Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Decentralized Stochastic Planning with Anonymity in Interactions

Pradeep Varakantham, Yossiri Adulyasak†, Patrick Jaillet‡
School of Information Systems, Singapore Management University
† Singapore-MIT Alliance for Research and Technology (SMART), Massachusetts Institute of Technology
‡ Department of Electrical Engineering and Computer Science, Massachussets Institute of Technology
pradeepv@smu.edu.sg, yossiri@smart.mit.edu, jaillet@mit.edu

Abstract

In this paper, we solve cooperative decentralized stochastic planning problems, where the interactions between agents (specified using transition and reward functions) are dependent on the number of agents (and not on the identity of the individual agents) involved in the interaction. A collision of robots in a narrow corridor, defender teams coordinating patrol activities to secure a target, etc. are examples of such anonymous interactions. Formally, we consider problems that are a subset of the well known Decentralized MDP (DEC-MDP) model, where the anonymity in interactions is specified within the joint reward and transition functions. In this paper, not only do we introduce a general model model called D-SPAIT to capture anonymity in interactions, but also provide optimization based optimal and local-optimal solutions for generalizable sub-categories of D-SPAIT.

Introduction

Decentralized Markov Decision Problem (Dec-MDP) model provides a rich framework to tackle decentralized decision-making problems. However, solving a Dec-MDP problem to create coordinated multi-agent policies in environments with uncertainty is NEXP-Hard (Bernstein et al. 2002). Researchers have typically employed two types of approaches to address this significant computational complexity: (1) approximate dynamic programming and policy iteration approaches (Seuken and Zilberstein 2007; Bernstein, Hansen, and Zilberstein 2005); (2) exploit static and dynamic sparsity in interactions (Becker et al. 2004; Nair et al. 2005; Velagapudi et al. 2011; Witwicki and Durfee 2012; Mostafa and Lesser 2012). In this paper, we pursue a third type of approach, where in we exploit anonymity in interactions. This is generalising on the notion of aggregate influences that has previously been considered in existing work (Witwicki and Durfee 2012; Mostafa and Lesser 2012; Varakantham et al. 2009).

More specifically, we exploit the fact that in many decentralised stochastic planning problems, interactions between agents (specified as joint transition or reward functions) are not dependent on the identity of the agents involved. Instead they are dependent only on the number of agents involved in the interaction. For instance, in the navigation domains of (Melo and Veloso 2011) and (Varakantham et al. 2009), rewards (and transitions) in narrow corridors are dependent only on the number of agents entering the narrow corridor simultaneously and not on the specific agents. Similarly, in the coordination problem introduced by (Yin and Tambe 2011) for Autonomous Underwater and Surface Vehicles (AUVs and ASVs), the rewards are associated with the number of agents sampling the underwater samples simultaneously and not which specific agents. In fact, most sensor network problems (Kumar, Zilberstein, and Toussaint 2011; Nair et al. 2005) have coordination problems with anonymous interactions, where reward is dependent on the number of agents tracking a region (and not which specific sensors). Finally, in the context of coordinating defenders in security patrolling problems (Shieh et al. 2013), the reward for patrolling a target by multiple agents is dependent on the number of agents patrolling a target.

While anonymity in interactions has been considered in the context of competitive games (Roughgarden and Tardos 2002; Varakantham et al. 2012; Ahmed, Varakantham, and Cheng 2012), it has not been considered in the context of decentralized and stochastic cooperative planning and that is the main focus of this paper. Concretely, we first provide a general model called Decentralized Stochastic Planning with Anonymous Interactions (D-SPAIT) to represent anonymity in interactions within the context of the Dec-MDPs framework. Secondly, we develop an optimization based formulation for solving the general D-SPAIT problems and in reference to this optimization, we prove a key theoretical result regarding the scalability of this formulation. Thirdly, we develop specific and scalable methods for generalizable sub-categories of the D-SPAIT problems and finally, we demonstrate the performance of these approaches on random D-SPAIT problems. In our experimental results, we also compare against Softmax based Flow Update (SMFU) algorithm, which was developed for competitive games and can be adapted to solve D-SPAIT problems.

Model: D-SPAIT

In this section, we describe the Decentralized Stochastic Planning with Anonymous InteracTions (D-SPAIT) model that combines the cooperative stochastic planning framework of Decentralized MDP (DEC-MDP) (Bernstein et al. 2002) with anonymity in interactions introduced in the context of multi-agent routing and planning models (Roughgarden and Tardos 2002; Varakantham et al. 2012) from competitive game theory. More formally, D-SPAIT is described

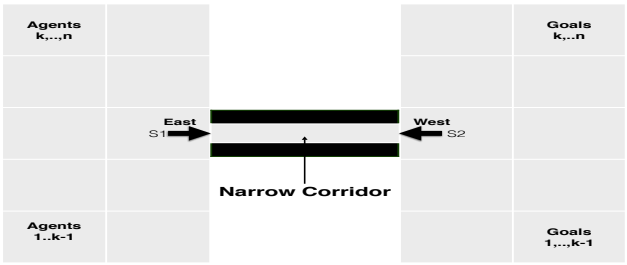

Figure 1: Example of navigation task, where agents are on the left side and goals are on the right side. For agents to achieve the goals, they have to pass through a narrow corridor. If multiple agents pass through the narrow corridor, then each agent receives a penalty that is dependent on the number of agents passing through the corridor. Except for interactions in the narrow corridor, agents can independently move through the grid world.

using the tuple of $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \phi, (\delta_i)_{i\in\mathcal{P}}, T\rangle$:
$\mathcal{P}$ is the agent population, $\mathcal{S}$ is the set of states and $\mathcal{A}$ is the action set of any individual agent in $\mathcal{P}$.
$\mathcal{R}$ is the joint reward and is expressed as the sum of individual rewards, R parameterized by the number of agents due to any anonymous interactions in the joint state, action pair:

$$\mathcal{R}^t(\langle s_1, .., s_{|\mathcal{P}|}\rangle, \langle a_1, .., a_{|\mathcal{P}|}\rangle) = \sum_i R^t(s_i, a_i, d^t_{s_i,a_i}) \quad (1)$$

where $d^t_{s_i,a_i}$ refers to the number of agents impacting the reward function for s_i, a_i in the anonymous interaction. For instance, in the example of Figure 1, number of agents impacting the reward for an agent entering the narrow corridor from S1 and taking action **East** will be those agents which are either entering the narrow corridor from S1 taking action **East** or S2 taking **West**. Therefore, if $x^t_i(s,a)$ denotes the number of times agent i takes action a in s at time step t, then $d^t_{s_1,"East"}$ corresponding to interaction in narrow corridor is given by:

$$d^t_{s_1,"East"} = \sum_i \Big[x^t_i(s_1,"East") + x^t_i(s_2,"West")\Big]$$

When an agent is not part of any interaction, then:
$R^t(s_i, a_i, d^t_{s_i,a_i}) = R^t(s_i, a_i)$
ϕ refers to the joint transition probability and is expressed as a product of individual transition functions, φ parameterized by the number of agents numbers due to any anonymous interactions in the joint state, action pair:

$$\phi^t(\langle s_1, \cdots, s_{|\mathcal{P}|}\rangle, \langle a_1, \cdots, a_{|\mathcal{P}|}\rangle, \langle \hat{s}_1, \cdots, \hat{s}_{|\mathcal{P}|}\rangle) = \prod_i \varphi^t(s_i, a_i, \hat{s}_i, d^t_{s_i,a_i}) \quad (2)$$

δ_i is the initial belief over states for agent i. $\delta^t_i = 0$ for all decision epochs $t > 0$. T is the time horizon. The goal is to compute a strategy for the individual agents so as to maximize joint expected reward over the time horizon. Some important aspects of the D-SPAIT model are:
(1) The individual rewards and transitions are parameterized by the number of agents in Equations 1 and 2 respectively. In general, $d^t_{s_i,a_i}$ can be different for reward and transition functions and we define $d^t_{s_i,a_i}$ for the reward function as:

$$d^t_{s_i,a_i} = \sum_i \sum_{(s,a)\in I^R_{s_i,a_i}} x^t_i(s,a) \quad (3)$$

where $I^R_{s_i,a_i}$ and $I^\varphi_{s_i,a_i}$ refer to the set of state, action pairs that impact (through the number of agents executing action a_i in state s_i) the reward and transition function respectively for (s_i, a_i) pair.
(2) Since the transition and reward functions of agents are independent given the number of agents, $d^t_{s_i,a_i}$ for each s_i, a_i and t values, the value function over all agents is given by:

$$V^0(\{\pi_i\}) = \sum_{i,s_i,a_i,t} R^t(s_i, a_i, d^t_{s_i,a_i}) x^t_i(s_i, a_i) \quad (4)$$

where $x^t_i(s_i, a_i)$ is the frequency of executing action a_i in state s_i for agent i at time t given joint policy $\{\pi_i\}$.

Optimization Approach

We first formulate the solution to a D-SPAIT problem as an optimization problem. Given that the individual reward and transition functions (parameterized by number of agents) can be any arbitrary functions, we cannot solve this optimization problem in the most general case. However, this optimization based formulation helps deduce an important property associated with optimal policies for D-SPAIT. Intuitively, the problem faced by every agent in D-SPAIT is an MDP with the reward and transition functions that depend on the number of agents involved in the anonymous interactions. Due to the dependence of reward and transition function on number of agents, this optimization problem becomes significantly more complicated than a single agent MDP. $x^t_i(s,a)$ in the formulation represents the fraction of times agent i takes action a in state s at time t.

Algorithm 1 SOLVEDSPAIT()

Inputs: R, φ, δ
Outputs: $\mathbf{x} = \{x^t_i(s,a)\}_{i,t,s,a}$

$$\max_{\mathbf{x}} \sum_{s,a,i,t} x^t_i(s,a) \cdot R^t(s, a, \sum_{j,(s',a')\in I^R_{s,a}} x^t_j(s',a'))$$

$$\sum_a x^t_i(s,a) - \sum_{s',a}\Bigg[x^{t-1}_i(s',a)\cdot \varphi^{t-1}(s',a,s, \sum_{j,(s',a')\in I^\varphi_{s,a}} x^{t-1}_j(s',a'))\Bigg] = \delta^t_i(s), \forall i, s \quad (5)$$

$$x^t_i(s,a)\begin{cases}\geq 0, & \forall i,s,a,t \geq 0\\ = 0, & \forall i,s,a,t < 0\end{cases}$$

The formulation in SOLVEDSPAIT() has three aspects:
(1) Since the goal is to compute optimal expected joint reward for all agents, objective includes a summation over all agents i.

(2) The reward function corresponding to a state, action pair in the objective has the number of agents impacting the reward value in that state, action pair, i.e., $\sum_{j,(s',a')\in I^R_{s,a}} x^t_j(s',a'))$ as a parameter. Since we have a finite horizon problem, $x^t_j(s',a')$ corresponds to the proportion of one agent (i.e. j) taking action a in state s' at time t' and consequently, $\sum_j x^t_j(s',a')$ corresponds to the number of agents taking action a' in state s' at time t.
(3)Similar to the reward function above, the transition function in the first constraint has an extra parameter corresponding to the number of agents executing a dependent action in dependent state.

We now prove an important property for populations, where the starting belief distribution, b over states is the same for all agents. We use $\mathbf{x}_i$ to represent the vector of flow values, $x^t_i(s,a)$ for all s,a,t.

Proposition 1 *If $\boldsymbol{x} = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_{|\mathcal{P}|}\right]$ is the optimal solution for the optimization problem in Algorithm 1 and $\forall i,j,s : \delta^0_i(s) = \delta^0_j(s)$, then $\hat{\boldsymbol{x}}$ defined below is also an optimal solution for the same optimization problem :*

$$\hat{\boldsymbol{x}} = \Big[\frac{\sum_i \boldsymbol{x}_i}{|\mathcal{P}|}, \frac{\sum_i \boldsymbol{x}_i}{|\mathcal{P}|}, \dots\Big]$$

Proof. We first show that $\hat{\mathbf{x}}$ belongs to the feasible region. By summing up the LHS and RHS of the Constraint 5 for all agents i and for each value of s and t, we have:

$$\sum_a \frac{\sum_i x^t_i(s,a)}{|\mathcal{P}|} - \sum_{s',a} \frac{\sum_i x^{t-1}_i(s',a)}{|\mathcal{P}|} \cdot \Big[\varphi^{t-1}\Big(s',a,s, \sum_{i,(s',a')\in I^{\varphi}_{s,a}} |\mathcal{P}| \cdot \frac{x^{t-1}_i(s',a')}{|\mathcal{P}|}\Big)\Big] = \delta^t(s)$$

Therefore, if $\mathbf{x}$ is a feasible solution to the optimization problem in SOLVEDSPAIT(), then so is $\hat{\mathbf{x}}$. Now, we show that $\hat{\mathbf{x}}$ is also an optimal solution. To demonstrate this, we calculate the objective value corresponding to $\mathbf{x}$ and show that it is equal to the value obtained with $\hat{\mathbf{x}}$.

$$\mathcal{F}(\mathbf{x}) = \sum_{s,a,t} \Big[\sum_i x^t_i(s,a)\Big] \cdot R^t\Big(s,a,\sum_j x^t_j(s,a)\Big)$$

$$= \sum_{s,a,t} \Big[|\mathcal{P}| \cdot \frac{\sum_i x^t_i(s,a)}{|\mathcal{P}|}\Big] \cdot R^t\Big(s,a,\Big[|\mathcal{P}| \cdot \frac{\sum_j x^t_j(s,a)}{|\mathcal{P}|}\Big) = \mathcal{F}(\hat{\mathbf{x}})$$

■

Proposition 1 provides a very important result that removes the dependence of SOLVEDSPAIT() on the number of agents. It indicates that the class of symmetric joint policies is guaranteed to contain at least one optimal joint policy. Hence, we can rewrite the optimization problem in SOLVEDSPAIT() as the one in SOLVEDSPAIT-HOMOGENEOUS(). While the scalability of SOLVEDSPAIT-HOMOGENOUS is independent of number of agents, it is dependent on two key factors.
(1) The structure of I^{φ} and I^R functions, i.e., the dependence of reward and transition for a given state action pair on number of agents in other state action pairs. We first describe the simpler case, where dependence of reward and transition for a given state action pair (s,a) at time t is dependent on number of agents executing action a in state s at the same time step t, i.e.,

$$I^{\varphi}_{s,a} = \{(s,a)\}; I^R_{s,a} = \{(s,a)\} \quad (6)$$

Algorithm 2 SOLVEDSPAIT-HOMOGENOUS()

Inputs: R, φ, δ
Outputs: $\mathbf{x} = \{x^t(s,a)\}_{t,s,a}$

$$\max_{\mathbf{x}} \sum_{s,a,t} |\mathcal{P}| \cdot x^t(s,a) \cdot \mathcal{R}^t(s,a,|\mathcal{P}| \cdot \sum_{(s',a')\in I^R_{s,a}} x^t(s',a'))$$

$$\sum_a x^t(s,a) - \sum_{s',a} x^{t-1}(s',a) \cdot \varphi^{t-1}\Big(s',a,s,|\mathcal{P}| \cdot \sum_{(\tilde{s},\tilde{a})\in I^{\varphi}_{s',a}} x^{t-1}(\tilde{s},\tilde{a})\Big) = \delta^t(s), \forall i,s$$

$$x^t(s,a) \begin{cases} \geq 0, & \forall i,s,a,t \geq 0 \\ = 0, & \forall i,s,a,t < 0 \end{cases}$$

However, our approaches are applicable to general I^{φ} and I^R structures and we provide specific comments in the subsection on "Discussion on generic I^R and I^{φ} Structures".
(2) Functional form of reward, R and transition, φ functions with respect to the number of agents, $d^t(s,a)$. For instance, it will be very difficult to solve SOLVEDSPAIT-HOMOGENOUS for non-linear and non-convex functions. We focus on functional forms for reward and transitions that can be used to approximate non-linear and non-convex functional forms, such as: (a) Linear; (b) Piecewise constant; (c) Piecewise linear and convex.

We now focus on functional forms where it is possible to provide scalable approaches (optimal and local-optimal).

Linear Rewards

Our first set of assumptions on the reward and transition functions where we can provide a scalable approach are:
(1) Transition function, ϕ is independent of d, i.e. $\phi^t(s,a,s',d^t_{s,a}) = \phi^t(s,a,s')$.
(2) Reward function, $\mathcal{R}$ is dependent linearly on $d^t_{s,a}$, i.e. $\mathcal{R}^t(s,a,d_{s,a}) = m^t_{s,a} \cdot d^t_{s,a} + c^t_{s,a}$. More specifically, we are interested in linear functions, where the reward decreases with increase in number of agents, that is to say, all $m^t_{s,a}$ are less than 0. It should be noted that a positive set of slopes imply that the problem is not concave and hence requires more sophisticated approximations.

Linearity of reward function and the structure of I (from Equation 6) reduces the objective in Algorithm 2 as follows:

$$\sum_{s,a,t} |\mathcal{P}| \cdot x^t(s,a) \cdot \mathcal{R}^t(s,a,|\mathcal{P}| \cdot x^t(s,a)) = \sum_{s,a,t} \Big[m^t_{s,a} \cdot |\mathcal{P}|^2 \cdot \big(x^t(s,a)\big)^2 + c^t_{s,a} \cdot |\mathcal{P}| \cdot x^t(s,a)\Big] \quad (7)$$

Thus, even when the reward function is linear, we have a quadratic objective. A general quadratic objective cannot be solved in polynomial time and typically requires non-linear solvers. However, in cases of resource congestion or collision interactions, reward decreases with number of agents and hence there is a negative slope on the reward function. In such cases, we can show that the objective is concave and hence has only one maxima.

Proposition 2 *If all slopes $m^t_{s,a}$ are negative, then the objective provided in Equation 7 is concave.*

Proof Sketch. We prove this proposition by showing that the Hessian matrix, $\mathcal{H}$ corresponding to the objective function is negative semidefinite. Let $\mathcal{F}$ denote the objective function, then

$$\mathcal{F}(\mathbf{x}) = \sum_{s,a,t} m^t_{s,a} \cdot |\mathcal{P}|^2 \cdot \left(x^t(s,a)\right)^2 + c^t_{s,a} \cdot |\mathcal{P}| \cdot x^t(s,a)$$

We first consider the diagonal element for the row corresponding to $\hat{s}, \hat{a}, \hat{t}$:

$$\mathcal{H}(\langle \hat{s}, \hat{a}, \hat{t} \rangle, \langle \hat{s}, \hat{a}, \hat{t} \rangle) = \frac{\partial^2 \mathcal{F}}{\partial^2 [x^{\hat{t}}(\hat{s}, \hat{a})]} = 2 \cdot m^{\hat{t}}_{\hat{s},\hat{a}} \cdot |\mathcal{P}|^2$$

We now consider a non diagonal element in the row corresponding to $\hat{s}, \hat{a}, \hat{t}$, and column corresponding to $\tilde{s}, \tilde{a}, \tilde{t}$:

$$\mathcal{H}(\langle \hat{s}, \hat{a}, \hat{t} \rangle, \langle \tilde{s}, \tilde{a}, \tilde{t} \rangle) = \frac{\partial^2 \mathcal{F}}{\partial [x^{\hat{t}}(\hat{s}, \hat{a})] \partial [x^{\tilde{t}}(\tilde{s}, \tilde{a})]} = 0$$

Therefore, all the diagonal elements of $\mathcal{H}$ are negative and other elements are zero. Hence, for any vector $\mathbf{y}$: $\mathbf{y}^T \mathcal{H} \mathbf{y} \leq 0$. Thus $\mathcal{F}$ is negative semi definite and concave. ■

Proposition 2 is important as concavity implies the maximization problem in Algorithm 3 can be solved in polynomial time. More practically, the optimization can be solved using standard LP solvers like CPLEX.

Algorithm 3 SOLVEDSPAIT-LINEAR ($\langle \mathbf{m}, \mathbf{c} \rangle$)

Inputs: $\{m^t_{s,a}, c^t_{s,a}\}_{s,a,t}, \varphi, \delta$
Outputs: $\mathbf{x} = \{x^t(s,a)\}_{t,s,a}$

$$\max_x \sum_{s,a,t} \left[m^t_{s,a} \cdot \left(|\mathcal{P}| \cdot x^t(s,a) \right)^2 + c^t_{s,a} \cdot |\mathcal{P}| \cdot x^t(s,a) \right] \textbf{ s.t.}$$

$$\sum_a x^t(s,a) - \sum_{s',a} x^{t-1}(s',a) \varphi^{t-1}(s',a,s) = \delta^t(s) \quad \forall s,t$$

$$x^t(s,a) \begin{cases} \geq 0, & \forall i,s,a,t \geq 0 \\ = 0, & \forall i,s,a,t < 0 \end{cases}$$

PieceWise Constant (PWC) Transition Function

From the assumptions in the previous sub section, we relax the assumption with respect to the transition function. More specifically, we no longer assume that transition function is independent of d, instead, we assume a piecewise constant dependency on d, i.e., $\forall s, a, s', t$, we have

$$\varphi^t(s,a,s',d) = \begin{cases} p^{t,1}(s,a,s'), & \textbf{if } \check{d}_1 \leq d \leq \hat{d}_1 \\ p^{t,2}(s,a,s'), & \textbf{if } \check{d}_2 \leq d \leq \hat{d}_2 \\ \ldots, & \ldots \end{cases} \quad (8)$$

A key advantage of piecewise constant functions is their ability to closely approximate very general functions. The impact of the piecewise constant dependency on d is on the flow preservation constraint $\forall s, a$:

$$\sum_a x^t(s,a) - \sum_{s',a} x^t(s',a) \cdot \varphi^t(s',a,s,|\mathcal{P}| \cdot x^t(s',a)) = \delta^t(s)$$

Since transition function is no longer a constant, the terms $x^t(s',a) \cdot \varphi^t(s',a,s,|\mathcal{P}| \cdot x^t(s',a))$ are a product of two variables and hence are non-linear. In this section, we contribute novel mechanisms to linearize the above terms. We use a set of new variables, $\{X^t(s',a,s)\}$ that represent the product terms, i.e.,:

$$X^t(s',a,s) = x^t(s',a) \cdot \varphi^t(s',a,s,|\mathcal{P}| \cdot x^t(s',a)) \quad (9)$$

To linearize these product terms, we first provide equivalent expressions for the new variables as follows:

$$X^t(s',a,s) = \sum_k X^{t,k}(s',a,s), \ \ where \quad (10)$$

$$X^{t,k}(s',a,s) =$$

$$\begin{cases} p^{t,k}(s',a,s) \cdot x^t(s',a), & \textbf{if } \check{d}_k \leq |\mathcal{P}| \cdot x^t(s',a) \leq \hat{d}_k \\ 0, & \textbf{otherwise} \end{cases} \quad (11)$$

The above expressions require two linearization steps. Firstly, we need to linearize the conditional required to check if the overall flow on state s' and a (= $|\mathcal{P}| \cdot x^t(s',a)$) belongs to an interval $[\check{d}_k, \hat{d}_k]$. The binary variables, $y^{t,k}(s',a)$ are used to represent the satisfaction of this condition, i.e., $y^{t,k}(s',a) = 1$ implies that the $|\mathcal{P}| \cdot x^t(s',a)$ value belongs to the interval $[\check{d}_k, \hat{d}_k]$. More specifically, the constraints that are used to achieve this assignment are as follows:

$$\sum_k y^{t,k}(s,a) = 1 \quad (12)$$

$$1 - y^{t,k}(s,a) \geq \frac{\check{d}_k - |P| \cdot x^t(s,a)}{M} \quad (13)$$

$$1 - y^{t,k}(s,a) \geq \frac{|P| \cdot x^t(s,a) - \hat{d}_k}{M} \quad (14)$$

Secondly, we need to set $X^{t,k}(s',a,s)$ to $p^{t,k}(s',a,s) \cdot x^t(s',a)$ if $y^{t,k}(s',a)$ is set to 1 and a value of zero otherwise. The linearized constraints corresponding to this conditional assignment are as follows:

$$X^{t,k}(s',a,s) \leq x^t(s',a) \cdot p^{t,k}(s',a,s) \quad (15)$$

$$X^{t,k}(s',a,s) \leq y^{t,k}(s',a) \cdot M \quad (16)$$

$$X^{t,k}(s',a,s) \geq x^t(s',a) \cdot p^{t,k}(s',a,s) - (1 - y^{t,k}(s',a)) \cdot M \quad (17)$$

Proposition 3 *Constraints 12, 13 and 14 ensure that:*
(a) $y^{t,k}(s,a) = 0$, if $|P| \cdot x^t(s,a) \notin [\check{d}_k, \hat{d}_k]$.
(b) $y^{t,k}(s,a) = 1$, if $|P| \cdot x^t(s,a) \in [\check{d}_k, \hat{d}_k]$.

Proposition 4 *Constraints 10, 15, 16, 17 ensure that definition of X variables in Equation 9 are satisfied.*

Proposition 5 *M in constraints 16 and 17 can be set to 1 without violating their correctness. Large values of M imply longer run-times (Hooker 1995) and thus, Proposition 5 is important in reducing the run-time required to solve the optimization problem significantly.*

Algorithm 4 SOLVEDSPAIT-PWC()

Inputs: $\{c_{s,a}^{t,k}\}_{t,k,s,a}, \varphi, \delta$

Outputs: $\mathbf{x} = \{x^t(s,a)\}_{i,t,s,a}$

$$\max_x \sum_{s,a,t,k} |\mathcal{P}| \cdot Z^{t,k}(s,a) \quad \textbf{s.t.}$$

$$Z^{t,k}(s,a) \leq z^{t,k}(s,a) \cdot M; Z^{t,k}(s,a) \leq x^t(s,a) * c_{s,a}^{t,k}$$

$$Z^{t,k}(s,a) \geq x^t(s,a) * c_{s,a}^{t,k} - (1 - z^{t,k}(s,a)) \cdot M$$

$$\sum_a x^t(s,a) - \sum_{s',a} x^{t-1}(s',a)\varphi^{t-1}(s',a,s) = \delta^t(s) \quad \forall s,t$$

$$\sum_k z^{t,k}(s,a) = 1; 1 - z^{t,k}(s,a) \geq \frac{\check{d}_k - |P| \cdot x^t(s,a)}{M}$$

$$1 - z^{t,k}(s,a) \geq \frac{|P| \cdot x^t(s,a) - \hat{d}_k}{M}$$

$$x^t(s,a) \begin{cases} \geq 0, & \forall i,s,a,t \geq 0 \\ = 0, & \forall i,s,a,t < 0 \end{cases}$$

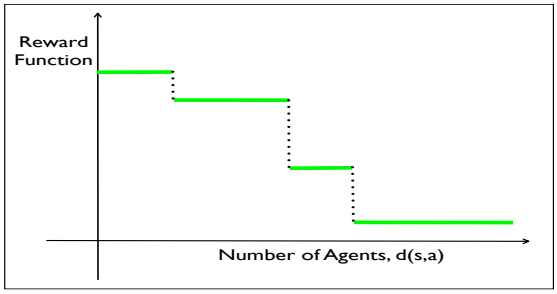

Figure 2: Example of PWC reward

PieceWise Constant (PWC) Rewards

We now consider piecewise constant reward functions (Figure 2) for $R(s,a,d)$ in a similar vein to the PWC transition function. We employ the same linearization techniques as the ones used for piecewise constant transition function. While we only provide the algorithm for piecewise constant rewards in SOLVEDSPAIT-PWC, the constraints introduced for piecewise constant transitions can be included directly into this algorithm.

PieceWise Linear and Convex (PWLC) Rewards

We now generalize the reward function to be piecewise linear and convex (Figure 3[a]). Unlike the previous cases where we have provided an exact linear program, we are only able to provide a local-optimal approach for this case. Formally, the piecewise linear and convex reward function is specified as follows:

$$\mathcal{R}^t(s,a,d) = \max_k \{m_{s,a}^{t,k} \cdot d + c_{s,a}^{t,k}\} \qquad (18)$$

Therefore, the objective reduces to:

$$\sum_{s,a,t} |\mathcal{P}| \cdot x^t(s,a) \cdot \max_k \left[m_{s,a}^{t,k} \cdot |\mathcal{P}| \cdot x^t(s,a) + c_{s,a}^{t,k}\right]$$

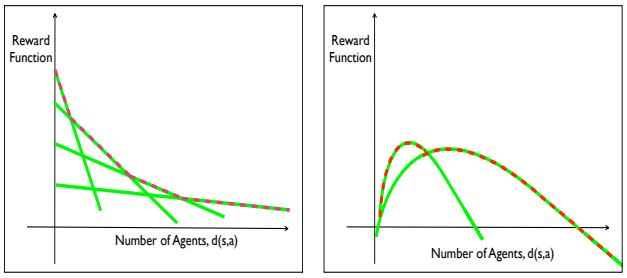

Figure 3: (a) Example of a PWLC reward function; (b) Maximum of two concave functions

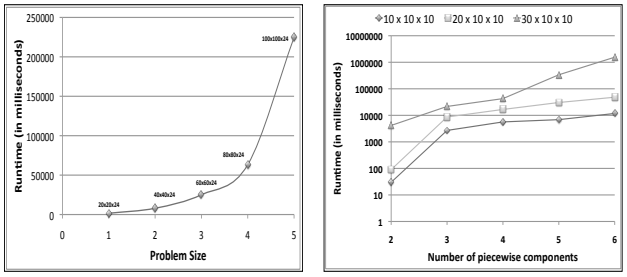

Figure 4: Run-time performance of (a) SOLVEDSPAIT-LINEAR(); (b) SOLVEDSPAIT-PWC().

$$= \sum_{s,a,t} \max_k \left[m_{s,a}^{t,k} \cdot |\mathcal{P}|^2 \cdot \left(x^t(s,a)\right)^2 + c_{s,a}^{t,k} \cdot |\mathcal{P}| \cdot x^t(s,a)\right]$$

Figure 3[b] provides an example to illustrate that the maximum of multiple concave functions is typically non-concave and has multiple local maxima. Hence, well known convex or concave programming solvers cannot be used directly to solve this optimization problem. Therefore, we pursue an approximate approach that iteratively improves on the flow variables **x** and the piecewise linear components, **k**.

Algorithm 5 SolveDSPAIT-PWLC()

1: $\mathbf{k} = \langle k_{0,0}^0, \ldots, k_{s,a}^t, \ldots \rangle \leftarrow$ GETRANDCOMPNNTS()
2: $\mathbf{x} \leftarrow$ SOLVEDSPAIT-LINEAR($\mathbf{k}$); $\mathbf{x}_1 \leftarrow \emptyset$
3: **while** $\mathbf{x} \neq \mathbf{x}_1$ **do**
4: $\quad \mathbf{x}_1 \leftarrow \mathbf{x}$; $\mathbf{k} \leftarrow$ GETBESTK($\mathbf{x}$)
5: $\quad \mathbf{x} \leftarrow$ SOLVEDSPAIT-LINEAR($\mathbf{k}$)
6: **return x**

Algorithm 5 provides the approximate approach for solving this category of problems. Initially, we select a random linear component (slope, m and intercept, c) from the available components for every state action pair at every time step. On line 1, we obtain these components **k** using the GETRANDCOMPNNTS() function. Given a fixed set of linear components, **k** for every state, action pair at each time step, the resulting problem has linear reward functions and hence we use the SOLVEDSPAIT-LINEAR() function of Algorithm 3 (line 2) to obtain the optimal flow, **x** . We then find the best component vector **k** corresponding to the flow vector **x** on line 4 using the GETBESTK() function. This

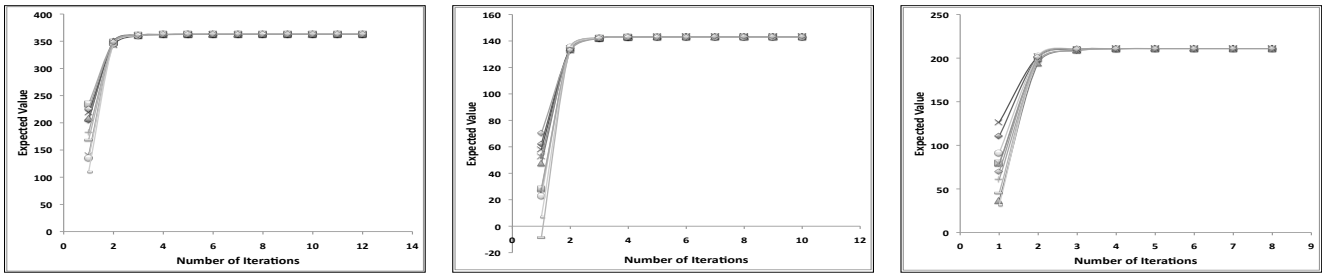

Figure 5: Convergence of SOLVEDSPAIT-PWLC function on three different example problems.

iterative improvement process over **k** and **x** (lines 4-5) is continued until the convergence of the flow vector.

In this algorithm, both GETBESTK() and SOLVEDSPAIT-LINEAR() functions will never find a solution that has lower value than the current solution as current solution is part of the search space. Each iteration thus does not reduce the solution quality and since there are a finite number of PWLC components, the algorithm will converge to a local optima.

Discussion on generic I^R and I^φ Structures

Our contributions earlier are provided for the case where $I^R(s,a) = \{(s,a)\}$ and $I^\varphi(s,a) = \{(s,a)\}$. We now focus on generic I^R and I^φ structures, where reward and transition functions can depend on sum of agent numbers in multiple different state, action pairs. We now comment on the generality of the our contributions in the context of generic I^R and I^φ structures.

(a) SOLVEDSPAIT-PWC() with PWC reward and transition functions can be used to solve with generic I^φ and I^R structure. There is no change in approach required.

(b) If we have linear reward function with a generic $I^R(s,a)$, the Hessian matrix for the objective is not negative semi definite with negative slopes. Hence, the problem is not concave. However, since the quadratic optimization problem contains products of the form $x^t(s,a) \cdot x^t(s',a')$ and each of the flow values are less than 1 (since finite horizon MDPs), separable programming (applied in the context of MDPs by (Ahmed et al. 2013)) can be used to provide tight approximations for the the quadratic terms. More specifically,

$$x^t(s,a) \cdot x^t(s',a') = A^2 - B^2$$
$$A = \frac{x^t(s,a) + x^t(s',a')}{2}; B = \frac{x^t(s,a) - x^t(s',a')}{2}$$

A and B can be approximated as follows, by dividing [0,1] into W intervals, $\{br_w\}$ and using SOS2 constraints:

$$A = \sum_w \lambda_w \cdot br_w; A^2 = \sum_w \lambda_w \cdot (br_w)^2$$
$$\sum_w \lambda_w = 1; SOS2(\{\lambda_w\})$$

SOLVEDSPAIT-LINEAR() and consequently SOLVEDSPAIT-PWLC will be updated to include this approximation of quadratic terms.

Experimental Results

In this section, we demonstrate the following:
(1) Run-time performance of SOLVEDSPAIT-LINEAR(), SOLVEDSPAIT-PWC approaches.
(2) Run-time performance, scalability and solution quality for the local optimal approach in SOLVEDSPAIT-PWLC.
(3) Performance comparison against the SMFU(Varakantham et al. 2012), which also exploits anonymous interactions in competitive settings.

We generated random DSPAIT problems, where both the reward and transition functions were generated randomly. For the transition function, we varied the reachability (number of states with positive probability) of the states and generated random transition probabilities. For the reward function, we generated random numbers between a range while satisfying the assumptions of the specific categories of functions (ex: negative slopes, monotonically non-increasing, etc.). We then used our optimization algorithms to solve [1] these random DSPAIT problems.

Figure 4(a) provides the runtime results for SOLVEDSPAIT-LINEAR(). A problem type is denoted by a cross product of number of states (zones), number of actions (zones) and number of decision epochs (time horizon). The five problem types we considered for the linear reward functions were: (20x20x24, 40x40x24, 60x60x24, 80x80x24,100x100x24). We randomly generated 25 instances for each problem type and each point in the graph Figure 4(a) is an average run-time for 25 instances. The biggest problem was solved within 4 minutes. The performance of the mixed integer linear program in SOLVEDSPAIT-PWC() is shown in Figure 4(b). We were able to solve problems with up to 30 states, 10 actions and 10 decision epochs with 6 piecewise constant components per state action pair (for both the reward and transition functions) within 20 mins. Every point in the graph is averaged over 25 instances.

We show the performance of the SOLVEDSPAIT-PWLC function by considering 10 piecewise linear components for every state action pair, time step corresponding to the reward function. We experimented with multiple sets of problems where the number of states, actions and time steps were varied from 20-100, 20-100 and 5-24 respectively. In all the cases, the number of iterations required for convergence

[1] All the linear and quadratic optimization problems were solved using the commercial optimization software CPLEX 12.2 on a 1.8 GHz Intel Core i5 machine with 8GB 1600 MHz DDR3 RAM.

was less than 15 ($\approx$ solving SOLVEDSPAIT-LINEAR() 15 times). Since SOLVEDSPAIT-PWLC does not provide optimal solutions, the key results are with respect to solution quality. While, SOLVEDSPAIT-PWLC converges to local optima , a very important and practically interesting phenomenon is observed with respect to the quality of the local optima. Figure 5 provides results on all our experiments with PWLC reward functions with number of components ranging from 5 - 20. Here are the key results:
(a) The total number of iterations for convergence (the number of times the while loop in SOLVEDSPAIT-PWLC() is executed) varied between 9-14. However, the number of iterations required to be near local optima was only 3 or 4.
(b) For each problem , we started with 10 random starting values of **k** and as witnessed in all the three graphs, the starting solution quality has a very high variance. However, the algorithm converged to local optima that were very close to each other on all the problems (3 shown here and numerous others that we experimented with). While we do not know if the global optima is close to these set of local optima, this is a unique result for a local optimal algorithm, especially since the problems were generated randomly.

Previously, SMFU was proposed by (Varakantham et al. 2012) to compute equilibrium solutions for stochastic decision making problems for competitive settings. SMFU exploits the anonymity in interactions while computing equilibrium and hence our comparison against SMFU. We compare against SMFU and not against D-TREMOR (Velagapudi et al. 2011) because:

- SMFU employs shaping of model based on influences of other agents, similar to D-TREMOR.
- SMFU exploits anonymity in interactions and scales to problems with thousands of agents.
- Unlike D-TREMOR, SMFU converges to local optimal.

Since SMFU's solution depends on the initial policy, performance was averaged over multiple initializations of the starting policy. The SOLVEDSPAIT-LINEAR() and SOLVEDSPAIT-PWLC() computed optimal policies at run-times (at least) an order of magnitude faster than the runtime by SMFU. For instance, on the 80x80x24 problem (equivalent in size to the real world taxi fleet optimization problem in (Varakantham et al. 2012)), the SOLVEDSPAIT-LINEAR problem generated optimal solutions in 70 seconds, whereas SMFU took close to 30 minutes. However, with respect to SOLVEDSPAIT-PWC(), SMFU computed solutions in runtimes that were an order of magnitude shorter on large problems (ex: 30x10x10). While SMFU returned optimal solutions in few cases, overall it returned solutions that were around 60% of the optimal.

Acknowledgements

This research is supported in part by the National Research Foundation (NRF) Singapore through the Singapore MIT Alliance for Research and Technology (SMART) and its Future Urban Mobility (FM) Interdisciplinary Research Group.

References

Ahmed, A.; Varakantham, P.; Adulyasak, Y.; and Jaillet, P. 2013. Regret based robust solutions for uncertain markov decision processes. In *NIPS'13*.

Ahmed, A.; Varakantham, P.; and Cheng, S.-F. 2012. Decision support for agent populations in uncertain and congested environments. In *UAI'12*, 44–53.

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

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

Bernstein, D. S.; Hansen, E. A.; and Zilberstein, S. 2005. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 1287–1292.

Hooker, J. 1995. Logic-based benders decomposition. *Mathematical Programming* 96:2003.

Kumar, A.; Zilberstein, S.; and Toussaint, M. 2011. Scalable multiagent planning using probabilistic inference. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2140–2146.

Melo, F. S., and Veloso, M. M. 2011. Decentralized mdps with sparse interactions. *Artif. Intell.* 175(11):1757–1789.

Mostafa, H., and Lesser, V. 2012. Offline planning for communication by exploiting structured interactions in decentralized mdps. In *IAT'09*.

Nair, R.; Varakantham, P.; Tambe, M.; and Yokoo, M. 2005. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 133–139.

Roughgarden, T., and Tardos, É. 2002. How bad is selfish routing? *Journal of the ACM* 49(2):236–259.

Seuken, S., and Zilberstein, S. 2007. Improved memory-bounded dynamic programming for decentralized POMDPs. In *UAI*.

Shieh, E.; Jain, M.; Jiang, A. X.; and Tambe, M. 2013. Ef?ciently solving joint activity based security games. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Varakantham, P.; Kwak, J. Y.; Taylor, M.; Marecki, J.; Scerri, P.; and Tambe, M. 2009. Exploiting coordination locales in distributed POMDPs via social model shaping. In *Nineteenth International Conference on Automated Planning and Scheduling*.

Varakantham, P.; Cheng, S.-F.; Gordon, G.; and Ahmed, A. 2012. Decision support for agent populations in uncertain and congested environments. In *AAAI*, 1471–1477.

Velagapudi, P.; Varakantham, P.; Sycara, K.; and Scerri, P. 2011. Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *AAMAS*, 955–962.

Witwicki, S., and Durfee, E. 2012. Influence-based policy abstraction for weakly-coupled dec-pomdps. In *ICAPS'10*.

Yin, Z., and Tambe, M. 2011. Continuous time planning for multiagent teams with temporal constraints. In *International Joint Conference on Artificial Intelligence (IJCAI)*.