Using DCOPs to Balance Exploration and Exploitation in Time-Critical Domains

Matthew E. Taylor, Manish Jain, Prateek Tandon, and Milind Tambe
http://teamcore.usc.edu/

The University of Southern California

Abstract. Substantial work has investigated balancing exploration and exploitation, but relatively little has addressed this tradeoff in the context of coordinated multi-agent interactions. This paper introduces a class of problems in which agents must maximize their on-line reward, a decomposable function dependent on pairs of agent's decisions. Unlike previous work, agents must both learn the reward function and exploit it on-line, critical properties for a class of physically-motivated systems, such as mobile wireless networks. This paper introduces algorithms motivated by the *Distributed Constraint Optimization Problem* framework and demonstrates when, and at what cost, increasing agents' coordination can improve the global reward on such problems.

1 Introduction

While there has been significant work studying the exploration vs. exploitation tradeoff, relatively little focuses on the more difficult multi-agent case. In this paper, we define a class of problems, *Distributed Coordination of Exploration and Exploitation* (D-CEE), where agents must balance coordinated exploration of an unknown environment with exploitation of its rewards. Methods in this paper assume a decomposable reward function (e.g., inter-agent constraints). Examples of such real world domains include network routing optimization, repeated task allocation and mobile network optimization (detailed in Section 2).

D-CEE problems extend the *Distributed Constraint Optimization Problem* (DCOP) formulation (Mailler & Lesser, 2004; Petcu & Faltings, 2005; Modi et al., 2005). Like DCOPs, D-CEEs are defined as collaborative problems in which the global reward function can be decomposed in terms of agent pairs (similar to a *coordination graph* (Guestrin et al., 2001)). The reward for each pair is called a *constraint* and depends on values that each agent has selected. The primary challenges are: 1) the reward function is unknown, 2) algorithms have limited time to run, making exhaustive exploration infeasible, 3) on-line reward is critical, and 4) agents must coordinate to explore or exploit.

Our problem formulation differs from existing work along multiple dimensions. Agents know their immediate constraints but not their reward function, whereas DCOPs

In *the annual workshop on Distributed Constraint Reasoning at IJCAI-09* (DCR-09), Pasadena, CA, July 2009.

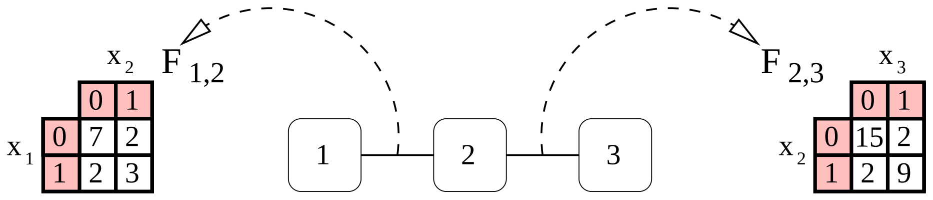


Fig. 1. This figure depicts three agents in a D-CEE. Each agent can select a value (0 or 1 in this example), and the global reward depends on pairs of agents' selected values.

assume the rewards are known. Additionally, we assume there are too many possible combinations of agent values for exhaustive exploration. D-CEE agents seek to maximize the global on-line (i.e., cumulative) reward over an episode of finite length, whereas DCOP and optimal stopping problems (c.f., the *Secretary Problem* (Freeman, 1983)) only maximize final reward. While single-agent approaches to exploration vs. exploitation may provide insight into D-CEE problems, this problem is more difficult because agents must coordinate. For instance, single-agent solutions (c.f., multi-armed bandits (Robbins, 1952)) do not account for the intrinsic coordinated nature of the problem. Because the reward function depends on *pairs* of agents, coordination is *critical* (see Figure 1, further explained in Section 2.1).

This paper introduces algorithms that differ along four dimensions. First, algorithms may be reactive or predictive. Second, in each *neighborhood* of the constraint network (any agent which shares a constraint with agent *i* is considered a neighbor of *i*), either a single agent may explore, or multiple agents may simultaneously explore. Third, algorithms are designed to account for agents with different abilities: those that can return to a previously visited setting and those that cannot. Fourth, in some situations, agents may know the experiment length *a priori*, while in others they may only know the probability of the experiment ending on the next time step.

Experiments on a simulated D-CEE domain demonstrate our algorithms' ability to successfully maximize the cumulative reward. Performance is analyzed over different graph topologies, numbers of agents, and experiment durations. In general, additional coordination among agents results in increased reward due to an increased amount of exploration. Surprisingly, sometimes the additional coordination *reduces* the total reward (not just the communication overhead), and we explain this counterintuitive result. After this analysis we introduce a set of extensions to demonstrate additional power and flexibility in the algorithms, discuss additional related work, and then conclude.

2 Domain

This section defines the D-CEE problem class and describes the *mobile wireless network* problem used as this paper's experimental testbed.

2.1 Problem Definition

This section introduces the terminology used to discuss D-CEE domains, which is based heavily on the DCOP formulation. A D-CEE consists of a set V of n variables,

$\{x_1, x_2, \ldots, x_n\}$, assigned to a set of agents, where each agent controls one variable's assignment. Variable x_i can take on any value from the finite domain D_i. The goal is to choose values for the variables such that the sum over a set of binary constraints and associated payoff or reward functions, $f_{ij} : D_i \times D_j \rightarrow N$, is maximized. More specifically, find an assignment, A, s.t. F(A) is maximized: $F(A) = \sum_{x_i, x_j \in V} f_{ij}(d_i, d_j)$, where $d_i \in D_i, d_j \in D_j$ and $x_i \leftarrow d_i, x_j \leftarrow d_j \in A$. Take the constraints in Figure 1 as an example. x_1, x_2, and x_3 are variables, each with a domain of $\{$0,1$\}$ and the global reward function as shown. If all agents choose the value 1, the total solution quality of this complete assignment is 12, which is locally optimal as no single agent can change its value to improve its reward (or the global reward). $F((x_1 \leftarrow 0), (x_2 \leftarrow 0), (x_3 \leftarrow 0))$ $= 22$ and is globally optimal.

The agents in a DCOP are traditionally assumed to have *a priori* knowledge of the corresponding reward functions. In order to more flexibly model a class of real world domains, D-CEE problems do not make this assumption. Thus, D-CEE problems appear similar to DCOPs, but with the following features absent from DCOPs: (1) agents initially know the constraint graph but only discover rewards when a pair of agents set their values to explicitly discover a reward value, (2) problems last a set amount of time, and (3) the agents' seek to maximize the on-line global reward over this time horizon T.

Existing DCOP algorithms can be divided into two classes. Complete algorithms find the optimal solution but are NP-hard and do not scale to large domains due to computation and communication costs (Modi et al., 2005). Incomplete algorithms scale much better but only provide locally optimal solutions. In particular, *k-optimal* algorithms guarantee that the final solution can only be improved if more than k agents change values simultaneously (Maheswaran et al., 2004; Pearce & Tambe, 2007). As k increases, communication and computation increase, but higher values of k typically result in higher final (instantaneous) rewards. How to best trade off k with performance considerations is an open problem — we investigate how D-CEE performance differs when extending $k = \{1, 2\}$ DCOP algorithms.

2.2 Problem Domain

A D-CEE approach is warranted when a set of agents need to coordinate to maximize their shared reward but are unable to explore the entire space of their possible assignments. Such a setting will most likely occur when there are many agents, a large number of possible assignments, and a limited amount of time (i.e., a set number of variable assignments may be explored). Furthermore, we limit ourselves to distributed approaches, which reduce communication, and improves robustness relative to centralization.

The D-CEE *mobile wireless network* problem is used as an example domain throughout this paper. During natural disasters and other ad-hoc situations, personnel may quickly form such a network by placing mobile robots in an area to relay information (e.g., about endangered victims or fires). The robots must optimize the signal strengths over the network to ensure reliable and effective communication for the duration of the task. Value settings correspond to agent locations, constraints are determined by the network topology, and rewards are signal strengths between robots.

Our formulation of the mobile wireless network problem is based on the capabilities of current low-cost physical robots (see Figure 2). An agent can measure the wireless signal strength between itself and each neighbor, corresponding to measuring the reward for the pair of agent assignments. Agents select from a set of possible physical locations (i.e., a variable assignment). Because the time for movement in physical robots dominates communication and calculation time (Jain et al., 2009), we measure experiment length by the number of *rounds*, defined by the period in which every agent may decide to move to a new position and then reach that position. All agents may choose to either `stay` in their current position or `explore`. We also consider agents which have the ability to `backtrack` to a previously explored position.

While wireless communication has predictable signal strengths when the receivers are within sight of each other, we assume a multi-path setting where *small scale fading* dominates, making signal strength prediction difficult or impossible (Molisch, 2005). Instead, agents can only determine signal strengths by moving to a location and sampling. Additionally, because such movements are small compared to the distance between robots, we assume that the constraint graph does not change over time. Signal strengths can be modeled as an independent random number drawn from some distribution, and initial experiments on physical hardware suggest that a Normal distribution is appropriate (Jain et al., 2009). However, all D-CEE algorithms are distribution-independent, allowing easy substitution by other (empirically measured) distributions.

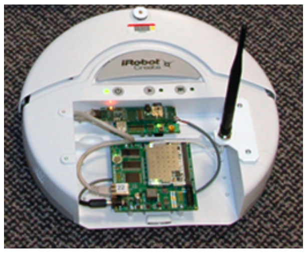


Fig. 2. iRobot Creates were successfully used to create a mobile wireless network testbed (Jain et al., 2009).

3 Algorithmic Contributions

This section describes our D-CEE algorithms. Our algorithms fall into two broad classes: static estimation (SE) and decision theoretic (i.e., balanced exploration, or BE). Our previous work in this area (Jain et al., 2009) was motivated by $k = 1$ DCOP algorithms in which agents coordinate to decide which (if any) agent in a neighborhood can select a new value (Section 3.1). Novel algorithms in Section 3.2 follow the framework of MGM-2 (Pearce & Tambe, 2007) and allow two agents in a neighborhood to coordinate a joint variable change. Section 4 will compare the performance of both sets of algorithms, as well as investigate a surprising result regarding the relative performance of $k = 1$ and $k = 2$ algorithms.

3.1 Algorithms with k = 1

MGM-Omniscient is an algorithm based on MGM (Maheswaran et al., 2004) which can be applied to D-CEE problems by artificially providing the full reward function (hence, omniscient, as no exploration is necessary). MGM-Omniscient is an upper bound on the solution quality for $k = 1$ agents. A round is defined as the period when every agent: (1) communicates its current value to all its neighbors (2) calculates and communicates its bid (the maximum *gain* in the reward it expects if allowed to change values) (3) changes its value.. An agent with the highest bid, per neighborhood, is allowed to change its value.

SE-Optimistic is a greedy approach that assumes agents will receive the maximum possible reward for unexplored value settings. **SE-Mean** is more conservative and assumes that an agent will receive the average reward for unexplored value settings. The two SE methods differ only in how they calculate the expected gain. Recall that we assume that a distribution over reward values is known.

BE-Rebid computes the expected utility of exploration based on the current reward, the number of remaining rounds, and the distribution of rewards. It assumes an agent may `backtrack`[1] to an explored value (provided its neighbors have not changed their values). Let R_b be the variable setting with the agent's highest discovered reward. The state of the agent is (R_b, T, n), where T is the number of rounds remaining in the experiment and n is the number of neighbors. Agents decide between the actions `backtrack`, with a utility of $V_{back} = R_b T$, and `explore`, with a utility of $V_{explore}(R_b, T, n) =$

$$\max_{0 \le t_e \le T} \left\{ t_e \mu(n) + t_s \int_{x > R_b} x Q(x, n, t_e) \partial x + t_s R_b F(R_b, n)^{t_e} \right\}. \tag{1}$$

The agent will bid to explore for t_e rounds and then exploit for $t_s = T - t_e$ rounds. $Q(x, n, t_e)$ gives the probability of x being the maximum reward among the t_e rewards explored computed via order statistics. $F(x)$ is the cumulative probability of drawing a sample less than or equal to x and $\mu(n)$ is the average reward over n neighbors (Full details can be found elsewhere (Jain et al., 2009)). The agent's utility of state (R_b, T, n) is:

$$V(R_b, T) = \max \left\{ V_{back}(R_b, T), V_{explore}(R_b, T, n) \right\}.$$

BE-Stay is applicable when agents are unable to backtrack. In this approach, every agent considers its current total reward and compares the expected reward it would receive if it kept the same variable assignment (V_{stay}) with the expected reward of exploring ($V_{explore}$). The reward of exploration, given the current reward R_c and T rounds remaining in the experiment is calculated as:

$$V(R_c, T, n) = \begin{cases} V_{stay}(R_c, 0) = V_{explore}(\cdot, 0, \cdot) = 0 & \text{for } T = 0 \\ \max(V_{stay}(R_c, T), V_{explore}(R_c, T, n)) & \text{for } T > 0 \end{cases}$$

The expected rewards of the two actions `stay` and `explore` are calculated recursively as:

$$V_{stay}(R_c, T) = R_c T \tag{2}$$

$$V_{explore}(R_c, T, n) = \int_{-\infty}^{\infty} Q(x, n)(V(x, T-1) + x) \partial x \tag{3}$$

[1] If the current reward equals the best reward found thus far, `backtrack` is equivalent to not changing any values.

3.2 Algorithms with k = 2

This section introduces our novel coordination algorithms, building upon the framework of MGM-2. Algorithms in this section are used to test the hypothesis that increasing coordination between agents accrues higher total reward in D-CEEs. Salient differences are summarized in Table 1.

MGM-Omniscient-2 is an implementation of MGM-2 (Maheswaran et al., 2004) which is artificially provided with the full reward matrix, similar to MGM-Omniscient. MGM-Omniscient-2 then finds a locally optimal solution where no combination of one agent or two neighboring agents can improve the global reward, representing an upper bound for $k = 2$ algorithms. A round is defined as the period in which each agent: (1) picks a neighbor and sends an *offer* for a joint variable change, based on the maximal gain. (2) for each received offer, send an *accept* or *reject* message based on whether the agents wants to pair with that particular neighbor or not. Agents choose the neighbor with the maximum offer. (3) calculate the *joint gain* of the pair if offer is accepted, otherwise calculate gain of individual change. (4) the agent with the highest gain per neighborhood may change its variable setting. If the agent was part of the pair and its offer had been accepted, then a *go* message is sent to the partnering agent. (5) execute the joint / individual assignment.

MGM-Omniscient-2 pays a higher communication cost than MGM-Omniscient, but it generally leads to a higher solution quality (Pearce & Tambe, 2007). Like MGM-Omniscient, MGM-Omniscient-2 monotonically increases its solution quality (Maheswaran et al., 2004). Note that single agents may act alone, subsuming MGM-Omniscient. The algorithms in the remainder of this section are based on this framework and thus we expect that these $k = 2$ algorithms will typically outperform their $k = 1$ counterparts, but with increased communication.

SE-Optimistic-2 makes the same assumption as SE-Optimistic: any unexplored reward is assumed to be optimal. Agents bid their gain on step 1 ($n \times R_{max} - R_c$, where R_{max} is the maximum possible reward per constraint and R_c is the current total reward) and the joint gain (step 3) is the sum of its bid with the chosen neighbor (without double-counting the shared constraint). **SE-Mean-2** makes the same assumption as SE-Mean. Agents bid the value $n \times \mu - R_c$ on step 1, but are otherwise unchanged from SE-Optimistic-2.

BE-Rebid-2 extends the BE-Rebid algorithm by allowing pairs of agents to select exploration and exploitation actions. Pairs of agents bid to (1) `explore-explore` (2) `explore-stay` (3) `stay-explore` (4) `coordinated backtrack`.[2] The `coordinated backtrack` action is unique to this algorithm, as it allows two agents to simultaneously backtrack to a previous setting — two agents simultaneously exploit their knowledge of the global reward function.

The gain of `explore` and `backtrack` actions are calculated like BE-Rebid; the utility of `explore`, or $V_{explore}(R_b, T, n)$ is given by Equation 1, and the utility of `backtrack` is again $V_{back} = R_b T$. The gain of an action is the difference between the expected utility of the action and $R_c T$. As in MGM-Omniscient-2, agents bid these

[2] An agent may not change its value if a neighbor is executing `backtrack` — `explore-backtrack` and `backtrack-explore` are not considered valid joint actions.

Table 1. This table summarizes D-CEE algorithms by the number of allowed simultaneous value changes, the type of reward information needed, whether agents can backtrack, and if agents require the rounds per experiment.

Method	$k = 2$?	Info. Required	Can Backtrack?	# Rounds
SE-Mean		μ		
SE-Mean-2	✓	μ		
SE-Opti		R_{max}		
SE-Opti-2	✓	R_{max}		
BE-Rebid		μ, σ	✓	✓
BE-Rebid-2	✓	μ, σ	✓	✓
BE-Stay		μ, σ		✓
BE-Stay-2	✓	μ, σ		✓
MGM-Omni		all rewards		
MGM-Omni-2	✓	all rewards		

individual gains to their neighbors and then agents bid to pair with the neighbor which offers the maximum gain. The utility of joint exploration by agents i and j, as calculated by agent j, is:

$$V_{exp-exp} = V_{explore:i}(R_{b:i}, T, n_i) + V_{explore:j}(R_{b:j}, T, n_j - 1)$$

which calculates the sum of two gains, not double counting the shared constraint (via $n_j - 1$). The gains of `explore-stay` and `stay-explore` are determined by the exploring agent. The gain of `coordinated backtrack` is the difference between the best joint reward of two agents and the current reward.

BE-Stay-2 may be used by agents that cannot `backtrack`. The utility of `explore` action is calculated as for BE-Stay in Section 3.1. Joint gains are calculated from a combination of these two actions. As in BE-Rebid-2, the gain of joint exploration is calculated as the sum of individual exploration bids, not double counting on the common constraint. Again, the gain of `explore-stay` and `stay-explore` is given by the gain of the exploring agent. The gain of `stay-stay` is 0, by definition. The bidding process followed is same as in BE-Rebid-2 and agents are again allowed to act individually if no coalition yields a higher expected utility.

4 Experimental Analysis

Experiments in this section compare the performance of our D-CEE algorithms in multiple instances of the mobile sensor wireless network problem. Our custom simulator allows us to generate random network topologies and random reward functions and then sequentially test our algorithms on the same problem instance. Lower and upper bounds were determined by disallowing all agent movement and using MGM-Omniscient-2, respectively. The results are reported as a scaled gain, scaled on [0,1]. Any gain greater

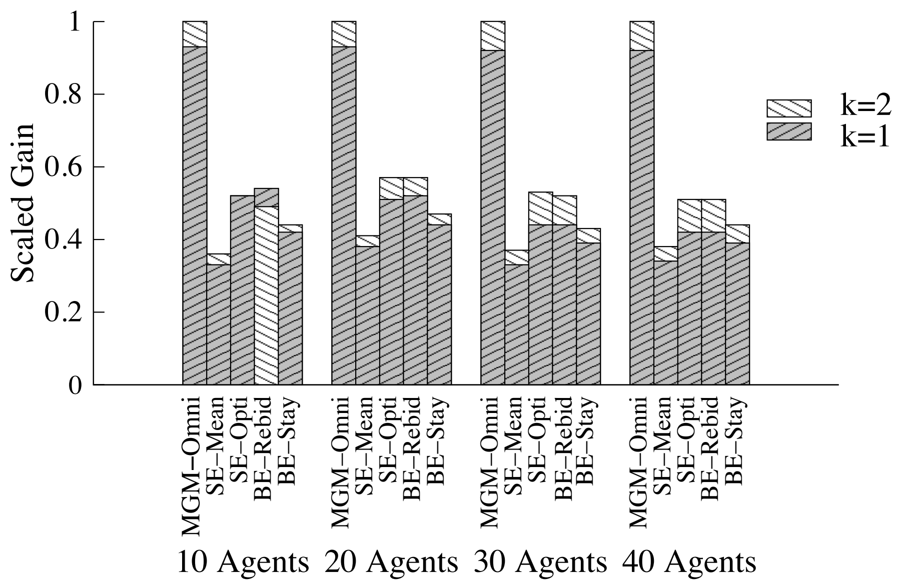


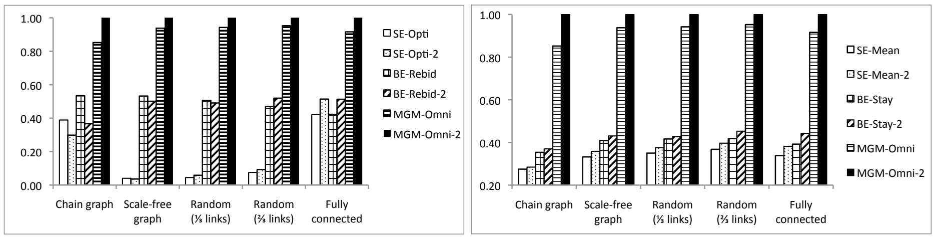
Fig. 3. This graph compares the scaled cumulative gain of different algorithms with different numbers of agents.

than zero represents an improvement in the total reward. Signal strengths used in the simulations are drawn from a normal distribution with a $\mu = 100$ and $\sigma = 16$.[3]

4.1 Scaling Tests

The results presented in Figure 3 show the relative performance of algorithms as the number of agents is varied. All trials were conducted with 100 rounds on fully connected graphs and each result is averaged over 30 independent runs. Notice that MGM-Omniscient underperforms MGM-Omniscient-2 (which scores 1.0 by definition), suggesting that $k = 2$ algorithms are able to provide improvements in D-CEE domains. In fact, the first important trend in these results is that for 20 or more agents, all $k = 2$ algorithms are statistically significantly better than the corresponding $k = 1$ algorithm (paired t-tests, $p < 0.05$). In experiments with 10 agents, BE-Rebid outperforms BE-Rebid-2, likely because there are relatively few constraints to optimize, and thus the magnitude of the variance is relatively low (see Section 4.3 for more discussion of this effect). The second trend to note is that Rebid algorithms outperform Stay algorithms, as the former allows agents to return to good configurations, improving the agents' ability to exploit their knowledge. We ran supplemental experiments (not shown) using 25, 50, 75, and 100 rounds and 20 agents, and found the relative performance unchanged.

[3] The range $\mu - 6\sigma$ to $\mu + 6\sigma$ covers 99.999% of the samples for a normal distribution; we considered signals with integral values within [0,200].



(a) Aggressive Algorithms (b) Conservative Algorithms

Fig. 4. This graph shows the scaled cumulative gain for algorithms on different network topologies. Algorithms are divided into "aggressive" and "conservative," where the former explore more of the value settings and the latter are more cautious (see the text).

4.2 Exploring Different Graph Topologies

Figure 4 shows the results when the topology of the agent network is changed. We investigate five topologies, including scale-free (i.e., the degree distribution asymptotically follows a power law) and random (where roughly $\frac{1}{3}$ or $\frac{2}{3}$ of the number of links in a fully connected graph are randomly added to the network). Algorithms with $k = 2$ often converge to a higher final reward (Maheswaran et al., 2004), due to their ability to make joint decisions. Additionally, joint decisions allow the agents to explore more combinations of values, enabling them to choose from more and potentially better options. In fact in our experiments, *all* $k = 2$ algorithms explored more entries in the reward matrix than their corresponding $k = 1$ algorithms, for all network topologies. That $k = 2$ is does not dominate other approaches is a particularly surprising result precisely because previous DCOP work showed that $k = 2$ algorithms reached higher final rewards; due to the on-line nature of D-CEE problems, $k = 1$ may sometimes be superior.

BE-Rebid and BE-Rebid-2 dominate other algorithms over all graph topologies, with the exception of full graphs, where the SE-Optimistic variants are not statistically different from the BE-Rebid variants. Also, note the precipitous drop in performance of the SE-Optimistic methods in scale-free and random graphs. As investigated elsewhere (Jain et al., 2009), agents in such graphs can have a high variance in their degree of network connectivity. Agents with many neighbors have a very large potential gain (due to the optimistic nature of the algorithm), causing them to move in the majority of rounds, to the detriment of other agents in the neighborhood.

4.3 *k=1* vs. *k=2* Analysis

Given prior results in DCOP work (Maheswaran et al., 2004; Pearce & Tambe, 2007), we expected that MGM-Omniscient-2 would always outperform MGM-Omniscient, which held true. However, $k = 2$ did not always outperform $k = 1$ on sparse graphs (i.e., scale-free, chain and random graphs with few constraints) even though $k = 2$ algorithms did learn more of the reward function.

The "aggressive exploration" of BE-Rebid-2 and SE-Optimistic-2 under-performs their $k = 1$ counterparts on many graphs due to how bids for pairs of agents are formed.

In particular, in $k = 1$ algorithms, the agent with the most to gain in a neighborhood will always be allowed to move. However, in $k = 2$ algorithms, sometimes an agent with a relatively high reward will pair with an agent with relatively low reward. While relatively high reward agents will only agree to move if it could improve the total reward, such an agent would not move in a $k = 1$ algorithm. We found that BE-Rebid-2 agents are not only more likely to change their values in general, but they are much more likely to leave a high-valued position by pairing with a neighbor, relative to BE-Rebid on chain graphs (see Figure 5). The x-axis in Figure 5 shows the instantaneous reward of the agent's assignment and the y-axis shows the percentage of times the agent changed its assignment when getting the corresponding reward. This figure demonstrates how agents with high reward can be coerced into moving out of their beneficial location by a low-valued neighbor in $k = 2$, whereas in $k = 1$ such high valued agents will move (as shown by the fact that no agents with a mean reward of 130 move in BE-Rebid). This effect is more pronounced in sparse graphs as the variance in agents' rewards is higher. We ran four sets of 30 trials of 40 agents for 100 rounds. BE-Rebid and BE-Rebid-2 agents in chain graphs found that the standard deviation of the average value per constraint was 14.2 and 15.5, respectively, while in the full graph it was 2.33 and 2.47, respectively. A higher variance leads to an increased probability that an agent with a very low reward will be able to "convince" a relatively high-valued neighbor to pair with it and change values.

4.4 Secondary Analysis

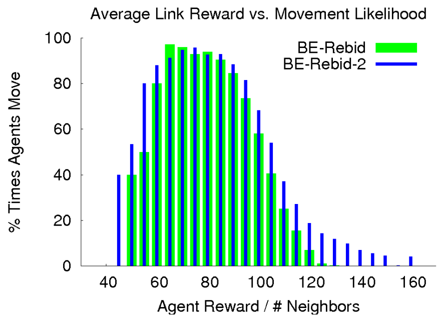


Fig. 5. This histogram shows how often a BE-Rebid agent moves vs. a BE-Rebid-2 agent, per the agent's normalized reward (total reward on its constraints divided by the number of constraints), where each bar averages five x-axis values. 30 independent trials were run on a chain topology with 40 agents for 100 rounds each.

In this section we compare the run time, memory, and communication requirements of our D-CEE algorithms. By instrumenting our simulator, we found that all $k = 2$ algorithms require more CPU time than their corresponding $k = 1$ algorithms. As expected, the more constraints there are, the more computation the algorithms require. Lastly, we found that SE-Mean and SE-Optimistic were faster than BE-Rebid, BE-Rebid was faster than MGM-Omniscient, and BE-Stay was the slowest (due to its recursive calculation), even though the two BE algorithms made use of pre-computed histograms based on the Normal distribution. This is not surprising, as the SE algorithms need only compute a multiply to determine their bid, whereas the BE algorithms must calculate multiple integrals and probabilities.

The memory requirements for $k = 1$ and $k = 2$ algorithms were not statistically significantly different and memory usage increases with the number of neighbors. $k = 2$

agents have a significant communication overhead, relative to $k = 1$ algorithms. $k = 2$ agents must send multiple messages per round, doubling the number of messages relative to $k = 1$, which need only send assignments and gains once per round. These results suggest that if D-CEE algorithms are embedded within mobile agents with limited computation, CPU usage may become critical, and the tradeoff in performance between $k = 1$ vs. $k = 2$ and SE vs. BE must be carefully weighted. We have assumed that movement cost dominates mobile sensor network domain because it requires the most wall clock time. However, since the network must carry messages, $k = 1$ is more appealing because reducing the coordination overhead leaves more bandwidth for the data.

Section 4 has empirically shown that all D-CEE methods can improve the total on-line reward for different graph topologies, different numbers of agents, and for different experiment lengths. The highest performing algorithms, reaching a cumulative gain of nearly 60% of optimal, are BE-Rebid and BE-Rebid-2. BE-Rebid performs best on sparse graphs, while BE-Rebid-2's performance is superior on dense graphs. Additionally, for full and chain graphs, SE-Optimistic also performs quite well, while it underperforms SE-Mean on scale-free and random graphs, and both SE methods have lower computational and communication overhead compared to the BE methods. Thus if agents have limited processing capabilities or low communication overhead is critical, one of the SE variants may be preferred to BE-Rebid or BE-Rebid-2.

5 D-CEE Algorithm Extensions

This section extends our D-CEE algorithms by showing that BE-Rebid can be extended to unknown experiment times, that our algorithms are robust to topology changes (e.g., agents unexpectedly fail), and that appropriate algorithms can be automatically selected per problem instance.

5.1 Extending to an Unknown Time Horizon

To extend BE-Rebid to unknown experiment lengths, we consider the case when a probability distribution, $P(t)$, is provided, defined as the probability that the length of the experiment is t rounds (i.e., the episode ends on round t). The probability that an experiment will reach round t, P_{reach}, is given by 1 - $\{$P(experiment ends on round$_1$) + ... + P(experiment ends on round$_{t-1}$)$\}$:

$$P_{reach}(t) = 1 - \sum_{1}^{t-1} P(i)$$

By Bayes Rule, the probability of reaching round t' given that round t has already been reached:

$$P_{reach}(t'|t) = \frac{P_{reach}(t' \cap t)}{P_{reach}(t)} = \frac{P_{reach}(t')}{P_{reach}(t)}$$

Thus, the expected number of remaining rounds, given that we have reached round t is:

$$\begin{aligned}T_{remain}(t) &= \left\{\sum_{i=t}^{\infty} iP(\text{episode is length } i|\text{have reached } t)\right\} - t\\ &= \left\{\sum_{i=t}^{\infty} iP(i|\text{have reached } t)\right\} - t\\ &= \left\{\sum_{i=t}^{\infty} i\frac{P(i)}{P_{reach}(t)}\right\} - t\end{aligned}$$

To incorporate these changes into BE-rebid, the value of backtrack is updated to:

$$V_{back}(R_b, t) = R_b T_{remain}(t)$$

The value of explore (Equation 1) has three terms, as described in Section 3.1. With an unknown experiment length, the equation is updated to include four terms:

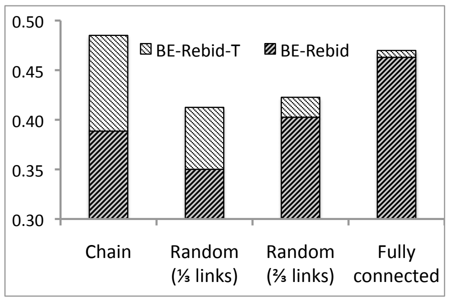
$$\begin{aligned}V_{explore}(R_b, t) = \max_{0\leq t_e\leq\infty} P_{reach}(t + t_e|t)\Bigg\{t_e\mu(n)\\ + T_{remain}(t + t_e)\Bigg[\int_{x>R_b} xQ(x, n, t_e)\partial x + R_b F(R_b, n)^{t_e}\Bigg]\Bigg\}\end{aligned}$$

An agent again commits to explore for t_e rounds. The first term describes the probability that the agent will reach round $t + t_e$, given it has reached round t. The second term is the utility of exploring for t_e rounds. Third and fourth terms are the utilities for backtracking in case of successful or unsuccessful exploration. In both the cases, the number of rounds for which the agent may exploit its discovered utility is given by $T_{remain}(t + t_e)$.

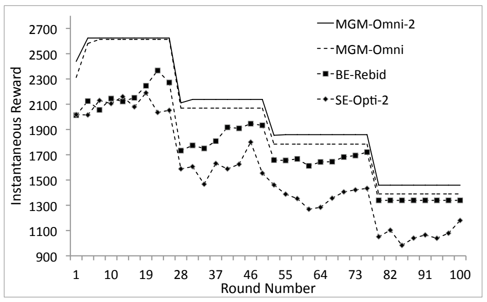
We compare this algorithm's performance with the BE-Rebid approach (see Figure 6a). The probability distribution used for the length of the experiment is a normal distribution with $\mu = 20$ and $\sigma = 5$. We compare BE-Rebid-T, which uses the above equations and probability distribution, with BE-Rebid which is told that the experiment will last for 100 rounds. We ran experiments on chain, random, and fully connected graphs, all with 10 agents, for 20 rounds, 30 trials each. In every case, the algorithms which were provided a probability distribution outperformed the algorithms provided an explicit (but incorrect) experiment length. The difference is statistically significant for all cases except full graphs — thus it may be beneficial to use BE-Rebid-T for unknown experiment lengths.

5.2 Dynamic Changes in Graph Topology

The algorithms presented are robust to agent failures and topology changes. The agents calculate their gains with the current set of neighbors on every round and act accordingly. Even if the algorithm has converged, it will begin optimizing again if needed, as can be seen in Figure 6b. The y-axis in the figure shows the global instantaneous reward and the x-axis shows the round. The graph shows an instance with 20 agents on a chain graph. Two random agents were killed after every 25 rounds, and the graph shows that agents resume their optimization (with varying degrees of success, depending on the algorithm employed). Only four algorithms have been showed for clarity.



(a) Unknown Time Horizon



(b) Crash Recovery

Fig. 6. Graph a (left) shows the improvement due to using the unknown time horizon equations (BE-Rebid-T). Graph b (right) shows how agents can recover after a failure of 2 agents after every 25 rounds.

5.3 Algorithmic Selection via Classification

BE-Rebid often achieves higher cumulative reward than SE-Optimistic (Sections 4.1 and 4.2), but SE-Optimistic is a substantially simpler algorithm with lower computational overhead (Section 4.4). However, performance on full graphs is often similar for these two algorithms. In this section, we use a trained decision tree to select an appropriate algorithm, showing that SE-Optimistic can be selected in many situations where it will do as well as BE-Rebid. Training data was collected from experiments with 43 different parameter settings,[4] and then a C4.5 decision tree (i.e., J48) was trained using the Weka toolkit (Witten & Frank, 2005). Training inputs were the number of agents and the number of allowed rounds. The target output was 0 if the performance of SE-Optimistic was better than, or statistically the same as, BE-Rebid, and 1 otherwise.

To test the effectiveness of our decision tree (which reached 92% training accuracy on 10-fold cross validation), we allow it to select which D-CEE algorithm to run at the beginning of an experiment, run the experiment, and compare the resulting cumulative reward to the algorithm not selected (keeping the random seed constant). We tested on 23 different parameter settings[5], and the decision tree selected SE-Optimistic 16 out of 23 times. When considering all 690 experiments (23 parameter settings, 30 trials each) we find that the difference between BE-Rebid and SE-Optimistic is statistically significant ($p < 4.7 \times 10^{-18}$) while the difference between BE-Rebid and the decision tree is not statistically significant ($p > 0.16$). Thus, the learned decision tree, when tested on experiments using a different number of agents and number of rounds, was able to use the less computationally demanding algorithm 67% of time *without* a statistically significant loss in cumulative reward. This preliminary result suggests that other predictions could also be successful, such as learning when BE-Rebid or BE-Rebid-2 would yield a higher total reward.

[4] Specifically, we used $\{10, 20, 30\}$ agents for $\{10, 20, 30, \ldots, 100\}$ rounds, 40 agents for $\{10, \ldots, 70\}$ rounds, and 50 agents for $\{10, \ldots, 60\}$ rounds, as simulating higher numbers of agents and rounds took prohibitively long.

[5] Specifically, we used $\{5, 15, 25\}$ agents for experiment lengths of $\{5, 15, 25, 45, 75, 95\}$ rounds and 45 agents for experiment lengths of $\{5, 15, 25, 45, 75\}$ rounds.

6 Related Work

Related work in DCOPs has been discussed in earlier sections. In addition, previous work in distributed constraint reasoning in sensor networks (Lesser et al., 2003; Zhang et al., 2003) uses a precursor method to the DCOP formulation and does not handle unknown reward matrices.

Cheng et al. (2005) suggest an approach for coordinating a set of robots based on swarm intelligence, however the objective of the work is to disperse the robots evenly within a specified shape, and not to optimize the signal strengths across the network. Correll et al. (2009) also look at optimizing a wireless network of mobile robots using a distributed swarm optimization, but are concerned with changing the topology (i.e., neighbors) of the network rather than optimizing signal strength. Marder et al. (2007) formulate dynamic sensor coverage as a "potential game," which is similar to a DCOP. However, like other DCOP work, the reward matrix is known, there is no time limit, and only final reward is considered. Gerkey et al. (2006) address a similar problem, but use auction mechanism and the goals of agents are significantly different (agents modify the topology of the network and on-line reward is not emphasized).

Kok and Vlassis (2006) use a reinforcement learning (RL) approach that applies to multi-agent tasks with coordination graphs. They model D-CEE-like problems in the RL framework by allowing agents to have an action for each value setting, but only a single state. Four algorithms are introduced which differ from our algorithms in three main ways: 1) all algorithms are centralized, 2) the number of allowed variables per agent is low but the algorithms take longer than ours to converge, primarily because 3) the algorithms do not explicitly reason about the length of the experiment. Their algorithms are also applicable in sequential decision making problems in addition to D-CEEs.

7 Conclusion

This paper has introduced D-CEE problems, a novel class of multi-agent problems based on physically-motivated domains, and a set of algorithms designed to exploit properties of such domains. In this paper, we (1) present an entire family of $k = 2$ algorithms, (2) provide detailed experimental comparison, (3) illustrate the robustness of these algorithms to agent failures, (4) extend the algorithms to handle unknown time horizons, and (5) provide a decision-tree learner to allow the appropriate use of less computationally intensive algorithms. In the Future, we plan to further explore the tradeoff between experiment length accuracy vs. algorithmic performance (Section 5.1) and to test our $k = 2$ algorithms on physical hardware, as we have done for our $k = 1$ algorithms (Jain et al., 2009).

Acknowledgements

We would like to thank Christopher Kiekintveld, W. Bradley Knox, and James Pita for helpful comments and suggestions on a previous draft of this paper. This work has been sponsored in part by DARPA contract FA8650-08-C-7811, in collaboration with the Lockheed Martin Corporation.

Bibliography

Cheng, J., Cheng, W., & Nagpal, R. (2005). Robust and self-repairing formation control for swarms of mobile agents. *AAAI*.

Correll, N., Bachrach, J., Vickery, D., & Rus, D. (2009). Ad-hoc wireless network coverage with networked robots that cannot localize. *ICRA*.

Freeman, P. R. (1983). The secretary problem and its extensions: A review. *International Statistical Review*, *51*.

Gerkey, B. P., Mailler, R., & Morisset, B. (2006). Commbots: Distributed control of mobile communication relays. *AAAI Workshop on Auction Mechanisms for Robot Coordination*.

Guestrin, C., Koller, D., & Parr, R. (2001). Multiagent planning with factored MDPs. *NIPS*.

Jain, M., Taylor, M. E., & Tambe, M. (2009). DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. *IJCAI*.

Kok, J. R., & Vlassis, N. (2006). Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, *7*, 1789–1828.

Lesser, V., Ortiz, C., & Tambe, M. (2003). *Distributed sensor nets: A multiagent perspective*. Kluwer Academic Publishers.

Maheswaran, R. T., Pearce, J. P., & Tambe, M. (2004). Distributed algorithms for DCOP: A graphical-game-based approach. *PDCS* (pp. 432–439).

Mailler, R., & Lesser, V. (2004). Solving distributed constraint optimization problems using cooperative mediation. *AAMAS*.

Marden, J., Arslan, G., & Shamma, J. (2007). Connections between cooperative control and potential games illustrated on the consensus problem. *European Control Conference*.

Modi, P. J., Shen, W., Tambe, M., & Yokoo, M. (2005). ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, *161*, 149–180.

Molisch, A. F. (2005). *Wireless communications*. IEEE Press.

Pearce, J., & Tambe, M. (2007). Quality guarantees on k-optimal solutions for distributed constraint optimization. *IJCAI*.

Petcu, A., & Faltings, B. (2005). A scalable method for multiagent constraint optimization. *IJCAI*.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, *58*, 527–535.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Zhang, W., Xing, Z., Wang, G., & Wittenburg, L. (2003). An analysis and application of distributed constraint satisfaction and optimization algorithms in sensor networks. *AAMAS*.