Stochastic Dominance in Stochastic DCOPs for Risk-Sensitive Applications

Duc Thien Nguyen
School of Information Systems
Singapore Management University
Singapore 178902
dtnguyen@smu.edu.sg

William Yeoh
School of Information Systems
Singapore Management University
Singapore 178902
williamyeoh@smu.edu.sg

Hoong Chuin Lau
School of Information Systems
Singapore Management University
Singapore 178902
hclau@smu.edu.sg

ABSTRACT

Distributed constraint optimization problems (DCOPs) are well-suited for modeling multi-agent coordination problems where the primary interactions are between local subsets of agents. However, one limitation of DCOPs is the assumption that the constraint rewards are without uncertainty. Researchers have thus extended DCOPs to Stochastic DCOPs (SDCOPs), where rewards are sampled from known probability distribution reward functions, and introduced algorithms to find solutions with the largest expected reward. Unfortunately, such a solution might be very *risky*, that is, very likely to result in a poor reward. Thus, in this paper, we make three contributions: (1) we propose a stricter objective for SDCOPs, namely to find a solution with the most stochastically dominating probability distribution reward function; (2) we introduce an algorithm to find such solutions; and (3) we show that stochastically dominating solutions can indeed be less risky than expected reward maximizing solutions.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

General Terms

Algorithms, Experimentation

Keywords

DCOP, DPOP, Uncertainty, Stochastic Dominance

1. INTRODUCTION

Distributed constraint optimization problems (DCOPs) are problems where agents need to coordinate their value assignments to maximize the sum of the resulting constraint rewards. They are well-suited for modeling multi-agent coordination problems where the primary interactions are between local subsets of agents, such as the scheduling of meetings [15], the coordination of sensors in networks [5, 14], the coordination of first responders in disasters [11], the management of power plant networks [10] and the generation of coalition structures [24].

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

One of the limitations of DCOPs is that they only model problems with (constraint) rewards that are both known a priori and without uncertainty. Thus, researchers have introduced various DCOP extensions to address this limitation. One example is the Distributed Coordination of Exploitation and Exploration (DCEE) model [8, 23], where agents initially do not know the constraint rewards and can only discover them through exploration. The agents in a DCEE problem coordinate to balance exploration and exploitation such that they accumulate the maximum amount of reward over a finite time horizon. Another example is the DCOP under Stochastic Uncertainty model [12], where there are random variables that take on values according to known probability distribution functions. The values of these random variables affect the overall reward. Finally, researchers have introduced a Stochastic DCOP (SDCOP) model where the constraint rewards are no longer deterministic values but are sampled from known probability distribution functions called reward functions [1]. Agents in these latter two problems coordinate to maximize the expected reward of the overall solution.

Finding a solution that maximizes the expected reward in an SDCOP can be done in a straightforward manner if the means of the probability distribution functions are known. Since the sum of the expected reward of two functions equals the expected reward of the convolution of the two functions according to the linearity of expectations, one can map an SDCOP to a DCOP by mapping the reward function in an SDCOP to the mean of that function in a DCOP. One can then use existing DCOP algorithms to solve the problem.

Our concern with the approach above is that a solution with the maximum expected reward might be very *risky*, that is, very likely to result in a poor reward, particularly in high variance environments. As an example, one might prefer to have a solution whose overall reward function follows a unimodal distribution rather than a bimodal distribution especially if the expected reward of the unimodal distribution is only slightly smaller than that of the bimodal distribution. Thus, in this paper, we propose a stricter objective for SDCOPs, namely to find a solution with the most stochastically dominating probability distribution function of the reward of the solution [6]. Intuitively, a stochastically dominating function is less risky than the dominated function.

We also introduce Stochastic Dominance DPOP (SD-DPOP), an extension of DPOP [18], to solve SDCOPs. Our results show that stochastic dominating solutions found by SD-DPOP are less risky compared to expected reward maximizing solutions found by DPOP for common risk functions.

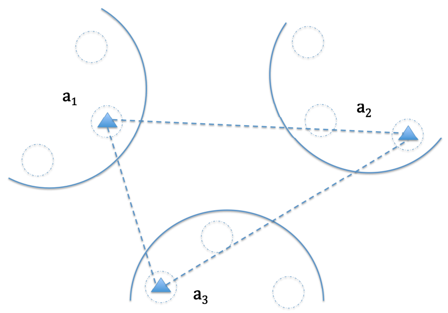


Figure 1: Illustration of Mobile Agents

2. MOTIVATING DOMAIN

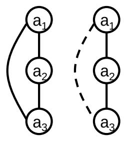
First responders in an earthquake hit area or soldiers in the battlefield might need to deploy mobile agents to create a mesh of communication network to coordinate their actions [16]. Therefore, we motivate our work in this paper with the problem of maximizing the signal strengths between neighboring mobile agents in a network [8]. Figure 1 shows an illustration of three mobile agents, whose positions are represented by triangles. Straight dotted lines connect neighboring agents. Each mobile agent can choose to be in one of three possible nearby locations, which are denoted by circular dotted lines. We assume that the mobile agents can only move within a small range from their starting locations and the topology of the network thus remains unchanged. The signal strength between two neighboring agents depends on their locations. For example, the further their distance, the weaker the signal strength. The small movements of the mobile agents can affect the signal strength significantly due to radio interference. However, the signal strength between a pair of neighboring agents can be modeled as an independent random number drawn from some distribution [9]. In this paper, we will use Gaussian functions to model the signal strengths. The objective in this problem is for the agents to coordinate their choice of locations such that the sum of the signal strength between every pair of neighboring agents is maximized. We use these two domains as our motivating domains because they are examples that call for one to be risk-averse. In both domains, losing the ability to communicate can result in the loss of lives.

3. BACKGROUND

In this section, we provide a brief description of the DCOP and Stochastic DCOP model as well as the DPOP algorithm.

3.1 DCOPs

A *distributed constraint optimization problem* (DCOP) [17, 18] is defined as a tuple $\langle A, X, D, F\rangle$. $A = \{a_i\}_0^n$ is the finite set of agents. $X = \{x_i\}_0^n$ is the set of variables, where x_i is owned by agent a_i. $D = \{d_i\}_0^n$ is the set of finite domains, where domain d_i is the set of possible values for agent $a_i \in A$. $F = \{f_i\}_0^m$ is the set of binary constraints, where each constraint $f_i : d_{i_1} \times d_{i_2} \rightarrow \mathbb{R}^+ \cup \{0\}$ specifies its non-negative reward as a function of the values of the two different variables $x_{i_1}, x_{i_2} \in X$ that share the constraint. Although the general DCOP definition allows one agent to own multiple variables as well as the existence of n-ary constraints, we restrict our definition here for simplification purposes. One can transform a general DCOP to our DCOP using pre-processing techniques [2]. A solution is a value assignment for a subset of variables. Its reward is the sum of the constraint rewards of all constraints shared by variables with assigned values. A solution is *complete* iff it is a value assignment for all variables. Solving a DCOP optimally means finding a reward-maximal complete solution.



(a) (b)

for $i < j$

x_i	x_j	Rewards	Reward Functions
0	0	5	$\mathcal{N}(5,1)$
0	1	8	$\mathcal{N}(8,1)$
1	0	20	$\mathcal{N}(20,1)$
1	1	3	$\mathcal{N}(3,1)$

(c)

Figure 2: Example DCOP

DCOPs are commonly visualized as *constraint graphs*, whose vertices are the agents and whose edges are the constraints. Most complete DCOP algorithms operate on *pseudo-trees*, which are spanning trees of fully connected constraint graphs such that no two vertices in different subtrees of the spanning tree are connected by an edge in the constraint graph. Figure 2(a) shows the constraint graph of an example DCOP with three agents controlling variables that can each be assigned the values 0 or 1, Figure 2(b) shows one possible pseudo-tree (the dotted line is called a *backedge*, which is an edge of the constraint graph that does not connect a pair of parent-child nodes), and the third column in Figure 2(c) shows the constraint rewards. We will use this problem as a running example in this paper.

3.2 Stochastic DCOPs

Stochastic DCOPs (SDCOPs) are extensions of DCOPs where each constraint $f_i : d_{i_1} \times d_{i_2} \rightarrow P(x_{i_1}, x_{i_2})$ now specifies a (potentially discretized) probability distribution function of the reward as a function of the values of the two different variables $x_{i_1}, x_{i_2} \in X$ that share the constraint [1]. We call these functions *reward functions* in this paper. For example, the fourth column in Figure 2(c) shows the Gaussian reward functions for the different pairs of values. We assume that the reward functions are all independent of each other. Solving an SDCOP optimally means finding a complete solution X^* that maximizes the expected sum of all rewards:

$$X^* = \underset{X \in d_1 \times d_2 \times \dots \times d_n}{\arg\max} \left\{ \mathbb{E}\left[\sum_i f_i(x_{i_1}, x_{i_2} \mid x_{i_j} \in X) \right] \right\} \quad (1)$$

One can solve for the this objective in a straightforward manner if the mean of each reward function is known. Solving an SDCOP optimally is then equivalent to solving a regular DCOP optimally, where the rewards of a constraint are the means of the respective reward functions specified by that constraint.

3.3 DPOP

Distributed Pseudo-tree Optimization Procedure (DPOP) [18] is a complete DCOP algorithm that can be viewed as a distributed version of the Bucket Elimination algorithm [3]. There are three phases in the operation of DPOP:

x_1	x_2	Rewards
0	0	max(5+ 5, 8+8) = 16
0	1	max(5+20, 8+3) = 25
1	0	max(20+ 5, 3+8) = 25
1	1	max(20+20, 3+3) = 40

(a)

x_1	Rewards
0	max(5+16, 8+25) = 33
1	max(20+25, 3+40) = 45

(b)

Table 1: UTIL Phase Computations in DPOP

(1) The first phase is the pseudo-tree generation phase, where DPOP calls existing distributed pseudo-tree construction algorithms like Distributed DFS [7] as a subroutine to construct its pseudo-tree.

(2) The second phase is the UTIL phase, where each agent, starting from the leafs of the pseudo-tree, computes the optimal sum of rewards in its subtree for each value combination of its ancestor agents. The agent does so by summing the rewards in the UTIL messages received from its child agents and projects out its own variable by optimizing over it. In our example problem, agent a_3 computes the optimal reward for each value combination of variables x_1 and x_2 as shown in Table 1(a) and sends the rewards to its parent agent a_2 in a UTIL message. Agent a_2 then computes the optimal reward for each value of variable x_1 as shown in Table 1(b) and sends the rewards to its parent agent a_2 in a UTIL message.

(3) The third phase is the VALUE phase, where each agent, starting from the root of the pseudo-tree, determines the optimal value for its variable. The root agent does so using the UTIL messages received in the second phase. In our example problem, agent a_1 determines from the UTIL message from agent a_2 that the value with the largest reward for its variable is 1 with a reward of 45. It then sends this information down to its child agent a_2 in a VALUE message. Upon receiving the VALUE message from its parent agent, agent a_2 determines that the value with the largest reward for its variable assuming that $x_1 = 1$ is 0 with a reward of 45. It then sends this information down to its child agent a_3 in a VALUE message. Finally, upon receiving the VALUE messages from its parent agent, agent a_3 determines that the value with the largest reward for its variable assuming that $x_1 = 1$ and $x_2 = 0$ is 0 with a reward of 25.

4. STOCHASTIC DOMINANCE

Although finding a solution that maximizes the expected reward in an SDCOP is a reasonable and intuitive objective, it fails to characterize "risk" very well. To illustrate this, Figure 3 shows two probability distribution functions, where the expected reward of the unimodal distribution is slightly smaller than that of the bimodal distribution but the variance is also smaller. Hence if the problem domain requires a decision maker to be risk-averse, then one might prefer the unimodal distribution over the bimodal one. In general, it is well-known (particularly in financial domains) that maximizing utility without considering risk does not yield good solutions in risky environments.

To incorporate the notion of risk, we propose a new (stricter) goal for SDCOPs, namely, our aim is to find a complete solution with the most stochastically dominating [6] overall reward function. Intuitively, a stochastically dominating function is less risky than any dominated function. In this paper, we will use the second-order stochastic dominance criteria [13] as the dominance criteria.

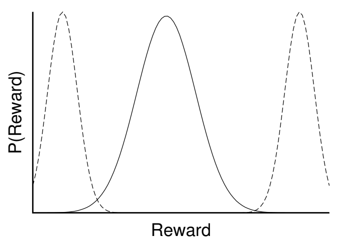


Figure 3: Unimodal and Bimodal Distributions

Let $f_{1+2+\ldots+m}(X)$ denote the overall reward function for a complete solution X, which is the convolution of the individual reward functions $f_i(x_{i_1}, x_{i_2} \mid x_{i_j} \in X)$ over all $i \in [0, m]$. And let $F_{X_k}(t) = \int_{-\infty}^{t} f_{1+2+\ldots+m}(X_k)(t)\, dt$ denote the corresponding cumulative distribution function. We say that a solution X_1 *dominates* (written $\prec$) another solution X_2 iff:

$$\int_{-\infty}^{x} F_{X_1}(t) - F_{X_2}(t)\, dt \leq 0 \tag{2}$$

for all values of x.

Our goal is hence to find a complete solution X^* with the most stochastically dominating overall reward function, i.e.:

$$f_{1+2+\ldots+m}(X^*) \not\prec f_{1+2+\ldots+m}(X) \tag{3}$$

Notice however that there may exist functions that do not dominate each other. As such, there can be a number of *pareto-optimal* solutions, where a pareto-optimal solution is a solution that is not dominated by any other solutions in the set.

Unfortunately, the agents in SDCOPs cannot compare the dominance of overall reward functions without transmitting the individual functions to a centralized agent. Therefore, we compute the dominance of the individual reward functions and find a complete solution X^* such that

$$\forall i : f_i(x^*_{i_1}, x^*_{i_2} \mid x^*_{i_j} \in X^*) \not\prec f_i(x_{i_1}, x_{i_2} \mid x_{i_j} \in X) \tag{4}$$

which implies Equation (3) according to Theorem 1, which we will prove in Section 5.4.

Moreover, it is guaranteed that for any arbitrary risk function, the optimal solution (i.e. the solution that maximizes the expected reward for that risk function) is a pareto-optimal solution [13]. Thus, finding the pareto set is useful in applications where the risk-averse function is not known *a priori* and users want to hedge against all possible risk functions. And once the risk function is known, we can find the optimal solution by evaluating only the pareto-optimal solutions (rather than all possible solutions).

5. SD-DPOP

We now introduce Stochastic Dominance DPOP (SD-DPOP), an extension of DPOP, to solve SDCOPs. The objective of SD-DPOP is to find the set of pareto-optimal solutions (when the risk function is not known *a priori*) and

x_1	x_2	Reward Functions
0	0	dom($\mathcal{N}$(5+ 5, 1+1), $\mathcal{N}$(8+8, 1+1)) = $\mathcal{N}$(16, 2)
0	1	dom($\mathcal{N}$(5+20, 1+1), $\mathcal{N}$(8+3, 1+1)) = $\mathcal{N}$(25, 2)
1	0	dom($\mathcal{N}$(20+ 5, 1+1), $\mathcal{N}$(3+8, 1+1)) = $\mathcal{N}$(25, 2)
1	1	dom($\mathcal{N}$(20+20, 1+1), $\mathcal{N}$(3+3, 1+1)) = $\mathcal{N}$(40, 2)

(a)

x_1	Reward Functions
0	dom($\mathcal{N}$(5+16, 1+2), $\mathcal{N}$(8+25, 1+2)) = $\mathcal{N}$(33, 3)
1	dom($\mathcal{N}$(20+25, 1+2), $\mathcal{N}$(3+40, 1+2)) = $\mathcal{N}$(45, 3)

(b)

Table 2: UTIL Phase Computations in SD-DPOP

Algorithm 1: SD-DPOP

```
   /* Phase 1: Pseudo-tree Generation Phase */
1  Generate pseudo-tree
   /* Phase 2: UTIL Phase */
2  if $x_i$ is a leaf node then
3      $UTIL_{x_i} \leftarrow CalcUtils()$;
4      Send $UTIL$ message ($UTIL_{x_i}$) to parent
5  end
6  Activate UTILMessageHandler( )
   /* Phase 3: VALUE Phase */
7  Activate VALUEMessageHandler( )
```

Procedure UTILMessageHandler($UTIL_{x_k}$)

```
8  Store $UTIL_{x_k}$
9  if received UTILmessage from every child then
10     if $x_i$ is a root node then
11         $v_i^* \leftarrow ChooseBestValue(NULL)$
12         $VALUE_{x_i} \leftarrow \{(x_i, v_i^*)\}$
13         Send $VALUE$ message ($VALUE_{x_i}$) to every child
14     else
15         $UTIL_{x_i} \leftarrow CalcUtils()$
16         Send $UTIL$ message ($UTIL_{x_i}$) to parent
17     end
18 end
```

Procedure VALUEMessageHandler($VALUE_{x_k}$)

```
19 $VALUE_{x_i} \leftarrow VALUE_{x_k}$
20 $v_i^* \leftarrow ChooseBestValue(VALUE_{x_i})$
21 $VALUE_{x_i} \leftarrow VALUE_{x_i} \cup \{(x_i, v_i^*)\}$
22 Send $VALUE$ message ($VALUE_{x_i}$) to every child
```

Function CalcUtils( )

```
23 $UTIL_{x_i} \leftarrow$ Reward functions for all value combinations
       of $x_i$, its parent and pseudo-parents
24 $UTIL_{x_i} \leftarrow Join(UTIL_{x_i}, UTIL_{x_c})$ for all children $x_c$
25 $UTIL_{x_i} \leftarrow Project(x_i, UTIL_{x_i})$
26 Return $UTIL_{x_i}$
```

then find an optimal solution for the risk function (once the function is known). It finds the set of pareto-optimal solutions using stochastic dominance. It finds an optimal solution by evaluating the set of pareto-optimal solutions.

At a high level, SD-DPOP is similar to DPOP except that the agents convolve the reward functions instead of sum the rewards, choose values with the dominating convolved reward function instead of values with the maximum reward, and potentially find multiple pareto-optimal solutions instead of one reward maximizing solution.

5.1 High Level Description

Algorithm 1 shows the pseudo-code of a simplified version of SD-DPOP, where we assume that there is only one stochastically dominating solution to ease readability.[1] Like DPOP, there are also three phases in the operation of SD-DPOP. The first phase [Line 01] is identical to that of DPOP. The second phase [Lines 2-6] is similar to that of DPOP except for three differences:

(1) The function *Join*() [Line 24] joins all constraints involving x_i by setting the reward function for each value combination in the joined constraint to the convolution of the individual reward functions. In DPOP, the reward for each value combination is the sum of the individual rewards. In our example problem, agent a_3 convolves $g_{13} = f_{13}(x_1 = 0, x_3 = 0)$ and $g_{23} = f_{23}(x_2 = 0, x_3 = 0)$ to determine the reward function when $x_1 = 0$, $x_2 = 0$ and $x_3 = 0$:

$$\begin{aligned}\int_{-\infty}^{\infty} g_{13}(s) g_{23}(t-s)\, ds &= (g_{13} * g_{23})(t) \\ &= \mathcal{N}(5,1) * \mathcal{N}(5,1) \\ &= \mathcal{N}(5+5, 1+1) \\ &= \mathcal{N}(10,2)\end{aligned}$$

(2) The function *Project*() [Line 25] projects the joined constraint down on x_i by setting the reward function for each value combination in the projected constraint to the stochastically dominating reward function between all corresponding functions in the pre-projected constraints. In DPOP, the reward for each value combination is maximum over all corresponding rewards in the pre-projected constraints. In our example problem, agent a_3 sets the reward function for $x_1 = 0$, $x_2 = 0$ to the stochastically dominating reward functions between the functions for $x_1 = 0, x_2 = 0, x_3 = 0$ (which is $\mathcal{N}(5+5, 1+1) = \mathcal{N}(10, 2)$) and $x_1 = 0$, $x_2 = 0, x_3 = 1$ (which is $\mathcal{N}(8+8, 1+1) = \mathcal{N}(16, 2)$). Tables 2(a) and 2(b) show the Project() computations of agent a_3 and a_2, respectively, where the function *dom*() returns the stochastically dominating solution.

(3) The agents send the projected convolved reward functions instead of the maximum summed rewards to their parent agents in UTIL messages.

Thus, by the end of the second phase, the root agent knows

[1]One can easily extend this simplified version to find multiple pareto optimal solutions by having each agent send in its UTIL message a vector of reward functions in its pareto set, and (b) send in its VALUE message a vector of values corresponding to its value for each reward function in its pareto set.

the overall reward function of each pareto-optimal solution. Once the risk function is known, the root agent evaluates the overall reward function of each pareto-optimal solution to find an optimal solution for the given risk function. It then starts the third phase.

The third phase [Line 7] is similar to that of DPOP except that the function *ChooseBestValue()* [Lines 11 and 20] returns the value of the optimal solution. In DPOP, the function returns the value with the maximum reward. In our example problem, there is only one pareto-optimal solution and, thus, that solution is also the optimal solution for any given risk function. Agent a_1 thus chooses its optimal value 1 for its variable and sends this information down to its child agent a_2 in a VALUE message.[2] Upon receiving the VALUE message from its parent agent, agent a_2 determines that the optimal value for its variable assuming that $x_1 = 1$ is 0. Finally, upon receiving the VALUE messages from its parent agent, agent a_3 determines that the optimal value for its variable assuming that $x_1 = 1$ and $x_2 = 0$ is 0.

5.2 Implementation Details

We now describe how one implement the convolution and stochastic dominance determination of continuous and discrete reward functions:

Case 1: The reward functions are continuous. One can use the *conv*() and *int*() functions in Matlab to convolve two functions and compute integrals to check for stochastic dominance, respectively. However, if the reward functions are Gaussian functions, one can more efficiently convolve two functions – $\mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) = \mathcal{N}(\mu_1, \sigma_1^2) * \mathcal{N}(\mu_2, \sigma_2^2)$ – and check the stochastic dominance of two functions – $\mathcal{N}(\mu_1, \sigma_1^2) \succ \mathcal{N}(\mu_2, \sigma_2^2)$ if $\mu_1 \geq \mu_2$ and $\sigma_1 \leq \sigma_2$ with at least either one being a strict inequality. Lastly, instead of checking all pairs of functions for stochastic dominance, one can also optimize this check by using the property that if $f_i \succ f_j$ and $f_j \succ f_k$, then $f_i \succ f_k$.

Case 2: The reward functions are discrete and are represented by samples in discretized bins. To convolve two reward functions, one can create a new sample whose value is the sum of a pair of samples, one from each reward function, for all possible pairs of samples. The new samples represent the convolved reward function. However, such a method does not scale up well with the number of samples since the number of operations is quadratic in the number of samples. We thus describe an optimized method whose number of operations is quadratic in the number of bins. Algorithm 2 shows the pseudocode, where function $Sample(k, f_i(j))$ returns the k-th sample in the j-th bin of function f_i and $n^j_{f_i}$ is the number of samples in that bin. We now associate a probability $P(x)$ with each sample x, which we use to compute the probability $P(f_i(j))$ and mean $\mu(f_i(j))$ of each bin j of each function f_i [Lines 1-4], which we use to create new samples [Lines 5-11]. One can further optimize the algorithm by ignoring empty bins and merging neighboring bins if the probability of one of the bin is less than a threshold.

To determine the stochastic dominance of two reward functions f_1 and f_2, one first computes the cumulative distribution function $F_i(t) = \sum_{k=1}^{t} f_i(k)$ of each reward function f_i. One then checks whether the following condition hold for all values of x: $\sum_{t \leq x} F_1(t) - F_2(t) \geq 0$. If so, then the reward function f_1 stochastically dominates f_2.

Algorithm 2: Convolve(f_1, f_2)

```
1  foreach function $f_i$ and bin $j$ do
2      $P(f_i(j)) \leftarrow \sum_{k=1}^{n^j_{f_i}} P(Sample(k, f_i(j)))$
3      $\mu(f_i(j)) \leftarrow \frac{\sum_{k=1}^{n^j_{f_i}} P(Sample(k, f_i(j))) \cdot Sample(k, f_i(j))}{P(f_i(j))}$
4  end
5  foreach bin $i$ of function $f_1$ do
6      foreach bin $j$ of function $f_2$ do
7          Create a new sample $x \leftarrow \mu(f_1(i)) + \mu(f_2(j))$
8          $P(x) \leftarrow P(f_1(i)) \cdot P(f_2(j))$
9          Place $x$ in the appropriate bin of the convolved
           function
10     end
11 end
```

5.3 Complexity Analysis

In DPOP, VALUE messages contain only the value of the sending agent and UTIL messages contain a reward value for each combination of values of the parent and pseudo-parent of the sending agent. Thus, its message complexity is $O(maxDom^w)$, where $maxDom = \arg\max_i |d_i|$ and w is the induced width of the pseudo-tree.

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-parent of the sending agent. Thus, if the reward functions are continuous and one can represent the function analytically with a constant number of parameters, such as the Gaussian function, which can be represented just by the mean and variance, then the message complexity is $O(p \cdot maxDom^w)$, where p is the size of the pareto set. If the reward functions are discrete and are represented by samples in discretized bins, then the message complexity is $O(b \cdot p \cdot maxDom^w)$, where b is the largest number of bins used to represent one reward function.

Therefore, like DPOP, SD-DPOP also suffers from an exponential memory requirement. However, one can likely extend SD-DPOP to make it memory-bounded, similar to how researchers have bounded the memory of DPOP in extensions like MB-DPOP [19] and PC-DPOP [20].

5.4 Correctness and Completeness

The completeness of SD-DPOP follow quite trivially from that of DPOP since the main differences are that agents now convolve reward functions instead of sum up reward values and it chooses stochastically dominating reward functions instead of the maximal reward value.

[2]In the more complicated case where there might be multiple pareto-optimal solutions, each agent stores all the reward functions sent by its child agents and sends back the reward function corresponding to the optimal solution down to its child agents in VALUE messages. To reduce memory and message complexity, the agents can store and send hash functions of the reward functions instead of the actual reward functions.

We now prove the correctness of SD-DPOP. Let f_{i+j} denote the convolution of reward functions f_i and f_j and F_i denote the cumulative distribution function of f_i.

LEMMA 1. *For any two arbitrary reward functions f_i and f_j, $F_{i+j}(t) = \int_{-\infty}^{\infty} F_j(t-x)\, f_i(x)\, dx$*

PROOF: Let f_i and f_j be two arbitrary reward functions.

$$\begin{aligned} F_{i+j}(t) &\overset{\text{def}}{=} \int_{-\infty}^{t} f_{i+j}(y)\, dy \\ &= \int_{-\infty}^{t}\int_{-\infty}^{\infty} f_j(y-x)\, f_i(x)\, dx\, dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{t} f_j(y-x)\, f_i(x)\, dy\, dx \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{t} f_j(y-x)\, dy\, f_i(x)\, dx \\ &= \int_{-\infty}^{\infty} F_j(t-x)\, f_i(x)\, dx \qquad \blacksquare \end{aligned}$$

LEMMA 2. *For any three arbitrary reward functions f_i, f_j and f_k, if $f_i \succ f_j$, then $f_{i+k} \succ f_{j+k}$.*

PROOF: Let f_i, f_j and f_k be three arbitrary reward functions where $f_i \succ f_j$.

$$\begin{aligned} &\int_{-\infty}^{z} F_{i+k}(t) - F_{j+k}(t)\, dt \\ &\quad = \int_{-\infty}^{z}\int_{-\infty}^{\infty} \big(F_i(t-x) - F_j(t-x)\big) f_k(x)\, dx\, dt \\ &\qquad\qquad \text{(Lemma 1)} \\ &\quad = \int_{-\infty}^{\infty} f_k(x) \int_{-\infty}^{z} F_i(t-x) - F_j(t-x)\, dt\, dx \\ &\quad = \int_{-\infty}^{\infty} f_k(x) \int_{-\infty}^{z-x} F_i(y) - F_j(y)\, dy\, dx \end{aligned}$$

We know that (1) $\int_{-\infty}^{k} F_i(y) - F_j(y)\, dy \leq 0$ for all values of k according to Equation 2 since $f_i \succ f_j$ and (2) $\int_{-\infty}^{\infty} f_k(x)\, dx \geq 0$ since it is an integral of a probability distribution function. Combining both inequalities, we get $\int_{-\infty}^{\infty} f_k(x) \int_{-\infty}^{z-x} F_i(y) - F_j(y)\, dy\, dx \leq 0$, which implies that $f_{i+k} \succ f_{j+k}$ according to Equation 2. $\blacksquare$

THEOREM 1. *For any $2m$ arbitrary reward functions $f_{i_1}, \ldots, f_{i_m}, f_{j_1}, \ldots, f_{j_m}$, if $f_{i_k} \succ f_{j_k}$ for all values of k, then $f_{i_1+\ldots+i_m} \succ f_{j_1+\ldots+j_m}$.*

PROOF: Let $f_{i_1}, \ldots, f_{i_m}, f_{j_1}, \ldots, f_{j_m}$ be arbitrary reward functions where $f_{i_k} \succ f_{j_k}$ for all values of k.

$$\begin{aligned} f_{i_1+\ldots+i_m} &\succ f_{j_1+i_2+i_3+\ldots+i_m} && (f_{i_1} \succ f_{j_1} \text{ and Lemma 2}) \\ &\succ f_{j_1+j_2+i_3+\ldots+i_m} && (f_{i_2} \succ f_{j_2} \text{ and Lemma 2}) \\ &\succ \cdots \\ &\succ f_{j_1+j_2+j_3+\ldots+j_m} && (f_{i_m} \succ f_{j_m} \text{ and Lemma 2}) \end{aligned}$$

Thus, $f_{i_1+\ldots+i_m} \succ f_{j_1+\ldots+j_m}$. $\blacksquare$

Therefore, since convolving stochastically dominating individual reward functions result in a stochastically dominating overall reward function according to Theorem 1, SD-DPOP is correct.

6. RELATED WORK

Motivated by risk-sensitive applications, other researchers have also independently investigated the use of dominance to solve SDCOPs. For example, Stranders *et al.* defined dominance with respect to a given risk function, where a random variable X dominates a random variables Y for a given function U iff $\mathbb{E}[U(X+Z)] \geq \mathbb{E}[U(Y+Z)]$ for all possible random variables Z with strict inequality for at least one Z [22]. They then derived sufficient and necessary conditions for such dominance to hold for one example risk function, proposed U-GDL, an extension of the GDL algorithm that uses the new conditions, and experimentally demonstrated the benefits of their approach. Our work is different from theirs in that SD-DPOP uses a stochastic dominance criteria, which is applicable for the class of concave risk functions.

Another piece of related work is the Multi-Objective Max-Sum (MOMS) algorithm by Delle Fave *et al.*, which is an algorithm that finds pareto-optimal solutions for a set of objective functions [4]. Our work is different from theirs in that MOMS needs to know the set of risk functions (set of objectives) prior to finding the pareto-optimal solutions while SD-DPOP does not. SD-DPOP actually finds the set of pareto-optimal solutions for all possible risk functions, which is an infinite set. However, MOMS can find pareto-optimal solutions for arbitrary utility functions, while SD-DPOP can only do so for concave risk functions. However, we expect all these extensions to easily apply to all GDL-based DCOP algorithms including DPOP, Max-Sum [5] and Action-GDL [25].

7. EXPERIMENTAL EVALUATION

We now compare the stochastically dominating solutions found by the SD-DPOP algorithm to the solutions that maximize the expected reward found by DPOP. We run our experiments on a MacBook Pro with a 2.7GHz Intel Core i7 and 8GB memory. We use the same sensor network problem that we used as our motivating domain, where we arranged the sensors in a grid and each sensor can move in the four cardinal directions or stay stationary. Thus, each sensor has 5 possible values. Additionally, each sensor can be constrained with any one of its four neighboring sensors. We varied the size of the problem by increasing the number of sensors in the grid and the number of constraints. Each constraint between two sensors consists of 25 reward functions corresponding to the 25 pairs of possible values of the two sensors. Each reward function is a Gaussian function whose mean and variance are randomly generated between the ranges of 80 to 100 and 0 to 80, respectively.

We use the following risk functions to evaluate the utilities of the solutions found:

$$g_1(x, \alpha, th) = \begin{cases} \alpha\, x - (\alpha - 1)\, th & x \leq th \\ th & \textit{otherwise} \end{cases}$$

$$g_2(x, \alpha, th) = -\exp(-\alpha\,(x - th)) + th$$

Both functions are commonly used risk functions in the literature [13, 21]. In both functions, the parameter α represents the level of risk aversion. The larger the value of α, the higher the level of risk aversion. The parameter th represents the threshold after which one obtains close to no utility with increasing x.

We run experiments for both continuous and discretized

(a) SD-DPOP with Stochastic Dominating Solutions

no. of	no. of	runtime	msg size	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
agents	constraints	(ms)	(kB)	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
9	12	539	368	1391	92	1160	1412	103	1292	1369	84	1194	1390	91	1199
16	24	2588	7240	2613	142	2176	2611	141	2395	2542	115	-3261	2571	123	2397
25	37	81821	65963	4055	177	3619	4058	179	3698	3971	151	-1596	4004	159	3698

(b) DPOP with Expected Reward Maximizing Solutions

no. of	no. of	runtime	msg size	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
agents	constraints	(ms)	(kB)	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
9	12	100	75	1455	167	949	1455	167	1283	1455	167	-2.13E8	1455	167	11260
16	24	172	139	2689	225	1840	2689	225	2389	2689	225	-1.66E14	2689	225	-7336
25	37	252	215	4162	278	3397	4162	278	3694	4162	278	-1.34E16	4162	278	-775615

Table 3: Experimental Results for Continuous Reward Functions

(a) SD-DPOP with Stochastic Dominating Solutions for 9 Agents and 9 Constraints

no. of	runtime	msg size	error	error	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
samples	(ms)	(kB)	in μ	in σ	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
10	3203	2417	91	185	971	86	373	971	86	889	962	80	-1472	965	81	897
100	7775	3032	25	44	971	73	544	972	74	893	959	67	849	965	69	898
1000	12786	2909	30	41	972	74	546	972	74	893	958	66	847	968	71	898
10000	15802	2803	31	43	972	74	546	972	75	893	957	66	854	966	70	898

(b) SD-DPOP with Stochastic Dominating Solutions for 16 Agents and 16 Constraints

no. of	runtime	msg size	error	error	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
samples	(ms)	(kB)	in μ	in σ	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
10	35845	9458	340	258	1751	132	1143	1751	132	1591	1741	128	-2.76E7	1741	127	1562
100	58125	9792	65	85	1750	105	1399	1750	105	1596	1721	93	1092	1736	97	1599
1000	69050	9088	80	59	1751	106	1402	1750	105	1596	1719	91	1057	1734	97	1599
10000	79235	8869	79	58	1750	104	1403	1750	105	1596	1717	90	1178	1734	96	159

Table 4: Experimental Results for Discretized Reward Functions

reward functions. Tables 3 and 4 show our experimental results, where we average over 50 problem instances for each configuration. For each problem instance, we run SD-DPOP to find the pareto set, and for each risk function, we evaluate the risk of a pareto-optimal solution by taking 10,000 samples of the overall reward function for that solution and evaluating those samples using the risk function. We then return the solution with the largest mean as the optimal solution. The root agent performs this sampling process since it knows the overall reward function for each pareto-optimal solution. After it determines the optimal solution, it propagates that information down the pseudo-tree to the other agents.

For each problem configuration, we use four risk functions: $g_1(x, 50, th)$, $g_1(x, 0, th)$, $g_2(x, 0.05, th)$ and $g_2(x, 0.025, th)$, where we set th to 100 times the number of constraints in the problem. For each risk function, we report the mean (μ) and standard deviation (σ) of the overall reward function of the optimal solution as well as the mean (μ_R) of the evaluation of overall reward function using the risk function. We also report the error in the mean and standard deviation (compared against the true mean and standard deviation) of the overall reward function of the optimal solution for experiments with discretized reward functions. We make the following observations:

- The runtime and message size of DPOP and SD-DPOP increases with the number of agents and constraints, which is to be expected. The runtime and message size of SD-DPOP is larger than that of DPOP. The runtime is larger because agents in SD-DPOP find multiple pareto-optimal solutions while agents in DPOP find only one solution. The message size is larger because agents in SD-DPOP sends more information in each message, as described in Section 5.3.
- The means (μ) of the expected reward maximizing solutions are larger than the means of the stochastic dominating solutions, which is to be expected. However, the means (μ_R) of the evaluations of the expected reward maximizing solutions using the risk functions are smaller than those of the stochastic dominating solutions, which implies that the solutions found by SD-DPOP is less risky. The difference in the means increases as α increases. Thus, the higher the level of risk aversion, the more important it is to find stochastic dominating solutions. An interesting future direction of research would be to compute a theoretical bound between μ and μ_R. However, this bound would be useful for risk functions with small values of α only. The reason is that if α is large or, equivalently, the risk function has an almost vertical slope, then μ is likely to be finite while μ_R is likely to be negative infinity.
- The runtime of SD-DPOP increases with the number of samples, which is to be expected. The message size increases as we increase the number of samples from 10 to 100. The reason is that the number of bins needed by the samples increases. However, the message size

does not increase further, and can sometimes decrease instead, as we increase the number of samples to 1000 or 10000. The reason is that as the number of samples increases significantly, the samples all converge near the mean of the distribution. As a result, the probability of bins at the tail ends become small. Since we optimize the number of bins by merging neighboring bins if the probability of one of the bin is less than a threshold of 0.001, these bins at each of the tail ends are merged into a single large bin.

- The errors in the mean and standard deviation decrease as we increase the number of samples from 10 to 100. The reason is that 10 samples is too small to accurately represent the reward function. However, the errors do not decrease further, and actually increase instead, as we increase the number of samples to 1000 or 10000. The reason is that 100 samples is sufficient to accurately represent the reward function and the increase in error is likely due to sampling approximations.

8. CONCLUSIONS

The Stochastic DCOP (SDCOP) model is useful in modeling multi-agent coordination problems where the constraint rewards are sampled from known reward functions. The goal of SDCOPs is to find a solution that maximizes the expected reward in such problems. However, these solutions might be very risky and hence unacceptable in risk-sensitive applications. In this paper, we make three contributions: (1) we propose a stricter objective for SDCOPs, namely to find a solution with the most stochastically dominating reward function; (2) we introduce SD-DPOP, an extension of DPOP that finds such solutions; and (3) we show that stochastically dominating solutions can indeed be less risky than expected reward maximizing solutions. For future work, we would like to bound the memory used by SD-DPOP in the same way that researchers have done for MB-DPOP and PC-DPOP. We believe that this is important since the exponential requirement on memory prohibits the use of this algorithm in large scale problems.

9. ACKNOWLEDGMENT

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. We would also like to thank our anonymous reviewers for their comments and suggestions, especially for pointing us to the work on the U-GDL and MOMS algorithms.

10. REFERENCES

[1] J. Atlas and K. Decker. Coordination for uncertain outcomes using distributed neighbor exchange. In *Proc. of AAMAS*, pages 1047–1054, 2010.

[2] D. Burke and K. Brown. Efficiently handling complex local problems in distributed constraint optimisation. In *Proc. of ECAI*, pages 701–702, 2006.

[3] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[4] F. Delle Fave, R. Stranders, A. Rogers, and N. Jennings. Bounded decentralised coordination over multiple objectives. In *Proc. of AAMAS*, pages 371–378, 2011.

[5] A. Farinelli, A. Rogers, A. Petcu, and N. Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *Proc. of AAMAS*, pages 639–646, 2008.

[6] S. Graves and J. Ringuest. Probabilistic dominance criteria for comparing uncertain alternatives: A tutorial. *Omega*, 37(2):346–357, 2009.

[7] Y. Hamadi, C. Bessière, and J. Quinqueton. Distributed intelligent backtracking. In *Proc. of ECAI*, pages 219–223, 1998.

[8] M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *Proc. of IJCAI*, pages 181–186, 2009.

[9] S. Kozono. Received signal-level characteristics in a wide-band mobile radio channel. *IEEE Transactions on Vehicular Technology*, 43(3):480–486, 1994.

[10] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimization with structured resource constraints. In *Proc. of AAMAS*, pages 923–930, 2009.

[11] R. Lass, J. Kopena, E. Sultanik, D. Nguyen, C. Dugan, P. Modi, and W. Regli. Coordination of first responders under communication and resource constraints (Short Paper). In *Proc. of AAMAS*, pages 1409–1413, 2008.

[12] T. Léauté and B. Faltings. $\mathbb{E}$[DPOP]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling. In *Proc. of DCR*, pages 87–101, 2009.

[13] H. Levy. *Stochastic Dominance Investment Decision Making under Uncertainty Studies in Risk and Uncertainty.* Springer, 1998.

[14] V. Lisý, R. Zivan, K. Sycara, and M. Péchoucek. Deception in networks of mobile sensing agents. In *Proc. of AAMAS*, pages 1031–1038, 2010.

[15] R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *Proc. of AAMAS*, pages 310–317, 2004.

[16] M. McClure, D. Corbett, and D. Gage. The DARPA LANdroids program. *SPIE*, 7332:73320A, 2009.

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

[18] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proc. of IJCAI*, pages 1413–1420, 2005.

[19] A. Petcu and B. Faltings. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In *Proc. of IJCAI*, pages 1452–1457, 2007.

[20] A. Petcu, B. Faltings, and R. Mailler. PC-DPOP: A new partial centralization algorithm for distributed optimization. In *Proc. of IJCAI*, pages 167–172, 2007.

[21] J. Pratt. Risk aversion in the small and in the large. *Econometrica*, 32(1–2):122–136, 1964.

[22] R. Stranders, F. Delle Fave, A. Rogers, and N. Jennings. U-GDL: A decentralised algorithm on DCOPs with uncertainty. Technical report, Department of Electronics and Computer Science, University of Southampton, 2011.

[23] M. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When should there be a "me" in "team"?: Distributed multi-agent optimization under uncertainty. In *Proc. of AAMAS*, pages 109–116, 2010.

[24] S. Ueda, A. Iwasaki, and M. Yokoo. Coalition structure generation based on distributed constraint optimization. In *Proc. of AAAI*, pages 197–203, 2010.

[25] M. Vinyals, J. Rodríguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the Generalized Distributive Law. *Autonomous Agents and Multi-Agent Systems*, 2010.