Regret-Based Multi-Agent Coordination with Uncertain Task Rewards

Feng Wu
School of Electronics and Computer Science
University of Southampton, United Kingdom
fw6e11@ecs.soton.ac.uk

Nicholas R. Jennings
School of Electronics and Computer Science
University of Southampton, United Kingdom
nrj@ecs.soton.ac.uk

Abstract

Many multi-agent coordination problems can be represented as DCOPs. Motivated by task allocation in disaster response, we extend standard DCOP models to consider *uncertain* task rewards where the outcome of completing a task depends on its current state, which is randomly drawn from *unknown* distributions. The goal of solving this problem is to find a solution for all agents that minimizes the overall worst-case loss. This is a challenging problem for centralized algorithms because the search space grows exponentially with the number of agents and is nontrivial for existing algorithms for standard DCOPs. To address this, we propose a novel decentralized algorithm that incorporates Max-Sum with iterative constraint generation to solve the problem by passing messages among agents. By so doing, our approach scales well and can solve instances of the task allocation problem with hundreds of agents and tasks.

Introduction

Distributed constraint optimization problems (DCOPs) are a popular representation for many multi-agent coordination problems. In this model, agents are represented as decision variables and the tasks that they can be assigned to are variable domains. The synergies between the agents' (joint) assignment are specified as constraint values. Now, some tasks may require a subgroup of the team to work together, either because a single agent has insufficient capabilities to complete the task or the teamwork can substantially improve performance. In either case, the constraints are the utilities of the agents' joint behaviors. Once the DCOP model of the problem is obtained, we can solve it efficiently using optimal methods such as ADOPT (Modi et al. 2005) and DPOP (Petcu and Faltings 2005) or approximate approaches such as DSA (Zhang et al. 2005), MGM (Maheswaran, Pearce, and Tambe 2004), or Max-Sum (Farinelli et al. 2008; Stranders et al. 2009; Rogers et al. 2011).

In DCOPs, the task rewards are often assumed to be completely known to the agents. However, this can make it difficult to model problems where the reward for completing a task depends on the task state, which is usually unobservable and uncontrollable by the agents. For example, in disaster response, a group of first responders may be sent out to an unknown area to search for survivors. However, the success of the search tasks (task rewards) will depend on many factors (task states) such as the local terrain, the weather condition, and the degree of damage in the search area. Initially, the responders may have very limited information about the task states, but must act quickly because time is critical for saving lives. In such cases, it is desirable to reason about the uncertainty of the task states (rewards) and assign the tasks to the agents in such a way that the worst-case loss (compared to the unknown optimal solution) is minimized. The aim of this is to perform as closely as possible to the optimal solution given the uncertain task rewards (caused by *unknown* task states).

Over recent years, a significant body of research has dealt with extending standard DCOPs to models with uncertainty. A common method is to introduce additional random variables (uncontrollable by the agents) to the constraint functions (Léauté and Faltings 2011). Another way to reason about the uncertainty is to randomly select a constraint function from a predefined function set (Atlas and Decker 2010; Stranders et al. 2011; Nguyen, Yeoh, and Lau 2012). However, most of the aforementioned approaches require the probability distributions of the random variables to be known (Atlas and Decker 2010) or the candidate functions to have certain properties (e.g., be Gaussian (Stranders et al. 2011) or concave (Nguyen, Yeoh, and Lau 2012)). Unfortunately, these assumptions are not common in our motivating domain because the agents have no or very limited information about the tasks as they start to respond to the crisis. In distributed settings, there are approaches that also consider robust optimization under uncertainty (Matsui et al. 2010; Léauté and Faltings 2011). However, they use a *maximin* strategy that could be overly pessimistic (e.g., the responders will decide to do nothing because all the tasks may fail in the worst case). Instead, we adopt *minimax* regret that usually offers more reasonable solutions in our problems (i.e., the responders must try their best even in the worst case). [1] Thus, the key challenge is to find a good solution (as close to the optimal as possible) given no or partial information about the associated task states (linked to the rewards).

To this end, we introduce a new model for multi-agent

[1]More discussion on using *minimax* regret as a criterion for decision making with utility uncertainty is in (Boutilier et al. 2006).

coordination with uncertain task rewards and propose an efficient algorithm for computing the *robust* solution (minimizing the worst-case loss) of this model. Our model, called *uncertain* reward DCOP (UR-DCOP), extends the standard DCOP to include random variables (task states), one for each constraint (task). We assume these random variables are independent from each other (the tasks are independent) and uncontrollable by the agents (e.g., the weather condition or unexpected damage). Furthermore, we assume the choice of these variables are drawn from finite domains with unknown distributions. For each such variable, we define a belief as a probability distribution over its domain. Thus, minimizing the worst-case loss is equivalent to computing the *minimax* regret solution in the joint belief space. Intuitively, this process can be viewed as a game between the agents and nature where the agents select a solution to minimize the loss, while nature chooses a belief in the space to maximize it.

For large UR-DCOPs with hundreds of agents and tasks, it is intractable for a centralized solver to compute the minimax regret solution for all the agents due to the huge joint belief and solution space. Thus, we turn to decentralized approaches because they can exploit the interaction structure and distribute the computation locally to each agent. However, it is challenging to compute the minimax regret in a decentralized manner because intuitively all the agents need to find the worst case (a point in the belief space) before they can minimize the loss. To address this, we borrow ideas from *iterative constraint generation*, first introduced by (Benders 1962) and recently adopted by (Regan and Boutilier 2010; 2011) for solving imprecise MDPs. Similar to their approaches, we decompose the overall problem into a *master* problem and a *subproblem* that are iteratively solved until they converge. The main contribution of our work lies in the development of two variations of Max-Sum to solve the master and sub-problems by passing messages among the agents. We adopt Max-Sum due to its performance and stability on large problems (i.e., hundreds of agents). We prove that our algorithm is optimal for acyclic factor graphs and error-bounded for cyclic graphs. In experiments, we show that our method can scale up to task allocation domains with hundreds of agents and tasks (intractable for centralized methods) and can outperform state-of-the-art decentralized approaches by having much higher values and lower regrets.

The UR-DCOP Model

Formally, a *distributed constraint optimization problem* (DCOP) can be defined as a tuple $\mathcal{M} = \langle \mathcal{I}, \mathcal{X}, \mathcal{D}, \mathcal{U} \rangle$, where:

- $\mathcal{I} = \{1, \cdots, n\}$ is a set of agents indexed by $1, 2, \cdots, n$;
- $\mathcal{X} = \{x_1, \cdots, x_n\}$ is a set of *decision variables* where x_i denotes the variable controlled by agent i;
- $\mathcal{D} = \{D_1, \cdots, D_n\}$ is a set of finite domains for the decision variables where domain D_i is a set of possible values for decision variable x_i;
- $\mathcal{U} = \{U_1, \cdots, U_m\}$ is a set of *soft* constraints where each constraint $U_j : D_{j1} \times \cdots \times D_{jk} \to \Re$ defines the value of possible assignments to subsets of decision variables where $U_j(x_{j1}, \cdots, x_{jk})$ is the value function for variables $x_{j1}, \cdots, x_{jk} \in \mathcal{X}$.

The goal of solving a DCOP is to find an assignment $\mathbf{x}^*$ of values in the domains of all decision variables $x_i \in \mathcal{X}$ that maximizes the sum of all constraints:

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}} \sum_{j=1}^{m} U_j(x_{j1}, \cdots, x_{jk}) \tag{1}$$

Turning to Max-Sum, this is a decentralized message-passing optimization approach for solving large DCOPs. To use Max-Sum, a DCOP needs to be encoded as a special bipartite graph, called a factor graph, where vertices represent variables x_i and functions U_j, and edges the dependencies between them. Specifically, it defines two types of messages that are exchanged between variables and functions:

- From variable x_i to function U_j:

$$q_{i\to j}(x_i) = \alpha_{i\to j} + \sum_{k\in M(i)\setminus j} r_{k\to i}(x_i) \tag{2}$$

  where $M(i)$ denotes the set of indices of the function nodes connected to variable x_i and $\alpha_{i\to j}$ is a scaler chosen such that $\sum_{x_i\in D_i} q_{i\to j}(x_i) = 0$.

- From function U_j to variable x_i:

$$r_{j\to i}(x_i) = \max_{\mathbf{x_j}\setminus x_i} \left[U_j(\mathbf{x_j}) + \sum_{k\in N(j)\setminus i} q_{k\to j}(x_k) \right] \tag{3}$$

  where $N(j)$ denotes the set of indices of the variable nodes connected to U_j and $\mathbf{x_j}$ is a variable vector $\langle x_{j1}, \cdots, x_{jk} \rangle$.

Notice that both $q_{i\to j}(x_i)$ and $r_{j\to i}(x_i)$ are scalar functions of variable $x_i \in D_i$. Thus, the marginal function of each variable x_i can be calculated by:

$$z_i(x_i) = \sum_{j\in M(i)} r_{j\to i}(x_i) \approx \max_{\mathbf{x}\setminus x_i} \sum_{j=1}^{m} U_j(\mathbf{x_j}) \tag{4}$$

after which the assignment of x_i can be selected by:

$$x_i^* = \operatorname*{argmax}_{x_i\in D_i} z_i(x_i) \tag{5}$$

From this background, we now turn to the UR-DCOP model itself. In particular, our work is mainly motivated by the task allocation problem in disaster response scenarios [2], where a group of first responders need to be assigned to a set of tasks in order to maximize saved lives. This problem can be straightforwardly modeled as a DCOP where: $\mathcal{I}$ is a set of first responders, x_i is the task assigned to responder i, D_i is a set of tasks that can be performed by responder i, and U_j is the reward for the completion of task j. However, in our domains, the value of U_j depends both on the joint choice of the agents and on the uncontrollable events such as fires, hurricanes, floods, or debris flows in the disaster area. These events can be formally abstracted as task states, which are usually unknown to the first responders, but critical for the

[2]Nevertheless, our results are broadly applicable to other domains that can be modeled as a UR-DCOP.

team performance. To model this, we introduce UR-DCOP — a new representation for multi-agent coordination with uncertain task rewards.

In more detail, UR-DCOP is an extension of the original DCOP model with two additional components:

- $\mathcal{E} = \{s_1, \cdots, s_m\}$ is a set of *random variables* modeling uncontrollable stochastic events, e.g., fires in a building or weather in a disaster area, for each constraint $U_j \in \mathcal{U}$;
- $\mathcal{S} = \{S_1, \cdots, S_m\}$ is a set of finite domains, e.g., levels of the fire damage or different weather conditions, for each random variable $s_j \in S_j$;

The value functions are augmented to consider both decision variables and random variables (task states), i.e., $U_j(s_j; x_{j1}, \cdots, x_{jk})$. We assume each value function only associates with one random variable. If multiple random variables are associated with a value function, without loss of generality, they can be merged into a single variable. Furthermore, we assume the random variables are not under the control of the agents and they are *independent* of the decision variables. Specifically, their values are *independently* drawn from unknown probability distributions.

Given a random variable s_j in UR-DCOPs, the probability distribution over domain S_j, denoted by $b_j \in \Delta(S_j)$, is called a *belief* of the random variable, and $\mathbf{b} = \langle b_1, \cdots, b_m \rangle$ is a *joint belief* of all random variables. Similarly, a *joint assignment* of all decision variables is denoted by $\mathbf{x} = \langle x_1, \cdots, x_n \rangle$ and a *partial joint assignment* for the value function U_j is denoted by $\mathbf{x_j} = \langle x_{j1}, \cdots, x_{jk} \rangle$. When the joint belief $\mathbf{b}$ is known, solving a UR-DCOP straightforwardly involves finding an assignment of all decision variables $\mathbf{x}$ that maximize the sum of the *expected* values:

$$V(\mathbf{b}, \mathbf{x}) = \sum_{j=1}^{m} \underbrace{\sum_{s_j \in S_j} b_j(s_j) U_j(s_j, \mathbf{x_j})}_{U_j(b_j, \mathbf{x_j})} \tag{6}$$

The key challenge in our problem is that the joint belief $\mathbf{b}$ is unknown. Therefore, we want to find a solution that is *robust* (minimizing the worst-case loss) to the uncertainty of the joint belief. As mentioned earlier, this objective is equivalent to the *minimax regret* given the belief space $\mathcal{B}$:

$$V_{regret} = \min_{\mathbf{x}'} \underbrace{\max_{\mathbf{b} \in \mathcal{B}} \underbrace{\max_{\mathbf{x}^*} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x}')]}_{R_1(\mathbf{x}', \mathbf{b})}}_{R_2(\mathbf{x}')} \tag{7}$$

where $\mathbf{x}^*$ is the optimal solution given belief $\mathbf{b}$. $R_1(\mathbf{x}', \mathbf{b})$ is the regret or loss of solution $\mathbf{x}$ relative to $\mathbf{b}$, i.e., the difference in expected value between $\mathbf{x}$ and the optimal solution $\mathbf{x}^*$ under belief $\mathbf{b}$. $R_2(\mathbf{x}')$ is the maximum regret of $\mathbf{x}$ with respect to the feasible belief space. Thus, the value of minimax regret, V_{regret}, minimizes the worst-case loss over all possible belief points.

As mentioned, first responders usually have very limited information about the response tasks when the disaster happens and there is significant uncertainty in the environment. In such cases, *minimax regret* minimizes the difference between the optimal value $V(\mathbf{b}, \mathbf{x}^*)$ and the actual value $V(\mathbf{b}, \mathbf{x})$ achieved by the current solution $\mathbf{x}$ in all possible beliefs $\mathbf{b} \in \mathcal{B}$. Thus, it is a good solution for the first responders given the limited information.

Solving UR-DCOPs

Generally, to compute the minimax regret in Equation 7, we first need to compute the optimal solution $\mathbf{x}^*$ given a belief point $\mathbf{b}$ and the current solution of agents $\mathbf{x}'$. Then, the whole belief space $\mathcal{B}$ is searched to find the worst-case belief $\mathbf{b}$. After that, we need to find the assignment $\mathbf{x}$ that minimizes the regret. On the one hand, it cannot be solved by standard DCOP algorithms given the uncertain rewards in our model. On the other hand, given a number of agents, it is very challenging for centralized algorithms to compute the minimax regret because the search space blows up exponentially with the number of agents.

Following the ideas of *iterative constraint generation* (ICG), two optimizations are alternatively solved at each iteration: the *master problem* and the *subproblem*. In more detail, the *master* problem solves a relaxation of Equation 7 by considering only a subset of all possible $\langle \mathbf{b}, \mathbf{x}^* \rangle$ pairs $\mathcal{G}$:

$$\begin{array}{ll} \min_{\mathbf{x},\delta} & \delta \\ \text{s.t.} & \forall \langle \mathbf{b}, \mathbf{x}^* \rangle \in \mathcal{G}, V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x}) \leq \delta \end{array} \tag{8}$$

Initially, this set can be arbitrary (e.g., empty or randomly generated). By giving $\mathcal{G}$, the master problem tries to minimize the loss for the worst case derived from $\mathcal{G}$.

The *subproblem* generates the maximally violated constraint relative to $\mathbf{x}$, the solution of the current master problem. More precisely, a new $\langle \mathbf{b}, \mathbf{x}^* \rangle$ pair is found by the subproblem. This pair is called a *witness* point because it indicates that the current $\mathbf{x}$ is not the best solution in terms of the minimax regret. In more detail, a program is solved to determine the witness $\langle \mathbf{b}, \mathbf{x}^* \rangle$ for the current solution $\mathbf{x}$:

$$\begin{array}{ll} \max_{\mathbf{b},\mathbf{x}^*,\delta'} & \delta' \\ \text{s.t.} & V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x}) \geq \delta' \end{array} \tag{9}$$

If $\delta' = \delta$ then the constraint for $\langle \mathbf{b}, \mathbf{x}^* \rangle$ in Equation 9 is satisfied at the current solution $\mathbf{x}$, and indeed all unexpressed constraints must be satisfied as well. Otherwise, $\delta' > \delta$, implying that the constraint for $\langle \mathbf{b}, \mathbf{x}^* \rangle$ is violated in the current relaxation. Thus, it is added to $\mathcal{G}$ and the master problem is solved again to compute a new $\mathbf{x}$. This process repeats until no new witness point can be found by the subproblem and the master problem terminates with the best solution $\mathbf{x}$.

Based on the ideas of ICG, we propose *iterative constraint generation Max-Sum* (ICG-Max-Sum) to solve UR-DCOPs. Similar to standard Max-Sum, our algorithm starts with encoding UR-DCOPs into a factor graph. Then, two Max-Sum algorithms are iteratively executed to solve the master and sub-problems. In the master problem, we run a Max-Sum to compute the current minimax solution $\mathbf{x}$ and minimax regret δ given the witness set $\mathcal{G}$. In the subproblem, we run another Max-Sum to generate a new witness point $\langle \mathbf{b}, \mathbf{x}^* \rangle$ and the corresponding minimax regret δ' given the current solution $\mathbf{x}$. Then, δ and δ' are compared by each

Algorithm 1: Iterative Constraint Generation Max-Sum

```
Input: $\mathcal{M}$: The UR-DCOP Model
1  Create a factor graph based on $\mathcal{M}$
2  Initialize the witness set $\mathcal{G} \leftarrow \emptyset$
3  repeat
       // The Master Problem
4      Run Max-Sum on the factor graph with $\mathcal{G}$
5      Compute the current minimax solution $\mathbf{x}$
6      Save each $x_i \in \mathbf{x}$ in variable node $i$
7      Compute the minimax regret $\delta$
       // The Subproblem
8      Run Max-Sum on the factor graph with $\mathbf{x}$
9      Compute the witness point $\langle \mathbf{b}, \mathbf{x}^* \rangle$
10     Compute the minimax regret $\delta'$
       // $\mathcal{G} \leftarrow \mathcal{G} \cup \{\langle \mathbf{b}, \mathbf{x}^* \rangle\}$ if $\delta' > \delta$
11     foreach variable node $i$ do
12         if $\delta' > \delta$ then
13             Save $x_i^* \in \mathbf{x}^*$ in variable node $i$
14         else Terminate variable node $i$
15     foreach function node $j$ do
16         if $\delta' > \delta$ then
17             Save $b_j \in \mathbf{b}$ in function node $j$
18         else Terminate function node $j$
19 until all nodes in the graph are terminated.
20 return the current minimax solution $\mathbf{x}$
```

node in the factor graph: If $\delta > \delta'$, the newly generated witness point $\langle \mathbf{b}, \mathbf{x}^* \rangle$ is added to $\mathcal{G}$; otherwise it terminates and returns the current minimax solution $\mathbf{x}$. These processes repeat until all nodes in the factor graph are terminated. Notice that in our algorithm the solutions $x_i \in \mathbf{x}$ and $x_i^* \in \mathbf{x}^*$ are computed and stored locally by variable i and belief $b_j \in \mathbf{b}$ is computed and stored locally by function j. The main procedures are shown in Algorithm 1.

The Master Problem

The *master* problem of Equation 8, given the witness set $\mathcal{G}$, can be equivalently written as:

$$\mathbf{x} = \arg\min_{\mathbf{x}'} \underbrace{\max_{\langle \mathbf{b}, \mathbf{x}^* \rangle \in \mathcal{G}} \sum_{j=1}^{m} [U_j(b_j, \mathbf{x}_\mathbf{j}^*) - U_j(b_j, \mathbf{x}_\mathbf{j}')]}_{\delta} \quad (10)$$

Note that the witness set $\mathcal{G}$ is known and the choice of $\langle b_j, \mathbf{x}_\mathbf{j}^* \rangle$ can be computed locally by function node j in Max-Sum because b_j is independent from other belief points and $\mathbf{x}_\mathbf{j}^*$ is only related to the variable nodes it connects. To do this, we consider the problem of minimizing a vector of regret functions for each witness point in $\mathcal{G}$:

$$\mathbb{V}(\mathbf{x}) = \left[\tilde{V}(\mathbf{x}, \langle \mathbf{b}, \mathbf{x}^* \rangle_1), \cdots, \tilde{V}(\mathbf{x}, \langle \mathbf{b}, \mathbf{x}^* \rangle_{|\mathcal{G}|}) \right] \quad (11)$$

where $\tilde{V}(\mathbf{x}, \langle \mathbf{b}, \mathbf{x}^* \rangle_g) = V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})$ and $\langle \mathbf{b}, \mathbf{x}^* \rangle_g$ is the gth element in $\mathcal{G}$. Accordingly, instead of $q_{j \to i}(x_i)$ and $r_{i \to j}(x_i)$ being scalar functions of x_i, these messages now map the domain of x_i to a set of regret vectors: $\forall x_i \in D_i, q_{j \to i}(x_i) = [q_1, \cdots, q_{|\mathcal{G}|}], r_{i \to j}(x_i) = [r_1, \cdots, r_{|\mathcal{G}|}]$.

To compute these messages, the two key operators required by Max-Sum (Equations 2 and 3) need to be redefined. In more detail, the operation of adding two messages is defined by adding each corresponding element in the two vectors: $q_{j \to i}^1(x_i) + q_{j \to i}^2(x_i) = [q_1^1 + q_1^2, \cdots, q_{|\mathcal{G}|}^1 + q_{|\mathcal{G}|}^2]$ and $r_{i \to j}^1(x_i) + r_{i \to j}^2(x_i) = [r_1^1 + r_1^2, \cdots, r_{|\mathcal{G}|}^1 + r_{|\mathcal{G}|}^2]$. For Equation 3, we need to minimize the regret of function node j with respect to its neighboring variables $\mathbf{x}_\mathbf{j}$ as:

$$r_{j \to i}(x_i) = \mathbb{U}_j(\tilde{\mathbf{x}}_\mathbf{j}) + \sum_{k \in N(j) \setminus i} q_{k \to j}(\tilde{x}_k) \quad (12)$$

where $\mathbb{U}_j(\tilde{\mathbf{x}}_\mathbf{j}) = [\tilde{U}_j(\tilde{\mathbf{x}}_\mathbf{j}, \langle b_j, \mathbf{x}_\mathbf{j}^* \rangle_1), \cdots, \tilde{U}_j(\tilde{\mathbf{x}}_\mathbf{j}, \langle b_j, \mathbf{x}_\mathbf{j}^* \rangle_{|\mathcal{G}|}]$, $\tilde{U}_j(\tilde{\mathbf{x}}_\mathbf{j}, \langle b_j, \mathbf{x}_\mathbf{j}^* \rangle_g) = U_j(b_j, \mathbf{x}_\mathbf{j}^*) - U_j(b_j, \mathbf{x}_\mathbf{j}')$, and

$$\tilde{\mathbf{x}}_\mathbf{j} = \arg\min_{\mathbf{x}_\mathbf{j} \setminus x_i} \max_{\langle b_j, \mathbf{x}_\mathbf{j}^* \rangle_g \in \mathcal{G}} [\tilde{U}_j(\mathbf{x}_\mathbf{j}, \langle b_j, \mathbf{x}_\mathbf{j}^* \rangle_g) + \sum_{k \in N(j) \setminus i} q_{k \to j}(x_k, g)] \quad (13)$$

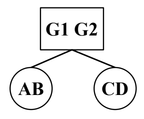
At the end of the message-passing phase, each variable x_i computes its marginal function $z_i(x_i)$ according to Equation 4. Obviously, the value of the marginal function is also a vector: $z_i(x_i) = [z_1, \cdots, z_{|\mathcal{G}|}]$. The best assignment of the variable x_i can be computed by:

$$x_i = \arg\min_{x_i' \in D_i} \max_{g} z_i(x_i', g) \quad (14)$$

where g is an index for the vector. After that, the minimax regret δ can be computed by propagating values in a (any) pre-defined tree structure of the factor graph: (1) Each variable node sends its assignment to its neighboring nodes; (2) On receipt of all the assignments from its neighboring nodes, each function node computes the regret value and sends the message to its neighboring nodes; (3) Each node propagates the regret values until all the regret values are computed and received by all the nodes. Then, δ can be computed by adding all the m messages in each node.

	G1	G2
AC	-57	64
AD	-96	-162
BC	54	72
BD	-4	55

(a) $\tilde{V}(\mathbf{x}, \langle \mathbf{b}, \mathbf{x}^* \rangle)$



(b) Factor Graph

	G1	G2
A	-96	-162
B	-4	55
C	-57	64
D	-96	-162

(c) $z_i(x_i, \langle \mathbf{b}, \mathbf{x}^* \rangle)$

Figure 1: Example of the Master Problem.

An example of the master problem with randomly generated $\mathbb{V}$ is shown in Figure 1. In this example, there are two variables with the domain $\{A, B\}$ and $\{C, D\}$ respectively and the witness set $\mathcal{G}$ is $\{G1, G2\}$. Clearly, the minimax solution is AD and the minimax regret is -96 since we have $\min\{\max\{-57, 64\}, \max\{-96, -162\}, \max\{54, 72\}, \max\{-4, 55\}\} = -96$. For our Max-Sum, according to Equation 12, the message $r_1(A) = \mathbb{V}(AD)$ since $AD = \arg\min_{AD,AC}\{\max\{-57, 64\}, \max\{-96, -162\}\}$. Similarly, we have the messages: $r_1(B) = \mathbb{V}(BD)$, $r_1(C)$

= $\mathbb{V}(AC)$, and $r_1(D)$ = $\mathbb{V}(AD)$. After the message-passing phase, the marginal functions $z_1(A) = \mathbb{V}(AD)$, $z_1(B) = \mathbb{V}(BD)$, $z_2(C) = \mathbb{V}(AC)$, $z_2(D) = \mathbb{V}(AD)$. Therefore, the best assignments of each variable are $x_1 = \arg\min_{A,B}\{\max\{-96, -162, \}, \max\{-4, 55\}\} = A$ and $x_2 = \arg\max_{C,D}\{\max\{-57, 64\}, \max\{-96, -162\}\} = D$. The joint solution is AD and the minimax regret is -96, which are equal to the minimax solution and regret that we computed earlier according to the definition.

The Subproblem

The *subproblem* in Equation 9 given the current solution $\mathbf{x}$ can be written as:

$$\langle \mathbf{b}, \mathbf{x}^* \rangle = \arg \underbrace{\max_{\mathbf{b}\in\mathcal{B}} \max_{\mathbf{x}'^*} \sum_{j=1}^{m} [U_j(b_j, \mathbf{x}'^*_\mathbf{j}) - U_j(b_j, \mathbf{x_j})]}_{\delta'} .$$

Since each belief b_j is independent from each other and from the decision variables, the calculation of each belief can be moved inside the utility function, shown as:

$$\langle \mathbf{b}, \mathbf{x}^* \rangle = \arg \underbrace{\max_{\mathbf{x}'^*} \sum_{j=1}^{m} \left\{ \max_{b_j} [U_j(b_j, \mathbf{x}'^*_\mathbf{j}) - U_j(b_j, \mathbf{x_j})] \right\}}_{\delta'}$$

Thus, we can define a new utility function as:

$$U_j(\mathbf{x}'^*_\mathbf{j}) = \max_{b_j} [U_j(b_j, \mathbf{x}'^*_\mathbf{j}) - U_j(b_j, \mathbf{x_j})] \qquad (15)$$

and rely on a linear program to compute the utility:

$$\begin{array}{ll} \max_{b_j} & U_j(b_j, \mathbf{x}'^*_\mathbf{j}) - U_j(b_j, \mathbf{x_j}) \\ \text{s.t.} & \forall s_j \in S_j, b_j(s_j) \geq 0 \\ & \sum_{s_j \in S_j} b_j(s_j) = 1 \end{array} \qquad (16)$$

This can be done locally and thereby the subproblem can be solved by standard DCOP algorithms. For Max-Sum, we need to implement a linear program (Equation 16) in each function node to compute the belief b_j when $U_j(\mathbf{x_j})$ is called in Equation 3. Once the optimal solution $\mathbf{x}^*$ is found, we can propagate $\mathbf{x}$ and $\mathbf{x}^*$ to the function nodes and apply Equation 16 for each function node j to compute the belief $\mathbf{b}$. Similar to the master problem, the minimax regret δ' can be computed by value propagation in the factor graph.

Analysis and Discussion

For the computation and communication complexity, the subproblem uses the standard Max-Sum except that a linear program is solved each time when the utility function is called in Equation 3. For the master problem, according to Equation 13, the computation is exponential only in the number of variables in the scope of U_j (similar to standard Max-Sum) but *linear* in the number of witness points in $\mathcal{G}$. The messages in the master problem are vectors with the length of $|\mathcal{G}|$ while the messages in the sub-problems are normal Max-Sum messages. In experiments, we observed $\mathcal{G}$ is usually very small ($<$10) for the tested problems.

Inherited from Max-Sum, the optimality of our algorithm depends on the structure of the factor graph. Specifically, for an acyclic factor graph, it is known that Max-Sum converges to the optimal solution of the DCOPs (Farinelli et al. 2008). When the factor graph is cyclic, the straightforward application of Max-Sum is not guaranteed to converge optimally. In this cases, we prune edges of the original graph and generate a tree graph using the methods in (Rogers et al. 2011) and then apply our algorithm on the remaining tree graph to compute the solution. Specifically, we define the weight of each dependency edge between variable x_i and function U_j in the original factor graph as:

$$w_{ij} = \max_{s_j} \max_{\mathbf{x_j}\backslash x_i} [\max_{x_i} U_j(s_j, \mathbf{x_j}) - \min_{x_i} U_j(s_j, \mathbf{x_j})] \qquad (17)$$

Given this, we could use the *maximum spanning tree* algorithm to form a tree structure graph (Rogers et al. 2011).

Theorem 1. *For acyclic factor graphs, ICG-Max-Sum guarantees to converge to the optimal minimax solution.*

Theorem 2. *For cyclic graphs, the error introduced by the minmax solution $\tilde{\mathbf{x}}$ of ICG-Max-Sum on the remaining tree graph is bounded by: $V_{regret}(\tilde{\mathbf{x}}) - V_{regret}(\mathbf{x}) \leq \varepsilon$ where $\mathbf{x}$ is the optimal minmax solution on the original graph and the error bound $\varepsilon = \sum_j \sum_{x_i \in \mathbf{x}^c_\mathbf{j}} w_{ij}$ where $\mathbf{x}^c_\mathbf{j}$ is the set of variable dependencies removed from U_j.*

The proofs of Theorems 1 and 2 can be found in the appendix. Note that for acyclic graphs $\varepsilon = 0$ and $\tilde{\mathbf{x}} = \mathbf{x}$ is the optimal solution. For cyclic graphs, the solution computed by our algorithms is error-bounded with pruning methods.

Empirical Evaluation

We tested the performance of our algorithm on a disaster scenario (similar to the Fukushima Daiichi nuclear disaster), in which radioactive explosions created expanding and moving radioactive clouds that pose a threat to people, food reserves, and other key assets around the area. Hence, a group of first responders are assigned to respond to the crisis. Because each task may require different teams of the responders to work together, they must be coordinated before entering the area. However, given the invisibility of radiation and the very short response time, information about the radioactive clouds is very limited. Thus, each task is highly uncertain about its possible outcomes (rewards).

Problem Setup We developed a simulator for the above scenario, in which tasks with any 4 types of targets (i.e., food, animal, victim, and fuel) were randomly generated on a 2D grid map. A target in this context is a person or asset that locates in the disaster area and needs to be saved by the responders. There were 4 types of responders (i.e., transporter, soldier, medic, and firefighter). However, our algorithms can be applied to problems with arbitrary numbers of target and responder types. In our settings, each task requires a team of responders with different skills (e.g., the task for saving a victim may require a firefighter to put out the fire and a medic to do first aid). Hence we randomized the requirements of each target type and kept them fixed for each instance. Given this, a factor graph was defined with variable nodes for the responders and function nodes for the tasks.

Table 1: Runtime Results of ICG vs. ICG-Max-Sum

#Agents	#Tasks	#States	ICG	ICG-Max-Sum
3	6	20	0.671s	2.219s
5	10	20	1.114s	4.783s
10	20	20	>2h	19.272s
100	200	20	>12h	628.7s

Here, a task was linked to a responder if she owned the skill required by the task. Since the radioactive clouds were invisible to the responders when allocating their tasks, we defined a set of states S_j for each task j. These states captured the possible situations that might happen during task execution. For instance, the road may be blocked, the target may have already been contaminated, or the responders may be killed during the process. For each state $s_j \in S_j$, we specified a utility $U_j(s_j, \mathbf{x_j})$ for the responders doing the task in a given state (e.g., if a resource has already been contaminated, there is little value in the responders saving it).

In the experiments, we ensured that there were more tasks than responders so that not all tasks can be performed at the same time since all tasks need at least one responder. Thus, the responders must make a good choice to maximize the overall team performance. Without loss of generality, we set the number of tasks to be twice the number of agents. However, this ratio can be arbitrary as long as there are more tasks than agents. For each instance, we defined a Markov chain for the states of each task with the transition matrix randomly initialized. When testing a solution for its true performance in the domain, we first randomly selected an initial state for every task and computed the value, following up with a state transition for all tasks according to the Markov chains. We repeated the process for 100 runs and reported the average values. Note that the task states were only used to evaluate a solution after it has been computed but hidden to the algorithms. The algorithms must compute a solution without knowing the task states or their distributions.

Experimental Results To date, none of the existing DCOPs solvers can solve our model so a directed comparison is not possible. Therefore, to test the scalability and solution quality of our algorithm, we compared it with two baselines: a centralized method based on ICG (Equations 8 and 9) and a decentralized method based on DSA (Zhang et al. 2005). Specifically, the two operators $\max_{\mathbf{x}^*}$ and $\min_{\mathbf{x}'}$ are alternatively solved by DSA and a linear program is used to solve the operator $\max_{\mathbf{b}\in\mathcal{B}}$ in Equation 7. We ran our experiments on a machine with a 2.66GHZ Intel Core 2 Duo and 4GB memory. All the algorithms were implemented in Java 1.6, and the linear programs are solved by CPLEX 12.4.

In more details, Table 1 shows the runtime of centralized ICG and ICG-Max-Sum. We can see from the table that the runtime of centralized ICG increases dramatically with the problem size and ran out of time (>2h) for problems with more than 10 agents, while ICG-Max-Sum only took few seconds to solve the same problems. As we can see from the table, large UR-DCOPs with many agents and tasks are intractable for centralized ICG. Intuitively, the reason for the stability of ICG-Max-Sum is that it can exploit the interaction structures of the task allocation problems (i.e., tasks only usually require a few agents to coordinate). Table 2 shows the average *true* values $V(\mathbf{s}, \mathbf{x})$ achieved by the solutions of DSA and ICG-Max-Sum respectively given the hidden task states. This is a useful measurement because it accounts for the real performance of the responders in the environment. From the table, we can see that the solutions of ICG-Max-Sum produced much higher values than the ones of DSA for all tested instances. For large problems, the performance of DSA dropped very quickly as the errors in the $\max_{x^*}$ and $\min_{x'}$ steps increased given more agents and tasks. Table 3 shows the *true* loss (i.e., the regret) $V_{regret}(\mathbf{x}) = V(\mathbf{s}, \mathbf{x}^*) - V(\mathbf{s}, \mathbf{x})$ achieved by the solutions of DSA and ICG-Max-Sum respectively given the hidden task states of the problem instance where $\mathbf{x}^*$ is the optimal solution for $\mathbf{s}$. Because it is intractable to compute the optimal solution $\mathbf{x}^*$ for large problems (e.g., the solution space for the instance with 200 agents and 400 tasks is 400^{200}), we only report the results for small instances. From the table, we can see that the actual regrets of ICG-Max-Sum are much lower than DSA especially for larger problems.

Table 2: Value Results of DSA vs. ICG-Max-Sum

#Agents	#Tasks	#States	DSA	ICG-Max-Sum
10	20	20	2117.57	5294.23
20	40	20	2556.95	6413.41
50	100	20	3939.11	7414.50
100	200	20	11461.69	23796.97
200	400	20	26243.01	46805.38

Table 3: Regret Results of DSA vs. ICG-Max-Sum

#Agents	#Tasks	#States	DSA	ICG-Max-Sum
2	4	20	65.42	55.14
3	6	20	941.12	10.37
5	10	20	1574.42	7.97
7	14	20	1766.38	13.22

Conclusions

We have presented the ICG-Max-Sum algorithm to find robust solutions for UR-DCOPs. Specifically, we assume the distributions of the task states are unknown and we use minimax regret to evaluate the worst-case loss. Building on the ideas of iterative constraint generation, we proposed a decentralized algorithm that can compute the minimax regret and solution using Max-Sum. Similar to Max-Sum, it can exploit the interaction structures among agents and scale up to problems with large number of agents. We proved that ICG-Max-Sum is optimal for acyclic graphs and the regret is error-bounded for cyclic graphs. Then, we empirically evaluated the performance of our algorithms on our motivating task allocation domains in a disaster response scenario. The experimental results confirm that ICG-Max-Sum has better scalability than centralized ICG and outperformed DSA — the state-of-the-art decentralized method — with much higher values and lower regrets in the tested domain. In the future, we plan to extend our work to more complex domains where the tasks are not completely independent.

Appendix

Lemma 1. *The master problems in ICG-Max-Sum will converge to the optimal solution for acyclic factor graphs.*

Proof. The messages (vectors) in the master problems represent the regret values of all the witness points in $|\mathcal{G}|$. The *sum* operator adds up all the regret components for each witness point, $U_j(b_j, \mathbf{x}_\mathbf{j}^*) - U_j(b_j, \mathbf{x}_\mathbf{j})$, sent from its neighboring nodes. The *max* operator selects the current minimax solution $\tilde{\mathbf{x}}_\mathbf{j}$ and sends out the corresponding regret values. Specifically, this operator is over matrices $[m_{ij}]$ with the row i indexed by witness points and column j by assignments. It compares two matrices and outputs the one with the smaller $\min_j \max_i [m_{ij}]$ value. This operator is associative and commutative with an identity element (matrix) $[\infty]$ (i.e., the algebra is a commutative semi-ring). Thus, since Max-Sum is a GDL algorithm (Aji and McEliece 2000), the results hold for acyclic factor graphs. □

Lemma 2. *The subproblems in ICG-Max-Sum will converge to the optimal solution for acyclic factor graphs.*

Proof. The subproblems are standard DCOPs given the utility function (Equation 15) that can be computed locally by each function node. Thus, Max-Sum will converge to the optimal solution for acyclic factor graphs. □

Proof of Theorem 1

Proof. According to Lemmas 1 and 2, the master problems and subproblems are optimal for acyclic factor graphs. Thus, this theorem can be proved by showing that the subproblem will enumerate all $\langle \mathbf{b}, \mathbf{x}^* \rangle$ pairs if $\mathbf{x}$ is not the minimax optimal solution. This is equivalent to proving that in the subproblem, $\delta' > \delta$ is always true and the new witness $\langle \mathbf{b}, \mathbf{x}^* \rangle \notin \mathcal{G}$ if $\mathbf{x} \neq \bar{\mathbf{x}}$ where $\bar{\mathbf{x}}$ is the minimax optimal solution. Suppose $\delta' = \delta$ and $\mathbf{x} \neq \bar{\mathbf{x}}$, then we have

$$\begin{array}{rcl} \delta' & = & \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})] \\ & > & \min_{\mathbf{x}'} \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x}')] \\ & = & \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \bar{\mathbf{x}})] \Longrightarrow \\ \delta & = & \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle \in \mathcal{G}} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})] = \delta' \\ & > & \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \bar{\mathbf{x}})]. \end{array} \quad (18)$$

Because $\mathcal{G}$ is only a subset of the whole space, we have

$$\max_{\langle \mathbf{b}, \mathbf{x}^* \rangle \in \mathcal{G}} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})] > \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle \in \mathcal{G}} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \bar{\mathbf{x}})].$$

Then, the current solution $\mathbf{x}$ computed by the master problem is $\mathbf{x} = \arg\min_{\mathbf{x}'} [\max_{\langle \mathbf{b}, \mathbf{x}^* \rangle \in \mathcal{G}} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x}')]] = \bar{\mathbf{x}}$. This is contradictory to the assumption $\mathbf{x} \neq \bar{\mathbf{x}}$. Furthermore, in the subproblem, the newly generated witness point must not be in $\mathcal{G}$, otherwise $\delta' = \delta$ due to the same $\mathbf{x}$ and $\langle \mathbf{b}, \mathbf{x}^* \rangle$ being in both problems because we have $\delta' = \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})] = \max_{\langle \mathbf{b}, \mathbf{x}^* \rangle \in \mathcal{G}} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})] = \delta$.

The algorithm will converge to the minimax optimal solution $\bar{\mathbf{x}}$ once all witness points $\langle \mathbf{b}, \mathbf{x}^* \rangle$ are enumerated and added to $\mathcal{G}$ by the subproblems. Thus, the results hold. □

Lemma 3. *For cyclic graphs, the error in values between the solution* $\tilde{\mathbf{x}}$ *computed on the remaining tree graph and the optimal solution* $\mathbf{x}^*$ *on the original graph is bounded by:*

$$\underbrace{\sum_j U_j(s_j, \mathbf{x}_\mathbf{j}^*)}_{\text{optimal value}} - \underbrace{\sum_j \min_{\mathbf{x}_\mathbf{j}^\mathbf{c}} U_j(s_j, \tilde{\mathbf{x}}_\mathbf{j})}_{\text{approximate value}} \leq \varepsilon \quad (19)$$

where $\mathbf{x}_\mathbf{j}^\mathbf{c}$ *represents the set of dependent variables that are originally connected to function* U_j *but have been removed in the tree graph and the error* $\varepsilon = \sum_j \varepsilon_j(\mathbf{x}_\mathbf{j}^\mathbf{c})$ *where the maximum impact of a set of removed dependencies* $\mathbf{x}_\mathbf{j}^\mathbf{c}$ *is:*

$$\varepsilon_j(\mathbf{x}_\mathbf{j}^\mathbf{c}) = \begin{cases} \max_{\mathbf{x}_\mathbf{j} \setminus \mathbf{x}_\mathbf{j}^\mathbf{c}} [\max_{\mathbf{x}_\mathbf{j}^\mathbf{c}} U_j(s_j, \mathbf{x}_\mathbf{j}) - \min_{\mathbf{x}_\mathbf{j}^\mathbf{c}} U_j(s_j, \mathbf{x}_\mathbf{j})] & \textit{if } \mathbf{x}_\mathbf{j}^\mathbf{c} \neq \emptyset \\ 0 & \textit{otherwise} \end{cases}$$

Proof. We can define new utility functions as $\forall j, F_j(\mathbf{x}_\mathbf{j}) = U_j(s_j, \mathbf{x}_\mathbf{j})$ for the factor graph and then the lemma holds according to Theorem 1 in (Rogers et al. 2011). □

Proof of Theorem 2

Proof. According to Theorem 1, the minmax solution computed by ICG-Max-Sum is optimal for the remaining tree graph. Thus, this theorem can be proved by showing the error introduced by removing edges is bounded comparing to the optimal minmax regret on the original graph. This is independent from the process of ICG and Max-Sum. Let the regret of $\tilde{\mathbf{x}}$ (i.e., the minmax solution computed by ICG-Max-Sum on the tree graph) on the original graph be:

$$V_{regret}(\tilde{\mathbf{x}}) \equiv \max_{\mathbf{b}} \max_{\mathbf{x}^*} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \tilde{\mathbf{x}})] \quad (20)$$

and the regret of $\mathbf{x}$ (i.e., the overall optimal minimax solution) on the original graph be:

$$V_{regret}(\mathbf{x}) \equiv \max_{\mathbf{b}} \max_{\mathbf{x}^*} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})] \quad (21)$$

According to Lemma 3, we have:

$$V(\mathbf{b}, \mathbf{x}) - V(\mathbf{b}, \tilde{\mathbf{x}}) \leq \varepsilon \quad (22)$$

where $V(\mathbf{b}, \mathbf{x}) = \sum_j U_j(\mathbf{b}_\mathbf{j}, \mathbf{x}_\mathbf{j})$ is the optimal value given $\mathbf{b}$ and $V(\mathbf{b}, \tilde{\mathbf{x}}) = \sum_j \min_{\mathbf{x}_\mathbf{j}^\mathbf{c}} U_j(\mathbf{b}_\mathbf{j}, \tilde{\mathbf{x}}_\mathbf{j})$ is the value of $\tilde{\mathbf{x}}$ on the original graph given $\mathbf{b}$ with $\mathbf{x}_\mathbf{j}^\mathbf{c}$ the set of dependency edges connected to U_j that have been removed in the tree graph. Note that $\mathbf{b}$ is the overall worst-case belief of the problem, which is independent from the choice of the agents as the task states are uncontrollable by the agents in our settings. Then, we have the following inequations:

$$\begin{aligned} V_{regret}(\mathbf{x}) &= \max_{\mathbf{b}} \max_{\mathbf{x}^*} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \mathbf{x})] \\ &\geq \max_{\mathbf{b}} \max_{\mathbf{x}^*} \{V(\mathbf{b}, \mathbf{x}^*) - [V(\mathbf{b}, \tilde{\mathbf{x}}) + \varepsilon]\} \\ &= \max_{\mathbf{b}} \max_{\mathbf{x}^*} [V(\mathbf{b}, \mathbf{x}^*) - V(\mathbf{b}, \tilde{\mathbf{x}})] - \varepsilon \\ &= V_{regret}(\tilde{\mathbf{x}}) - \varepsilon \end{aligned} \quad (23)$$

Thus, the theorem holds because we have:

$$V_{regret}(\tilde{\mathbf{x}}) - V_{regret}(\mathbf{x}) \leq \varepsilon \quad (24)$$

□

Acknowledgments

We thank all the anonymous reviewers for their helpful comments and suggestions. This work was supported by the ORCHID project (`http://www.orchid.ac.uk`).

References

Aji, S. M., and McEliece, R. J. 2000. The generalized distributive law. *IEEE Transactions on Information Theory* 46(2):325–343.

Atlas, J., and Decker, K. 2010. Coordination for uncertain outcomes using distributed neighbor exchange. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 1047–1054.

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4:238–252.

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence* 170(8):686–713.

Farinelli, A.; Rogers, A.; Petcu, A.; and Jennings, N. R. 2008. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 639–646.

Léauté, T., and Faltings, B. 2011. Distributed constraint optimization under stochastic uncertainty. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 68–73.

Maheswaran, R.; Pearce, J.; and Tambe, M. 2004. Distributed algorithms for dcop: A graphical-game-based approach. 432–439.

Matsui, T.; Matsuo, H.; Silaghi, M.; Hirayama, K.; Yokoo, M.; and Baba, S. 2010. A quantified distributed constraint optimization problem. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 1023–1030.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yoko, M. 2005. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161(1–2):149–180.

Nguyen, D. T.; Yeoh, W.; and Lau, H. C. 2012. Stochastic dominance in stochastic DCOPs for risk-sensitive applications. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 257–264.

Petcu, A., and Faltings, B. 2005. DPOP: A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 266–271.

Regan, K., and Boutilier, C. 2010. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 1127–1133.

Regan, K., and Boutilier, C. 2011. Eliciting additive reward functions for markov decision processes. In *Proceedings of the 22nd international joint conference on Artificial Intelligence*, 2159–2164.

Rogers, A.; Farinelli, A.; Stranders, R.; and Jennings, N. R. 2011. Bounded approximate decentralised coordination via the Max-Sum algorithm. *Artificial Intelligence* 175(2):730–759.

Stranders, R.; Farinelli, A.; Rogers, A.; and Jennings, N. R. 2009. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, volume 9, 299–304.

Stranders, R.; Fave, F. M. D.; Rogers, A.; and Jennings, N. R. 2011. U-GDL: A decentralised algorithm for DCOPs with uncertainty. Technical report, University of Southampton.

Tambe, M. 2012. *Security and game theory: algorithms, deployed systems, lessons learned.* Cambridge University Press.

Wu, F.; Jennings, N. R.; and Chen, X. 2012. Sample-based policy iteration for constrained DEC-POMDPs. In *Proceedings of the 20th European Conference on Artificial Intelligence*, 858–863.

Zhang, W.; Wang, G.; Xing, Z.; and Wittenburg, L. 2005. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161(1):55–87.