Infinite-Horizon Proactive Dynamic DCOPs

Khoi D. Hoang†, Ping Hou†, Ferdinando Fioretto‡, William Yeoh†, Roie Zivan⋄, Makoto Yokoo⋆
†Department of Computer Science, New Mexico State University, USA
{khoang, phou, wyeoh}@cs.nmsu.edu
‡Department of Industrial and Operations Engineering, University of Michigan, USA
fioretto@umich.edu
⋄Department of Industrial Engineering and Management, Ben Gurion University of the Negev, Israel
zivanr@cs.bgu.ac.il
⋆Department of Informatics, Kyushu University, Japan
yokoo@inf.kyushu-u.ac.jp

ABSTRACT

The Distributed Constraint Optimization Problem (DCOP) formulation is a powerful tool for modeling multi-agent coordination problems. Researchers have recently extended this model to Proactive Dynamic DCOPs (PD-DCOPs) to capture the inherent dynamism present in many coordination problems. The PD-DCOP formulation is a *finite-horizon* model that assumes a finite horizon is known a priori. It ignores changes to the problem after the horizon and is thus not guaranteed to find optimal solutions for infinite-horizon problems, which often occur in the real world. Therefore, we *(i)* propose the *Infinite-Horizon PD-DCOP* (IPD-DCOP) model, which extends PD-DCOPs to handle *infinite horizons*; *(ii)* exploit the convergence properties of Markov chains to determine the optimal solution to the problem after it has converged; *(iii)* propose three distributed greedy algorithms to solve IPD-DCOPs; *(iv)* provide theoretical quality guarantees on the new model; and *(v)* empirically evaluate both proactive and reactive algorithms to determine the tradeoffs between the two classes. The final contribution is important as, thus far, researchers have exclusively evaluated the two classes of algorithms in isolation. As a result, it is difficult to identify the characteristics of problems that they excel in. Our results are the first in this important direction.

Keywords

Distributed Constraint Optimization; Dynamic DCOPs; Stochastic DCOPs

1. INTRODUCTION

Distributed Constraint Optimization Problems (DCOPs) [17, 21, 34] are problems where agents need to coordinate their value assignments to maximize the sum of the resulting constraint rewards. This model can be applied to solve a number of multi-agent coordination problems including distributed meeting scheduling, coordination of sensors or robots, coalition formation, smart grids, and smart homes [3, 6, 7, 8, 12, 14, 15, 16, 25, 28, 32, 35]. However, DCOPs address and solve a single (static) problem as they assume that the problem does not change over time.

This limiting assumption prompted researchers to propose the *Dynamic DCOP* model [22], which is a sequence of (static) DCOPs with changes between them. There are a number of algorithms that handle various changes such as addition/removal of agents [26] or changes in the topology of the coordination graph [35]. In this paper, we focus on the problem where *only the reward functions can change*. This is a popular Dynamic DCOP variant, and researchers have proposed a number of algorithms to solve it. Dynamic DCOP algorithms can be categorized as *(1) reactive*, which are *online* algorithms reacting to the changes of the problem by solving the DCOP every time such changes occur [22, 26, 33], or *(2) proactive*, which are *offline* algorithms that take into account prior knowledge on the evolution of the problem when finding solutions. The recently introduced *Proactive Dynamic DCOP* (PD-DCOP) algorithms [10] belong to this category.

The PD-DCOP formulation is a *finite-horizon* Dynamic DCOP model that assumes a finite horizon is known a priori. It ignores changes to the problem after the horizon and optimizes the problems within the horizon only. It is suitable in problems where there is a deadline (i.e., horizon) after which the problem either ends or changes to the problem are inconsequential. However, in some applications, there may not be such a deadline or the deadline is not known. In such problems, infinite-horizon models are necessary.

Therefore, we *(i)* propose the *Infinite-Horizon PD-DCOP* (IPD-DCOP) model, which extends PD-DCOPs to handle *infinite horizons*; *(ii)* exploit the convergence properties of Markov chains to determine the optimal solution to the problem after it has converged; *(iii)* propose three distributed greedy algorithms to solve IPD-DCOPs; *(iv)* provide theoretical quality guarantees on the new model; and *(v)* empirically evaluate both proactive and reactive algorithms to determine the tradeoffs between the two classes. The final contribution is important as, thus far, researchers have exclusively evaluated the two classes of algorithms in isolation. As a result, it is difficult to identify the characteristics of problems that they excel in. Our results are the first in this important direction.

2. BACKGROUND

We now provide background on the regular and dynamic DCOPs as well as on key Markov chain properties that form the foundations for our proposed algorithm.

2.1 DCOPs

A *Distributed Constraint Optimization Problem* (DCOP) [17, 34] is a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F}, \alpha \rangle$, where $\mathbf{A} = \{a_i\}_{i=1}^{p}$ is a set of *agents*; $\mathbf{X} = \{x_i\}_{i=1}^{n}$ is a set of decision *variables*; $\mathbf{D} = \{D_x\}_{x \in \mathbf{X}}$ is a set of finite *domains* and each variable $x \in \mathbf{X}$ takes values from the set $D_x \in \mathbf{D}$; $\mathbf{F} = \{f_i\}_{i=1}^{k}$ is a set of *reward functions*, each defined

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

over a set of decision variables: $f_i : \prod_{x\in\mathbf{x}^{f_i}} D_x \to \mathbb{R}_0^+ \cup \{-\infty\}$, where infeasible configurations have $-\infty$ rewards, $\mathbf{x}^{f_i} \subseteq \mathbf{X}$ is the *scope* of f_i, and $\alpha : \mathbf{X} \to \mathbf{A}$ is a function that associates each decision variable to one agent.

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

2.2 Dynamic DCOPs

A *Dynamic DCOP* (D-DCOP) [22] is a sequence of DCOPs with changes between them, without an explicit model for how the problem will change over time. Solving a D-DCOP optimally means finding a reward-maximal solution for each DCOP in the sequence. Therefore, this approach is *reactive* since it does not consider future changes. Its advantage is that solving a D-DCOP is no harder than solving h DCOPs, where h is the horizon of the problem. Researcher have used this approach to solve D-DCOPs, where they introduce a super-stabilizing algorithm that reuses information from previous searches to speed up its current search [22]. Reactive algorithms are *online* algorithms by definition.

Alternatively, a *proactive* approach predicts future changes in the D-DCOP and finds robust solutions that require little or no refinements despite future changes in the problem. Recently, researchers have introduced a *Proactive D-DCOP* (PD-DCOP) formulation that incorporates a switching cost for changing solutions between subsequent DCOPs and proposed several proactive approaches to solve this problem [10]. Existing proactive algorithms are all *offline* algorithms as they solve the entire sequence of D-DCOPs before the first change occurs.

Finally, researchers have also proposed other models for D-DCOPs including one where agents have deadlines to choose their values [23], a model where agents can have imperfect knowledge about their environment [13, 19], and a model where changes in the constraint graph depends on the value assignments of agents [35].

2.3 Markov Chains

A *Markov chain* [9] is a sequence of random variables $\langle x^0, x^1, \dots, x^T\rangle$, where the transition from state x^{t-1} to state x^t depends exclusively on the previous state. More formally,

$$\begin{aligned}\Pr(x^t = j \mid x^{t-1} = i, x^{t-2} = r, \dots, x^0 = s)\\ = \Pr(x^t = j \mid x^{t-1} = i)\end{aligned} \quad (1)$$

for all time steps $t > 0$. We use Pr to denote the probability measure.

A Markov chain with a finite state space is said to be *time-homogeneous* if the transition $\Pr(x^t = j \mid x^{t-1} = i)$ is identical for all time steps t. We use the time-invariant notation of P_{ij} to denote such transitions.

A time-homogeneous Markov chain with a finite state space converges to a *stationary distribution* (i.e., the probability distribution of the random variable no longer changes over all states) when:

$$\pi^{t-1} \cdot P = \pi^t = \pi^* \quad (2)$$

where $\pi^t = [\pi_1^t, \dots, \pi_M^t]$ is the probability distribution at time t over M states in the chain and P is the transition matrix where each element P_{ij} in the matrix is the time-homogeneous transition probability from state i to state j.

A state j is said to be *accessible* from i, denoted by $i \to j$, if there exists a sequence of t-step transitions ($t \geq 1$) such that:

$$\Pr(x^t = j \mid x^0 = i) > 0 \quad (3)$$

We use the notation $P_{ij}^t = \Pr(x^t = j \mid x^0 = i)$ to denote the probability of such transitions. Two states i and j *communicate*, denoted by $i \leftrightarrow j$, if both states are accessible from each other. A *class* C of communicating states is a non-empty set of states such that each state $i \in C$ in the class communicates with every other state $j \in C$ in the class but does not communicates with any state $j \notin C$ outside the class. The *period* of a state i, denoted by $d(i)$, is the greatest common divisor (gcd) of the time steps t for which $P_{ii}^t > 0$.

$$d(i) = \gcd\{t : P_{ii}^t > 0\} \quad (4)$$

The state is said to be *aperiodic* if it has period $d(i) = 1$, and *periodic* if $d(i) > 1$. In finite-state time-homogeneous Markov chain, all states in the same class have the same period. If all the states of a Markov chain form a single class, then that chain has the period of that class.

A state i is said to be *recurrent* if it is accessible from all states j that are accessible from i. In other words, $i \to j$ implies $j \to i$. Otherwise, it is *transient*. All states in the same class are either recurrent or transient. A class of states are said to be *ergodic* if they are both recurrent and aperiodic. A *unichain* is a chain that contains a single recurrent class and may be some transient states. A unichain is called *ergodic unichain* if the recurrent class is ergodic.

In this paper, we consider the case where each Markov chain is guaranteed to converge to a *unique* stationary distribution π^* given any initial distribution. Specifically, the Markov chain should follow one of the following conditions (from strict to loose conditions): (i) $P_{ij} > 0$ for all states i and j, (ii) all states are in one single class and they are ergodic, (iii) the Markov chain is an ergodic unichain [9]. Throughout the paper, we interchangeably use the terms *converged distribution* and *stationary distribution*.

3. IPD-DCOP MODEL

At a high level, the *Infinite-Horizon Proactive Dynamic DCOP* (IPD-DCOP) model is a straightforward, but essential and significant, extension of the PD-DCOP model. Both IPD-DCOP and PD-DCOP models assume that a finite horizon h is given. However, PD-DCOPs ignore all changes to the problem after its finite horizon h and, thus, do not optimize for them, while IPD-DCOPs assume that the Markov chains will converge to stationary distributions after that horizon and an optimal solution for those stationary distributions should be adopted after the horizon. Since both models are similar, we follow and use the same PD-DCOP notations to describe IPD-DCOPs.

Formally, an IPD-DCOP is a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{Y}, \mathbf{D}, \Omega, \mathbf{F}, p_{\mathbf{Y}}^0, \mathbf{T}, h, c, \alpha\rangle$, where:

- $\mathbf{A} = \{a_i\}_{i=1}^p$ is a set of *agents*.
- $\mathbf{X} = \{x_i\}_{i=1}^n$ is a set of *decision variables*, which are variables controlled by the agents.
- $\mathbf{Y} = \{y_i\}_{i=1}^m$ is a set of *random variables*, which are variables that are uncontrollable and model stochastic events (e.g., traffic, weather, malfunctioning devices).
- $\mathbf{D} = \{D_x\}_{x\in\mathbf{X}}$ is a set of finite *domains* of the decision variables. Each variable $x \in \mathbf{X}$ takes values from the set $D_x \in \mathbf{D}$.
- $\Omega = \{\Omega_y\}_{y\in\mathbf{Y}}$ is a set of finite *domains* of the random variables (e.g., different traffic conditions, weather conditions). Each variable $y \in \mathbf{Y}$ takes values from the set $\Omega_y \in \Omega$.
- $\mathbf{F} = \{f_i\}_{i=1}^k$ is a set of *reward functions*, each defined over a mixed set of decision and random variables: f_i :

$\prod_{x \in \mathbf{X} \cap \mathbf{x}^{f_i}} D_x \times \prod_{y \in \mathbf{Y} \cap \mathbf{x}^{f_i}} \Omega_y \to \mathbb{R}_0^+ \cup \{-\infty\}$, where $\mathbf{x}^{f_i} \subseteq \mathbf{X} \cup \mathbf{Y}$ is the *scope* of f_i and infeasible value combinations for the variables in $\mathbf{x}^{f_i}$ have $-\infty$ rewards.

- $p_{\mathbf{Y}}^0 = \{p_y^0\}_{y \in \mathbf{Y}}$ is a set of initial *probability distributions* for the random variables $y \in \mathbf{Y}$.
- $\mathbf{T} = \{T_y\}_{y \in \mathbf{Y}}$ is a set of *transition functions* $T_y : \Omega_y \times \Omega_y \to [0, 1]$ for the random variables $y \in \mathbf{Y}$, describing the probability for a random variable to change its value in successive time steps. For each time step $t > 0$ and values $\omega, \omega' \in \Omega_y$, $T_y(\omega, \omega') = \Pr(y^t = \omega' | y^{t-1} = \omega)$, where y^t denotes the value of the variable y at time step t. Thus, $T_y(\omega, \omega')$ describes the probability for the random variable y to change its value from ω at time step $t - 1$ to ω' at time step t. Note that $T_y(\omega, \omega')$ is identical for every time step. Finally, $\sum_{\omega' \in \Omega_y} T_y(\omega, \omega') = 1$ for all $\omega \in \Omega_y$.
- $h \in \mathbb{N}$ is the *horizon* where the agents solve the problem with stationary distributions and keep the same solution onwards. It is guaranteed to maximize the expected reward from h onwards to the infinite horizon. In general, it is preferred that $h = \infty$ so as not to impose restriction on when agents are allowed to change their values.
- $c \in \mathbb{R}_0^+$ is a *switching cost*, which is the cost associated with the change in the value of a decision variable from one time step to the next.
- $\alpha : \mathbf{X} \to \mathbf{A}$ is a function that associates each decision variable to one agent. We assume that the random variables are not under the control of the agents and are independent of decision variables. Thus, their values are solely determined according to their transition functions.

Throughout this paper, we refer to decision (resp. random) variables with the letter x (resp. y). We also assume that each agent controls exactly one decision variable (thus, α is a bijection), and that each reward function $f_i \in \mathbf{F}$ is associated with at most one random variable y_i.[1]

As agents will keep their optimal solution from horizon h onwards, the goal of an IPD-DCOP is to find a sequence of $h + 1$ assignments $\bar{\mathbf{x}}^*$ for all the decision variables in $\mathbf{X}$:

$$\bar{\mathbf{x}}^* = \underset{\bar{\mathbf{x}} = \langle \mathbf{x}^0, \ldots, \mathbf{x}^h \rangle \in \Sigma^{h+1}}{\operatorname{argmax}} \mathcal{F}(\bar{\mathbf{x}}) \tag{5}$$

$$\mathcal{F}(\bar{\mathbf{x}}) = \sum_{t=0}^{h} \mathcal{F}^t(\mathbf{x}^t) \tag{6}$$

$$- \sum_{t=0}^{h-1} \left[c \cdot \Delta(\mathbf{x}^t, \mathbf{x}^{t+1}) \right] \tag{7}$$

where Σ is the assignment space for the decision variables of the IPD-DCOP at each time step, $\Delta : \Sigma \times \Sigma \to \{0\} \cup \mathbb{N}$ is a function counting the number of assignments to decision variables that differs from one time step to the next.

Equation (6) refers to the reward optimization for each time step:

$$\mathcal{F}^t(\mathbf{x}^t) = \mathcal{F}_x^t(\mathbf{x}^t) + \mathcal{F}_y^t(\mathbf{x}^t) \tag{8}$$

$$\mathcal{F}_x^t(\mathbf{x}) = \sum_{f_i \in \mathbf{F} \setminus \mathbf{F}_Y} f_i(\mathbf{x}_i) \tag{9}$$

$$\mathcal{F}_y^t(\mathbf{x}) = \sum_{f_i \in \mathbf{F}_Y} \sum_{\omega \in \Omega_{y_i}} f_i(\mathbf{x}_i|_{y_i = \omega}) \cdot p_{y_i}^t(\omega) \tag{10}$$

where $\mathbf{x}_i$ is an assignment for all the variables in the scope $\mathbf{x}^{f_i}$ of the function f_i, $\mathbf{x}_i|_{y_i=\omega}$ indicates that the random variable $y_i \in \mathbf{x}^{f_i}$ takes on the event $\omega \in \Omega_{y_i}$, $\mathbf{F}_Y = \{f_i \in \mathbf{F} \mid \mathbf{x}^{f_i} \cap \mathbf{Y} \neq \emptyset\}$ is the set of functions in $\mathbf{F}$ that involve random variables, $p_{y_i}^t(\omega)$ is the probability for the random variable y_i to assume value ω at time t:

$$p_{y_i}^t(\omega) = \begin{cases} \sum_{\omega' \in \Omega_{y_i}} p_{y_i}^{t-1}(\omega') \cdot T_{y_i}(\omega', \omega) & \text{if } t < h \\ p_{y_i}^*(\omega) & \text{if } t = h \end{cases} \tag{11}$$

where $p_{y_i}^*(\omega)$ is the stationary distribution of random variable y_i taking on value ω, and $p_{y_i}^*$ is the solution of the following system of linear equations. For each $\omega \in \Omega_{y_i}$:

$$\sum_{\omega' \in \Omega_{y_i}} p_{y_i}^*(\omega') \cdot T_{y_i}(\omega', \omega) = p_{y_i}^*(\omega) \tag{12}$$

and $p_{y_i}^*$ is a probability vector:

$$\sum_{\omega \in \Omega_{y_i}} p_{y_i}^*(\omega) = 1 \tag{13}$$

In this paper, we assume that each Markov chain is guaranteed to converge to a unique stationary distribution π^* given any initial distribution.

Note that at horizon h, agents solve the problem with the stationary distribution of random variables and keep this optimal solution onwards. This would maximize the expected reward at each time step after the distributions have converged to the stationary distribution. If choosing any other assignment, then, it will result in a smaller reward (see Theorem 3).

Finally, Equation (7) considers the penalties due to the changes in the decision variables' values during the optimization process.

4. IPD-DCOP ALGORITHMS

In general, IPD-DCOPs can be solved in an online or offline manner. Online approaches have the benefit of observing the actual values of the random variables during execution and can thus exploit these observations to improve the overall solution quality. However, the downside to online approaches is that they may have a limited amount of time to find a solution as the problem may change rapidly. Offline approaches, on the other hand, have the benefit of having sufficient time to take into account all possible changes to the problem. However, the finite horizon h needs to be known a priori or pre-determined before execution. Offline approaches may seek to find *open-loop* solutions, which are solutions that do not depend on observation of the actual values of the random variables during execution, or *closed-loop* solutions, which are solutions that do depend on such observations. IPD-DCOP solutions are open-loop solutions by definition as they do not depend on observations (see Equation (5)). It is possible to find closed-loop solutions as well, and one such type of solutions are MDP policies. Readers are refered to Section 6 for a discussion on how MDPs and their variants relate to IPD-DCOPs.

As the changes from PD-DCOPs to IPD-DCOPs are straightforward, one can modify existing PD-DCOP algorithms, which are all offline algorithms, to solve IPD-DCOPs. We also introduce three new greedy algorithms, called FORWARD, BACKWARD, and HYBRID to solve IPD-DCOPs. FORWARD can be implemented as either an offline or online algorithm, BACKWARD is an offline algorithm, and HYBRID is an online algorithm. The fact that we introduce online proactive algorithms is important as it allows us to empirically evaluate and compare the empirical performance of online proactive algorithms and online reactive algorithms. Comparisons between proactive and reactive algorithms have never been done, to the best of our knowledge, and a better understanding of the problem characteristics that each algorithm type excels in is important.

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

4.1 PD-DCOP Algorithm Extensions

As PD-DCOP algorithms solve finite-horizon problems, one can extend them to solve IPD-DCOPs offline. To fully understand these extensions, familiarity with PD-DCOP algorithms may be necessary. Due to the lack of space, we only briefly elaborate on them here and refer the reader to the original PD-DCOP paper [10]. Two PD-DCOP algorithms that we extend and later compare against are S-DPOP and LS-SDPOP. S-DPOP ignores the switching costs between subsequent DCOPs and solves each DCOP in the dynamic problem optimally. LS-SDPOP is a local search algorithm that first runs S-DPOP to find an initial PD-DCOP solution, and then iteratively improves the solution by incorporating switching costs.

The key changes to extend these algorithms to solve IPD-DCOPs are as follows: (1) Find the stationary distribution of the random variables in the problem; and (2) Solve the problem at horizon h with this stationary distribution and adopt it at that time step. Details for these changes are described in Step 1 of the FORWARD algorithm in the next subsection. Aside from these changes, all other components of the algorithms remain unchanged. Note that we solve the problem at horizon h with the stationary distribution so that we can guarantee that we are adopting the optimal solution at the infinite horizon.

4.2 FORWARD

We now introduce FORWARD, a distributed greedy algorithm to solve IPD-DCOPs. We will first introduce FORWARD as an offline algorithm that solves IPD-DCOPs within a given amount of time. We summarize below the high-level ideas for this algorithm:

- STEP 1: It assumes that the Markov chains of all the random variables will converge to their own stationary distributions and searches for an optimal DCOP solution for these stationary distributions.
- STEP 2: It greedily solves the IPD-DCOP one time step at a time starting from the initial time step $t = 0$. In other words, it successively solves the DCOP for each time step starting from $t = 0$. When solving each DCOP optimally, it takes into account the switching costs of changing values from the solution in the previous time step.
- STEP 3: If it is about to run out of time (e.g., it can only solve the DCOP for the current time step $h - 1$ before it terminates), it also takes into account the switching costs of changing values from the solution in the current time step to the solution for the converged distribution.

Therefore, it can potentially have different solutions for all time steps $0 \leq t < h$, and incurring switching costs if the solutions are different, but the solutions for all subsequent time steps $t \geq h$ will be the optimal solution for the stationary distribution.

STEP 1: Observe that the m random variables in the IPD-DCOP form m independent Markov chains since the transition function $T_y \in \mathbf{T}$ of each random variable $y \in \mathbf{Y}$ is independent of the transition functions for the other random variables. Furthermore, these Markov chains are *time-homogeneous* – the transition functions are identical for all time steps – and has *finite state spaces* – the domain of each random variable y is a finite set $\Omega_y \in \Omega$. In this paper, we assume that all the Markov chains will converge to a *unique* stationary distribution given any initial distribution [9]. The computation of the unique distribution for each random variable y, computed using a system of linear equations (Equations (12) and (13)), can be done independently by each agent a that controls the decision variable x that is constrained with random variable y. In other words, the computation for random variable y is performed by the agent a such that $\exists\, x \in \mathbf{X}, f \in \mathbf{F} : y \in \mathbf{x}^f \wedge x \in \mathbf{x}^f \wedge \alpha(x) = a$.

Once the stationary distribution for each random variable is found, the agents run a pre-processing step to reformulate the constraint between decision and random variables into constraints between decision variables only. Specifically, for each constraint $f \in \mathbf{F}_Y$ between decision variables $\mathbf{x}$ and a random variable y (i.e., scope $\mathbf{x}^f = \mathbf{x} \cup \{y\}$), the following new constraint is created:

$$F^h(\mathbf{x}) = \sum_{\omega \in \Omega_y} f(\mathbf{x}|_{y=\omega}) \cdot p^*_y(\omega) \tag{14}$$

where $p^*_y(\omega)$ is the probability of random variable y having state ω in the stationary distribution. Note that the new scope of this new constraint is exclusively the decision variables $\mathbf{x}$ only. After this pre-processing step, they run a complete DCOP algorithm to find an optimal solution given the converged distribution of the random variables.

STEP 2: The agents optimally solve each sub-problem associated with time steps t $(0 \leq t < h)$ using a complete DCOP algorithm. However, before solving each DCOP, they run a pre-processing step, where they (1) reformulate the constraint between decision and random variables similar to Step 1, and (2) capture the cost of switching values between time steps in new unary constraints of decision variables. Similar to Step 1, for each constraint $f \in \mathbf{F}_Y$ between decision variables $\mathbf{x}$ and a random variable y, the following new constraint is created for each time step $0 \leq t < h$:

$$F^t(\mathbf{x}) = \sum_{\omega \in \Omega_y} f(\mathbf{x}|_{y=\omega}) \cdot p^t_y(\omega) \tag{15}$$

where $p^t_y(\omega)$ is the probability of random variable y in state ω at time step t.

To capture the cost of switching values across time steps, for each decision variable $x \in \mathbf{X}$, the following new unary constraint is created for each time step $0 < t < h - 1$:

$$C^t(x) = -c \cdot \Delta(x^{t-1}, x^t) \tag{16}$$

After these pre-processing steps, the agents repeatedly run a complete DCOP algorithm to successively solve the DCOPs from time step $t = 0$ onwards.

STEP 3: Finally, if there is only enough time to complete one more complete DCOP run, before the run, the agents also incorporate the switching cost from the current solution to the solution for the converged distribution as unary constraints in a pre-processing step:

$$C^{h-1}(x) = -c \cdot \Big(\Delta(x^{h-2}, x^{h-1}) + \Delta(x^{h-1}, x^*)\Big) \tag{17}$$

where $h - 1$ is the current time step and x^* is the value of the variable x in the solution for the converged distribution of random variables.

Finally, to change FORWARD from the above offline version to an online version, it simply skips Step 3 as it will never run out of time. Since it skips Step 3, it will continue to greedily solve each time step indefinitely. Therefore, there is also no need to run Step 1 as it will never assume that the Markov chains have converged.

4.3 BACKWARD

Instead of greedily solving the IPD-DCOP one time step at a time forward starting from $t = 0$ towards the horizon, one can also greedily solve the problem backwards from $t = h$ towards the first time step. The BACKWARD algorithm implements this key difference. It too has the similar three steps of FORWARD with small differences:

- STEP 1: Both algorithms solve the problem at horizon $t = h$ first with stationary distribution.
- STEP 2: While FORWARD successively solves each sub-problem from horizon $t = 0$ forwards, BACKWARD successively solves each sub-problem from horizon $t = h - 1$ backwards. At time step t, while FORWARD takes into account the switching cost from the solution in the previous time step $t - 1$, BACKWARD takes into account the switching cost to the solution in the next time step $t + 1$. Specifically, before solving each sub-problem optimally, BACKWARD runs a pre-processing step by creating a unary constraint for each time step $0 \leq t \leq h - 1$ as (15) and:

$$C^t(x) = -c \cdot \Delta(x^{t+1}, x^t) \tag{18}$$

- STEP 3: The key difference occurs when both algorithms solve the last sub-problem. For FORWARD, it solves the last sub-problem at $t = h - 1$ and considers the switching cost from the solution at $t = h - 2$ as well as the switching cost to the solution at $t = h$ (optimal solution for the stationary distribution). On the other hand, BACKWARD solves the problem at $t = 0$ while taking into account the switching cost to the solution at $t = 1$.

4.4 HYBRID

Finally, HYBRID is a combination between the proactive FORWARD algorithm and reactive algorithms. Similar to FORWARD, HYBRID greedily solves the problem from the first horizon $t = 0$ onwards. The difference is that it will observe the values of the random variables at each time step $t \geq 0$, and take that information into account when computing the probability distribution of the random variables in the next time step. It then solves the problem for the next time step with the updated probability distributions, thereby finding better solutions than the FORWARD algorithm.

5. THEORETICAL RESULTS

Let $\mathcal{F}^{\mathbf{y}}(\mathbf{x})$ denote the reward of a DCOP where the random variables are assigned $\mathbf{y}$ and the decision variables are assigned $\mathbf{x}$; $\mathbf{x}^*$ denote the optimal assignment of decision variables for the converged distribution of random variables. Then, $\mathcal{F}^{\mathbf{y}}_{\Delta} = \max_{\mathbf{x}\in\Sigma} \mathcal{F}^{\mathbf{y}}(\mathbf{x}) - \mathcal{F}^{\mathbf{y}}(\mathbf{x}^*)$ is the largest possible solution quality loss for assigning $\mathbf{x}^*$; and $\beta = \Pi_{y\in\mathbf{Y}} \min_{\omega,\omega'} T_y(\omega,\omega')$ is the smallest probability for transitioning between two joint states $\mathbf{y}$ and $\mathbf{y}'$ of the joint random variables in $\mathbf{Y}$.

THEOREM 1. *When $\beta > 0$, the error between the optimal solution qualities of an IPD-DCOP with horizon h and an IPD-DCOP with horizon ∞ is bounded from above by:*

$$c \cdot |\mathbf{X}| + \sum_{\mathbf{y}\in\prod_{y\in\mathbf{Y}}\Omega_y} \frac{(1-2\beta)^h}{2\beta} \mathcal{F}^{\mathbf{y}}_{\Delta}$$

PROOF SKETCH: First, given a random variable y, the following inequality holds if the Markov chain converges to the stationary distribution p^*_y [9]. For a given $\omega \in \Omega_y$:

$$|T^t_y(\omega', \omega) - p^*_y(\omega)| \leq (1 - 2\alpha)^t$$

for all $\omega' \in \Omega_y$, where T is the transition matrix and $\alpha = \min_{\omega,\omega'} T_y(\omega, \omega')$.

As T^* and T^t are the stationary transition matrix and the transition matrix at time step t, respectively:

$$p^0_y \cdot T^* = p^*_y$$
$$p^0_y \cdot T^t = p^t_y$$

For $\omega \in \Omega_y$:

$$\begin{aligned}
p^*_y(\omega) &= \sum_{\omega'\in\Omega_y} p^0_y(\omega') \cdot T^*_y(\omega', \omega)\\
p^t_y(\omega) &= \sum_{\omega'\in\Omega_y} p^0_y(\omega') \cdot T^t_y(\omega', \omega)\\
|p^*_y(\omega) - p^t_y(\omega)| &= |\sum_{\omega'\in\Omega_y} p^0_y(\omega') \cdot (T^*_y(\omega', \omega) - T^t_y(\omega', \omega))|\\
&= |\sum_{\omega'\in\Omega_y} p^0_y(\omega') \cdot (p^*_y(\omega) - T^t_y(\omega', \omega))|\\
&\leq \sum_{\omega'\in\Omega_y} p^0_y(\omega') \cdot |(p^*_y(\omega) - T^t_y(\omega', \omega))|\\
&\leq \sum_{\omega'\in\Omega_y} p^0_y(\omega') \cdot (1-2\alpha)^t\\
&\leq (1-2\alpha)^t
\end{aligned}$$

where $T^*_y(\omega', \omega) = p^*_y(\omega)$ for all $\omega' \in \Omega_y$. Similarly, with $\mathbf{y} \in \prod_{y\in\mathbf{Y}} \Omega_y$, we have:

$$\delta^t_{\mathbf{Y}}(\mathbf{y}) = |p^*_{\mathbf{Y}}(\mathbf{y}) - p^t_{\mathbf{Y}}(\mathbf{y})| \leq (1-2\beta)^t$$

Then, the solution quality loss for assigning $\mathbf{x}^*$:

$$\begin{aligned}
\mathcal{F}^t_{\Delta} &= \sum_{\mathbf{y}\in\prod_{y\in\mathbf{Y}}\Omega_y} \delta^t_{\mathbf{Y}}(\mathbf{y}) \cdot F^{\mathbf{y}}_{\Delta}\\
&\leq \sum_{\mathbf{y}\in\prod_{y\in\mathbf{Y}}\Omega_y} (1-2\beta)^t \cdot F^{\mathbf{y}}_{\Delta}
\end{aligned}$$

Next, let $\hat{\mathbf{x}}^* = \langle\hat{\mathbf{x}}^*_0, \ldots, \hat{\mathbf{x}}^*_h\rangle$ be the optimal solution for DCOPs from time step 0 to h, $\bar{\mathbf{x}}^* = \langle\mathbf{x}^*_0, \ldots, \mathbf{x}^*_h\rangle$ be the optimal solution given $\mathbf{x}^*_h = \mathbf{x}^*$, and let $\check{\mathbf{x}}^* = \langle\check{\mathbf{x}}^*_0 = \hat{\mathbf{x}}^*_0, \check{\mathbf{x}}^*_1 = \hat{\mathbf{x}}^*_1, \ldots, \check{\mathbf{x}}^*_{h-1} = \hat{\mathbf{x}}^*_{h-1}, \check{\mathbf{x}}^*_h = \mathbf{x}^*_h = \mathbf{x}^*\rangle$. We then have the following solution qualities:

$$\begin{aligned}
U_+ &= \sum_{t=0}^{h} \mathcal{F}^t(\hat{\mathbf{x}}^*_t) - \sum_{t=0}^{h-1} [c \cdot \Delta(\hat{\mathbf{x}}^*_t, \hat{\mathbf{x}}^*_{t+1})]\\
U &= \sum_{t=0}^{h} \mathcal{F}^t(\mathbf{x}^*_t) - \sum_{t=0}^{h-1} [c \cdot \Delta(\mathbf{x}^*_t, \mathbf{x}^*_{t+1})]\\
U_- &= \sum_{t=0}^{h} \mathcal{F}^t(\check{\mathbf{x}}^*_t) - \sum_{t=0}^{h-1} [c \cdot \Delta(\check{\mathbf{x}}^*_t, \check{\mathbf{x}}^*_{t+1})]
\end{aligned}$$

As $\bar{\mathbf{x}}^*$ is the optimal solution for the IPD-DCOP and $\check{\mathbf{x}}^*_h = \mathbf{x}^*_h = \mathbf{x}^*$, so $U_- \leq U$. Moreover, $\check{\mathbf{x}}^*_t = \hat{\mathbf{x}}^*_t$ for all time step between 0 and $h - 1$, so we have:

$$\begin{aligned}
U_+ - U \leq U_+ - U_- = \Big[\mathcal{F}^h(\hat{\mathbf{x}}^*_h) - \mathcal{F}^h(\mathbf{x}^*)\Big]&\\
- [c \cdot \Delta(\hat{\mathbf{x}}^*_{h-1}, \hat{\mathbf{x}}^*_h) - c \cdot \Delta(\check{\mathbf{x}}^*_{h-1}, \mathbf{x}^*)]&
\end{aligned}$$

and:

$$c \cdot \Delta(\hat{\mathbf{x}}^*_{h-1}, \hat{\mathbf{x}}^*_h) - c \cdot \Delta(\check{\mathbf{x}}^*_{h-1}, \mathbf{x}^*) \leq c \cdot |\mathbf{X}|$$

Moreover, $\mathcal{F}^h(\hat{\mathbf{x}}^*_h) - \mathcal{F}^h(\mathbf{x}^*)$ is the reward difference at time step h for applying $\mathbf{x}^*$. So, it is bounded by $\mathcal{F}^h_{\Delta}$:

$$\mathcal{F}^h(\hat{\mathbf{x}}^*_h) - \mathcal{F}^h(\mathbf{x}^*) \leq \mathcal{F}^h_{\Delta}$$

The error bound is $U_+ - U \leq \mathcal{F}^h_{\Delta} + c \cdot |\mathbf{X}|$. In addition, from $t = h + 1$ to ∞, sum of the error bounds is $\sum_{t=h+1}^{\infty} \mathcal{F}^t_{\Delta}$.

Summing up the two error bounds for $0 \rightarrow h$ and $h+1 \rightarrow \infty$, we get:

$$\mathcal{F}^h_\Delta + c \cdot |\mathbf{X}| + \sum_{t=h+1}^{\infty} \mathcal{F}^t_\Delta = c \cdot |\mathbf{X}| + \sum_{t=h}^{\infty} \mathcal{F}^t_\Delta$$

and:

$$\sum_{t=h}^{\infty} \mathcal{F}^t_\Delta = \sum_{t=h}^{\infty} \left(\sum_{\mathbf{y} \in \prod_{y \in \mathbf{Y}} \Omega_y} \delta^t_{\mathbf{Y}}(\mathbf{y}) \cdot F^{\mathbf{y}}_\Delta \right)$$
$$\leq \sum_{\mathbf{y} \in \prod_{y \in \mathbf{Y}} \Omega_y} \sum_{t=h}^{\infty} (1-2\beta)^t \cdot F^{\mathbf{y}}_\Delta$$
$$\leq \sum_{\mathbf{y} \in \prod_{y \in \mathbf{Y}} \Omega_y} \frac{(1-2\beta)^h}{2\beta} F^{\mathbf{y}}_\Delta$$

Then, the bound can be rewritten as:

$$c \cdot |\mathbf{X}| + \sum_{\mathbf{y} \in \prod_{y \in \mathbf{Y}} \Omega_y} \frac{(1-2\beta)^h}{2\beta} \mathcal{F}^{\mathbf{y}}_\Delta$$

□

THEOREM 2. *Optimally solving an* IPD-DCOP *with a horizon that is polynomial (exponential) in* $|\mathbf{X}|$ *is PSPACE-complete (PSPACE-hard).*

PROOF SKETCH: Similar to the complexity proof of PD-DCOPs [10], there exists a naive depth-first search to solve IPD-DCOPs with horizon h, where the algorithm requires linear space in the number of variables and horizon length. Also, one can reduce a *satisfiability of quantified Boolean formula* (QSAT) to an IPD-DCOP with $h = 1$. If the time horizon h is only polynomial in the size of $|\mathbf{X}|$, solving PD-DCOPs is PSPACE-complete. If the time horizon h is exponential in the size of $|\mathbf{X}|$, solving PD-DCOPs is PSPACE-hard. □

THEOREM 3. *From time step h onwards, adopting the optimal solution for the stationary distribution, instead of any other solution, will maximize the expected reward from that time step onwards.*

PROOF SKETCH: As p^*_y is the stationary distribution of random variable y and it is also the converged distribution:

$$\lim_{t \rightarrow \infty} p^t_y = p^*_y \tag{19}$$

$$p^*_y \cdot T = p^*_y \tag{20}$$

After convergence, as p^*_y does not change for every $y \in \mathbf{Y}$, the optimal solution for the successive DCOPs remain the same. Let h^* be the horizon when the stationary distribution converges, $\mathbf{x}^*$ be the optimal solution, $\mathbf{x}'$ be any sub-optimal solution, and $\mathcal{F}^*(\mathbf{x})$ be the quality of solution $\mathbf{x}$ for the regular DCOP with stationary distribution. As the stationary distribution at h^* is the actual distribution at h^*, the solution $\mathbf{x}^*$ is optimal for DCOP at h^* and also optimal for all DCOPs from h^*:

$$\mathcal{F}^*(\mathbf{x}^*) > \mathcal{F}^*(\mathbf{x}') \tag{21}$$

The difference in quality between two solutions for DCOPs after h^* is:

$$\Delta^\infty_{h^*} = \sum_{t=h^*}^{\infty} \left[\mathcal{F}^*(\mathbf{x}^*) - \mathcal{F}^*(\mathbf{x}') \right] \tag{22}$$

As the difference in solution quality from h to h^* is finite, it is dominated by $\Delta^\infty_{h^*} = \infty$. In other words, if the solution at time step is $\mathbf{x}'$, the accumulated expected reward is smaller than the expected reward with optimal solution $\mathbf{x}^*$. □

6. RELATED WORK

There is a large body of work on dynamic DCOPs, which we summarized in Section 2.2. Among them, the PD-DCOP model [10] is the one that is most closely related to ours. There are two key differences. The first is that IPD-DCOPs model infinite-horizon problems while PD-DCOPs model finite-horizon problems. As such, the IPD-DCOP solution at the last time step h (after which the solution does not change) is one that is optimized for the (converged) stationary distribution of random variables $p^*_{y_i}(\omega)$ (see Equations (10) and (11)). In contrast, the PD-DCOP solution at the last time step (after which the problem is assumed to no longer change) is one that is optimized for the actual distribution of random variables at that time step.

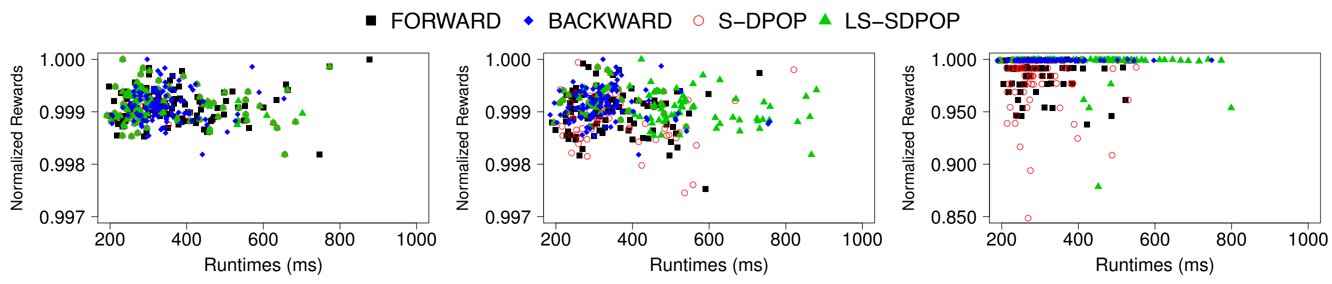
The second difference is in the objective functions. IPD-DCOPs optimize the expected cumulative *undiscounted* reward function (see Equations (6) and (7)) while PD-DCOPs optimize the expected cumulative *discounted* reward function, where the reward at time step t is discounted by a discount factor $\gamma^t < 1$. While it is possible to extend the objective of PD-DCOPs to the undiscounted case, we believe that for infinite-horizon problems, optimizing the expected undiscounted reward fits more naturally with many multi-agent applications, where achieving a goal in a future time step is as important as achieving it in the current time step. For example, in a sensor network application, detecting an intruder next week should not be any less valuable than detecting an intruder today. In some cases, optimizing the undiscounted reward is harder and the motivation for using discounted utilities is mostly a convenience, not a better match with the real objective. It is well known that discount factors are often selected arbitrarily despite the fact that they could affect the final policy [11].

Like PD-DCOPs, IPD-DCOPs are also related to Dynamic/Mixed/Stochastic CSPs [27, 29, 30] as well as to Dec-(PO)MDPs [1, 2, 20, 31]. The generality of Dec-POMDPs would allow them to represent (I)PD-DCOPs. For example, their goal of finding closed-loop solutions subsume open-loop solutions. Further, they assume very general transition functions, while IPD-DCOPs assume that the transition functions are independent of the agents' decisions. However, as discussed in the original PD-DCOP paper, such generality is achieved at the cost of a high complexity – solving Dec-POMDPs optimally is NEXP-hard for finite-horizon problems and undecidable for infinite-horizon problems [2]. On the other hand, IPD-DCOPs are PSPACE-hard/complete (Theorem 2).

Additionally, due to the high complexity of Dec-POMDPs, they are typically solved in a centralized manner [1, 2, 4, 5]. In contrast, the specialization of the IPD-DCOP model allows it to computationally exploit the specific structures of the model and to solve it in a decentralized manner. IPD-DCOPs also exploit the convergence properties of Markov chain, which cannot be done if transition functions are affected by the agents' decisions.

Thus, one ought to view IPD-DCOPs as a model between DCOPs and Dec-POMDPs, which we hope is a step towards greater synergy between these two research fields. In fact, researchers have proposed ND-POMDPs [18], which combine elements of the two models. However, ND-POMDPs share the same worst-case complexity as regular Dec-POMDPs.

Additionally, a new area that is related to IPD-DCOPs is the work on (PO)MDPs and their decentralized models using the average-reward criterion [24]. Both IPD-DCOPs and these Markovian models optimize the expected reward at each time step after the random variables have converged. However, IPD-DCOP solutions also optimize the cumulative reward before the random variables are assumed to have converged while average-reward solutions do not.



(a) Switching Cost = 10 (b) Switching Cost = 1,000 (c) Switching Cost = 100,000

Figure 1: Comparison between Offline Proactive Algorithms

7. EXPERIMENTAL RESULTS

As IPD-DCOPs can be solved in an offline or online manner, we perform empirical evaluations for both settings. In contrast to many experiments in the literature, our experiments are performed in an *actual distributed system*, where each agent is an Intel i7 Quadcore 3.4GHz machine with 16GB of RAM, connected in a local area network. We thus report actual distributed runtimes.

Experiment with Offline Algorithms: We empirically evaluate all the offline algorithms: The offline version of FORWARD, BACKWARD, and two PD-DCOP algorithms (S-DPOP and LS-DPOP) [10] that we adapt to solve IPD-DCOPs. Figure 1 shows the results, where we varied the switching cost c of the problem from 10 to 100,000. We used the following settings for the problem: $|\mathbf{A}| = |\mathbf{X}| = |\mathbf{Y}| = 8$, where each agent owns exactly one decision variable, and each decision variable is constrained with exactly one random variable; $|D_x| = |\Omega_y| = 3$ for all decision variables x and random variables y; constraint density $p_1 = 0.5$;[2] and $h = 5$. The initial probability distributions for the random variables and their transition functions are randomly generated and then normalized.

Each of the three scatter plots normalized rewards (y-axis), where we normalize the rewards with the largest reward over all instances in all three plots, and runtimes (x-axis) of the three algorithms on 100 instances. We make the following observations:

- When the switching cost is small ($c = 10$), the agents have little incentive to keep their values and will change them to the ones that gain the largest rewards. Thus, all four algorithms find solutions of similar quality in the same amount of time.
- As the switching cost increases ($c = 1,000$), S-DPOP finds worse solutions since it does not take into account switching costs. Thus, LS-SDPOP is able to make iterative improvements, resulting in better solutions but at the cost of larger runtimes. Also, S-DPOP, FORWARD, and BACKWARD start to differ in their solutions and runtimes, but without any statistically significant differences.
- When the switching cost is large ($c = 100,000$), BACKWARD finds the best solutions, followed by FORWARD. The reason is the following: At each time step t, BACKWARD only needs to consider the switching cost between two solutions: its solution for the current time step t and the solution at time step $t + 1$. In contrast, FORWARD needs to consider the switching cost between three solutions at time step $h - 1$: its solution for the current time step $h - 1$, the solution at time step $h - 2$, and the solution at time step h. Therefore, FORWARD may have greedily chosen bad solutions at time step $h - 2$ that incur very high switching costs regardless of the solution at time step $h - 1$. This case is exacerbated when the switching costs are very high. In such a case, the optimal solution may be that the solution for the converged solution be adopted for all time steps from $t = 0$ onwards. BACKWARD is able to take this factor into account while FORWARD is not able to do so since it does not consider the optimal solution at the converged solution when searching for solutions at time steps $t = 0$ to $t = h - 2$.

[2] p_1 is the density of functions between two decision variables.

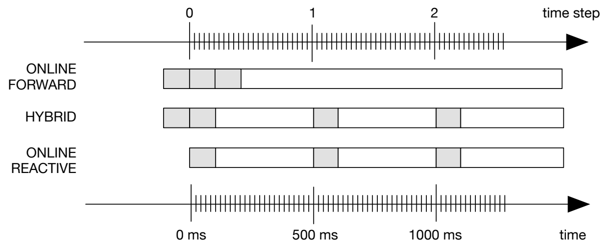


Figure 2: Search Time vs. Solution Adoption Time

Both BACKWARD and FORWARD find better solutions than LS-SDPOP and S-DPOP. The reason is the following: S-DPOP does not take into account switching costs when solving each sub-problem DCOP optimally. Therefore, S-DPOP has worse solutions. While LS-SDPOP improves the SDPOP solution through local search improvements, it is hampered by the overly inferior solutions of S-DPOP. Therefore, LS-SDPOP has better solutions than S-DPOP but are still not as good as those found by BACKWARD and FORWARD.

In terms of runtime, LS-SDPOP has the largest runtime, while BACKWARD, FORWARD and S-DPOP have similar runtimes. These trends show that BACKWARD and FORWARD find better solutions than S-DPOP and LS-SDPOP with runtimes that are smaller than LS-SDPOP and comparable to that of S-DPOP.

Experiment with Online Algorithms: We empirically evaluate all the online algorithms: The online version of FORWARD, HYBRID, and a reactive algorithm that waits for the problem to change before solving it. In order to fairly compare them, we evaluated them in a simulator that emulates an actual physical deployment.

Figure 2 illustrates the time both algorithms spent searching for solutions (denoted by gray rectangles) as well as the time they adopted their solutions (denoted by white rectangles) when the time duration between iterations is 500ms. FORWARD starts searching for optimal solutions before the problem starts, and adopts the solution later. HYBRID solves the first sub-problem at $t = 0$ based on the initial distribution of random variables, which is known a priori. When the problem starts, HYBRID adopts the solution while observing the values of random variables, using the observation to find its solution for the next time step. Finally, the reactive algorithm solves the problem each time the problem changes.

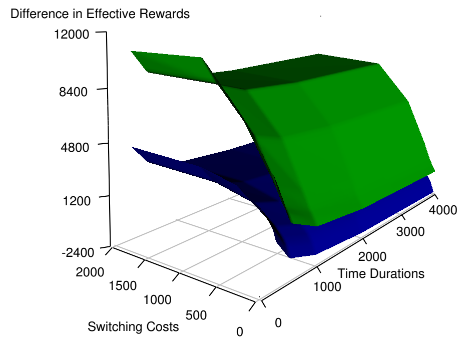


Figure 3: Comparison between FORWARD and Reactive

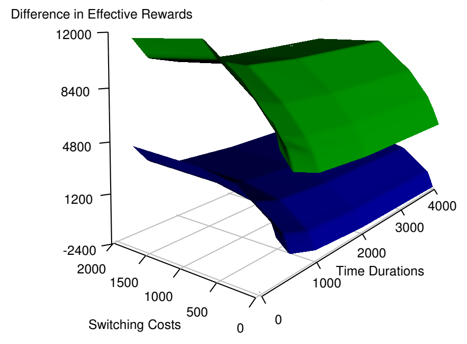


Figure 4: Comparison between HYBRID and Reactive

The *effective* reward R_{eff} of the reactive algorithm in each time step t is thus the weighted sum

$$R_{eff} = w_1^t \cdot q_{t-1}^t + w_2^t \cdot q_t^t - (w_1^t + w_2^t) \cdot c_{t-1,t} \tag{23}$$

where w_1^t is the time it spent searching for a solution at time step t;[3] w_2^t is the time it adopted the solution found; q_{t-1}^t is the quality of solution found in the previous time step $t - 1$; q_t^t is the quality of solution found in the current time step t; and $c_{t-1,t}$ is the switching cost incurred between the two time steps. For FORWARD and HYBRID, since they find a solution for each time step before the start of that time step, $w_1^t = 0$ for all time steps and $R_{eff} = w_2^t \cdot (q_t^t - c_{t-1,t})$. However, the solution found for each time step by the three algorithms are likely to differ and we aim to experimentally evaluate the conditions in which one class of algorithms is preferred over the other.

Figure 3 shows the results comparing FORWARD and the reactive algorithm, where we varied two parameters – the time duration between subsequent time steps of the dynamic DCOP (i.e., the time before the DCOP changes) and the switching cost c of the dynamic DCOP. The vertical axis shows the difference between the effective rewards of the FORWARD and reactive algorithms. The blue manifold shows the average difference over 50 runs[4] and the green manifold shows the largest difference across runs. These results are averaged over 20 problem instances and all other problem settings are identical to those in Experiment 1.

When the switching cost is 0, the reactive algorithm is able to find an optimal solution at every time step. However, when the cost increases, it may myopically choose a solution that is good for the current time step but bad for future time steps. Thus, the reactive algorithm is best when the switching cost is small and deteriorates with larger switching costs. When the time duration between subsequent time steps is small, the reactive algorithm spends most of the time searching for the solution and little time adopting it; vice versa when the time duration is large. Thus, the reactive algorithm is worst when the time duration is small and improves with longer durations. Finally, the worst-case results have similar trends except that the effects are amplified.

Figure 4 shows the result between HYBRID and the reactive algorithm. The trends are similar to those in Figure 3, except that the largest differences, shown by the green manifold, are larger than those in Figure 3. The reason is that HYBRID uses its observation of the random variables in the current time step to compute a more accurate probability distribution of random variables for the next time step. By observing and getting better predictions on the values of random variables, HYBRID can find better solutions in many runs. Moreover, unlike the reactive algorithm, HYBRID is able to adopt the solution immediately when the problem changes.

These experimental results, therefore, shed light – for the first time to the best of our knowledge – on the identification of characteristics for which each class of algorithms excels in. To summarize, reactive algorithms are well suited for problems with small switching costs and that changes slowly. In contrast, proactive algorithms are well suited for problems with large switching costs and that changes quickly. Finally, our hybrid algorithm combines the strengths of both approaches – it works well in the same type of problems that proactive algorithms work well in and it exploits observations to improve its solutions like reactive algorithms.

8. CONCLUSIONS

Researchers recently proposed the Proactive Dynamic DCOP (PD-DCOP) formulation to model dynamically changing multi-agent coordination problems [10]. It assumes that the problem can only change for a finite number of times *and* this number is known in advanced. Unfortunately, this assumption is unrealistic in many applications. Therefore, in this paper, we propose the Infinite-Horizon PD-DCOP (IPD-DCOP) model, which extends PD-DCOPs to optimize the cumulative reward obtained across an infinite number of time steps. It exploits the convergence properties of Markov chains and assumes that the underlying Markov chain in the problem is guaranteed to converge to the unique stationary distribution. We also show how to compute an optimal solution for this converged chain and adopting it from some horizon h onwards. This method can also be adopted by PD-DCOP algorithms to solve the new IPD-DCOP model. When solving IPD-DCOPs offline, our new distributed offline greedy algorithms FORWARD and BACKWARD find better solutions with comparable or smaller runtimes compared to extensions of PD-DCOP algorithms. When solving IPD-DCOPs online, our new distributed online greedy algorithms FORWARD and HYBRID outperform reactive algorithms in problems with large switching costs and that changes quickly. Our empirical findings on the tradeoffs between proactive and reactive algorithms are the first, to the best of our knowledge, that shed light on this important, but often ignored, issue.

Acknowledgements

The team from NMSU is partially supported by NSF grants 1345232, 1540168, and 1550662. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government. Makoto Yokoo is partially supported by JSPS KAKENHI Grant Number 24220003.

[3]We discretize time into 50ms intervals.

[4]We conducted multiple runs as the actual states of the random variables across time steps can be different across runs.

REFERENCES

[1] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[2] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[3] F. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. Jennings. Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection. In *Proceedings of ICRA*, pages 469–476, 2012.

[4] J. S. Dibangoye, C. Amato, and A. Doniec. Scaling up decentralized MDPs through heuristic search. In *Proceedings of UAI*, pages 217–226, 2012.

[5] J. S. Dibangoye, C. Amato, A. Doniec, and F. Charpillet. Producing efficient error-bounded solutions for transition independent decentralized MDPs. In *Proceedings of AAMAS*, pages 539–546, 2013.

[6] A. Farinelli, A. Rogers, and N. Jennings. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous Agents and Multi-Agent Systems*, 28(3):337–380, 2014.

[7] F. Fioretto, W. Yeoh, and E. Pontelli. A multiagent system approach to scheduling devices in smart homes. In *Proceedings of AAMAS*, 2017.

[8] F. Fioretto, W. Yeoh, E. Pontelli, Y. Ma, and S. Ranade. A DCOP approach to the economic dispatch with demand response. In *Proceedings of AAMAS*, 2017.

[9] R. Gallager. *Stochastic processes: theory for applications*. Cambridge University Press, 2013.

[10] K. Hoang, F. Fioretto, P. Hou, M. Yokoo, W. Yeoh, and R. Zivan. Proactive dynamic distributed constraint optimization. In *Proceedings of AAMAS*, pages 597–605, 2016.

[11] N. Jiang, A. Kulesza, S. Singh, and R. Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of AAMAS*, pages 1181–1189, 2015.

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

[13] R. Lass, E. Sultanik, and W. Regli. Dynamic distributed constraint reasoning. In *Proceedings of AAAI*, pages 1466–1469, 2008.

[14] T. Léauté and B. Faltings. Coordinating logistics operations with privacy guarantees. In *Proceedings of IJCAI*, pages 2482–2487, 2011.

[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 *Proceedings of AAMAS*, pages 310–317, 2004.

[16] S. Miller, S. Ramchurn, and A. Rogers. Optimal decentralised dispatch of embedded generation in the smart grid. In *Proceedings of AAMAS*, pages 281–288, 2012.

[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] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of AAAI*, pages 133–139, 2005.

[19] D. T. Nguyen, W. Yeoh, H. C. Lau, S. Zilberstein, and C. Zhang. Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs. In *Proceedings of AAAI*, pages 1447–1455, 2014.

[20] F. Oliehoek, M. Spaan, C. Amato, and S. Whiteson. Incremental clustering and expansion for faster optimal planning in Dec-POMDPs. *Journal of Artificial Intelligence Research*, 46:449–509, 2013.

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

[22] A. Petcu and B. Faltings. Superstabilizing, fault-containing multiagent combinatorial optimization. In *Proceedings of AAAI*, pages 449–454, 2005.

[23] A. Petcu and B. Faltings. Optimal solution stability in dynamic, distributed constraint optimization. In *Proceedings of IAT*, pages 321–327, 2007.

[24] M. Petrik and S. Zilberstein. Average-reward decentralized Markov decision processes. In *Proceedings of IJCAI*, pages 1997–2002, 2007.

[25] P. Rust, G. Picard, and F. Ramparany. Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems. In *Proceedings of IJCAI*, pages 468–474, 2016.

[26] E. Sultanik, R. Lass, and W. Regli. Dynamic configuration of agent organizations. In *Proceedings of IJCAI*, pages 305–311, 2009.

[27] S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.

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

[29] R. Wallace and E. Freuder. Stable solutions for dynamic constraint satisfaction problems. In *Proceedings of CP*, pages 447–461, 1998.

[30] T. Walsh. Stochastic constraint programming. In *Proceedings of ECAI*, pages 111–115, 2002.

[31] S. Witwicki and E. Durfee. Towards a unifying characterization for quantifying weak coupling in Dec-POMDPs. In *Proceedings of AAMAS*, pages 29–36, 2011.

[32] W. Yeoh, A. Felner, and S. Koenig. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 38:85–133, 2010.

[33] W. Yeoh, P. Varakantham, X. Sun, and S. Koenig. Incremental DCOP search algorithms for solving dynamic DCOPs. In *Proceedings of IAT*, pages 257–264, 2015.

[34] W. Yeoh and M. Yokoo. Distributed problem solving. *AI Magazine*, 33(3):53–65, 2012.

[35] R. Zivan, H. Yedidsion, S. Okamoto, R. Glinton, and K. Sycara. Distributed constraint optimization for teams of mobile sensing agents. *Autonomous Agents and Multi-Agent Systems*, 29(3):495–536, 2015.