal with various sources of uncertainty. *Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs)* [2] are one of the most general multi-agent frameworks focused on team coordination in presence of uncertainty about agents' actions and observations. The ability to capture a wide range of complex scenarios makes Dec-POMDPs of central interest within MAS research. However, the result of this generality is a high complexity for generating optimal solutions. Dec-POMDPs are non-deterministic exponential (NEXP) complete [2], even for two-agent problems, and scalability remains a critical challenge [3].

Game Theory (GT) [4] studies interactions between self-interested agents, aiming at maximizing the welfare of the participants. Some of the most compelling applications of game theory to MAS have been in the area of *auctions* and *negotiations* [5, 6, 7]. Such approaches model the trading process by which agents can reach agreements on matters of common interest, using market oriented and cooperative mechanisms, such as reaching *Nash equilibria (NEs)*. Typical resolution approaches aim at deriving a set of equilibrium strategies for each agent, such that, when these strategies are employed, no agent can profit by unilaterally deviating from their strategies. A limitation of game-theoretical approaches is the lack of an agent's ability to reason upon a global objective, as the underlying model relies on the interactions of self-interested agents.

Constraint Programming (CP) [8] aims at solving decision-making problems formulated as optimization problems of some real-world objective. Constraint programs use the notion of constraints—i.e., relations among entities of the problems—in both problem modeling and problem solving. CP relies on inference techniques which prevent the exploration of those parts of the solution search space whose assignments are inconsistent with the constraints and/or dominated with respect to the objective function. *Distributed Constraint Optimization Problems (DCOPs)* [9, 10, 11] are problems where agents need to coordinate their value assignments, in a decentralized manner, to optimize their objective functions. DCOPs focus on attaining a global optimum given the interaction graph of a collection of agents. This approach is flexible and can effectively model a wide range of problems. In general, problem solving and communication strategies are directly linked in DCOPs. This feature makes DCOP's algorithmic components suitable for exploiting the structure of the interaction graph to generate efficient solutions.

The absence of a framework to model dynamic problems and uncertainty makes DCOPs unsuitable at solving certain classes of multi-agent problems, such as those characterized by actions uncertainty and dynamic environments. However, since its original introduction, the DCOP model has undergone a continuous evolution, to capture a variety of characteristics regarding agents' behavior and the environment in which agents act. Researchers have proposed a number of DCOP frameworks that differ from each other in terms of expressivity and classes of problem they can target, extending the DCOP model to handle both dynamic and uncertain environments. However, current research has not explored how the different DCOP frameworks related within the general MAS context, which is critical to understand: **(a)** what resolution methods could be borrowed from other MAS paradigms, and **(b)** what applications can be most effectively modeled within each framework. This survey aims to comprehensively analyze and categorize the different DCOP frameworks proposed by the AAMAS community. We do so by presenting an extensive review of the DCOP model and its extensions, the different resolution methods, as well as a number of applications modeled within each particular DCOP extension. This analysis provides also opportunities to identify and discuss future directions in the general DCOP research area.

This survey paper is organized as follows. In the next section we introduce DCOPs, give an overview on the representation and coordination models adopted in DCOP resolution, and propose a classification of the different

variants of DCOPs based on the characteristics of the agents and the environment. Section 3 presents the classical DCOP model, as well as two notable extensions, characterized respectively by asymmetric function utilities and multi-objective optimization. In Section 4, we discuss DCOP models in which agents act in an uncertain environment and agents have only partial knowledge of the environment in which they act. In Section 5, we present DCOP models where the environment changes over time. Section 6 discusses DCOP models in which agents are non-cooperative. For each of these models, we introduce their formal definitions, related concepts, resolution algorithms, and possible application domains. In Section 7, we describe a number of applications that have been proposed in the DCOP literature. Section 8 provides some potential future directions for research and Section 9 concludes the work. To facilitate the reading of this survey, we have provided in Table 1 a summary of the most commonly used notations.

List of key symbols			
a_i	Agent	$\pi(\cdot)$	Projection operator
x_i	Decision variable	$p_i(\cdot)$	Probability function
r_i	Random variable	L_{a_i}	a_i 's local variables
D_i	Domain of x_i	N_{a_i}	a_i 's neighbors
Ω_i	Event space of r_i	C_{a_i}	a_i 's children
f_i	Reward function	PC_i	a_i 's pseudo-children
\mathbf{x}^i	Scope of f_i	P_{a_i}	a_i 's parent
m	Number of agents	PP_{a_i}	a_i 's pseudo-parents
n	Number of variables	$\alpha(f_i)$	agents whose variables are in \mathbf{x}^i
q	Number of random variables	E_C	Set of edges of the constraint graph
k	Number of reward functions	E_T	Tree edges of the pseudo-tree
\mathbf{F}_g	Global objective function	E_F	Set of edges of the factor graph
$\vec{\mathbf{F}}$	Vector of objective functions	w^*	Induced width of the pseudo-tree
F_i	Objective function in $\vec{\mathbf{F}}$	d	Size of the largest domain
$\vec{\mathbf{F}}^o$	Utopia point	l	Size of the largest neighborhood
\perp	Infeasible value	z	Size of the largest local variable set
σ	Complete solution	s	Maximal sample size
$\sigma_{\mathbf{x}^i}$	Partial solution of scope \mathbf{x}^i in σ	p	Size of the Pareto set
Σ	State space	b	Size of the largest bin

Table 1: Commonly Used Symbols and Notations

2. DCOP Classification

The DCOP model has undergone a process of continuous evolution to capture diverse characteristics of agents behavior and the environment in which they operate. We propose a classification of DCOP models from a Multi-Agent Systems perspective, that accounts for the different assumptions made about the behavior of agents and their interactions with the environment. The classification is based on the following elements (summarized in Table 2):

- **Agent Behavior:** This parameter captures the stochastic nature of the effects of an action being executed. In particular, we distinguish between *deterministic* and *stochastic* effects.
- **Agent Knowledge:** This parameter captures the knowledge of an agent about its own state and the environment—distinguishing between *total* and *partial* knowledge.
- **Agent Teamwork:** This parameter characterizes the approach undertaken by (teams of) agents to solve a distributed problem. It can be either a *cooperative* resolution approach or a *competitive* resolution approach. In the former class, all agents cooperate to achieve a common goal (i.e., optimize a utility function). In the latter class, each agent (or team of agents) seeks to achieve its own individual goal.
- **Environment Behavior:** This parameter captures the exogenous properties of the environment. For example, it is possible to distinguish between *deterministic* and *stochastic* responses of the environment to the execution of an action.

ELEMENT		CHARACTERIZATION	
AGENT(S)	BEHAVIOR	Deterministic	Stochastic
	KNOWLEDGE	Total	Partial
	TEAMWORK	Cooperative	Competitive
ENVIRONMENT	BEHAVIOR	Deterministic	Stochastic
	EVOLUTION	Static	Dynamic

Table 2: DCOP classification elements.

		Environment Behavior	
		DETERMINISTIC	STOCHASTIC
Environment Evolution	STATIC	<i>classical-DCOP</i>	<i>probabilistic-DCOP</i>
	DYNAMIC	<i>dynamic-DCOP</i>	—

Table 3: DCOPs Models.

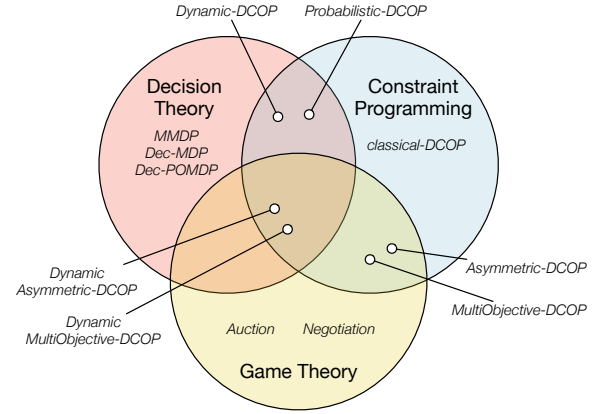


Figure 3: DCOPs within a MAS perspective.

- **Environment Evolution:** This parameter captures whether the DCOP is *static* (i.e., it does not change over time) or *dynamic* (i.e., it changes over time).

Figure 3 illustrates a categorization of the DCOP models proposed to date from a MAS perspective. In particular, we focus on the DCOP models proposed at the junction of *Constraint Programming* (CP), *Game Theory* (GT), and *Decision Theory* (DT). The *classical* DCOP model is directly inherited from CP and characterized by a static model, a deterministic environment and agent behavior, total agent knowledge, and with cooperative agents. Concepts from auctions and negotiations, traditionally explored in GT, have influenced the DCOP framework, leading to Asymmetric DCOPs, which has asymmetric agent payoffs, and Multi-Objective DCOPs. The DCOP framework has borrowed fundamental DT concepts related to modeling uncertain and dynamic environments, resulting in models like Probabilistic DCOPs and Dynamic DCOPs. Researchers from the DCOP community have also designed solutions that inherit from all of the three communities.

In the next sections, we will describe the different DCOP frameworks starting from classical DCOPs before proceeding to its various extensions. We will focus on a categorization based on three dimensions: *Agent knowledge*, *environment behavior*, and *environment evolution*. We assume a *deterministic agent behavior*, *fully cooperative agent teamwork*, and *total agent knowledge* (unless otherwise specified), as they are, by far, common assumptions adopted by the DCOP community. The DCOP models associated to such categorization are summarized in Table 3. The bottom-right entry of the table is left empty, indicating a promising model with dynamic and uncertain environments that, to the best of our knowledge, has not been explored yet. There has been only a modest amount of effort in modeling the different aspects of teamwork within the DCOP community. We will describe a formalism that has been adopted to model DCOPs with mixed cooperative and competitive agents in Section 6.

3. Classical DCOP

With respect to our categorization, in the *classical DCOP* model [9, 10, 11], the agents are completely cooperative and they have deterministic behavior and total knowledge. Additionally, the environment is static and deterministic.

In this section, we review the formal definitions of classical DCOPs, present some relevant solving algorithms, and provide details of selected variants of classical DCOPs of particular interest.

3.1. Definition

Definition 1 (DCOP). A classical DCOP is described by a tuple $P = \langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F}, \alpha \rangle$, where:

- $\mathbf{A} = \{a_1, \dots, a_p\}$ is a finite set of agents.
- $\mathbf{X} = \{x_1, \dots, x_n\}$ is a finite set of variables, with $n \geq p$.
- $\mathbf{D} = \{D_1, \dots, D_n\}$, the set of domains for the variables in \mathbf{X} , with D_i being the domain of variable x_i .
- $\mathbf{F} = \{f_1, \dots, f_k\}$ is a finite set of objective functions, with $f_i : \times_{x_j \in \mathbf{x}^i} D_j \rightarrow \mathbb{R}^+ \cup \{\perp\}$, where $\mathbf{x}^i \subseteq \mathbf{X}$ is the set of variables relevant to f_i , referred to as the scope of f_i , and \perp is a special element used to denote that a given combination of values for the variables in \mathbf{x}^i is not allowed.¹ Each function f_i represents a factor in a global objective function, $\mathbf{F}_g(\mathbf{X}) = \sum_{i=1}^k f_i(\mathbf{x}^i)$. In the DCOP literature, the functions f_i are also called constraints, cost functions, utility functions, payoff functions, or reward functions.
- $\alpha : \mathbf{X} \rightarrow \mathbf{A}$ is an onto total function, from variables to agents, which assigns the control of each variable $x \in \mathbf{X}$ to an agent $\alpha(x)$.

With a slight abuse of notation, we will denote with $\alpha(f_i)$ the set of agents whose variables are involved in the scope of f_i , i.e., $\alpha(f_i) = \{\alpha(x) \mid x \in \mathbf{x}^i\}$. A solution is a value assignment for a subset of variables of \mathbf{X} . A solution is *complete* if it assigns a value to each variable in \mathbf{X} . We will use the notation σ to denote a complete solution and $\sigma_{\mathbf{x}^i}$ to denote the partial solution obtained by projecting σ on the variables in the scope \mathbf{x}^i . The goal in a DCOP is to find a complete solution that maximizes the total problem reward expressed by its reward functions:

$$\sigma^* = \operatorname{argmax}_{\sigma \in \Sigma} \mathbf{F}_g(\sigma) = \operatorname{argmax}_{\sigma \in \Sigma} \sum_{f_i \in \mathbf{F}} f_i(\sigma_{\mathbf{x}^i}), \quad (1)$$

where Σ is the *state space*, defined as the set of all possible complete solutions. Let us also introduce the following notations. Given an agent a_i , we denote with $L_{a_i} = \{x_j \in \mathbf{X} \mid \alpha(x_j) = a_i\}$ the set of variables controlled by agent a_i , or its *local variables*, and we denote with $N_{a_i} = \{a'_i \in \mathbf{A} \mid a_i \neq a'_i, \exists f_j \in \mathbf{F}, x_r, x_s \in \mathbf{x}^j, \alpha(x_r) = a_i \wedge \alpha(x_s) = a'_i\}$ the set of its *neighboring agents*.

A constraint f_i is said to be *hard* if $\forall \sigma \in \Sigma$ we have that $f_i(\sigma_{\mathbf{x}^i}) \subseteq \{0, \perp\}$. Otherwise, the constraint is said to be *soft*.

3.2. DCOP: Representation and Coordination

Representation in DCOPs plays a fundamental role, both from an agent coordination perspective and from an algorithmic perspective. We discuss here the most predominant representations adopted in various DCOP algorithms. Let us start by describing some widely adopted assumptions regarding agent knowledge and coordination, which will apply throughout this document, unless otherwise stated:

- A variable and its domain are known exclusively to the agent controlling it and its neighboring agents.
- Each agent knows the reward values of the constraints involving at least one of its local variables. No other agent has knowledge about such constraints.
- Each agent knows exclusively (and it may communicate with) its own neighboring agents.

3.2.1. Constraint Graph

Given a DCOP P , $G_P = (\mathbf{X}, E_C)$ is the *constraint graph* of P , where an undirected edge $\{x, y\} \in E_C$ exists if and only if there exists $f_j \in \mathbf{F}$ such that $\{x, y\} \subseteq \mathbf{x}^j$. A constraint graph is a standard way to visualize a DCOP instance. It underlines the agents' locality of interactions and therefore it is commonly adopted by DCOP resolution algorithms.

Given an ordering o on \mathbf{X} , we say that a variable x_i has a higher *priority* with respect to a variable x_j if x_i appears before x_j in o . Given a constraint graph G_P and an ordering o on its nodes, the *induced graph* G_P^* on o , is the graph obtained by connecting nodes, processed in increasing order of priority, to all their higher-priority neighbors. For a

¹We assume sets of variables to be sorted according to a fixed order of \mathbf{X} .

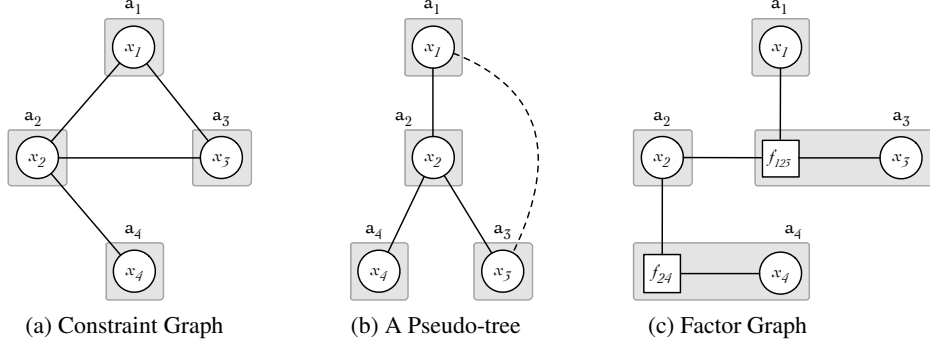


Figure 4: DCOP representations.

given node, the number of higher-priority neighbors is referred to as its *width*. The *induced width* w_o^* of G_P is the maximum width over all the nodes of G_P^* on ordering o .

Figure 4(a) shows an example constraint graph of a DCOP with four agents a_1 through a_4 , each controlling one variable with domain $\{0,1\}$. There are two constraint: a ternary constraint, f_{123} with scope $\mathbf{x}^{123} = \{x_1, x_2, x_3\}$ and represented by a clique among x_1, x_2 and x_3 , and a binary constraint f_{24} with scope $\mathbf{x}^{24} = \{x_2, x_4\}$.

3.2.2. Pseudo-Tree

A number of DCOP algorithms require a partial ordering among the agents. In particular, when such order is derived from a depth-first search exploration, the resulting structure is known as (*DFS*) *pseudo-tree*. A pseudo-tree arrangement for a DCOP P is a subgraph $T_P = \langle \mathbf{X}, E_T \rangle$ of G_P such that T_P is a spanning tree of G_P —i.e., a connected subgraph of G_P containing all the nodes and being a rooted tree—with the following additional condition: for each $x, y \in \mathbf{X}$, if $\{x, y\} \subseteq \mathbf{x}^i$ for some $f_i \in \mathbf{F}$, then x, y appear in the same branch of T_P (i.e., x is an ancestor of y in T_P or vice versa). Edges of G_P that are *in* (respectively *out* of) E_T are called *tree edges* (respectively *backedges*). The tree edges connect parent-child nodes, while backedges connect a node with its *pseudo-parents* and its *pseudo-children*. We use C_{a_i} , PC_{a_i} , P_{a_i} , PP_{a_i} , to denote the set of children, pseudo-children, parent and pseudo-parents of the agent a_i .

Both constraint graph and pseudo-tree representations cannot deal explicitly with n -ary constraints (functions whose scope has more than two variables). A typical artifact to deal with n -ary constraints in a pseudo-tree representation is to introduce a virtual variable which monitors the value assignments for all the variables in the scope of the constraint, and generates the reward values [12]—the role of the virtual variables can be delegated to one of the variables participating in the constraint [13, 14].

Figure 4(b) shows one possible pseudo-tree of the example DCOP in Figure 4(a), where $C_{a_1} = \{x_2\}$, $PC_{a_1} = \{x_3\}$, $P_{a_4} = \{x_2\}$, and $PP_{a_3} = \{x_1\}$. The solid lines are tree edges and dotted lines are backedges.

3.2.3. Factor Graph

Another way to represent DCOPs is through a *factor graph* [15]. A *factor graph* is a bipartite graph used to represent the factorization of a function. In particular, given the global objective function \mathbf{F}_g , the corresponding factor graph $F_P = \langle \mathbf{X}, \mathbf{F}, E_F \rangle$ is composed of variable nodes $x_i \in \mathbf{X}$, factor nodes $f_j \in \mathbf{F}$ and edges E_F such that there is an undirected edge between factor node f_j and variable node x_i if $x_i \in \mathbf{x}^j$.

Factor graphs can handle n -ary constraints explicitly. To do so, they use a similar method as that adopted within pseudo-trees with n -ary constraints: they delegate the control of a factor node to one of the agents controlling a variable in the scope of the constraint. From an algorithmic perspective, the algorithms designed over factor graphs can directly handle n -ary constraints, while algorithms designed over pseudo-trees require changes in the algorithm design so to delegate the control of the n -ary constraints to some particular entity.

Figure 4(c) shows the factor graph of the example DCOP in Figure 4(a), where each agent a_i controls its variable x_i and, in addition, a_3 controls the constraint f_{123} and a_4 controls f_{24} .

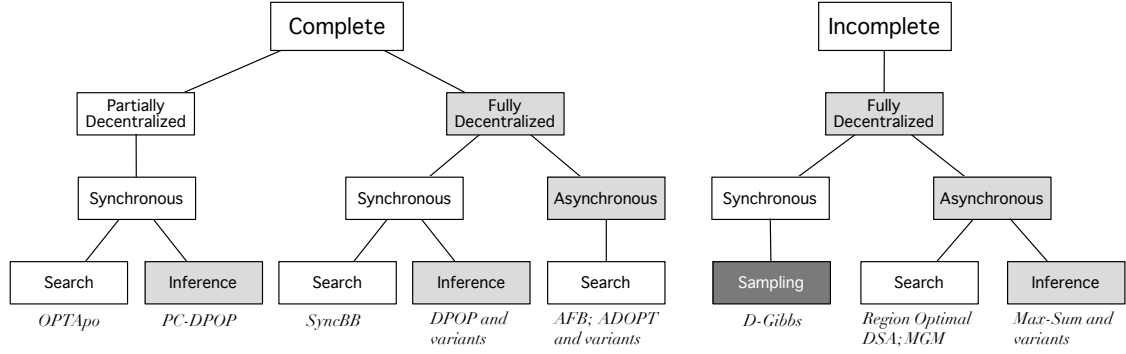


Figure 5: Classical DCOP Algorithm Taxonomy.

3.3. Algorithms

The field of classical DCOPs is mature and a number of different algorithms have been proposed. DCOP algorithms can be classified as being either *complete* or *incomplete*, based on whether they can guarantee the optimal solution or they trade optimality for smaller execution times, producing approximated solutions. In addition, each of these classes can be categorized into several groups, such as: **(1)** *partially* or *fully decentralized*, depending on the degree of locality exploited by the algorithms; and **(2)** *synchronous* or *asynchronous*, based on the way local information is updated. Finally, the resolution process adopted by each algorithm can be classified in three categories [16]:

- *Search-based methods*, which are based on the use of search techniques to explore the space of possible solutions. These techniques are often derived from corresponding search techniques developed for centralized AI search problems, such as best-first search and depth-first search.
- *Inference-based methods*, which are inspired from dynamic programming and belief propagation techniques. These techniques allow agents to exploit the structure of the constraint graph to aggregate rewards from their neighbors, effectively reducing the problem size at each step of the algorithm.
- *Sampling-based methods*, which are incomplete approaches that sample the search space to approximate a function (usually a probability distribution) as a product of statistical inference.

Figure 5 illustrates a taxonomy of classical DCOP algorithms. In the following subsections, we briefly describe some representative complete and incomplete algorithms of each of the classes introduced above. A detailed description of the DCOP algorithms is out of the scope of this document. We refer the interested readers to the original articles that introduce each algorithm.

Throughout this document, we will often refer to the following notation when discussing the complexity of the algorithms: the size of the largest domain is denoted by $d = \max_{D_i \in \mathbf{D}} |D_i|$, and w^* refers to the induced width of the pseudo-tree.

COMPLETE ALGORITHMS

SynchBB [17]. *Synchronous Branch-and-Bound* (SynchBB) is a complete, synchronous, search-based algorithm that can be considered as a distributed version of a branch-and-bound algorithm. It uses a complete ordering of the agents in order 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. SynchBB agents space requirement and maximum size of message are in $O(n)$, while they require, in the worst case, to perform $O(d^m)$ number of operations. The network load is also in $O(d^m)$.

AFB [18]. *Asynchronous Forward Bounding* (AFB) is a complete, asynchronous, search-based algorithm that can be considered as the asynchronous version of SynchBB. In this algorithm, agents communicate their reward estimates,

which in turn are used to compute bounds and prune the search space. In AFB, agents extend a CPA sequentially, provided that the lower bound on its reward does not exceed the global bound, that is, the reward of the best solution found so far. Each agent performing an assignment (the “assigning” agent) triggers asynchronous checks of bounds, by sending *forward* messages containing copies of the CPA to agents that have not yet assigned their variables. The unassigned agents that receive a CPA, estimate the lower bound of the CPA, given their local view of the constraint graph. They also send such estimates back to the agent that originated the *forward* message. This assigning agent will receive these estimates asynchronously and aggregate them into an updated lower bound. If the updated lower bound exceeds the current upper bound, the agent initiates a backtracking phase. As in SynchBB, the worst case complexity for network load and agent’s operations is $O(d^m)$, while the size of messages and each agent’s space requirement are in $O(n)$.

ADOPT [9]. *Asynchronous Distributed OPTimization (ADOPT)* is a complete, asynchronous, search-based algorithm that makes use of a DFS pseudo-tree ordering of the agents. The algorithm relies on maintaining, in each agent, lower and upper bounds on the solution reward for the subtree rooted at its node(s) in the DFS tree. Agents explore partial solutions in best-first order, that is, in increasing lower bound order. Agents use *COST* messages (propagated upwards in the DFS pseudotree) and *THRESHOLD* and *VALUE* messages (propagated downwards in the tree) to iteratively tighten the lower and upper bounds, until the lower bound of the minimum cost solution is equal to its upper bound. ADOPT agents store lower bounds as thresholds, which can be used to prune partial solutions that are provably sub-optimal. ADOPT agents need to maintain a *context* which stores the assignments of higher priority neighbors, and a lower bound and an upper bound for each domain value and child; thus, the space requirement for each agent is in $O(d(l+1))$, where $l = \max_{a_i \in A} |N_{a_i}|$. Its worst case network load and agent complexity is $O(d^m)$, while its maximum message size is in $O(h)$. ADOPT has been extended in several ways. In particular, *BnB-ADOPT* [19, 20] uses a branch-and-bound method to reduce the amount of computation performed during search, and *ADOPT(k)* combines both ADOPT and BnB-ADOPT into an integrated algorithm [21]. There are also extensions that trade solution optimality for smaller runtimes [22], extensions that use more memory for smaller runtimes [23], and extensions that maintain soft arc-consistency [24, 25, 26, 27].

DPOP [10]. *Distributed Pseudo-tree Optimization Procedure (DPOP)* is a complete, synchronous, inference-based algorithm that makes use of a DFS pseudo-tree ordering of the agents. It involves three phases. In the first phase, the agents order themselves into a DFS pseudo-tree. In the second phase, each agent, starting from the leaves of the pseudo-tree, aggregates the rewards in its subtree for each value combination of variables in its separator.² The aggregated rewards are encoded in a *UTIL* message, which is propagated from children to their parents, up to the root. In the third phase, each agent, starting from the root of the pseudo-tree, selects the optimal values for its variables. The optimal values are calculated based on the *UTIL* messages received from the agent’s children and the *VALUE* message received from its parent. The *VALUE* messages contain the optimal values of the agents and are propagated from parents to their children, down to the leaves of the pseudo-tree. Thus, DPOP generates a number of messages that is in $O(m)$. However, the size of the messages and the agent’s space requirement are exponential in the induced width of the pseudo-tree: $O(d^{w^*})$. Finally, the number of operations performed by DCOP agents is in the order complexity of $O(d^{w^*+z})$, with $z = \max_{a_i \in A} |L_i|$. DPOP has also been extended in several ways to enhance its performance and capabilities. O-DPOP and MB-DPOP trade runtimes for smaller memory requirements [28, 29], A-DPOP trades solution optimality for smaller runtimes [30], SS-DPOP trades runtime for increased privacy [31], PC-DPOP trades privacy for smaller runtimes [32], H-DPOP propagates hard constraints for smaller runtimes [33], BrC-DPOP enforces branch consistency for smaller runtimes [34], and ASP-DPOP is a declarative version of DPOP that uses Answer Set Programming [35].

OptAPO [36]. *Optimal Asynchronous Partial Overlay (OptAPO)* is a complete, synchronous, search-based algorithm. It trades agent privacy for smaller runtimes through partial centralization. It employs a cooperative mediation schema, where agents can act as mediators and propose value assignments to other agents. In particular, agents check if there is a conflict with some neighboring agent. If a conflict is found, the agent with the highest priority acts as a mediator. During mediation, OptAPO solves subproblems using a centralized branch-and-bound-based search, and

²The separator of a_i contains all ancestors of a_i in the pseudo-tree (through tree edges or back edges) that are connected to a_i or one of its descendants.

when solutions of overlapping subproblems still have conflicting assignments, the solving agents increase the centralization to resolve them. By sharing their knowledge with centralized entities, agents can improve their local decisions, reducing the communication costs. For instance, the algorithm has been shown to be superior to ADOPT on simple combinatorial problems. However, it is possible that several mediators solve overlapping problems, duplicating efforts [32], which can be a bottleneck especially for dense problems. The worst case agent complexity is in $O(d^n)$, as an agent might solve the entire problem. The agent space requirement is in $O(nd)$, as a mediator agent needs to maintain the domains of all the variables involved in the mediation section, while the message size is in the order of $O(d)$. The network load decreases with the amount of partial centralization required, however, its worst case order complexity is exponential in the number of agents $O(d^m)$. The original version of OptAPO has been shown to be incomplete [37], but a complete variant has been proposed [37].

INCOMPLETE ALGORITHMS

Max-Sum [38]. *Max-Sum* is an incomplete, asynchronous, inference-based algorithm based on belief propagation. It operates on factor graphs by performing a marginalization process of the reward functions, and optimizing the rewards for each given variable. This process is performed by recursively propagating messages between variable nodes and functions nodes. The value assignments take into account their impact on the marginalized reward function. Max-Sum is guaranteed to converge to an optimal solution in acyclic graphs, but convergence is not guaranteed in cyclic graphs. Nevertheless, it has been shown to often converge in practice. Max-Sum has also been extended in several ways to improve it. Bounded Max-Sum is able to bound the quality of the solutions found by removing a subset of edges from a cyclic DCOP graph to make it acyclic, and by running Max-Sum to solve the acyclic problem [39]. Improved Bounded Max-Sum improves on the error bounds [40], and Max-Sum_AD guarantees convergence in acyclic graphs through a two-phase value propagation phase [41]. Max-Sum and its extensions have been successfully employed to solve a number of large scale, complex MAS applications (see Section 7).

Region Optimal [42]. *Region-optimal* algorithms are incomplete, synchronous, search-based algorithms that allow users to specify regions of the constraint graph (e.g., regions with a maximum size of k agents [42], t hops from each agent [43], or a combination of both size and hops [44]) and solve the subproblem within each region optimally. The concept of k -optimality is defined with respect to the number of agents whose assignments conflict, whose set is denoted by $c(\sigma, \sigma')$, for two assignments σ and σ' . The deviating cost of σ with respect to σ' , denoted by $\Delta(\sigma, \sigma')$, is defined as the difference of the aggregated reward associated to the assignment σ ($F(\sigma)$) minus the reward associated to σ' ($F(\sigma')$). An assignment σ is k -optimal if $\forall \sigma' \in \Sigma$, such that $|c(\sigma, \sigma')| \leq k$, we have that $\Delta(\sigma, \sigma') \geq 0$. In contrast, the concept of t -distance emphasizes the number of hops from a central agent a of the region $\Omega_t(a)$, that is the set of agents which are separated from a by at most t hops. An assignment σ is t -distance optimal if, $\forall \sigma' \in \Sigma$, $F(\sigma) \geq F(\sigma')$ with $c(\sigma, \sigma') \subseteq \Omega_t(a)$, for any $a \in \mathbf{A}$. The *Distributed Asynchronous Local Optimization* (DALO) simulator provides a mechanism to coordinate the decision of local groups of agents based on the concepts of k -optimality and t -distance [43]. The quality of the solutions found is bounded by a function of k or t [44].

MGM [45]. The *Maximum Gain Message* (MGM) is an incomplete, synchronous, search-based algorithm that performs a distributed local search. Each agent starts by assigning a random value to each of its variables. Then, it sends this information to all its neighbors. Upon receiving the values of its neighbors, it calculates the maximum gain in reward if it changes its value and sends this information to all its neighbors as well. Upon receiving the gains of its neighbors, it changes its value if its gain is the largest among its neighbors. This process repeats until a termination condition is met.

DSA [46]. The *Distributed Stochastic Algorithm* (DSA) is an incomplete, synchronous, search-based algorithm that is similar to MGM, except that each agent does not send its gains to its neighbors and it does not change its value to the value with the maximum gain. Instead, it decides stochastically if it takes on the value with the maximum gain or other values with smaller gains. This stochasticity allows DSA to escape from local minima. Similarly to MGM, it repeats until a termination condition is met.

D-Gibbs [47]. The *Distributed Gibbs* (D-Gibbs) algorithm is an incomplete, synchronous, sampling-based algorithm that extends the Gibbs sampling process [48] by tailoring it to solve DCOPs in a decentralized manner. The Gibbs sampling process is a centralized Markov chain Monte Carlo algorithm that can be used to approximate joint

probability distributions. By mapping DCOPs to maximum a-posteriori estimation problems, probabilistic inference algorithms like Gibbs sampling can be used to solve DCOPs.

3.4. Relevant Uses

The classical DCOP model is capable of representing a wide range of MAS applications, especially those where agents in a team need to work cooperatively to achieve a single goal in a static, deterministic, and fully observable environment. Exploring the domain structural properties, as well as understanding the requirements of the problem designer, is crucial to design and apply effective DCOP algorithms. When an optimal solution is required, then a complete algorithm can be used to solve the problem. However, if particular assumptions can be made on the problem structure, more efficient solutions can be adopted. For instance, if the constraint graph of the DCOP is always a *Direct Acyclic Graph (DAG)*, then an incomplete inference-based algorithm, like Max-Sum, is sufficient to guarantee the optimality of the solution found.

Complete algorithms are often unsuitable for tackling large-scale problems, due to their exponential requirements in time or memory. In contrast, incomplete algorithms are more appropriate to rapidly find solutions, at the cost of sacrificing optimality. The communication requirements also need to be taken into account. For example, when communication is unreliable, it is not recommended to employ search-based solutions, such as ADOPT or AFB, where communication requirements are exponential in the size of the problem. In contrast, inference-based algorithms are more reliable in the presence of uncertain communication networks as they, in general, require only a linear number of messages to complete their computations.

A popular application that is often referenced in the classical DCOP literature is the *Distributed Multi-Event Scheduling* (also known as *Meeting Scheduling*) [49]. It captures generic scheduling problems where one wishes to schedule a set of events within a time range. Each event is defined by (i) the time required to complete the event, (ii) the resources required to complete the event, and (iii) the cost of using such resources at a given time. A scheduling conflict occurs if two events with at least one common resource are scheduled in overlapping time slots. The goal is to maximize the overall reward, defined as the net gain between the opportunity benefit and opportunity cost of scheduling various events.

For an in-depth description of the above problem and for an extensive list of applications and their mappings to the suitable DCOP classes, we refer the reader to Section 7.

3.5. Notable Variant: Asymmetric DCOPs

Asymmetric DCOPs [50] are used to model multi-agent problems where two variables in the scope of the same reward function can receive different rewards from each other. Such a problem cannot be naturally represented by classical DCOPs, which require that all variables in the scope of the same reward function receive the same rewards as each other.

3.5.1. Definition

An *Asymmetric DCOP* is defined by a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F}, \alpha \rangle$, where $\mathbf{A}, \mathbf{X}, \mathbf{D}$ and α are defined as in Definition 1, and each $f_i \in \mathbf{F}$ is defined as: $f_i : \times_{x_j \in \mathbf{X}^i} D_j \times \alpha(f_i) \rightarrow (\mathbb{R}^+ \cup \{\perp\})$. In other words, an Asymmetric DCOP is a DCOP where the reward that an agent obtains from a reward function may differ from the reward another agent obtains from the same reward function.

As rewards for participating agents may differ from each other, the goal in Asymmetric DCOPs is also different than the goal in classical DCOPs. Given a reward function $f_j \in \mathbf{F}$ and complete solution σ , let $f_j(\sigma, a_i)$ denote the reward obtained by agent a_i from reward function f_j with solution σ . Then, the goal in Asymmetric DCOPs is to find the complete solution σ^* :

$$\sigma^* = \operatorname{argmax}_{\sigma \in \Sigma} \sum_{f_j \in \mathbf{F}} \sum_{a_i \in \alpha(f_j)} f_j(\sigma, a_i) \quad (2)$$

3.5.2. Relation to Classical DCOPs

One way to solve MAS problems with asymmetric rewards via classical DCOPs is through the *Private Event As Variables* (PEAV) model [45]. It can capture asymmetric rewards by introducing, for each agent, as many “mirror” variables as the number of variables held by neighboring agents. The consistency with the neighbors’ state variables is imposed by a set of equality constraints. However such formalism suffers from scalability problems, as it may result in a significant increase in the number of variables in a DCOP. In addition, Grinshpoun *et al.* showed that most of the existing incomplete classical DCOP algorithms cannot be used to effectively solve Asymmetric DCOPs, even when the problems are reformulated through the PEAV model [50]. They show that such algorithms are unable to distinguish between different solutions that satisfies all hard constraints, resulting in a convergence to one of those solutions and the inability to escape that local optimum. Therefore, it is important to generate ad-hoc algorithms to solve Asymmetric DCOPs.

3.5.3. Algorithms

The current research direction in the design of Asymmetric DCOP algorithms has focused on adapting existing classical DCOP algorithms to handle the asymmetric rewards. Asymmetric DCOPs require that each agent, whose variables participate in a reward function, coordinate the aggregation of their individual rewards. To do so, two approaches have been identified [51]:

- A *two-phase strategy*, where only one side of the constraint (i.e., the reward of one agent) is considered in the first phase. The other side(s) (i.e., the reward of the other agent(s)) is considered in the second phase once a full assignment is produced. As a result, the rewards of all agents are aggregated.
- A *single-phase strategy*, which requires a systematic check of each side of the constraint before reaching a full assignment. Checking each side of the constraint is often referred to as *back checking*, a process that can be performed either synchronously or asynchronously.

COMPLETE ALGORITHMS

SyncABB-2ph [50]. *Synchronous Asymmetric Branch and Bound - 2-phase (SyncABB-2ph)* is a complete, asymmetric, search-based algorithm that extends SynchBB with the two-phase strategy. Phase 1 emulates SynchBB, where each agent considers the rewards of its constraints with higher-priority agents. Phase 2 starts once a full assignment is found. During this phase, each agent aggregates the sides of the constraints that were not considered during Phase 1 and verifies that the known bound is not exceeded. If the bound is exceeded, Phase 2 ends and the agents restart Phase 1 by backtracking and resuming the search from the lower priority agent that exceeded the bound. The worst case complexity of this algorithm reflects that of SynchBB in terms of space requirement, network load, size of messages, and number of operations performed by each agent.

SyncABB-1ph [50, 52]. *Synchronous Asymmetric Branch and Bound - 1-phase (SyncABB-1ph)* is a complete, asymmetric, search-based algorithm that extends SynchBB with the one-phase strategy. Each agent, after having extended the CPA, updates the bound with its local reward associated to the constraints involving its variables—as done in SynchBB. In addition, the CPA is sent back to the assigned agents to update its bound via a sequence of back checking operations. Its worst case order complexity reflects that of SynchBB in terms of space requirement, network load, size of messages, and number of operations performed by each agent.

ATWB [50]. The *Asymmetric Two-Way Bounding (ATWB)* algorithm is a complete, asynchronous, search-based algorithm that extends AFB to accommodate both forward bounding and backward bounding. The forward bounding is performed analogously to AFB. The backward bounding, instead, is achieved by sending copies of the CPA backward to the agents whose assignments are included in the CPA. Similarly to what is done in AFB, agents that receive a copy of the CPA compute their estimates and send them forward to the assigning agent. Its worst case order complexity reflects that of AFB in terms of space requirement, network load, size of messages, and number of operations performed by each agent.

INCOMPLETE ALGORITHMS

ACLS [50]. *Asymmetric Coordinated Local Search (ACLS)* is an incomplete, synchronous, search-based algorithm. After a random value initialization, each agent exchanges its values with all its neighboring agents. At the end of this

step, each agent knows the values of all its neighboring agents, and identifies all possible improving assignments for its own variables, given the current neighbors choices. Each agent then picks one such assignments, according to the distribution of gains from each proposal assignment, and exchanges it with its neighbors. When an agent receives a proposal assignment, it responds with the evaluation of its side of the constraints, resulting from its current assignment and the proposal assignments of the other agents participating in the constraint. After receiving the evaluations from each of its neighbors, each agent estimates the potential gain or loss derived from its assignment, and commits to a change with a given probability, similarly to agents in DSA, to escape from local minima.

MCS-MGM [50]. *Minimal Constraint Sharing MGM (MCS-MGM)* is an incomplete, synchronous, search-based algorithm that extends MGM by considering each side of the constraint. Like MGM, the agents operate in an iterative fashion, where they exchange their current values at the start of each iteration. Afterwards, each agent sends the utility for its side of each constraint to its neighboring agents that participate in the same constraint.³ Upon receiving this information, each agent knows the total utility for each constraint—by adding together the utilities of both sides of the constraint. Therefore, like in MGM, the agent can calculate the maximum gain in utility if it changes its values, and will send this information to all its neighbors. Upon receiving the gains of its neighbors, each agent changes its value if its gain is the largest among its neighbors.

3.5.4. Relevant Uses of Asymmetric DCOPs

In addition to the requirements given in Section 3.4, Asymmetric DCOPs are suitable when constrained agents receive different rewards for a joint action. Examples of problems that can be suitably modeled as Asymmetric DCOPs are resource allocation problems where agents may have different gains from using the same resource, and where their preferences and constraints regarding usage time slots and durations are expected to be different. For example, in the meeting scheduling problems, agents attending the same meeting may derive different utilities from it. Other examples include supply chain management problems [53], where different consumers have different levels of urgency, and smart grid applications [54], where consumption costs are dictated by a varying pricing scheme, but the price depends on the agents' power loads, which are not evenly distributed among users—as happens, for example, in the case of charging/discharging schedule of electric vehicles [55].

It is important to note that, even though the Asymmetric DCOP model bears similarities with many game-theoretic approaches, these two models are fundamentally different. While game-theoretic agents are self-interested, and their non-cooperative actions lead to a desirable global target, Asymmetric DCOP agents are cooperative and seek to maximize the global reward even at the expense of smaller local rewards.

3.6. Notable Variant: Multi-Objective DCOPs

Multi-objective optimization (MOO) [56, 57] aims at solving problems involving more than one objective function to be optimized simultaneously. In a MOO problem, optimal decisions need to accommodate conflicting objectives. Examples of MOO problems include optimization of electrical power generation in a power grid while minimizing emission of pollutants and minimization of the costs of buying a vehicle while maximizing comfort. *Multi-objective DCOPs* extend MOO problems and DCOPs.

3.6.1. Definition

A *Multi-objective DCOP* (MO-DCOP) is defined by a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \vec{\mathbf{F}}, \alpha \rangle$, where $\mathbf{A}, \mathbf{X}, \mathbf{D}$, and α are defined as in Definition 1, and $\vec{\mathbf{F}} = [F_1, \dots, F_h]^T$ is a vector of multi-objective functions, where each F_i is a set of optimization functions f_j defined as in Definition 1. For a solution σ of a MO-DCOP, let the *reward* for σ according to the i^{th} multi-objective optimization function set F_i ($1 \leq i \leq h$) be

$$F_i(\sigma) = \sum_{f_j \in F_i} f_j(\sigma_{\mathbf{x}_i}) \quad (3)$$

³We actually describe a version of the algorithm with a guarantee that it will converge to a local optima. In the original version of the algorithm, which does not have such guarantee, each agent sends the utility only if its gain with the neighbor's new values is larger than the neighbor's last known gain.

The goal of a MO-DCOP is to find an assignment σ^* , such that:

$$\sigma^* = \operatorname{argmax}_{\sigma \in \Sigma} \vec{F}(\sigma) = \operatorname{argmax}_{\sigma \in \Sigma} [F_1(\sigma), \dots, F_h(\sigma)]^T \quad (4)$$

where $\vec{F}(\sigma)$ is a *reward vector* for the MO-DCOP. A solution to a MO-DCOP involves the optimization of a set of partially-ordered assignments. Note that we consider, in the above definition, point-wise comparison of vectors—i.e., $\vec{F}(\sigma) \geq \vec{F}(\sigma')$ if $F_i(\sigma) \geq F_i(\sigma')$ for all $1 \leq i \leq h$. Typically, there is no single global solution where all the objectives are optimized at the same time. Thus, solutions of a MO-DCOP are characterized by the concept of *Pareto optimality*, which can be defined through the concept of *dominance*:

Definition 2 (Dominance). *A complete solution $\sigma \in \Sigma$ is dominated by a complete solution $\sigma^* \in \Sigma$ iff $\vec{F}(\sigma^*) \geq \vec{F}(\sigma)$ and $F_i(\sigma^*) > F_i(\sigma)$ for at least one F_i .*

Definition 3 (Pareto Optimality). *A complete solution $\sigma^* \in \Sigma$ is Pareto optimal iff it is not dominated by any other complete solution.*

Therefore, a solution is Pareto optimal iff there is no other solution that improves at least one objective function without deteriorating the reward of another function. Another important concept is the *Pareto front*:

Definition 4 (Pareto Front). *The Pareto front is the set of all reward vectors of all Pareto optimal solutions.*

Solving an MO-DCOP is equivalent to finding the Pareto front. Even for tree-structured MO-DCOPs, the size of the Pareto front may be exponential in the number of variables.⁴ Thus, multi-objective algorithms often provide solutions that may not be Pareto optimal but may satisfy other criteria that are significant for practical applications. A widely-adopted criteria is that of the *weakly Pareto optimality*:

Definition 5 (Weakly Pareto Optimal). *A complete solution $\sigma^* \in \Sigma$ is weakly Pareto optimal iff there is no other complete solution $\sigma \in \Sigma$ such that $\vec{F}(\sigma) > \vec{F}(\sigma^*)$.*

In other words, a solution is weakly Pareto optimal if there is no other solution that improves *all* of the objective functions simultaneously. An alternative approach to Pareto optimality is one that uses the concept of *utopia points*:

Definition 6 (Utopia Point). *A reward vector $\vec{F}^\circ = [F_1^\circ, \dots, F_h^\circ]^T$ is an utopia point iff $F_i^\circ = \max_{\sigma \in \Sigma} F_i(\sigma)$ for all $1 \leq i \leq h$.*

Thus, a utopia point is the vector of rewards obtained by independently optimizing h DCOPs, each associated to one objective of the multi-objective function vector. In general \vec{F}° is unattainable. Therefore, different approaches focus on finding a *compromise solution* [58], which is a Pareto optimal solution that is *close* to the utopia point. The concept of *close* is dependent on the approach adopted.

3.6.2. Algorithms

We categorize the proposed MO-DCOP algorithms into two classes: *complete* and *incomplete* algorithms, according to their ability to find the complete set of Pareto optimal solutions or only a subset of it.

COMPLETE ALGORITHMS

MO-SBB [59]. *Multi-Objective Synchronous Branch and Bound (MO-SBB)* is a complete, synchronous, search-based algorithm that extends SynchBB. It uses an analogous search strategy as the mono-objective SynchBB: after establishing a complete ordering, MO-SBB agents extend a Current Partial Assignment with their own value assignments and the current associated reward vectors. Once a non-dominated solution is found, it is broadcast to all agents, which add the solution to a list of global bounds. Thus, agents maintains an approximation of the Pareto front, which is used to bound the exploration, and extend the CPA only if the new partial assignment is not dominated by solutions in the list of global bounds. When the algorithm terminates, it returns the set of Pareto optimal solutions obtained

⁴In the worst case, every possible solution can be a Pareto optimal solution.

by filtering the list of global bounds by dominance. Its worst case complexity reflects that of SynchBB in terms of network load, size of messages, and number of operations performed by each agent, while the agent's memory requirement is $O(np)$, where p is the size of the Pareto set.

Pseudo-tree Based Algorithm [60]. The proposed algorithm is a complete, asynchronous, search-based algorithm that extends ADOPT. It introduces the notion of *boundaries* on the vectors of multi-objective values, which extends the concept of lower and upper bounds to vectors of values. The proposed approach starts with the assumption that $|F_1| = \dots = |F_h| = k$; furthermore, functions within each F_i are sorted according to a predefined ordering, and for each $1 \leq j \leq k$, we have that the scope of f_j^i (i.e., the j^{th} function in F_i) is the same for each i (i.e., all functions in the same position in different F_i have the same scope). Thus, without loss of generality, we will refer to the scope of f_j^i as \mathbf{x}^j .

In such context, for $1 \leq j \leq k$, given a complete assignment σ we define the *vector of reward values* $\vec{\sigma}_j = [\sigma_{f_j^1}^1, \dots, \sigma_{f_j^h}^h] = [f_j^1(\sigma_{\mathbf{x}^j}), \dots, f_j^h(\sigma_{\mathbf{x}^j})]$. The notion of non-dominance is applied to these vectors, where a vector $\vec{\sigma}_j$ is non-dominated *iff* there is no other vector $\vec{\sigma}'_j$ such that $ub(\sigma_j^s) \leq lb(\sigma'^s_j)$ for all $1 \leq s \leq h$ and $ub(\sigma_j^s) < lb(\sigma'^s_j)$ for at least one s . The algorithm uses the notion of non-dominance for bounded vectors to retain exclusively non-dominated vectors. As in ADOPT, the network load is in $O(d^m)$. The worst case computational and memory complexity at each agent is in $O(p)$, as the number of combinations of utility vectors grows exponentially with the number of tuples of utility values, in the worst case.

Such method has also been proposed to solve Asymmetric MO-DCOPs [61], which is an extension of both Asymmetric DCOPs and MO-DCOPs.

INCOMPLETE ALGORITHMS

B-MOMS [62]. *Bounded multi-objective max-sum (B-MOMS)* is an incomplete, asynchronous, inference-based algorithm, and was the first MO-DCOP algorithm introduced. It extends Bounded Max-Sum to compute bound approximations for multi-objective DCOPs. It consists of three phases. The *Bounding Phase* generates an acyclic subgraph of the multi-objective factor graph, using a generalization of the maximum spanning tree problem to vector weights. During the *Max-sum Phase*, the agents coordinate to find the Pareto optimal set of solutions to the acyclic factor graph generated in the bounding phase. This is achieved by extending the addition and marginal maximization operators adopted in Max-Sum to the case of multiple objectives. Finally, the *Value Propagation Phase* allows agents to select a consistent variable assignment, as there may multiple Pareto optimal solutions. The bounds provided by the algorithm are computed using the notion of utopia points. B-MOMS requires $O(\bar{d}_i^{|\mathbf{x}^i|})$ evaluations of function f_i , where \bar{d}_i is the largest domain among variables in \mathbf{x}^i . Furthermore, since in the worst case, every variable assignment of x is Pareto optimal, the message size complexity is $O(h d^{n+1})$, where h is the size of each non-dominated vector.

DP-AOF [63]. *Dynamic Programming based on Aggregate Objective Functions (DP-AOF)* is an incomplete, synchronous, inference-based algorithm. It adapts the AOF technique [56], designed to solve centralized multi-objective optimization problems, to solve MO-DCOPs. Centralized AOF adopts a scalarization to convert a MOO problem into a single objective optimization. This is done by assigning weights $(\alpha_1, \dots, \alpha_h)$ to each of the functions in the objective vector $[F_1, \dots, F_h]^T$ such that $\sum_{i=1}^h \alpha_i = 1$ and $\alpha_i > 0$ for all $1 \leq i \leq h$. The resulting mono-objective function $\sum_{i=1}^h \alpha_i F_i$ can be solved using any mono-objective optimization technique with guarantee to find a Pareto optimal solution [56].

DP-AOF proceeds in two phases. First, it computes the utopia point $\vec{\mathbf{F}}^\circ$ by solving as many mono-objective DCOPs as the number of objective functions in the MO-DCOP. It then constructs a new problem building upon the solutions obtained from the first phase. Such a problem is used to assign weights to each objective function of the MO-DCOP to construct the new mono-objective function in the same way as centralized AOF, which then can be solved optimally. The complexity of DP-AOF, in terms of number of operations, is given by $O(h d^{w^*})$, as the algorithms solves h objectives using DPOP.

MO-DPOP_{L_p} [64]. *Multi-Objective L_p-norm based Distributed Pseudo-tree Optimization Procedure (MO-DPOP_{L_p})* is an incomplete, synchronous, inference-based algorithm. It adapts DPOP using a scalarization measure based on the L_p -norm to find a subset of the Pareto front of a MO-DCOP. Similar to DP-AOF, the algorithm proceeds in two phases. During the first phase, it uses DPOP to find the utopia point $\vec{\mathbf{F}}^\circ$. In the second phase, the agents coordinate to find a complete solution that minimizes the distance from $\vec{\mathbf{F}}^\circ$ according to the L_p -norm. The algorithm is guaranteed

to find a Pareto optimal solution only when the L_1 -norm (Manhattan norm) is adopted. In this case, MO-DPOP_{L_1} finds a Pareto optimal solution that maximizes the average of the reward values of all objectives.

DIPLS [65]. *Distributed Iterated Pareto Local Search (DIPLS)* is an incomplete, synchronous, search-based algorithm. It extends the *Pareto Local Search (PLS)* algorithm [66], which is a hill climbing algorithm designed to solve centralized multi-objective optimization problems, to solve MO-DCOPs. The idea behind DIPLS is to evolve an initial solution toward the Pareto front. To do so, it starts from an initial set of random assignments, and applies PLS iteratively to generate new non-dominated solutions. DIPLS requires a total ordering of agents and elects one agent as the *controller*. At each iteration, the controller filters the set of solutions by dominance and broadcasts them to the agents in the MO-DCOP. Upon receiving a solution, an agent generates a list of *neighboring solutions* by modifying the assignments of the variables that it controls, and sends them back to the controller. When the controller receives the messages from all agents, it proceeds to filter (by dominance) the set of solutions received, and if a new non-dominated solution is found, it repeats the process. DIPLS is shown to outperform B-MOMS on random graph problems [65].

3.6.3. Relevant Uses

The MO-DCOP model is tailored to represent those classes of distributed problems that cannot be modeled with a single optimization function. The observations reported in Section 3.4 still hold for MO-DCOPs and, in addition, agents cooperate to optimize multiple objectives. A number of MAS applications fall in the category of multi-objective optimization. One example is that of *disaster management and coordination problems*, where agents coordinate to effectively respond to an emergency scenario [67]. Due to the low-powered mobile devices, and limited bandwidth consumption, the MO-DCOP approach fits naturally with the coordinated decision making required in disaster management, where agents, for instance, may try to evacuate the largest group of people and at the same time preserve their battery life as long as possible [68]. Another application is *cyber security*, where agents coordinate to inspect data and detect, in advance, a possible cyber attack. In cyber security problems, multiple optimization criteria arise naturally when agents attempt to minimize risk (security), surveillance (privacy), and costs [69].

4. Probabilistic DCOPs

So far, we have discussed DCOP models that can model MAS problems in environments that are static and deterministic. However, many real-world applications are characterized by environments with stochastic behavior. In other words, there are exogenous events that can influence the outcome of agent actions. For example, weather conditions or the state of a malfunctioning device can affect the reward of agent actions. To cope with such scenarios, researchers have introduced *Probabilistic DCOP (P-DCOP)* models, where the uncertainty in the state of the environment is modeled through stochasticity in the reward functions. With respect to our categorization, in the P-DCOP model the agents are completely cooperative and they have deterministic behavior. Additionally, the environment is static and stochastic. While a large body of research has focused on problems where agents have *total knowledge*, we will discuss a subclass of P-DCOPs where the agents' knowledge of the environment is limited, and agents must balance *exploration* of the unknown environment and the *exploitation* of the known rewards.

4.1. Definition

A common strategy to model uncertainty is to augment the outcome of the reward functions with a stochastic character [70, 71, 72]. Another method is to introduce additional random variables to the reward functions, which simulate exogenous uncontrollable traits [73, 74, 75]. To cope with such a variety, we introduce the *Probabilistic DCOP (P-DCOP)* model, which generalizes the proposed models of uncertainty. A P-DCOP is defined by a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F}, \alpha, \mathcal{I}, \Omega, \mathbf{P}, \mathcal{U}, \mathcal{E} \rangle$, where \mathbf{A} and \mathbf{D} are defined as in Definition 1. In addition,

- \mathbf{X} is a mixed set of decision variables and random variables.
- $\mathcal{I} = \{r_1, \dots, r_q\} \subseteq \mathbf{X}$ is a set of *random variables* modeling uncontrollable stochastic events, such as weather or a malfunctioning device.

- \mathbf{F} is the set of reward functions, each defined over a mixed set of decision variables and random variables, and such that each value combination of the decision variables on the reward function, results in a probability distribution. As a result, the local value assignment $\sigma_{\mathbf{x}^i \setminus \mathcal{I}}$, given an outcome for the random variables involved in f_i , is itself a random variable.
- $\alpha : \mathbf{X} \setminus \mathcal{I} \rightarrow \mathbf{A}$ is a mapping from decision variables to agents. Notice that random variables are not controlled by any agent, as their outcomes do not depend on the agents' actions.
- $\Omega = \{\Omega_1, \dots, \Omega_q\}$ is the (possibly discrete) set of events for the random variables (e.g., different weather conditions or stress levels a device is subjected to) such that each random variable $r_i \in \mathcal{I}$ takes values in Ω_i . In other words, Ω_i is the domain of random variable r_i .
- $\mathbf{P} = \{p_1, \dots, p_q\}$ is a set of probability distributions for the random variables, such that $p_i : \Omega_i \rightarrow [0, 1] \subseteq \mathbb{R}$, assigns a probability value to an event for r_i , and $\int_{\omega \in \Omega_i} p_i(\omega) d\omega = 1$, for each random variable $r_i \in \mathcal{I}$.
- \mathcal{U} is a *utility function* from random variables to random variables, that ranks different outcomes based on the decision maker preferences. This function is needed when the reward functions have uncertain outcomes, and thus these distribution are not readily comparable.
- \mathcal{E} is an *evaluator function* from random variables to real values, that, given an assignment of values to the decision variables, summarizes the distribution of the aggregated reward functions.

The goal in a P-DCOP is to find a complete solution σ^* , that is, an assignment of values to all the decision variables, such that:

$$\sigma^* = \operatorname{argmax}_{\sigma \in \Sigma} \mathcal{E} \left[\mathcal{U} \left(\sum_{f_i \in \mathbf{F}} f_i(\sigma_{\mathbf{x}^i \setminus \mathcal{I}}) \right) \right] \quad (5)$$

In other words, agents attempt to maximize the utility of the cumulative reward functions of the P-DCOP, with respect to the evaluator function \mathcal{E} .

The probability distribution over the domain of random variables $r_i \in \mathcal{I}$ is called a *belief*. An assignments of all random variables in \mathcal{I} describes a (possible) *scenario* governed by the environment. As the random variables are not under the control of the agents, they *act* independently of the decision variables. Specifically, their beliefs are drawn from probability distributions. Furthermore, they are assumed to be independent of each other and, thus, they model independent sources of exogenous uncertainty.

The utility function \mathcal{U} enables us to compare the uncertain reward outcomes of the reward functions. In general, the utility function is non-decreasing, that is, the higher the reward, the higher the utility. However, the utility function should be defined for the specific application of interest. For example, in farming, the utility increases with the amount of produce harvested. However, farmers may prefer a smaller but highly certain amount of produce harvested over a larger but highly uncertain and, thus, risky outcome. The evaluation function \mathcal{E} is used to summarize in one criterion the rewards of a given assignment that depends on the random variables. A possible evaluation function is the *expectation* function: $\mathcal{E}[\cdot] = \mathbb{E}[\cdot]$.

Let us now introduce some concepts that are commonly adopted in the study of P-DCOPs.

Definition 7 (Convolution). *The convolution of two functions f and g , is a kind of integral transform, defined as the integral of the product of the two functions after one is reversed and shifted:*

$$(f * g)(x) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(x - \tau) d\tau \quad (6)$$

In other words, the convolution can be viewed as a weighted average of the function $f(\tau)$ at x , where the weighting is given by $g(-\tau)$ simply shifted by amount x . It produces a new function that defines the overlapping area between f and g as a function of the quantity that one of the original functions is translated. The convolution of f and g exists if f and g are both *Lebesgue integrable* functions, that is if their integrals are finite: $\int |f| d\tau < +\infty$ and $\int |g| d\tau < +\infty$.

In a P-DCOP, the value returned by a function f_i , for an assignment on its scope \mathbf{x}^i , is a random variable V_i . That is $V_i \sim f_i(\mathbf{x}^i)$. Thus the global value $\sum_{f_i \in \mathbf{F}} V_i$ is also a random variable, whose *probability density function (PDF)* is the convolution of the PDFs of the individual V_i . Thus, the concept of convolution of two PDFs in a P-DCOP is related to the summing of utilities of two reward functions in classical DCOPs.

A common concept in optimization with uncertainty is that of *ranking* a set of random variables $\{r_1, r_2, \dots\}$ with *cumulative PDFs (CDFs)* $\{F_1(x), F_2(x), \dots\}$. Such distributions are also commonly called *lotteries*, a concept related to that of *stochastic dominance*, which is a form of stochastic ordering based on preference regarding outcomes. It refers to situations where a probability distribution over possible outcomes can be ranked as superior to another.

The first-order stochastic dominance refers to the situation when one lottery is unambiguously better than another:

Definition 8 (First-Order Stochastic Dominance). *Given two random variables r_i and r_j with CDFs $F_i(x)$ and $F_j(x)$, respectively, F_i first-order stochastically dominates F_j iff:*

$$F_i(x) \leq F_j(x), \quad (7)$$

for all x with a strict inequality over some interval.

If F_i first-order stochastically dominates F_j , then F_i necessarily has a strictly larger expected value: $\mathbb{E}[F_i(x)] > \mathbb{E}[F_j(x)]$. In other words, if F_i dominates F_j , then the decision maker prefers F_i over F_j regardless of his utility function \mathcal{U} is, as long as it is weakly increasing.

It is not always the case that one CDF will first-order stochastically dominate another. In such a case, one can use the second-order stochastic dominance to compare them. The latter refers to the situation when one lottery is unambiguously less *risky* than another:

Definition 9 (Second-Order Stochastic Dominance). *Given two random variables r_i and r_j with CDFs $F_i(x)$ and $F_j(x)$, respectively, F_i second-order stochastically dominates F_j iff:*

$$\int_{-\infty}^c F_i(x) dx \leq \int_{-\infty}^c F_j(x) dx, \quad (8)$$

for all c with a strict inequality for some values of c .

If F_i second-order stochastically dominates F_j , then $\mathbb{E}[F_i(x)] \geq \mathbb{E}[F_j(x)]$. If Equation 8 holds for all $c \geq c'$, for some sufficiently large c' , then $\mathbb{E}[F_i(x)] = \mathbb{E}[F_j(x)]$. In this case, as both lotteries are equal in expectation, the decision maker prefers the lottery F_i , which has less variance and is, thus, less risky.

Another common concept in P-DCOPs is that of *regret*. In decision theory, regret expresses the negative *emotion* arising from learning that a different solution than the one adopted, would have had a more favorable outcome. In P-DCOPs the regret of a given solution is typically defined as the difference between its associated reward and that of the theoretical optimal solution. The notion of regret is especially useful in allowing agents to make robust decisions in settings where they have limited information about the reward functions.

An important type of regret is the *minimax regret*. Minimax regret is a decision rule used to minimize the possible loss for a worst case (i.e., maximum) regret. As opposed to the (expected) regret, minimax regret is independent of the probabilities of the various outcomes. Thus, minimax regret could be used when the probabilities of the outcomes are unknown or difficult to estimate.

4.2. Algorithms

We categorize P-DCOP algorithms into *complete* and *incomplete* algorithms, according to their ability to guarantee to find the optimal solutions or not, for a given evaluator and utility functions.

COMPLETE ALGORITHMS

$\mathbb{E}[\text{DPOP}]$ [74]. $\mathbb{E}[\text{DPOP}]$ is an incomplete, synchronous, sampling-based and inference-based algorithm. It uses a *collaborative sampling* strategy, where all agents concerned with a given random variable agree on a common sample set that will be used to estimate the PDF of that random variable. Agents performing collaborative sampling independently propose sample sets for the random variables they control, and elect one agent among themselves as responsible for combining the proposed sample sets into one. The algorithm is defined over P-DCOPs with $\mathcal{I} \neq \emptyset$, and deterministic reward functions outcome, that is, for each combination of values for the variables in \mathbf{x}^i , $f_i(\mathbf{x}^i)$ is a degenerate distribution (i.e., a distribution that results in a single value) and the utility function \mathcal{U} is the identity function. \mathcal{E} is an arbitrary evaluator function.

\mathbb{E} [DPOP] builds on top of DPOP, and proceeds in four phases: In Phase 1, the agents order themselves into a pseudo-tree ignoring the random variables. In Phase 2, the agents bind random variables to some decision variable. In Phases 3 and 4, the agents run the UTIL and VALUE propagation phases like in DPOP except that random variables are sampled. Based on different strategies adopted in binding the random variables in Phase 2, the algorithm has two variants [73]. In *Local- \mathbb{E} [DPOP]*, a random variable $r_i \in \mathcal{I}$ is assigned to each decision variable responsible for enforcing a constraint involving r_i . In this approach, the agents do not collaborate by exchanging information about how their utilities depend on the random variables. In contrast, *Global- \mathbb{E} [DPOP]* assigns r_i to the *lowest common ancestor* agent,⁵ which is responsible for combining the proposed samples. While this additional information can produce higher-quality solutions, both algorithms are generally incomplete. One exception is when the evaluation function \mathcal{E} adopted is *linear*, as in the case of the *expectation* function, in which case the algorithms are complete.

In terms of complexity, \mathbb{E} [DPOP] has the same order complexity of DPOP, for both number and size of messages. *Local- \mathbb{E} [DPOP]* produces messages of size $O(d^{w^*})$, where d is the size of the largest decision variable domain and w^* is the induced width of the DFS pseudo-tree. In contrast, *Global- \mathbb{E} [DPOP]* generates, in the worst case, UTIL messages of size $O(d^{w^*} s^q)$, where s is the largest sample set size; this occurs when the root as well as all leaves are constrained with all q random variables.

SD-DPOP [72]. *Stochastic Dominance DPOP (SD-DPOP)* also operates on a P-DCOP model, where $\mathcal{I} = \emptyset$ and \mathcal{E} is the identity function, while \mathcal{U} is a stochastic dominance criteria, which is applicable for the class of concave risk functions. It is a complete synchronous inference-based algorithm that extends DPOP to solve P-DCOPs. Similar to DPOP, it has three phases. In Phase 1, like DPOP, it constructs the pseudo-tree. In Phase 2, instead of summing up utilities, the agents convolve the reward functions. And instead of propagating utilities up the pseudo-tree, the agents propagate convolved reward functions. In Phase 3, like DPOP, the agents choose values for their variables. However, instead of choosing values that maximizes the utility, the agents choose values with the second-order stochastically dominating reward function.

Like DPOP, SD-DPOP requires a linear number of messages. In addition, in SD-DPOP, VALUE messages contain each Pareto optimal value of the sending agent, and UTIL messages contain a representation of the reward function for each Pareto optimal solution and each combination of values of the parent and pseudo-parents of the sending agents. Thus, for continuous PDFs that could be represented by mean and variance, the message complexity is $O(p d^{w^*})$, where p is the size of the Pareto set. If the reward functions are represented by discretized bins, then the message complexity is $O(b p d^{w^*})$, with b being the maximum number of bins used to represent a reward function.

INCOMPLETE ALGORITHMS

DNEA [70]. The P-DCOP model proposed by Atlas and Decker is characterized by uncertainty exclusively at the level of function outcome and not due to random variables [70]. Thus, $\mathcal{I} = \emptyset$. In addition, the utility function \mathcal{U} is the identity function, while \mathcal{E} is a given evaluator function. In such settings, by employing the evaluation function, they show that one can reduce the uncertainty associated to each reward function to the deterministic case. Thus, one can solve the proposed P-DCOP problems using classical DCOP approaches.

In particular, they propose the *Distributed Neighbor Exchange Algorithm (DNEA)*, which is an incomplete synchronous search-based algorithm that is similar to DSA. Each agent starts by assigning a random value to each of its variables and sends this information to all its neighbors. Upon receiving the values of its neighbors, it computes a reward vector, which contains the reward for each possible combination of values for all its variables under the assumption that its neighbors' values are those in the messages received. It then sends this reward vector to all its neighbors. Upon receiving the reward vector of its neighbors, it computes the best value for each of its variable, assigns those values to its variable probabilistically, and sends the assigned values to all its neighbors. This process repeats until a termination condition is satisfied. At each iteration, each agent a_i has complexity $O(l d^2)$, where $l = \max_{a_i \in \mathbf{A}} |N_{a_i}|$, and $d = \max_{D_i \in \mathbf{D}} |D_i|$, and the total number of messages exchanged is $O(2|E_C|)$.

U-GDL [71]. The P-DCOP model proposed by Stranders *et al.* also assumes that the reward functions are not dependent on random variables [71]. Thus, $\mathcal{I} = \emptyset$. Additionally, they assume that $\mathcal{E} = \mathbb{E}$ is the expectation function, and \mathcal{U} is a given risk function.

⁵The agent that is separated by the smallest number of tree edges from all variables constrained with the given random variable.

They propose the *Uncertain Generalized Distributive Law (U-GDL)* algorithm, which is an incomplete asynchronous inference-based algorithm similar to Max-Sum. It extends the *Generalized Distributive Law (GDL)* algorithm [76] by redefining the $(\max, +)$ algebra to the setting where rewards are random variables rather than scalars. The $+$ operator is extended to perform convolution of two random variables. To cope with the potential issue on that not all PDFs are closed under convolution, the authors suggest to resort to sampling methods to approximate such operations. The max operator is defined to distribute over convolution and to select the maximal elements from a set of random variables based on their expected utility. In order to filter partial potential solutions that can never achieve global optimality, the authors introduce a first-order stochastic dominance condition, which is employed in the context of the max operator. They also discuss *necessary* and *sufficient* conditions for dominance, where the former discards all dominated solutions, but it might also discard some non-dominated solution as well—this is equivalent to using a standard DCOP to solve the P-DCOP model adopted in their work. The latter preserves optimal solutions, but retains in general sub-optimal ones as well. Prior to convergence, U-GDL requires a number of messages equals to twice the diameter of the resulting acyclic graph, while the message size is exponential in the maximum size of the merged variables’ domain.

4.3. Relevant Uses

Extending the classical DCOP model to include the capability of handling uncertain events allows P-DCOPs to extend the set of applications that can be modeled. In particular, P-DCOPs are suitable to capture those applications characterized by a static environment evolution with exogenous uncertain events (e.g., when the actions of agents on the environment can have different outcomes, based on external, uncertain factors) and, yet, agents have total knowledge of their own actions and of the observable environment, and act in a fully cooperative context.

The domain of *multi-agent task planning and scheduling* encompasses diverse problems that require complex models and robust solutions under uncertainty [77]. In a scheduling problem, there is a set of tasks, each associated with a duration, which may vary probabilistically, and constrained by a release time and a deadline. In order to complete a task, a sequence of complex actions must be enforced. Agents can execute actions (also called methods) that use some resources. The quantity of the resources employed and the quality of the outcome of each action are uncertain, and they can be described using probability distributions. Agents need to coordinate in the execution of sequences of actions, in order to maximize the number of tasks completed before their deadlines [78]. Uncertainty of various task characteristics, such as completion time, solution quality, and total cost, is one of the major complexities underlying this type of scheduling problems.

Another typical application that benefits from the rich P-DCOP model is demand response in smart grid applications [79]. For example, the use of electric vehicles (EV) in a smart grid requires an effective schedule for charging vehicles to fulfill the next-day desired trips. The number of next-day trips is often estimated using a probabilistic model [80]. At the same time, users attempt to minimize the charging cost, which is directly proportional to the power demand in a real-time pricing mechanism, and can also be modeled under uncertainty.

4.4. Notable Variant: P-DCOPs with Partial Agent Knowledge

We now describe a class of probabilistic DCOPs where agents have *partial* knowledge about the environment. That is, the reward functions are only partially known and, therefore, agents may discover the unknown rewards via *exploration* [81]. The new model aims at capturing those domains where agents have an “explorative nature,” i.e., one of the agents’ goals is to acquire knowledge about the environment in which they act. Agents are concerned with a total, online reward achievable in a limited time frame. In this context, agents must balance the coordinated *exploration* of the unknown environment and the *exploitation* of the known portion of the rewards, in order to maximize the global utility. This model was originally called *Distributed Coordination of Exploration and Exploitation (DCEE)* [82].

4.4.1. Definition

The P-DCOP model for agents with partial knowledge is described by extending the P-DCOP model introduced in Section 4.1, as follows: $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F}, \alpha, \mathcal{I}, \Omega, \mathbf{P}, \mathcal{U}, \mathcal{E}, T \rangle$, where $T > 0$ is a finite time horizon characterizing the time within which the agents can exploit the unknown reward functions and explore the search space. The goal in such a

P-DCOP problem is to find a set of assignment $\vec{\sigma}^* = [\sigma_1^*, \dots, \sigma_T^*]$ that maximizes the utility of the cumulative reward within the finite time horizon T :

$$\vec{\sigma}^* = \underset{\sigma^1, \dots, \sigma^T}{\operatorname{argmax}} \mathcal{E} \left[\sum_{t=0}^T \mathcal{U} \left(\sum_{f_i \in \mathbf{F}} f_i(\sigma_{\mathbf{x}^i}^t) \right) \right] \quad (9)$$

where $\sigma^t \in \Sigma$ denotes a complete solution at time t . In other words, agents have at most T steps to modify the value of their decision variables, and solve T P-DCOP problems by acquiring more and more knowledge on the environment as the time unrolls.

4.4.2. Algorithms

In a stochastic and a priori unknown environment, the reward functions need to be learned online through interactions between the agents and their environment. Thus, the algorithms presented in this section are targeted to coordinate agents to solve a sequence of optimization problems in order to simultaneously reduce uncertainty about the local reward functions (exploration) and optimize the global objective (exploitation). In addition, the following algorithms are incomplete.

BE-Rebid [82]. The *Balanced Exploration Rebid (BE-Rebid)* is a synchronous, search-based algorithm that solves P-DCOPs with $\mathcal{I} = \emptyset$, \mathcal{E} and \mathcal{U} are the identity functions. It extends MGM as it calculates and communicates its expected gain. The algorithm is introduced in the context of a *wireless sensor network* problem, where agents can perform small movements in order to enhance their communication capabilities, which are characterized by the distance between pairs of agents. Each agent can perform three actions: *stay* in the current position, *explore* another position, or *backtrack* to a previous explored position and halt movement. In each time step, BE-Rebid computes the expected reward of executing the *explore* or *backtrack* actions, assuming complete knowledge of the underlying distribution of the reward functions. Exploring is evaluated by using order statistics and is based on the reward of the best value found during exploration. Backtracking to a known position results in a reward associated to the backtracked state for the remainder of the time steps (i.e., it *stays* in that state).

Following the region-optimal approaches presented in the context of classical DCOPs, the authors propose a version of the algorithm, called *BE-Rebid-2* [81], that allows pairs of agents to explore in a coordinated fashion. Interestingly, in such settings, the authors find that increasing coordination (measured by the number of agents that can execute a joint action) can decrease solution quality. This phenomenon is referred to as *team uncertainty penalty*.

HEIST [83]. HEIST is a synchronous, inference-based algorithm that solves P-DCOPs with $\mathcal{I} = \emptyset$, \mathcal{E} is the expectation function, and \mathcal{U} is the identity function. Thus, it aims at maximizing the expected utility of the cumulative reward function \mathbf{F}_g , within the finite time horizon T . It does so by modifying a *multi-armed bandit (MAB)* approach [84] to a distributed scenario. A MAB is a slot machine with multiple arms, each of which yields a reward, drawn from an unknown but fixed probability distribution. It trades exploration and exploitation by pulling the arms in order to maximize the cumulative reward over a finite horizon. To cope with the uncertain and stochastic nature of the reward functions, HEIST models each utility function as a MAB, such that the joined assignment of the variables in the scope of the given utility function becomes an arm of that bandit. It seeks to maximize the expected cumulative optimization reward received over a finite time horizon by repeatedly pulling the MAB arms to select the joint action with the highest estimated *upper confidence bound (UCB)* [85] on the sum of the local gains received in a single time step. To do so, it employs a belief propagation algorithm, known as *generalized distributive law (GDL)* [76], in order to maximize the UCB in a decentralized fashion. The authors show that HEIST enables agents to balance between exploration and exploitation, and derive optimal asymptotic bounds on the regret of the global cumulative reward attained.

ICG-MaxSum [86]. The *Iterative Constraint Generation Max-Sum (ICG-MaxSum)* algorithm is a synchronous, inference-based algorithm that solves P-DCOPs with $\mathcal{I} \neq \emptyset$, \mathcal{E} is the minimax regret function, and \mathcal{U} is the identity function. Furthermore, the horizon is $T = 1$. Thus, unlike the previous algorithms, ICG-MaxSum does not attempt to learn the outcome of the reward functions. Its objective is to find *robust* solutions to the uncertainty of the joint belief. The algorithm extends the *Iterative Constraint Generation (ICG)* method [87, 88] to the decentralized case, by decomposing the overall problem into a *master* problem and a *subproblem* that are iteratively solved until convergence. At each iteration, the resolutions of these problems are attained by using Max-Sum. The *master* problem solves a relaxation of the minimax regret goal, where only a subset of all possible joint beliefs is considered, attempting

to minimize the loss for the worst case derived from the considered joint belief. Once it generates a solution, the *subproblem* finds the maximally violated constraint associated to such a solution. This is referred to as the *witness* point, indicating that the current solution is not the best one in terms of the minimax regret. This point is added to the set of joint beliefs considered by the master problem, and the process is repeated until no new witness points can be found. The computational complexity of the algorithm is dominated by the master problem, whose computation is exponential in the number of variables in the scope of the associated reward function (similarly to the standard Max-Sum) and linear in the number of witness points. The messages in the master problem have size proportional to the size of the set of the joint beliefs considered during the master problem, while the messages in the subproblems are normal Max-Sum messages.

4.4.3. Relevant Uses

The P-DCOP model for agents with partial knowledge is suitable to model those applications where agents have no prior knowledge of how the environment reacts to some of the actions. In such model, the agents are aware of their own actions, which are performed deterministically. However, there is uncertainty in the reward associated to such actions, which is influenced by uncertain events that can be discovered over time. Thus, a common approach in such cases is to resort to *sampling* strategies, in order to obtain simple approximations of the probability distributions—in the form of sample realizations of the probabilistic rewards.

A common application domain for such class of P-DCOPs is the *distributed coordination in disaster scenarios* [82]. In such a scenario, a set of mobile agents needs to be deployed in the affected area in order to create a communication network mesh [89].⁶ The goal is to maximize the signal strengths between neighboring mobile agents in the network. Mobile agents can commit to small movements to enhance their signal strengths, which in turn depends on their locations: signal strength diminishes with increasing distance and it is affected by other agents' radio interference and obstacles between agents. Thus, the signal strength between a pair of neighboring agents is an example of the unknown reward that must be sampled [90].

5. Dynamic DCOPs

Within a real-world MAS application, agents often act in dynamic environments that evolve over time. For instance, in a disaster management search and rescue scenario, new information (e.g., the number of victims in particular locations, or priorities on the buildings to evacuate) typically becomes available in an incremental manner. Thus, the information flow modifies the environment over time. To cope with such requirement, researchers have introduced the *Dynamic DCOP (D-DCOP)* model, where reward functions can change during the problem solving process, agents may fail, and new agents may be added to the DCOP being solved. With respect to our categorization, in the D-DCOP model, the agents are completely cooperative and they have deterministic behavior and total knowledge. On the other hand, the environment is dynamic and deterministic.

5.1. Definition

The *Dynamic DCOP (D-DCOP)* model is defined as a sequence of classical DCOPs: $\mathcal{D}_1, \dots, \mathcal{D}_T$, where each $\mathcal{D}_t = \langle \mathbf{A}^t, \mathbf{X}^t, \mathbf{D}^t, \mathbf{F}^t, \alpha^t \rangle$ is a DCOP, representing the DCOP at time step t , for $1 \leq t \leq T$. The goal in a D-DCOP is to solve optimally the DCOP at each time step. We assume that the agents have total knowledge about their current environment (i.e., the current DCOP), but they are unaware of changes to the problem in future time steps.

In a dynamic system, agents are required to adapt as fast as possible to environmental changes. *Stability* [91, 92] is a core algorithmic concept, where an algorithm seeks to minimize the number of steps that it requires to converge to a solution each time the problem changes. In such a context, these converged solutions are also called *stable solutions*. *Self-stabilization* is a related concept derived from the the area of fault-tolerance:

Definition 10 (Self-stabilization). *A system is self-stabilizing if and only if the following two properties hold:*

- *Convergence: The system reaches a stable solution in a finite number of steps, starting from any given state.*

⁶This is used by first responders to coordinate their actions.

- Closure: *The system remains in a stable solution, provided that no changes in the environment happens.*

An extension of the concept of self-stabilization is that of *super-stabilization* [93], which focuses on stabilization after topological changes. In the context of D-DCOPs, differently from self-stabilizing algorithms, where convergence after a single change in the constraint graph can be as slow as the convergence from an arbitrary starting state, super-stabilizing algorithms take special care of the time required to adapt to a single change in the constraint graph.

5.2. Algorithms

In principle, one could use classical DCOP algorithms to solve each DCOP \mathcal{D}_t at each time step $1 \leq t \leq T$. However, the dynamic environment evolution encourages firm requirements on the algorithm design, in order for the agents to respond automatically and efficiently to environmental changes over time. In particular, D-DCOP algorithms often follow the *self-stabilizing* property. As in the previous sections, we categorize the algorithms as being either complete or incomplete, according to their ability to determine the optimal solution at each time step.

COMPLETE ALGORITHMS

SDPOP [94]. *Self-stabilizing DPOP (SDPOP)* is a synchronous, inference-based algorithm that extends DPOP to handle dynamic environments. It is composed of three self-stabilizing phases: **(1)** A self-stabilizing DFS tree generation, whose goal is to create and maintain a DFS pseudo-tree structure; **(2)** A self-stabilizing algorithm for populating the utility messages; and **(3)** A self-stabilizing algorithm for the value propagation phase. Such procedures work as in DPOP and they are invoked whenever any change in the DCOP problem sequence is revealed. In [94], the authors discuss two self-stabilizing extensions, namely *super-stabilization* and *fault-containment*, which can be used to provide guarantees about the way the system transitions from a valid state to the next, after an environment change. The complexity of SDPOP is similar to that of DPOP: at each time step, it uses $O(m)$ messages of maximal size $O(d^{w^*})$. Upon individual changes, SDPOP stabilizes after at most ℓ UTIL messages and k VALUE messages, where ℓ is the length of the longest branch in the pseudo-tree, and k the number of reward functions of the problem.

RSDPOP [95]. *RSDPOP*⁷ is a synchronous, inference-based that extends the SDPOP algorithm by introducing the possibility of specifying *commitment deadlines* and *stability constraints*. In the proposed model, some of the variables may be unassigned at a given point in time, while others must be assigned within a specific deadline. Commitment deadlines can be either hard or soft. *Hard commitments* model irreversible processes. When a hard committed variable is assigned, its value cannot be changed. *Soft commitments* model contracts with penalties. If a soft committed variable x_i^t has been assigned at time t , its value can be changed at time $t' > t$, at the price of a cost penalty. Such costs are modeled via *stability constraints*, which are defined as binary relations, $s_i : D_i \times D_i \rightarrow \mathbb{R}^+$, representing the cost of changing value of variable x_i from time t to time $t + 1$. Given the set of stability constraints $\mathbf{S} \subseteq \mathbf{F}$, at each time t , the goal is to find a solution

$$\sigma_t^* = \underset{\sigma}{\operatorname{argmax}} \left(\mathbf{F}_g(\sigma) - \sum_{s_j \in \mathbf{S}} s_j(\sigma_{t-1}^*(x_j), \sigma(x_j)) \right).$$

The latter term accounts for the penalties associated to the value assignment updates for the soft committed variables. RSDPOP has the same order complexity as SDPOP.

I-ADOPT and I-BnB-ADOPT [96]. *Incremental Any-space ADOPT (I-ADOPT)* and *Incremental Any-space BnB-ADOPT (I-BnB-ADOPT)* are asynchronous, search-based algorithms that extend ADOPT and BnB-ADOPT, respectively. In the incremental any-space versions of the algorithms, each agent maintains bounds for multiple contexts (in contrast agents in ADOPT and BnB-ADOPT maintain bounds for one context only). By doing so, when solving the next DCOP in the sequence, agents may reuse the bounds information computed in the previous DCOP. In particular, the algorithms identify *affected agents*, which are agents that cannot reuse the information computed in the previous iterations, and they recompute bounds exclusively for such agents.

The algorithm worst case complexity is analogous to that of ADOPT, in terms of network load, size of messages, and number of operations performed by each agent. In contrast, the agent space requirements increase with the respect

⁷The full name of the algorithm was not provided by the authors in [95].

to that of ADOPT, by a factor proportional to the amount of cache used by the agents to reuse bounds of the previous DCOP.

INCOMPLETE ALGORITHMS

SBDO [97]. *Support Based Distributed Optimization (SBDO)* is an asynchronous search-based algorithm that extends the *Support Based Distributed Search* algorithm [98] to the multi-agent case. It uses two types of messages: *is-good* and *no-good*. Is-good messages contain an ordered partial assignment; These messages are exchanged among neighboring agents upon a change in their value assignments. Each agent, upon receiving a message, decides what value to assign to its own variables, attempting to maximize their local utilities, and communicates such decisions to its neighboring agents via is-good messages. No-good messages are used in response to violations of hard constraints, or in response to obsolete assignments. A no-good message is augmented with a *justification*, that is, the set of hard constraints that are violated, and are saved locally within each agent. This information is used to discard partial assignments, when they are supersets of one of the known no-goods. The changes of the dynamic environment are communicated via messages, which are sent from the *environment* to the agents. In particular, changes in hard constraints require the update of all the justifications in all no-goods.

FMS [99]. *Fast Max-Sum (FMS)* is an asynchronous inference-based algorithm that extends Max-Sum to the dynamic DCOP model. As in Max-Sum, the algorithm operates on a factor graph. Solution stability is maintained by recomputing only those factors that changed between the previous DCOP \mathcal{D}_{t-1} and the current DCOP \mathcal{D}_t . In [99], the authors exploit domain-specific properties in a task allocation problem to reduce the number of states over which each factor has to compute its solution. In addition, FMS is able to efficiently manage addition or removal of tasks (e.g., factors), by incurring in message propagation exclusively on the factor graph regions affected by such topological changes. FMS has been extended in several ways. *Bounded Fast Max-Sum* provides bounds on the solution found, as well as it guarantees super-stabilization [100]. *Branch-and-Bound Fast Max-Sum (BnB-FMS)* extends FMS providing online domain pruning using a branch-and-bound technique [101].

Mobed [102]. *Multiagent Organization with Bounded Edit Distance (Mobed)* does not directly solves D-DCOPs, but instead is a self-stabilizing pseudo-tree generation algorithm. It operates in dynamic environments where constraints may change, new agents may appear, and existing agents may be dropped. As virtually all algorithms solving D-DCOPs are required to deal with network structural reconfigurations, we believe that describing this approach in this section is appropriate. Unlike existing distributed DFS algorithms, Mobed is able to bound the *edit distance*, a metric that measures the difference between two pseudo-trees, between pseudo-trees for subsequent DCOPs in a D-DCOP. Mobed is able to do so by exploiting the concept of agent hierarchies, which are defined as subtrees in the original multi-agent constraint graph, during the operation of addition and removal of an agent in the constraint graph. In particular the addition of an agent connected to all nodes in a pseudo-tree, consisting of a root with ℓ children, has a worst case edit distance of $\ell + 1$, for a DFS algorithm. In Mobed this operation is performed determining the insertion point (defined as a neighbor of the agent being added) that minimizes the tree depth when inserting it as the parent or the child of the insertion point. The number of edits for each agent addition is thus bounded by two.

Distributed Q-learning and R-learning [103]. The *Distributed Q-learning and R-learning* algorithms are synchronous reinforcement-learning-based algorithms that extend the centralized Q-learning [104] and centralized R-learning [105, 106] algorithms. The algorithms solve a variant of D-DCOPs, called *Markovian Dynamic DCOPs (MD-DCOPs)*, where the DCOP in the current time step \mathcal{D}_{t+1} depends on the variables assignment performed by the agents in the DCOP in the current time step \mathcal{D}_t . However, the transition function between these two DCOPs are not known to the agents and the agents must, thus, learn them. Each agent maintains Q-values and R-values for each σ_{t-1}, d_i^t pair, where σ_{t-1} is the solution for the DCOP \mathcal{D}_{t-1} and d_i^t is the value of its variables in the reward function $f_i^t \in \mathbf{F}^t$. These Q-values and R-values represent the predicted reward the agent will get if it assigns its variables values according to d_i^t when σ_{t-1} is the previous solution. The agents repeatedly refine these values at every iteration and choose the values with the maximum Q-value or R-value at each iteration.

5.3. Relevant Uses

Dynamic environments play a fundamental role in real-world MAS applications. Virtually all complex MAS applications involve dynamic situations, which may restructure the network topology due to agent movements, or

bring additional information to the problem being solved. For example, in a search and rescue operation during disaster management, as the environment evolves over time, new information becomes available about the civilians to be rescued and new agencies may arrive at any time to help conduct rescue operations. In a smart grid domain, real-time pricing is commonly enforced, thus, agent preferences need to be adapted over time, while energy costs are updated. This dynamic behavior is naturally captured by the D-DCOP framework.

A well-studied D-DCOP application is *mobile sensing teams (MST)* [107, 108]. In MST, agents are equipped with sensors that can be used to detect targets, and they are free to move in any feasible direction in a grid. Each agent has a *sensing range*, which determines the coverage range within which an agent can detect targets, and a *mobility range*, which determines the maximum distance that an agent can travel within a time step. The common goal of the team is for the agents to maximize the number of targets detected.

6. Quantified DCOPs

The various extensions of the DCOP model discussed so far differ from each other in terms of agent behavior (deterministic vs. stochastic), agent knowledge (total vs. partial) environmental behavior (deterministic vs. stochastic), and environmental evolution (static vs. dynamic). But in terms of the agent teamwork, all the models assume that the agents are completely cooperative. Recently, researchers have introduced the *Quantified DCOP (QDCOP)* model [109], which assumes that a subset of agents are adversarial, that is, the agents are *partially cooperative* or *competitive*.

6.1. Definition

The *Quantified DCOP (QDCOP)* model [109] adapts the *Quantified Constraint Satisfaction Problem (QCSP)* [110] and *Quantified Distributed CSP (QDCSP)* [111, 112] models to DCOPs. In QCSPs and QDCSPs, all variables are associated to quantifiers and the constraints should be satisfied independently of the value taken by universally quantified variables. Analogously, in QDCOPs, existential (\exists) and universal (\forall) quantifiers are introduced to differentiate the cooperative agents from the adversarial ones.

A *QDCOP* has the form $Q(\mathbf{F}) = q_0x_0 \dots q_nx_n$.⁸ Q is a sequence of quantified variables, where each $q_i \in \{\exists, \forall\}$ quantifies the variable x_i . The goal of a QDCOP is to find a global optimal solution of the corresponding DCOP. However, a universally quantified variable is not coordinated nor assigned, as the result has to hold when it takes any value from its domain. In contrast, an existentially quantified variable takes exactly one value from its domain, as in (*cooperative*) DCOPs. Thus, the optimal solution of a QDCOP may be different from that of the corresponding DCOP. While a DCOP solution defines a single value, associated to its utility, a QDCOP defines *upper* and *lower bounds* to the optimal solution. In particular, the best choice in a QDCOP defines the smallest *lower bound*. In the worst case, the universally quantified variables can worsen the overall objective as much as possible. Therefore, the worst case defines the smallest *upper bound*. While solving a classical DCOP is NP-complete, solving a QDCOP is in general PSPACE-complete.

6.2. Algorithms

QDCOPs impose a rigid order on the variables, which reflects the correct order of evaluation of the quantifiers. Therefore, classical DCOP algorithms cannot be directly applied to solve QDCOPs. Matsui et al. [109] proposed several variations of ADOPT to solve QDCOPs, which are all based on a DFS pseudo-tree ordering. To keep the ordering of the quantifiers unchanged, the pseudo-tree can be reshaped by applying extra null edges for each pair of nodes, if necessary.

The algorithms presented here are based on the intuition that universally quantified variables can be seen as adversarial virtual agents, whose goal is to minimize the overall objective. Following this intuition and the pseudo-tree modifications discussed above, pseudo-tree-based DCOP algorithms can be extended to solve QDCOP.

Min-max ADOPT [109]. *Min-max ADOPT* is a synchronous, search-based algorithm that extends ADOPT to solve QDCOPs. It uses VALUE messages to communicate values of the variables, and COST messages to announce

⁸In the original proposal, the set \mathbf{F} is separated in a set of constraints C , representing relationships among variables, and a set of functions F , assigning values to each valid assignment.

their utilities, similarly to ADOPT. Each agent, starting from the root of the pseudo-tree, assigns values to its variables and propagates them to its neighboring agents with lower priority. Upon receiving VALUE messages from all higher-priority neighbors, the agent updates its context and repeats the same process by choosing an assignment that maximizes its local utility. In Min-max ADOPT, the existentially quantified variables are used to compute the lower bounds, while the universally quantified variables are used to compute the upper bounds. This process is executed until the root agent detects that the upper bound is equal to the lower bound. This algorithm has a relatively simple structure and does not adopt any major pruning strategy.

Alpha-beta ADOPT [109]. *Alpha-beta ADOPT* is a synchronous, search-based algorithm that extends Min-max ADOPT by adapting the *alpha-beta* search strategy, a common pruning strategy adopted in *game-tree search*. This strategy employs two boundary parameters, *alpha* and *beta*, representing the lower bound and the upper bound for each possible utility of an assignment, respectively. Alpha represents the lower bound, controlled by the universally quantified variables, while beta represents the upper bound, controlled by the existentially quantified variables. Lower bound and upper bound can be modified exclusively by equally universally quantified and existentially quantified variables, respectively. In Alpha-beta ADOPT, when an agent reports the utility value of the current partial solution, its parent reduces the alpha/beta threshold accordingly. Thus, the new alpha/beta values are used to prune the search when an agent detects that the current solution cannot be better than any other solution already evaluated. Alpha and beta values are obtained using a backtracking technique similar to how thresholds are obtained through backtracking in the original ADOPT.

Bi-threshold ADOPT [109]. *Bi-threshold ADOPT* extends ADOPT by employing two *backtracking thresholds* instead of one as in ADOPT. In ADOPT, each agent a_i maintains the threshold invariant $lb_i^* \leq t_i \leq ub_i^*$, where lb_i^* and ub_i^* are the smallest lower and upper bounds, respectively, of the agent over all of its values and t_i is the threshold of the agent. In contrast, in Bi-threshold ADOPT, each agent maintains the threshold invariant $lb_i^* \leq t_i^\alpha \leq t_i^\beta \leq ub_i^*$, where t_i^α is a lower bound on the threshold, similar to alpha in Alpha-beta ADOPT, and t_i^β is an upper bound on the threshold, similar to beta in Alpha-beta ADOPT.

6.3. Relevant Uses

Adversarial agents are common in many MAS applications. In a QDCOP, universally quantified variables can be considered as the choice of the nature or an adversary. QDCOPs can thus formalize application problems where a team of agents seeks to limit the effect of the adversarial agents, as well as problems associated to planning under uncertainty. Examples of relevant applications include *smart grid contingency planning problems*, where suppliers need to satisfy consumer loads, which are unknown. Thus, the consumer can be seen adversarial events, and the suppliers try to find a robust plan that can handle any request from the consumers. Another domain is *distributed surveillance*, where sensors in a network need to coordinate to come up with a robust plan to detect intruders, who are trying to avoid detection.

7. DCOP Applications

DCOP models have been adopted to represent a wide range of MAS applications, thanks to their ability to capture essential and fundamental MAS aspects as well as the support for the development of general domain-independent algorithms, including applications in wireless sensor networks, power networks, and scheduling. We provide here a description of some of the most compelling applications as well as a general overview of their corresponding DCOP models. A comprehensive list of DCOP applications, categorized according to the DCOP classification of Table 2, is given in Table 4 with related references.

7.1. Disaster Management and Coordination Problems

Disaster management and coordination problems refer to how to efficiently and effectively respond to an emergency. In such scenarios, low-powered mobile devices that require limited bandwidth are often deployed and utilized. Due to their decentralized nature, the DCOP approach fits naturally with this application. We now describe several problems within this application domain.

Disaster Evacuation Problems. In a disaster scenario, moving evacuees to the closest refuge shelter can quickly overwhelm shelter capacities. A number of researchers have proposed a DCOP model for disaster evacuation, in which several groups of evacuees have to be led to available shelter [89, 113, 67, 114, 115]. Group leaders can communicate via mobile devices to monitor and coordinate actions. Each group is represented by a DCOP agent managing variables that represent shelter allocations. Thus, the domain of each variable corresponds to the available shelters. Group sizes and shelter capacity, as well as additional group requirements (e.g., medical needs) and the distance of a group to shelters, are encoded as cost functions. Solving the DCOP ensures an assignment of all groups to shelters that minimizes overflow, such that groups receive the services they need and their travel distances are minimal.

Coalition Formation with Spatial and Temporal Constraints Problems. In a Coalition Formation with Spatial and Temporal Constraints (CFST) problem [99, 116, 117], ambulance and fire brigade agents cooperate in order to react efficiently to an emergency scenario so as to rescue victims and extinguish fires located in different parts of a city. Agents can travel from one location to another in a given time. Each task (i.e., rescuing a victim, extinguishing a fire) has a deadline, representing the time until which the victim will survive, and a workload, denoting the amount of time necessary to rescue the victim or put out the fire. The locations of the victims and the fires may be unknown to the agents, and need to be discovered at run time, which requires agents to dynamically update the sequence of the tasks they will attempt, taking account of two main constraints: (1) Spatial constraints, which model where an agent can route and at what time; and (2) Temporal constraints, which model task deadlines and completion times. Agents may also form coalitions to execute a given task faster or if the requirements of a given task cannot be met by one single agent. Hence, the agents' arrival times at each task need to be coordinated in order to form the desired coalition. The objective is to maximize the number of tasks to be completed.

A DCOP formalization for the CFST is described in [99], where ambulances and fire brigades are modeled as DCOP agents. Each agent controls a variable that encodes the current task that the agent will attempt, and whose domain represents task locations. Unary constraints restrict the set of reachable locations, according to distance from a destination and the victim's deadline. Agents' coalitions are defined as groups of agents traveling to the same location. Each task is associated to a utility, which encodes the success for a coalition to complete such task. The goal is to find an assignment of agents to tasks that maximizes the overall utility.

7.2. Radio Frequency Allocation Problems

The performance of a wireless local area network (WLAN) depends on the channel assignments among neighboring access points (APs). Neighboring transmissions occurring in APs on the same channel or adjacent channels degrade network performance due to transmission interference. Typically, in dense urban areas, APs may belong to different administrative domains, whose control are delegated to different entities. Thus, a distributed approach to the channel assignment is necessary.

Cooperative Channel Assignment Problems. In a cooperative channel assignment problem [118, 119, 120], APs need to be configured in order to reduce the overall interference between simultaneous transmissions on neighboring channels. In [119, 120], a DCOP-based approach is proposed for cooperative channel assignment in WLANs where APs may belong to different administration entities. In the proposed model, each AP is represented by a DCOP agent, which controls a decision variable modeling a choice for the AP's channels. The signal-to-interference-and-noise ratio (SINR) perceived by an AP or a client is modeled as a cost function, as the overall concurrent transmissions occurring in the same channel and in partially overlapped adjacent channels. The goal is to find an assignment of channels to APs that minimizes the total interference experienced in the WLAN.

In [121, 122], the authors study dynamic solutions to the problem of allocating and utilizing the wireless network's available spectrum. In such a problem, the agents operate in a dynamic radio frequency environment that is composed of time-varying interference sources, which are periodically sampled and measured.

7.3. Recommendation Systems

Recommendation systems are tools that provide user-tailored information about items to users. These systems provide information that is tailored to the characteristics and preferences of the users.

Group Recommendation Problems. Like individual recommendations, group recommendations need to take into account the preferences of all group members and formulate a recommendation that suits the whole group. In [123], the authors proposed a DCOP-based travel package recommendation system for groups of users. The users in the group share a common goal (the travel package recommendation), and have individual preferences for each travel service (hotel, flight companies, tour operators, etc). In such a problem, the objective is to find a group recommendation that optimizes the users' preferences. The proposed DCOP solution is composed of two types of agents: user agents and recommender agents. Each user agent controls a decision variable that models that user's travel choices, while each recommender agent controls a decision variable that models a travel service supplier's recommendations. User travel preferences are modeled via unary constraints on the user agents' decision variables. Binary constraints between user agents in a group and their associated recommender agent ensure that each user's choice in the group is compatible to the recommendation of the recommender agent. The goal is to find the best recommendation for the entire group.

7.4. Scheduling Problems

Scheduling problems are an important class of problems, and they have been long studied in the area of Constraint Programming and Operations Research [124, 125, 126, 127]. It is a problem where one associates time schedules to resource usage. The problem is made particularly difficult when scheduling needs to occur in a distributed manner across several entities. In such a context, many scheduling problems can be naturally mapped to distributed constraint optimization problems.

Radar Network System Scheduling Problems. The *NetRad* system is a system of radars designed for quick detection of low-lying meteorological phenomena, such as tornadoes [128]. They are short-range radars used in dense networks, which can scan a particular volume of space. In the *NetRad* system, information collected on weather phenomena is actively used to create new scanning strategies, which can be dynamically modified. The *NetRad* system is organized in multiple meteorological command and control (MCCs) units, each controlling a set of radars. End users such as forecasters and emergency managers can provide preferences on the phenomena that they would like to track. Thus, the MCC ranks the importance of scanning tasks accordingly. This scheduling problem focuses on selecting a weather-scanning strategy for radars to use in the next radar scanning cycle that maximizes end user preferences.

In the context of a DCOP [129, 130], each MCC is modeled as an agent whose decision variables represent the scanning strategies of each radar controlled by that MCC. Each radar can choose where to scan by choosing a subset of phenomena in its range. The utility of a scan of a phenomenon depends on the priority of the phenomenon determined by the requesting user as well as the quality of the scan. The goal is to find the joint policy for each radar that maximizes the system utility.

Distributed Meeting Scheduling Problems. The distributed meeting scheduling problem captures generic scheduling problems where one wishes to schedule a set of events within a time range [131, 132]. Each event is defined by: (i) the resources required for it to be completed, (ii) the time required for it to be completed, within which it holds the required resources, and (iii) the cost of using such resources at a given time. A scheduling conflict occurs if two events with at least one common resource are scheduled in overlapping time slots. The goal is to maximize the utilities over all the resources, defined as the net gain between the opportunity benefit and opportunity cost of scheduling various events.

In [49], the authors discuss three possible DCOP formulations for this problem: time slots as variables (TSAV), events as variables (EAV), and private events as variables (PEAV). We describe the EAV formulation and refer the reader to the original article for the other two formulations and additional details. In the EAV formulation, events are considered as decision variables. Each variable can take on a value from the time slot range that is sufficiently early to schedule the required resources for the required amount of time or zero to denote that an event is not scheduled. If a variable takes on a non-zero value, then all its required resources cannot be assigned to any other overlapping event.

Water Allocation Scheduling Problems. The management of water resources in large-scale systems is often associated with multiple institutionally-independent decision makers (DMs), which may represent different and conflicting interests, such as flood prevention, hydropower production, and water supply [133]. The aim of such problems is to find an efficient use of water allocation and distribution according to the different users' interests.

In [133], the authors formalize a regulatory mechanism in water management as a DCOP. The model involves several *active* human agents and *passive* ecological agents. Each agent is associated with an optimization function that

it seeks to maximize. Active agents make decisions about the amount of water to divert from the river or to be released from a dam in order to maximize their corresponding objective functions. Passive agents, on the other hand, represent ecological interests through their associated objective functions and do not make decisions. The agents model different water supplies for cities and agricultural districts, hydropower productions, and ecological preservation. The goal is to optimize the agents' objective functions, satisfying hard (physical) constraints and maximizing the soft (normative) constraints, which aim at protecting the interests of the passive agents. A solution to such problem, which makes use of a multi-objective DCOP formalization, is presented in [134].

Patient Scheduling Problems. Medical appointments scheduling problems are related to meeting scheduling problems, as they need to associate patients to resources (e.g., doctors, medical machinery) and times, but they require different types of constraints. Patients may require several services from different departments within the same hospital or in multiple hospitals. In general, the objective is to minimize the patient treatment waiting time under limited resource conditions, as well as ensuring efficient resource usage, taking into account patient preferences.

In [135], the authors formulate the problem of scheduling patients to diagnostic units in an hospital as a DCOP, where appointments are modeled as variables, whose domains describe times, durations, and locations. The constraints of the problem model the schedule feasibility, the patient preferences over hospitalization times, the workplace constraints, which restricts the types of appointment for a given workplace, and diagnostic unit constraints, which model resource usage.

In [136, 137], the authors propose a Dynamic DCOP model for a radiotherapy patients scheduling problem. In this problem, each agent represents a patient, and it controls variables that represent private information (e.g., type of tumor, number of radiation doses per day, the use of chemotherapy) and public information (e.g., current schedule of the radiotherapy machine). The constraints of the problem model the duration of each daily treatment, as well as tumor-specific treatment restrictions. The problem objective considers patient waiting times to receive their treatment, patient priorities (based on tumor aggressiveness), and patient preferences.

7.5. Sensor Network Problems

Sensor networks typically consist of a large number of inexpensive and autonomous sensor nodes, constrained by a limited communication range and battery life. These networks have been deployed for environmental sensing (temperature, humidity, etc.), military applications (e.g., battlefield surveillance), and target tracking [138]. When deploying sensor networks, it may not be possible to pre-determine the position of each sensor node. The distributed nature of the problem and the presence of several communication and sensing constraints create a natural fit for DCOPs to solve a wide range of related applications.

Target Tracking Problems. In a typical target tracking application [46, 139, 140, 141, 142, 143], a collection of small Doppler sensors are scattered in an area to detect possible moving targets in the region. Each sensor is battery-powered, can communicate with one another through radio communication, and can scan and detect an object within a fixed range. Communication incurs an energy cost. Thus, to save energy, a sensor may turn itself off. Multiple sensors may be necessary to detect a single target with high accuracy. The overall objective is to maximize the number of targets detected, as quickly as possible and, at the same time, preserve energy so as to prolong the system's lifetime.

In [46], a simplified version of the above problem is viewed as a weighted graph coloring problem, where the total weight of violated constraints need to be minimized. A node corresponds to a sensor, an edge between two nodes represents the constraint of a shared region between agents, and the weight captures the importance of the common region. The size of the common region reflects the amount of energy loss when two sensors scan the shared region at the same. Each color corresponds to a time slot in which a sector is scanned. A node must have at least one color so that the corresponding sector is scanned at least once. This graph coloring problem is mapped to a DCOP, where agents represent nodes, agent's variables represent the agents decision on their color, and reward functions represent the graph edges.

In [143], the authors use a hierarchical DCOP approach to scale to larger problems. The authors partition the original problem into n local regions, and use n DCOPs to solve the smaller subproblems. Their solutions are then combined in a hierarchical approach, solved by a DCOP that encompasses variables and constraints shared among the connected regions of the lower hierarchy DCOPs.

Robotic Network Optimization Problems. The robotic network optimization problem describes a sensor network problem where sensors are placed on top of robots that have limited movement capability. In such a problem, robots

can make small movements to optimize the wireless connectivity with their neighbors, without affecting the network topology [144, 140].

In [140], the authors propose a DCOP formulation where each robot is represented by an agent. Each agent controls one variable describing the decision on the robots' possible movements. Thus, the variables' domains consist of the valid positions the agent can move to. The reward functions of the problem model the power gain (or loss, depending on the optimization criteria) of the wireless link from a transmitter and a receiver robot, and depend on their positions. Radio communication in wireless sensor networks has a predictable signal strength loss that is inversely proportional to the square of the distance between transmitter and receiver. However, radio wave interference, is very difficult to predict [145]. Thus, in [140], the authors use a P-DCOP-based approach with partial agent knowledge to capture the robot's partial knowledge on its reward functions, and to balance *exploration* of the unknown rewards and *exploitation* of the known portion of the rewards.

Mobile Sensor Team Problem. The *Mobile Sensor Team* (MST) problem is similar to the target tracking problem with the difference that agents are capable of moving autonomously within the environment and that time is modeled explicitly as a discrete sequence of time steps. In an MST, agents are placed on a grid. For an agent a_i , cur_pos_i denotes the agent's current position; SR_i denotes the agent's perception *sensing range*, which determines the coverage range within which an agent can detect targets; MR_i denotes the agent's mobility range, which defines the maximum distance that the agent can move within a single time step; and $cred_i$ denotes the agent's credibility, which reflects the likelihood of the correctness of the detected targets. The targets are defined implicitly through an *environmental requirement* (ER) function, which defines, for each point in space, the minimum *joint credibility* value (the sum of the credibility variables) required for that point to be sensed. In such a representation, targets are points p with $ER(p) > 0$. Given a set of agents SR_p whose sensing range covers a target p , the *remaining coverage requirement* of p is the environmental requirement diminished by the joint credibility of the agents currently covering p : $Cur_REQ(p) = \max\{0, ER(p) \ominus \sum_{a_i \in SR_p} cred_i\}$, where $\ominus : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is an operator that defines how the environmental requirement decreases by the joint credibility. The goal of the agents is to find positions that minimize the values of Cur_REQ for all targets.

MST problems are modeled through a subclass of dynamic DCOPs, named *DCOP_MST* [107, 108, 146]. Each agent a_i controls one variable x_i representing its position, and whose domain contains all locations within MR_i of cur_pos_i . Thus, the domains are updated each time the agent moves.⁹ The constraint C_p of a target p involves exclusively those agents a_i whose variable's domain includes a location within SR_i of p . Thus, at each time step, both domains and constraints may change. As a consequence, the constraint graph changes as well—the neighbors of each agent has to be updated at each time step. Finally, in a *DCOP_MST* two agents are neighbors if their sensing areas overlap.

Sensor Sleep Scheduling Problem. Wireless sensor nodes are equipped with a radio, which can be used to communicate with neighboring nodes, and a limited power source. These sensor nodes are often deployed in inaccessible terrains, thus replacing their power sources may not be possible. The wireless sensor sleeping scheduling problem aims at switching on/off a particular sensor node component (such as the sensor or the radio) for a certain period of time, so to ensure power conservation, maximizing the lifetime of the sensor network.

In [147], the authors propose a DCOP model for this problem, where each sensor is an agent whose variables denote its status (on or off) for each time step. Hard constraints are employed to enforced that if a sensor is on, then all its neighbors should be off, and that sensors cannot stay on for two consecutive time steps. The overall objective is to minimize the delay induced in the network.

A similar problem is solved in [142], where sensors are also able to harvest energy from the environment (e.g. using a photo-voltaic cell or vibration-harvesting microgenerators). In such a context, the goal is to find a schedule that maximizes the probability of detecting events while maintaining energy neutral operations (that is, exhibit an indefinite lifetime for each of the agents).

⁹An alternative representation is that of modeling the domain of each variable as the entire grid, and to constrain the agent variable, at each agent move, in order to hide those points lying outside MR_i .

7.6. Service-Oriented Computing Problems

The service-oriented computing paradigm is one that relies on sharing resources over a network, focusing on maximizing the effectiveness of the shared resources, which are used by multiple applications. Efficient solutions with optimal use of resources is crucial in this paradigm and has a wide industrial impact [148]. The distributed nature of the resources and the privacy concerns arising when different clouds are involved in the deployment, makes DCOP appealing to solve a range of problems in this paradigm [149].

Application Component Placement Problems. An application component placement (ACP) problem is defined over a network of servers offering storage devices with various capabilities, and component-based application with requirements for processing, communication, and storage [150, 151]. The ACP problem is a problem of deciding which server to assign to each application component. The component-based application is described by a set of characteristics that establish their requirements in terms of hardware (e.g., CPU speed, storage capacity) as well as constraints between components of the same application (e.g., minimum bandwidth, secure communication channel requirement).

In [150], the authors propose a DCOP model for the ACP problem where servers bid for a component to host, with an emphasis that is proportional to the affinity of the server characteristics and the component hardware and software requirements. Each server is modeled by an agent, which controls a decision variable representing the server bids. Thus, the domain of each variable is the set of possible components that may be deployed on the server. Unary functions express utility for each components. Hard constraints are employed to ensure that each component is deployed exactly on a single server, and that two components are placed between servers satisfying the required communication bandwidth. The objective is to find a feasible assignment of component to servers that maximizes the utilities.

Server Allocation Problems. Services-oriented middleware networks are composed of entities that can both provide and require multiple service connected within a physical network. In turn, each service can be provided by multiple servers and can serve multiple clients. A service request from a given client, takes into account various quality of service (QoS) parameters (e.g., service response time, service completion time). When a client generates a service request, it can be satisfied by any of the servers offering such requests. The server allocation problem is a problem of selecting servers to allocate services, ensuring maximum social welfare, while meeting the QoS requirements of all clients.

In [152], the authors present a DCOP model for this problem where agents correspond to network entities, variables correspond to services, and their values are either 1, if the associated agent is willing to provide/forward the service, or 0, otherwise. Clients' service requests are mapped to servers' service offers, accounting for the delays that occur when traversing between intermediate nodes using a routing multicast protocol [153]. Moreover, in order to provide a service, all requested QoS requirements need to be satisfied. The utility associated to each variable is the combination of the utility for such a node when acting as a service consumer, a service provider, and a service forwarder, and depends on several parameters, such as available GPU cycles, battery power, memory, and bandwidth. The problem may change dynamically when a new request is made, or when a new service is offered or released, and as such can be modeled as a D-DCOP.

7.7. Smart Grid Problems

The smart grid is a vision of the future electricity grid, where bidirectional flow of both electricity and information, in an automated fashion, improves efficiency and reliability of energy production and distribution. The development of smart grids poses several challenges: (1) How to deal with the increasing electric grid utilization due to growth of loads, such as electric vehicles (EVs) and heat pumps; (2) How to efficiently integrate a diverse range of energy sources, including intermittent renewable generators, into the power network; and (3) How to deal with the uncertainty in the equipment as well as in the participation of consumers through demand side technologies. Due to the distributed and dynamic nature of loads and generators participating in the smart grid, agent-based decentralized autonomous control of smaller distributed microgrids is a very compelling solution [154, 155]. In particular, several solutions based on agent-based decentralized optimization have been explored to deliver this vision [156, 157, 158, 159]. The following is a list of the most prominent DCOP approaches for smart grid applications.

Optimal Dispatch Problems. This problem is the problem of coordinating network generators with respect to network cost, while satisfying the load and network constraints [160]. Researchers have cast this problem as a DCOP [156, 161, 162, 158], where the authors consider a network of nodes (agents), each of which relays power to other nodes, but can also contain a combination of generators and loads. Generators are distributed across nodes, and are represented through variables whose domain describe a certain set of discrete power outputs. The distribution cables connecting nodes of the networks, are also associated to DCOP variables, each of which has a thermal capacity describing the maximum power that it can safely carry. The DCOP function describes a particular optimization criteria, such as minimizing the carbon emissions of generators within the network, as well as imposes load and network constraints. In particular, the constraints ensure that the overall demand and supply are in balance and that the thermal capacity constraints of the cables are satisfied.

Power Supply Restoration Problems. After (multiple) line failures, a power grid must to be reconfigured to ensure restoration of power supply. A power network distribution is a network of power lines connected by switching devices (SDs) and fed by circuit breakers (CBs). SDs are analogous to sinks (transformer stations), while CBs are analogous to power sources. Both of these devices can operate in two states: open or closed. Closed SDs consume some power and forward the rest of it on other lines. Open SDs stop power flow. CBs feed the network when they are closed. The configuration of the devices' state is such that energy flow traversing CBs takes the form of a (*feeder*) *tree*, and that no SD is powered by more than a power line. Flow conservation and transmission line capacity constraints must be enforced. The power supply restoration problem is the problem of finding a configuration that ensures power restoration for the maximum number of sinks affected by the line failures.

Researchers have proposed a DCOP formulation for this problem in [159, 163]. In such a framework, each node of the distribution network is controlled by an agent, which owns all variables and constraints corresponding to that node. Two DCOP variables are associated to each network node: A load variable and a direction variable. Load variables model the amount of incoming flow for sink nodes, and the number of sinks fed for power source nodes. Direction variables model all the possibilities of feeding a node, as the set of possible configurations in which its neighboring nodes can forward power to it. The acyclicity of the power flow and the flow conservation are modeled as constraints. The former restricts the power path to be a tree as well as defines the optimization criterion. The latter enforces Kirchhoff's law, that the amount of incoming power flow to the node i must equal the sum of power consumed at i and the amount of power forwarded to other nodes.

Microgrid Islanding Problems. A microgrid islanding problem is the problem of creating islands (i.e., clusters of generator units and loads able to operate without external energy supply) in response to major power outages and blackouts. In [164], the authors formalized this problem as a DCOP, where agents represent nodes in the network, and each agent has its own power generation and power consumption capabilities. Variables represent the amount of power that an agent generates and consumes, as well as transmission line flows and switch status between network nodes. Flow variables are constrained by their maximum transmission line capacities, while switches are modeled as binary variables that can be turned on or off. Flow conservation are modeled as constraints to enforce Kirchhoff's law. The goal is to find a switching configuration that minimizes the unserved load of the system.

Prosumer Energy Trading Problems. In its more general form, a smart grid is populated by prosumers capable of both generating and consuming resources. The prosumer energy trading problem aims at setting market-based prices for prosumers to directly trade over the smart grid, while taking account of the grid constraints. This problem has been cast as an optimization problem called the energy allocation problem, where given a graph with nodes representing prosumers, and edges describing transmission lines connecting adjacent prosumers, the goal is to find the allocation which maximizes the benefits of all the prosumers while satisfying the capacity constraints of the network.

In [165], the authors propose a DCOP formalization for this problem, where each prosumer is an agent. Variables are associated with edges of the energy trading network (i, j) and describe the number of units of energy that prosumer i sells to/buys from prosumer j . Thus, two variables (i, j) and (j, i) are associated to each edge of the network. For each prosumer, an energy balance constraint models the utility of a given instantiation of its offers as the sum of the offers associated to the energy traded with each of its neighbor. Line capacity and flow conservation constraints ensure that the energy traded along the transmission lines is within their maximum capacity and is consistent with the Kirchhoff's law.

7.8. Supply Chain Management Problems

The management of large businesses involves the management of the flow of goods from suppliers to customers. This flow of goods is called a supply chain. Supply chains have to be carefully managed to ensure that a sufficient quantity of raw material is available at factories for production and a sufficient quantity of processed goods is available at stores for consumers to buy. Additionally, since goods can be purchased from different producers and sold to different consumers, there is also the need to consider how much to buy/sell the goods and who to buy/sell the goods to. In such an environment, information, decision making and control are inherently decentralized.

Supply Chain Formation Problems. A supply chain formation problem is the process of determining the participants in a supply chain, who will exchange what, with whom, and the terms of the exchange. Several DCOP-based approaches for this problem have been proposed in the literature [166, 167, 168, 169, 170]. In general, they rely on the notion of a *Task Dependency Network* (TDN) introduced by Walsh and Wellman [171], which is a graph-based representation to capture dependencies among production processes. A TDN is a bipartite directed acyclic graph representing exchanges, where nodes in the graph correspond to producers and consumers and edges in the graph correspond to feasible exchange of goods between the producers and consumers. A path from potential producers and consumers defines a feasible supply chain configuration and the goal is to find the feasible configuration that optimizes a particular cost function. A DCOP encoding for this problem models producers and consumers as agents, each of which controls a variable describing the agent's decision regarding whom to buy from/sell to as well as its associated quantities and costs. Cost functions are associated with each edge of the TDN encoding the willingness of the producer and consumer to trade with each other.

A dynamic version of the above formalization has been investigated in [172]. Such a model allows for the entry and departure of producers and consumers as well as changes in properties of the problem (e.g., prices of goods, production capacity of producers, consumption requirements of consumers, etc.).

7.9. Traffic Flow Control Problems

A challenge for the increase of transportation demand is to enforce traffic flow control using the existing infrastructure, such as traffic lights, loop detectors, and cameras. Coordinating the actions of these individual devices aims at smoothing the traffic flow at the network level. Such coordinated actions often generates coherent traffic control plans faster and more accurately compared to those of a human traffic operator [173]. Due to the distributed nature of such devices, multi-agent solutions are particularly suitable for this class of problems.

Traffic Light Synchronization Problem. This problem is the problem of finding a synchronization schema for the traffic lights in adjacent intersections that creates green waves, which are waves of vehicles that are traveling at a given speed and are able to cross multiple intersections without stopping at red lights.

In [174, 175], the authors model this problem as a DCOP, where agents represent traffic lights, each controlling one variable that models the coordination direction for the associated traffic signal. Thus, the domain of the variables is given by two possible directions of coordination (north-south/south-north, east-west/west-east). Conflicts that arise when two neighboring traffic signals choose different directions are modeled as constraints. Cost functions are defined to model the number of incoming vehicles in a junction, and the costs are influenced by whether two adjacent agents agree on a direction of coordination or not. The goal is to minimize the global cost. Due to the dynamic nature of the problem, the proposed algorithms fall under the umbrella of D-DCOP algorithms.

In [176, 177], the authors propose a P-DCOP-based approach with partial agent knowledge to solve the traffic light synchronization problem in the context where agents have partial information about their cost functions. In particular, the authors argue that traffic patterns may vary during time and, thus, the agents should learn them to update their cost functions. In this context, agents are given a limited amount of time to explore different signal duration intervals, whose associated rewards are learned by evaluating the average travel time of the first 100 cars traveling across the agent's traffic light.

8. Potential Directions for Future Work

Despite the fact that classical DCOPs have reached a sufficient level of maturity from the algorithmic perspective, most of the other proposed formalisms fall short on both theoretical and algorithmic foundations. The proposed algorithms mostly extend classical DCOP algorithms and, therefore, they result in similar performances. Investigating

Group	Problem	Model			References
		AK	EB	EE	
DISASTER MANAGEMENT & COORDINATION	<i>Disaster Evacuation</i>	T	D	S	[89, 67, 113, 115]
		T	D	D	[67, 114]
	<i>Coalition Formation</i>	P	D	D	[99, 116]
RADIO FREQUENCY ALLOCATION	<i>Cooperative Channel Assignment</i>	T	D	S	[119, 120]
		T	D	D	[121, 122]
RECOMMENDATION SYSTEMS	<i>Group Recommendation</i>	T	D	S	[123]
SCHEDULING	<i>Radar Network System Scheduling</i>	T/P	D/S	D	[129, 130]
	<i>Distributed Meeting Scheduling</i>	T	D	S	[49, 178]
	<i>Water Allocation Scheduling</i>	T	D	S	[133, 134]
		T	D	S	[135]
	<i>Patient Scheduling</i>	T	D	D	[136, 137, 179, 180]
SENSOR NETWORK	<i>Target Tracking</i>	T	D/S	D	[46, 139, 141, 143]
	<i>Robotic Network Optimization</i>	P	D	D	[144, 140]
	<i>Mobile Sensor Team</i>	P	S	D	[107, 108, 146]
	<i>Sensor Sleep Scheduling</i>	T	D	S	[147, 142]
SERVICE-ORIENTED COMPUTING	<i>Application Component Placement</i>	T	D	S	[150, 151]
	<i>Server Allocation</i>	T	D	D	[152]
SMART GRID	<i>Optimal Dispatch</i>	T	D/S	D	[156, 161, 162, 158]
	<i>Power Supply Restoration</i>	T	D	S	[164]
	<i>Microgrid Islanding</i>	T	D	S	[159, 163]
	<i>Prosumer Energy Trading</i>	T	D/S	S	[165]
SUPPLY CHAIN MANAGEMENT	<i>Supply Chain Formation</i>	T	D	S	[166, 167, 168, 169, 170]
		T	D	D	[172]
TRAFFIC CONTROL	<i>Traffic Light Synchronization</i>	T	S	D	[174, 175]
		P	S	D	[176, 177]

Table 4: DCOP Applications. AK = Agent Knowledge, it can be [T]otal or [P]artial; EB = Environment Behavior, it can be [D]eterministic or [S]tochastic; EE = Environment Evolution, it can be [S]tatic or [D]ynamic.

strategies that are based on different backgrounds could help propel the evolution of this area. In particular, we envision that insights from machine learning and data mining could be integrated within the dynamic and probabilistic DCOP algorithms to supply information on the unknown and dynamic environment characteristics. Orthogonally, further investigations on relating DCOPs to the areas of game theory and decision theory are also necessary. In particular, similar to the work done by Chapman *et al.* [181], where they studied relationship between DCOPs and potential games, analyzing the relationship of DCOPs to cooperative and non-cooperative games could shed light on how to effectively address coordination and reasoning strategies in DCOP with partially cooperative agents. Additionally, merging insights from decision theory, such as handling partial observability, with the inherent DCOP ability of naturally exploiting problem structure could result in improved performance and refined models.

Another open question is related to agent coordination. It has been observed that simple coordination strategies give good results in extremes (high or low) agent workload environments. However, at intermediate workload levels, such strategies lose their effectiveness, and more complex coordination strategies are necessary [182, 183]. For example, Zhang and Lesser [183] study agent learning, where coordination is driven by a dynamic decomposition of a DCOP, each solving smaller independent subproblems. Their proposal effectively produced near-optimal agent policies for learning and significantly reduced the amount of communication. These empirical observations suggest the existence of phase transition behaviors occurring at the level of agent coordination. A formal understanding of these phenomena, which looks at the environment, agents' local reasoning strategies, their assumptions on other agents states, as well as team coordination could help researchers to understand the inherent complexity of coordination problems. Such a formal framework could be useful to build new and more efficient coordination strategies for a wide variety of multi-agent applications, perhaps resembling the way studies of phase transitions of NP-hard problems [184] led to the understanding of problem complexity and creation of effective heuristics and search strategies.

Modeling many real-world problems as DCOPs often require each agent to solve large complex subproblems, each requiring many variables and constraints. A limitation of most DCOP algorithms is the assumption that each agent controls exactly one variable, and that all constraints are binary. Such assumptions simplify the algorithm organization and presentation. However, in order to operate under this assumption, DCOP algorithms have to reformulate the problem using pre-processing techniques [185, 186], which can negatively impact their performance. A more realistic view is to allow each agent to solve its local subproblem (in a centralized fashion) since it is independent of the subproblems of other agents. The agent’s subproblem resolution can then explore techniques from centralized reasoning, such as constraint optimization problems (COPs), linear programming (LP), and graphical models. One can even exploit novel hardware platforms, such as general-purpose graphical processing units (GPGPUs) to parallelize such solvers.

Another open question in this research area concerns the definition of a systematic process to evaluate and compare the DCOP algorithms. First, there are multiple metrics that can be used to measure the runtime of an algorithm, such as the number of non-concurrent constraint checks (NCCCs) [187], the simulated runtime [188], and the number of cycles [189]. However, there is no consensus on a standard metric. Second, there is no general language being used to formally specify a DCOP. By and large, most stand-alone algorithms specify DCOPs in an ad-hoc manner. Due to this two factors, it is often inconvenient to experimentally compare different algorithms. In addition, new proposed algorithms are, in general, evaluated on arbitrary benchmarks, some inherited from the CSP literature, some other from approximations of more concrete problems. To cope with these issues, it will be useful to develop a benchmark repository, perhaps by taking inspiration from the efforts made by the constraint programming (CP) community with CSPLib [190], a library of test problems for constraint solvers, or by the planning community with their international planning competitions [191].

9. Conclusions

DCOPs have emerged as a popular formalism for distributed reasoning and coordination in multi-agent systems. Due to its limitation to support complex, real-time, and uncertain environment, researchers have introduced several model extensions to handle dynamic and uncertain environments, as well as different levels of cooperation within the agents. In this article, we provided an analysis of the recent advances made by the AAMAS community within the DCOP framework and propose a categorization based on agent characteristics, environment properties, and type of teamwork adopted. Within the proposed classification, we (1) presented a comprehensive algorithmic review, analyzing inter-relation and performances; (2) discussed a number of application domains that can be naturally modeled within each DCOP framework; and (3) pondered on some potential directions for future work with regards to agent coordination, algorithm scalability, modeling language, and evaluation criteria of the DCOP models and algorithms.

Acknowledgments

The research in this paper has been partially supported by NSF grants DBI-1458595, CBET-1401639, HRD-1345232 and DGE-0947465.

References

- [1] H. Raiffa, *Decision analysis: introductory lectures on choices under uncertainty*, Addison-Wesley, 1968.
- [2] D. S. Bernstein, R. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of markov decision processes, *Mathematics of operations research* 27 (4) (2002) 819–840.
- [3] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, M. J. Kochenderfer, Decentralized control of partially observable markov decision processes, in: *Proceedings of the Annual Conference on Decision and Control (CDC)*, IEEE, 2013, pp. 2398–2405.
- [4] K. Binmore, *Fun and games, a text on game theory*, DC Heath and Company, 1992.
- [5] S. Kraus, Negotiation and cooperation in multi-agent environments, *Artificial Intelligence* 94 (1) (1997) 79–97.
- [6] P. Noriega, C. Sierra, Auctions and multi-agent systems, in: *Intelligent Information Agents*, Springer, 1999, pp. 153–175.
- [7] S. Parsons, M. Wooldridge, Game theory and decision theory in multi-agent systems, *Autonomous Agents and Multi-Agent Systems* 5 (3) (2002) 243–254.
- [8] F. Rossi, P. v. Beek, T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*, Elsevier Science Inc., New York, NY, USA, 2006.

- [9] P. Modi, W.-M. Shen, M. Tambe, M. Yokoo, ADOPT: Asynchronous distributed constraint optimization with quality guarantees, *Artificial Intelligence* 161 (1–2) (2005) 149–180.
- [10] A. Petcu, B. Faltings, A scalable method for multiagent constraint optimization, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 1413–1420.
- [11] W. Yeoh, M. Yokoo, Distributed problem solving, *AI Magazine* 33 (3) (2012) 53–65.
- [12] E. Bowring, M. Tambe, M. Yokoo, Multiply-constrained distributed constraint optimization, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, ACM, 2006, pp. 1413–1420.
- [13] F. Pecora, P. Modi, P. Scerri, Reasoning about and dynamically posting n-ary constraints in adopt, in: *International Workshop on Distributed Constraint Reasoning (DCR)*, Vol. 7, 2006.
- [14] T. Matsui, H. Matsuo, M. Silaghi, K. Hirayama, M. Yokoo, Resource constrained distributed constraint optimization with virtual variables, in: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2008, pp. 120–125.
- [15] F. R. Kschischang, B. J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, *IEEE Transactions on Information Theory* 47 (2) (2001) 498–519.
- [16] W. Yeoh, Speeding up distributed constraint optimization search algorithms, Ph.D. thesis, University of Southern California, Los Angeles (United States) (2010).
- [17] K. Hirayama, M. Yokoo, Distributed partial constraint satisfaction problem, in: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 1997, pp. 222–236.
- [18] A. Gershman, A. Meisels, R. Zivan, Asynchronous Forward-Bounding for distributed COPs, *Journal of Artificial Intelligence Research* 34 (2009) 61–88.
- [19] W. Yeoh, A. Felner, S. Koenig, Bnb-adopt: An asynchronous branch-and-bound dcop algorithm, *Journal of Artificial Intelligence Research* 38 (2010) 85–133.
- [20] P. Gutierrez, P. Meseguer, Removing redundant messages in n-ary BnB-ADOPT, *Journal of Artificial Intelligence Research* 45 (2012) 287–304.
- [21] P. Gutierrez, P. Meseguer, W. Yeoh, Generalizing ADOPT and BnB-ADOPT, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, pp. 554–559.
- [22] W. Yeoh, X. Sun, S. Koenig, Trading off solution quality for faster computation in DCOP search algorithms, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2009, pp. 354–360.
- [23] W. Yeoh, P. Varakantham, S. Koenig, Caching schemes for DCOP search algorithms, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009, pp. 609–616.
- [24] C. Bessiere, P. Gutierrez, P. Meseguer, Including soft global constraints in DCOPs, in: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2012, pp. 175–190.
- [25] C. Bessiere, I. Brito, P. Gutierrez, P. Meseguer, Global constraints in distributed constraint satisfaction and optimization, *Computer Journal* 57 (6) (2014) 906–923.
- [26] P. Gutierrez, P. Meseguer, Improving BnB-ADOPT⁺-AC, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012, pp. 273–280.
- [27] P. Gutierrez, J. Lee, K. M. Lei, T. Mak, P. Meseguer, Maintaining soft arc consistencies in BnB-ADOPT⁺ during search, in: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2013, pp. 365–380.
- [28] A. Petcu, B. Faltings, ODPOP: An algorithm for open/distributed constraint optimization, in: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2006, pp. 703–708.
- [29] A. Petcu, B. Faltings, MB-DPOP: A new memory-bounded algorithm for distributed optimization, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 1452–1457.
- [30] A. Petcu, B. Faltings, Approximations in distributed optimization, in: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2005, pp. 802–806.
- [31] R. Greenstadt, B. Grosz, M. Smith, SSDPOP: Improving the privacy of DCOP with secret sharing, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2007, pp. 1098–1100.
- [32] A. Petcu, B. Faltings, R. Mailler, PC-DPOP: A new partial centralization algorithm for distributed optimization, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 167–172.
- [33] A. Kumar, A. Petcu, B. Faltings, H-DPOP: Using hard constraints for search space pruning in DCOP, in: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2008, pp. 325–330.
- [34] F. Fioretto, T. Le, W. Yeoh, E. Pontelli, T. C. Son, Improving DPOP with branch consistency for solving distributed constraint optimization problems, in: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2014, pp. 307–323.
- [35] T. Le, T. C. Son, E. Pontelli, W. Yeoh, Solving distributed constraint optimization problems with logic programming, in: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2015, pp. 1174–1181.
- [36] R. Mailler, V. Lesser, Solving distributed constraint optimization problems using cooperative mediation, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2004, pp. 438–445.
- [37] T. Grinshpoun, A. Meisels, Completeness and performance of the apo algorithm., *Journal of Artificial Intelligence Research* 33 (2008) 223–258.
- [38] A. Farinelli, A. Rogers, A. Petcu, N. Jennings, Decentralised coordination of low-power embedded devices using the Max-Sum algorithm, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008, pp. 639–646.
- [39] A. Rogers, A. Farinelli, R. Stranders, N. Jennings, Bounded approximate decentralised coordination via the max-sum algorithm, *Artificial Intelligence* 175 (2) (2011) 730–759.
- [40] E. Rollon, J. Larrosa, Improved Bounded Max-Sum for distributed constraint optimization, in: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2012, pp. 624–632.
- [41] R. Zivan, H. Peled, Max/min-sum distributed constraint optimization through value propagation on an alternating DAG, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012, pp. 265–272.

- [42] J. Pearce, M. Tambe, Quality guarantees on k-optimal solutions for distributed constraint optimization problems, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 1446–1451.
- [43] C. Kiekintveld, Z. Yin, A. Kumar, M. Tambe, Asynchronous algorithms for approximate distributed constraint optimization with quality bounds, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2010, pp. 133–140.
- [44] M. Vinyals, E. Shieh, J. Cerquides, J. Rodriguez-Aguilar, Z. Yin, M. Tambe, E. Bowring, Quality guarantees for region optimal DCOP algorithms, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2011, pp. 133–140.
- [45] R. Maheswaran, J. Pearce, M. Tambe, Distributed algorithms for DCOP: A graphical game-based approach, in: Proceedings of the International Conference on Parallel and Distributed Computing Systems (PDCS), 2004, pp. 432–439.
- [46] W. Zhang, G. Wang, Z. Xing, L. Wittenberg, Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks, *Artificial Intelligence* 161 (1–2) (2005) 55–87.
- [47] D. T. Nguyen, W. Yeoh, H. C. Lau, Distributed Gibbs: A memory-bounded sampling-based DCOP algorithm, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2013, pp. 167–174.
- [48] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (6) (1984) 721–741.
- [49] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, P. Varakantham, Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2004, pp. 310–317.
- [50] T. Grinshpoun, A. Grubshtein, R. Zivan, A. Netzer, A. Meisels, Asymmetric distributed constraint optimization problems, *Journal of Artificial Intelligence Research* 47 (2013) 613–647.
- [51] I. Brito, A. Meisels, P. Meseguer, R. Zivan, Distributed constraint satisfaction with partially known constraints, *Constraints* 14 (2) (2009) 199–234.
- [52] V. Levit, T. Grinshpoun, A. Meisels, A. L. Bazzan, Taxation search in boolean games, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2013, pp. 183–190.
- [53] D. A. Burke, K. N. Brown, M. Dogru, B. Lowe, Supply chain coordination through distributed constraint optimization, in: International Workshop on Distributed Constraint Reasoning (DCR), 2007.
- [54] S. D. Ramchurn, P. Vytelingum, A. Rogers, N. Jennings, Agent-based control for decentralised demand side management in the smart grid, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2011, pp. 5–12.
- [55] B. Lutati, V. Levit, T. Grinshpoun, A. Meisels, Congestion games for v2g-enabled ev charging, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2014, pp. 1440–1446.
- [56] K. Miettinen, *Nonlinear multiobjective optimization*, Vol. 12, Springer Berlin Heidelberg, 1999.
- [57] R. T. Marler, J. S. Arora, Survey of multi-objective optimization methods for engineering, *Structural and multidisciplinary optimization* 26 (6) (2004) 369–395.
- [58] M. E. Salukvad, Optimization of vector functionals. i. programming of optimal trajectories, *Automation and remote Control* 32 (8) (1971) 5–15.
- [59] A. Medi, T. Okimoto, K. Inoue, A two-phase complete algorithm for multi-objective distributed constraint optimization, *JACIII* 18 (4) (2014) 573–580.
- [60] T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, H. Matsuo, Distributed search method with bounded cost vectors on multiple objective dcops, in: Proceedings of the Principles and Practice of Multi-Agent Systems (PRIMA), 2012, pp. 137–152.
- [61] T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, H. Matsuo, Leximin multiple objective optimization for preferences of agents, in: Proceedings of the Principles and Practice of Multi-Agent Systems (PRIMA), 2014, pp. 423–438.
- [62] F. M. Delle Fave, R. Stranders, A. Rogers, N. R. Jennings, Bounded decentralised coordination over multiple objectives, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2011, pp. 371–378.
- [63] T. Okimoto, M. Clement, K. Inoue, Aof-based algorithm for dynamic multi-objective distributed constraint optimization, in: *Multi-disciplinary Trends in Artificial Intelligence (MIWAI)*, 2013, pp. 175–186.
- [64] T. Okimoto, N. Schwind, M. Clement, K. Inoue, Lp-norm based algorithm for multi-objective distributed constraint optimization Extended Abstract, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2014, pp. 1427–1428.
- [65] M. Wack, T. Okimoto, M. Clement, K. Inoue, Local search based approximate algorithm for multi-objective dcops, in: Proceedings of the Principles and Practice of Multi-Agent Systems (PRIMA), 2014, pp. 390–406.
- [66] L. Paquete, M. Chiarandini, T. Sttze, Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study, in: *Metaheuristics for Multiobjective Optimisation*, Vol. 535, 2004, pp. 177–199.
- [67] R. N. Lass, J. B. Kopena, E. A. Sultanik, D. N. Nguyen, C. P. Dugan, P. J. Modi, W. C. Regli, Coordination of first responders under communication and resource constraints, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2008, pp. 1409–1412.
- [68] M. Tambe, E. Bowring, H. Jung, G. Kaminka, R. Maheswaran, J. Marecki, P. J. Modi, R. Nair, S. Okamoto, J. P. Pearce, et al., Conflicts in teamwork: Hybrids to the rescue, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2005, pp. 3–10.
- [69] T. Okimoto, N. Ikegai, K. Inoue, H. Okada, T. Ribeiro, H. Maruyama, Cyber security problem based on multi-objective distributed constraint optimization technique, in: Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W), IEEE, 2013, pp. 1–7.
- [70] J. Atlas, K. Decker, Coordination for uncertain outcomes using distributed neighbor exchange, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2010, pp. 1047–1054.
- [71] R. Stranders, F. M. Delle Fave, A. Rogers, N. Jennings, U-gdl: A decentralised algorithm for dcops with uncertainty, Tech. rep., University of Southampton, Department of Electronics and Computer Science (2011).
- [72] D. T. Nguyen, W. Yeoh, H. C. Lau, Stochastic dominance in stochastic dcops for risk-sensitive applications, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2012, pp. 257–264.

- [73] T. Léauté, B. Faltings, E [dpop]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling, in: International Workshop on Distributed Constraint Reasoning (DCR), 2009, pp. 87–101.
- [74] T. Léauté, B. Faltings, Distributed constraint optimization under stochastic uncertainty., in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), Vol. 11, 2011, pp. 68–73.
- [75] Y. Wang, K. Sycara, P. Scerri, Towards an understanding of the value of cooperation in uncertain world, in: Proceedings of the International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (WI-IAT), Vol. 2, IEEE/WIC/ACM, 2011, pp. 212–215.
- [76] S. M. Aji, R. J. McEliece, The generalized distributive law, IEEE Transactions on Information Theory 46 (2) (2000) 325–343.
- [77] R. N. Lass, E. A. Sultanik, R. Greenstadt, W. C. Regli, Robust distributed constraint reasoning, in: International Workshop on Distributed Constraint Reasoning (DCR), 2009, pp. 75–86.
- [78] J. Atlas, K. Decker, Task scheduling using constraint optimization with uncertainty, Proceedings of CAPS 7 (2007) 25–27.
- [79] P. Scott, S. Thiébaux, M. Van Den Briel, P. Van Hentenryck, Residential demand response under uncertainty, in: Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP), Springer, 2013, pp. 645–660.
- [80] K. Turitsyn, N. Sinityn, S. Backhaus, M. Chertkov, Robust broadcast-communication control of electric vehicle charging, in: International Conference on Smart Grid Communications (SmartGridComm), IEEE, 2010, pp. 203–207.
- [81] M. E. Taylor, M. Jain, P. Tandon, M. Yokoo, M. Tambe, Distributed on-line multi-agent optimization under uncertainty: Balancing exploration and exploitation, Advances in Complex Systems 14 (03) (2011) 471–528.
- [82] M. E. Taylor, M. Jain, Y. Jin, M. Yokoo, M. Tambe, When should there be a me in team?: distributed multi-agent optimization under uncertainty, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2010, pp. 109–116.
- [83] R. Stranders, L. Tran-Thanh, F. M. D. Fave, A. Rogers, N. R. Jennings, DCOPs and bandits: Exploration and exploitation in decentralised coordination, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2012, pp. 289–296.
- [84] J. Vermorel, M. Mohri, Multi-armed bandit algorithms and empirical evaluation, in: Proceedings of the European Conference on Machine Learning (ECML), Springer, 2005, pp. 437–448.
- [85] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Machine learning 47 (2-3) (2002) 235–256.
- [86] F. Wu, N. R. Jennings, Regret-based multi-agent coordination with uncertain task rewards, arXiv preprint arXiv:1309.1973.
- [87] J. F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische mathematik 4 (1) (1962) 238–252.
- [88] K. Regan, C. Boutilier, Robust policy computation in reward-uncertain mdps using nondominated policies., in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2010, pp. 1127–1133.
- [89] J. B. Kopena, E. A. Sultanik, R. N. Lass, D. N. Nguyen, C. J. Dugan, P. J. Modi, W. C. Regli, Distributed coordination of first responders, Internet Computing 12 (1) (2008) 45–47.
- [90] S. Kozono, Received signal-level characteristics in a wide-band mobile radio channel, Vehicular Technology, IEEE Transactions on 43 (3) (1994) 480–486.
- [91] E. W. Dijkstra, Self-stabilization in spite of distributed control, Communication of the ACM 17 (11) (1974) 643–644.
- [92] G. Verfaillie, N. Jussien, Constraint solving in uncertain and dynamic environments: A survey, Constraints 10 (3) (2005) 253–281.
- [93] S. Dolev, T. Herman, Superstabilizing protocols for dynamic distributed systems, in: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing, ACM, 1995, p. 255.
- [94] A. Petcu, B. Faltings, Superstabilizing, fault-containing distributed combinatorial optimization., in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2005, pp. 449–454.
- [95] A. Petcu, B. Faltings, Optimal solution stability in dynamic, distributed constraint optimization, in: Proceedings of the International Conference on Intelligent Agent Technology (IAT), IEEE/WIC/ACM, 2007, pp. 321–327.
- [96] W. Yeoh, P. Varakantham, X. Sun, S. Koenig, Incremental dcop search algorithms for solving dynamic dcops (Extended Abstract), in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2011, pp. 1069–1070.
- [97] G. Billiau, C. F. Chang, A. Ghose, Sbdop: A new robust approach to dynamic distributed constraint optimisation, in: Proceedings of the Principles and Practice of Multi-Agent Systems (PRIMA), Springer Berlin Heidelberg, 2012, pp. 11–26.
- [98] P. Harvey, C. F. Chang, A. Ghose, Support-based distributed search: a new approach for multiagent constraint processing, in: Argumentation in Multi-Agent Systems, Springer Berlin Heidelberg, 2007, pp. 91–106.
- [99] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, N. R. Jennings, Decentralized coordination in robocup rescue, Computer Journal 53 (9) (2010) 1447–1461.
- [100] K. Macarthur, A. Farinelli, S. Ramchurn, N. Jennings, Efficient, superstabilizing decentralised optimisation for dynamic task allocation environments, in: International Workshop on Optimization In Multi-Agent Systems, 2010, pp. 25–32.
- [101] K. Macarthur, A. Farinelli, S. Ramchurn, N. Jennings, A distributed anytime algorithm for dynamic task allocation in multi-agent systems, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2011, pp. 701–706.
- [102] E. Sultanik, R. N. Lass, W. C. Regli, Dynamic configuration of agent organizations., in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2009, pp. 305–311.
- [103] D. T. Nguyen, W. Yeoh, H. C. Lau, S. Zilberstein, C. Zhang, Decentralized multi-agent reinforcement learning in average-reward dynamic dcops, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2014, pp. 1341–1342.
- [104] J. Abounadi, D. Bertsekas, V. Borkar, Learning algorithms for Markov decision processes with average cost, SIAM Journal on Control and Optimization 40 (3) (2001) 681–698.
- [105] A. Schwartz, A reinforcement learning method for maximizing undiscounted rewards, in: Proceedings of the International Conference on Machine Learning (ICML), 1993, pp. 298–305.
- [106] S. Mahadevan, Average reward reinforcement learning: Foundations, algorithms, and empirical results, Machine Learning 22 (1–3) (1996) 159–195.
- [107] R. Zivan, R. Glinton, K. Sycara, Distributed constraint optimization for large teams of mobile sensing agents, in: Proceedings of the International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (WI-IAT), IEEE/WIC/ACM, 2009, pp. 347–354.

- [108] H. Yedidsion, R. Zivan, A. Farinelli, Explorative max-sum for teams of mobile sensing agents, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014, pp. 549–556.
- [109] T. Matsui, H. Matsuo, M. C. Silaghi, K. Hirayama, M. Yokoo, S. Baba, A quantified distributed constraint optimization problem, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010, pp. 1023–1030.
- [110] M. Benedetti, A. Lallouet, J. Vautard, Quantified constraint optimization, in: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2008, pp. 463–477.
- [111] S. Baba, A. Iwasaki, M. Yokoo, M. C. Silaghi, K. Hirayama, T. Matsui, Cooperative problem solving against adversary: Quantified distributed constraint satisfaction problem, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010, pp. 781–788.
- [112] S. Baba, Y. Joe, A. Iwasaki, M. Yokoo, Real-time solving of quantified cps based on monte-carlo game tree search, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, pp. 655–661.
- [113] C. J. Carpenter, C. J. Dugan, J. B. Kopena, R. N. Lass, G. Naik, D. N. Nguyen, E. Sultanik, P. J. Modi, W. C. Regli, Intelligent systems demonstration: disaster evacuation support, in: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2007, pp. 1964–1965.
- [114] R. N. Lass, W. C. Regli, A. Kaplan, M. Mitkus, J. J. Sim, Facilitating communication for first responders using dynamic distributed constraint optimization, in: *Proceedings of the Symposium on Technologies for Homeland Security (IEEE HST)*, IEEE, 2008, pp. 316–320.
- [115] K. Kinoshita, K. Iizuka, Y. Iizuka, Effective disaster evacuation by solving the distributed constraint optimization problem, in: *Proceedings of the International Conference on Advanced Applied Informatics (IIAIAI)*, IEEE, 2013, pp. 399–400.
- [116] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, J. A. Rodriguez-Aguilar, Efficient inter-team task allocation in robocup rescue, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015, pp. 413–421.
- [117] G. Steinbauer, A. Kleiner, Towards csp-based mission dispatching in c2/c4i systems, in: *Safety, Security, and Rescue Robotics (SSRR)*, 2012 IEEE International Symposium on, IEEE, 2012, pp. 1–6.
- [118] D. Hollos, H. Karl, A. Wolisz, Regionalizing global optimization algorithms to improve the operation of large ad hoc networks, in: *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, Vol. 2, IEEE, 2004, pp. 819–824.
- [119] T. L. Monteiro, M. E. Pellenz, M. C. Penna, F. Enembreck, R. D. Souza, G. Pujolle, Channel allocation algorithms for wlans using distributed optimization, *AEU-International Journal of Electronics and Communications* 66 (6) (2012) 480–490.
- [120] T. L. Monteiro, G. Pujolle, M. E. Pellenz, M. C. Penna, F. Enembreck, R. Demo Souza, A multi-agent approach to optimal channel assignment in wlans, in: *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, IEEE, 2012, pp. 2637–2642.
- [121] J. Xie, I. Howitt, A. Raja, Cognitive radio resource management using multi-agent systems, in: *Proceedings of the Consumer Communications and Networking Conference (CCNC)*, IEEE, 2007, pp. 1123–1127.
- [122] U. Mir, L. Merghem-Boulahia, D. Gaïti, A cooperative multiagent based spectrum sharing, in: *Advanced International Conference on Telecommunications (AICT)*, IEEE, 2010, pp. 124–130.
- [123] F. Lorenzi, F. dos Santos, P. R. Ferreira Jr, A. L. Bazzan, Optimizing preferences within groups: A case study on travel recommendation, in: *Advances in Artificial Intelligence (SBIA)*, 2008, pp. 103–112.
- [124] B. Giffler, G. L. Thompson, Algorithms for solving production-scheduling problems, *Operations research* 8 (4) (1960) 487–503.
- [125] M. M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, *Operations research* 35 (2) (1987) 254–265.
- [126] S. Minton, M. D. Johnston, A. B. Philips, P. Laird, Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems, *Artificial Intelligence* 58 (1) (1992) 161–205.
- [127] P. V. Hentenryck, L. Michel, *Constraint-based local search*, The MIT Press, 2009.
- [128] J. Kurose, E. Lyons, D. McLaughlin, D. Pepyne, B. Philips, D. Westbrook, M. Zink, An end-user-responsive sensor network architecture for hazardous weather detection, prediction and response, in: *In Proceedings of the Second Asian Internet Engineering Conference*, Springer, 2006, pp. 1–15.
- [129] Y. Kim, M. Krainin, V. Lesser, Application of max-sum algorithm to radar coordination and scheduling, in: *International Workshop on Distributed Constraint Reasoning (DCR)*, 2010.
- [130] E. P. Vargo, E. J. Bass, R. Cogill, Belief propagation for large-variable-domain optimization on factor graphs: An application to decentralized weather-radar coordination, *Systems, Man, and Cybernetics: Systems*, IEEE Transactions on 43 (2) (2013) 460–466.
- [131] L. Garrido, K. Sycara, Multi-agent meeting scheduling: Preliminary experimental results, in: *Proceedings of the Second International Conference on Multiagent Systems*, 1996, pp. 95–102.
- [132] N. R. Jennings, A. Jackson, Agent-based meeting scheduling: A design and implementation, *Electronics letters* 31 (5) (1995) 350–352.
- [133] M. Giuliani, A. Castelletti, F. Amigoni, X. Cai, Multiagent systems and distributed constraint reasoning for regulatory mechanism design in water management, *Journal of Water Resources Planning and Management*.
- [134] F. Amigoni, A. Castelletti, M. Giuliani, Modeling the management of water resources systems using multi-objective dcops, in: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 821–829.
- [135] M. Hannebauer, S. Müller, Distributed constraint optimization for medical appointment scheduling, in: *Proceedings of the fifth international conference on Autonomous agents*, ACM, 2001, pp. 139–140.
- [136] G. Billiau, C. F. Chang, A. Ghose, A. A. Miller, Using distributed agents for patient scheduling, in: *Proceedings of the Principles and Practice of Multi-Agent Systems (PRIMA)*, 2012, pp. 551–560.
- [137] G. Billiau, C. F. Chang, A. Ghose, A. Miller, Support-based distributed optimisation: An approach to radiotherapy scheduling, in: *Electronic Healthcare*, 2012, pp. 327–334.
- [138] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, *Computer networks* 38 (4) (2002) 393–422.
- [139] T. Matsui, H. Matsuo, A formalization for distributed cooperative sensor resource allocation, in: *Agent and Multi-Agent Systems: Technologies and Applications*, 2008, pp. 292–301.
- [140] M. Jain, M. E. Taylor, M. Tambe, M. Yokoo, Dcops meet the real world: Exploring unknown reward matrices with applications to mobile

- sensor networks., in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Vol. 9, 2009, pp. 181–186.
- [141] K. Ota, T. Matsui, H. Matsuo, Layered distributed constraint optimization problem for resource allocation problem in distributed sensor networks, in: Proceedings of the Principles and Practice of Multi-Agent Systems (PRIMA), 2009, pp. 245–260.
 - [142] R. Stranders, A. Farinelli, A. Rogers, N. R. Jennings, Decentralised coordination of continuously valued control parameters using the max-sum algorithm, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2009, pp. 601–608.
 - [143] S. Hosseini Semnani, O. A. Basir, Target to sensor allocation: A hierarchical dynamic distributed constraint optimization approach, *Computer Communications* 36 (9) (2013) 1024–1038.
 - [144] H. Choxi, P. Modi, A distributed constraint optimization approach to wireless network optimization, in: Proceedings of the AAAI-07 Workshop on Configuration, 2007, pp. 1–8.
 - [145] A. F. Molisch, *Wireless Communications*, Vol. 34, John Wiley & Sons, 2012.
 - [146] H. Yedidsion, R. Zivan, Applying dcop_mst to a team of mobile robots with directional sensing abilities, in: International Joint Workshop on Optimization In Multi-Agent Systems and Distributed Constraint Reasoning (OPTMAS-DCR), 2014.
 - [147] S. Chachra, M. Marefat, Distributed algorithms for sleep scheduling in wireless sensor networks, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2006, pp. 3101–3107.
 - [148] R. Moreno-Vozmediano, R. S. Montero, I. M. Llorente, Key challenges in cloud computing: Enabling the future internet of services, *Internet Computing* 17 (4) (2013) 18–25.
 - [149] B. Mejias, P. V. Roy, From mini-clouds to cloud computing, in: International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW), IEEE, 2010, pp. 234–238.
 - [150] Z. Jin, J. Cao, M. Li, A distributed application component placement approach for cloud computing environment, in: Proceedings of the International Conference on Dependable, Autonomic and Secure Computing (DASC), IEEE, 2011, pp. 488–495.
 - [151] X. Li, H. Wang, B. Ding, X. Li, Mabp: an optimal resource allocation approach in data center networks, *Science China Information Sciences* 57 (10) (2014) 1–16.
 - [152] B. Choudhury, P. Dey, A. Dutta, S. Choudhury, A multi-agent based optimised server selection scheme for soc in pervasive environment, in: Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection, 2014, pp. 50–61.
 - [153] Z. Yan, J.-H. Lee, S. Shen, C. Qiao, Novel branching-router-based multicast routing protocol with mobility support, *Parallel and Distributed Systems, IEEE Transactions on* 24 (10) (2013) 2060–2068.
 - [154] E. Davidson, M. Dolan, S. McArthur, G. Ault, The use of constraint programming for the autonomous management of power flows, in: Intelligent System Applications to Power Systems, 2009. ISAP'09. 15th International Conference on, IEEE, 2009, pp. 1–7.
 - [155] S. D. Ramchurn, P. Vytelingum, A. Rogers, N. R. Jennings, Putting the 'smarts' into the smart grid: A grand challenge for artificial intelligence, *Commun. ACM* 55 (4) (2012) 86–97.
 - [156] S. Miller, S. D. Ramchurn, A. Rogers, Optimal decentralised dispatch of embedded generation in the smart grid, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2012, pp. 281–288.
 - [157] T. Matsui, H. Matsuo, A distributed cooperative model for resource supply networks, in: Proceedings of the International MultiConference of Engineers and Computer Scientists, Vol. 1, 2011, pp. 7–14.
 - [158] P. Jain, S. J. Ranade, S. Gupta, E. Pontelli, Optimum operation of a customer-driven microgrid: A comprehensive approach, in: Power Electronics, Drives and Energy Systems (PEDES), 2012 IEEE International Conference on, IEEE, 2012, pp. 1–6.
 - [159] A. Kumar, B. Faltings, A. Petcu, Distributed constraint optimization with structured resource constraints, in: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 923–930.
 - [160] A. Gupta, Network management: Current trends and future perspectives, *Journal of Network and Systems Management* 14 (4) (2006) 483–491.
 - [161] S. Gupta, P. Jain, W. Yeoh, S. Ranade, E. Pontelli, Solving customer-driven microgrid optimization problems as dcops, in: Proc. of the Distributed Constraint Reasoning Workshop, 2013, pp. 45–59.
 - [162] D. Athanasiadis, I. Kockar, S. McArthur, Distributed constraint optimisation for flexible network management, in: Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES, IEEE, 2013, pp. 1–5.
 - [163] P. Agrawal, A. Kumar, P. Varakantham, Near-optimal decentralized power supply restoration in smart grids, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1275–1283.
 - [164] S. Gupta, W. Yeoh, E. Pontelli, P. Jain, S. Ranade, Modeling microgrid islanding problems as dcops, in: North American Power Symposium (NAPS), 2013, pp. 1–6.
 - [165] J. Cerquides, G. Picard, J. A. Rodríguez-Aguilar, Designing a marketplace for the trading and distribution of energy in the smart grid, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1285–1293.
 - [166] J. Gaudreault, J.-M. Frayret, G. Pesant, Distributed search for supply chain coordination, *Computers in Industry* 60 (6) (2009) 441–451.
 - [167] T. Peña-Alba, J. Cerquides, J. A. Rodríguez-Aguilar, M. Vinyals, Scalable decentralized supply chain formation through binarized belief propagation, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2012, pp. 1275–1276.
 - [168] T. Peña-Alba, M. Vinyals, J. Cerquides, J. A. Rodríguez-Aguilar, A scalable message-passing algorithm for supply chain formation, in: AAAI, 2012, pp. 1436–1442.
 - [169] M. Winsper, M. Chli, Decentralized supply chain formation using max-sum loopy belief propagation, *Computational Intelligence* 29 (2) (2013) 281–309.
 - [170] T. Peña-Alba, M. Vinyals, J. Cerquides, J. A. Rodríguez-Aguilar, Exploiting max-sum for the decentralized assembly of high-valued supply chains, in: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2014, pp. 373–380.
 - [171] W. E. Walsh, M. P. Wellman, Modeling supply chain formation in multiagent systems, in: Agent mediated electronic commerce II, Springer,

- 2000, pp. 94–101.
- [172] M. Chli, M. Winsper, Using the max-sum algorithm for supply chain emergence in dynamic multiunit environments, *IEEE T. Systems, Man, and Cybernetics: Systems* 45 (3) (2015) 422–435.
 - [173] R. Van Katwijk, B. De Schutter, J. Hellendoorn, Multi-agent control of traffic networks: Algorithm and case study, in: *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2009, pp. 1–6.
 - [174] D. de Oliveira, A. L. Bazzan, V. Lesser, Using cooperative mediation to coordinate traffic lights: a case study, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2005, pp. 463–470.
 - [175] R. Junges, A. L. Bazzan, Evaluating the performance of dcop algorithms in a real world, dynamic problem, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008, pp. 599–606.
 - [176] T. Pham, A. Tawfik, M. Taylor, A simple, naive agent-based model for the optimization of a system of traffic lights: Insights from an exploratory experiment, in: *Proceedings of the conference on agent-based modeling in transportation planning and operations*, 2013, pp. 1–21.
 - [177] T. Brys, T. T. Pham, M. E. Taylor, Distributed learning and multi-objectivity in traffic light control, *Connection Science* 26 (1) (2014) 65–83.
 - [178] P. M. Berry, C. Albright, E. Bowring, K. Conley, K. Nitz, J. P. Pearce, B. Peintner, S. Saadati, M. Tambe, T. Uribe, et al., Conflict negotiation among personal calendar agents (Extended Abstract), in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006, pp. 1467–1468.
 - [179] S. Khanna, A. Sattar, D. Hansen, B. Stantic, An efficient algorithm for solving dynamic complex dcop problems, in: *Proceedings of the International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (WI-IAT)*, Vol. 2, IEEE/WIC/ACM, 2009, pp. 339–346.
 - [180] S. Khanna, A. Sattar, J. Boyle, D. Hansen, B. Stantic, An intelligent approach to surgery scheduling, in: *Proceedings of the Principles and Practice of Multi-Agent Systems (PRIMA)*, 2012, pp. 535–550.
 - [181] A. Chapman, A. Rogers, N. Jennings, A parameterisation of algorithms for distributed constraint optimisation via potential games (2008) 99–113.
 - [182] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. N. Prasad, A. Raja, et al., Evolution of the gpgp/taems domain-independent coordination framework, *Journal of Autonomous Agents and Multi-Agent Systems* 9 (1-2) (2004) 87–143.
 - [183] C. Zhang, V. Lesser, Coordinating multi-agent reinforcement learning with limited communication, in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013, pp. 1101–1108.
 - [184] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, Determining computational complexity from characteristic phase transitions, *Nature* 400 (6740) (1999) 133–137.
 - [185] D. Burke, K. Brown, Efficiently handling complex local problems in distributed constraint optimisation, in: *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 2006, pp. 701–702.
 - [186] M. Yokoo (Ed.), *Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems*, Springer Berlin Heidelberg, 2001.
 - [187] L. Lamport, Time, clocks, and the ordering of events in a distributed system, *Communications of the ACM* 21 (7) (1978) 558–565.
 - [188] E. Sultanik, P. J. Modi, W. C. Regli, On modeling multiagent task scheduling as a distributed constraint optimization problem., in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 1531–1536.
 - [189] N. A. Lynch, *Distributed algorithms*, Morgan Kaufmann, 1996.
 - [190] CSPLib: A problem library for constraints, <http://www.csplib.org/>.
 - [191] ICAPS international planning competition, <http://icaps-conference.org/index.php/Main/Competitions>.