8-2017

Proactive and reactive coordination of non-dedicated agent teams operating in uncertain environments

Pritee AGRAWAL
Singapore Management University, priteea.2013@phdis.smu.edu.sg

Pradeep VARAKANTHAM
Singapore Management University, pradeepv@smu.edu.sg

Citation

AGRAWAL, Pritee and VARAKANTHAM, Pradeep. Proactive and reactive coordination of non-dedicated agent teams operating in uncertain environments. (2017). *Proceedings of the 26th International Joint Conference on Artificial Intelligence IJCAI-17, Melbourne, Australia, August 19-25*. 28-34.
Available at: https://ink.library.smu.edu.sg/sis_research/3850

Proactive and Reactive Coordination of Non-dedicated Agent Teams Operating in Uncertain Environments

Pritee Agrawal, Pradeep Varakantham
School of Information Systems, Singapore Management University, Singapore 188065
priteea.2013@phdis.smu.edu.sg, pradeepv@smu.edu.sg

Abstract

Domains such as disaster rescue, security patrolling etc. often feature dynamic environments where allocations of tasks to agents become ineffective due to unforeseen conditions that may require agents to leave the team. Agents leave the team either due to arrival of high priority tasks (e.g., emergency, accident or violation) or due to some damage to the agent. Existing research in task allocation has only considered fixed number of agents and in some instances arrival of new agents on the team. However, there is little or no literature that considers situations where agents leave the team after task allocation. To that end, we first provide a general model to represent non-dedicated teams. Second, we provide a proactive approach based on sample average approximation to generate a strategy that works well across different feasible scenarios of agents leaving the team. Furthermore, we also provide a 2-stage approach that provides a 2-stage policy that changes allocation based on observed state of the team. Third, we provide a reactive approach that rearranges the allocated tasks to better adapt to leaving agents. Finally, we provide a detailed evaluation of our approaches on existing benchmark problems.

1 Introduction

In delivery of services or goods [Dantzig and Ramser, 1959; Dolgov and Durfee, 2006], task allocations to individual vehicles are based on uncertain travel times to delivery locations. Also, in disaster rescue scenarios [Varakantham *et al.*, 2014; Velagapudi *et al.*, 2011; Varakantham *et al.*, 2009], victims have to be allocated to robots while considering the uncertainty in travelling through disaster prone areas. Furthermore, in large warehouses [Hazard *et al.*, 2006; Wurman *et al.*, 2007] of online portals such as Amazon, movement of automated robots fetching goods based on online orders (uncertainty) have to be coordinated in the usage of pathways (resources). Finally, in problems associated with safety and security, recent research [Brown *et al.*, 2014; Shieh *et al.*, 2014; Varakantham *et al.*, 2013] has considered patrolling problems where a team of defenders coordinate to secure a set of targets against an observing adversary (e.g., traffic police coordination for patrolling roads in order to reduce violations).

These domains have the following common characteristics: (a) A team of agents (e.g., ambulances/fire trucks) coordinate plans to achieve a goal or to optimize a certain criterion (e.g., save victims); (b) There is transition uncertainty in planning problems of individual agents, either due to travelling on roads (due to traffic) or due to uncertain demand (online orders) or physical constraints (e.g., robots); (c) Actions of agents either require the availability of resources (roads, paths, tools, etc.) or completion of tasks allocated (target surveillance, delivery of items, etc.). Furthermore, there is usually a hard constraint on the number of tasks/resources available and this causes agent plans to be dependent on each other; and most importantly (d) The individual agents have a chance of leaving the team at any time step due to either a breakdown (e.g., in the case of large warehouses, individual robots leave the system either to get charged or because of malfunction) or to address a higher priority task (e.g., in case of traffic patrolling problems, traffic police have to attend to incidents/accidents in addition to patrolling roads).

We are interested in application problems with above mentioned characteristics. Due to the non-dedicated nature of the team members (as agents can leave the team at any time), we refer to these teams as non-dedicated agent teams. The notion of non-dedicated teams was introduced and studied extensively in the context of Belief Desire Intention (BDI) frameworks [Cohen and Levesque, 1991; Grosz and Kraus, 1996; Tambe, 1997] for multi-agent planning. However, those works relied on existence of plan libraries and considered primarily deterministic outcomes to actions. In this work, we generate plans automatically and focus on domains with probabilistic outcomes to actions.

Another closely related thread of research is on adhoc teams [Barrett *et al.*, 2017], which focuses on a newly added team member that cooperates with a variety of teammates, without directly altering the behavior of teammates. We focus on non-dedicated teams where the configuration of the team is altered to accommodate the leaving team member. More recently, Shieh *et al.* [Shieh *et al.*, 2014] re-introduced the notion of non-dedicated teams in the context of defender teams patrolling against an observing adversary. While Shieh *et al.*'s work considered non-deterministic outcomes to actions, their exhaustive offline approach is not scalable.

To that end, we first provide a general model for the study of non-dedicated agent teams operating in uncertain environments. We then provide multiple proactive and reactive approaches to generate policies for individual agents in non-dedicated agent teams. One of our main contributions is a proactive approach that relies on sampling and decomposition to efficiently generate policies for individual agents. Another major contribution involves the formulation of a two stage approach where the state of the team (i.e., agent exits) at the end of first stage is input for the second stage to provide a two stage policy for the agents. Finally, our exhaustive evaluation on benchmark problems illustrate the improved performance of our proactive and reactive approaches.

2 Model: ND-TasC-MDP

We build on the TasC-MDP model introduced by Dolgov *et al.* [Dolgov and Durfee, 2006] and extended by Agrawal *et al.* [Agrawal *et al.*, 2016] to represent the problems with non-dedicated teams operating in uncertain environments. We refer to this model as Non Dedicated TasC-MDP (ND-TasC-MDP) and is characterised by the tuple:

$$\left\langle \mathcal{A}g, \Gamma, \mathcal{C}, D, \langle \mathbf{M}_i \rangle_{i \in \mathcal{A}g}, \{\Delta_i\}_{i \in \mathcal{A}g}, H \right\rangle$$

- $\mathcal{A}g$ is the set of agents.
- Γ is the different types of tasks.
- $\mathcal{C} = \bigcup_{\tau \in \Gamma} \mathcal{C}(\tau)$ corresponds to the set of all tasks, where $\mathcal{C}(\tau)$ is the set of tasks of type τ. $|\mathcal{C}(\tau)|$ is the capacity bound for tasks of type τ.
- $D = \{\tau_1 \prec \tau_2, \tau_1 \parallel \tau_2, \ldots\}$ represent the set of allocation constraining dependencies and temporal dependencies (refer [Agrawal *et al.*, 2016] for more details). For purposes of easy exposition, we focus on problems with independent tasks, however, all our approaches are easily extendable to cases with dependencies.
- $\mathbf{M}_i = \langle S_i, A_i, P_i, R_i, \rho_i, \alpha_i^0 \rangle$ is the MDP model for agent i along with the task associations of actions.
  - S_i, A_i, P_i, R_i are the standard MDP elements.
  - $\rho_i : A_i \times \Gamma \rightarrow \mathbb{R}_{\{0,1\}}$ is a function that specifies the binary task association of all actions. In states where tasks can be accomplished, there are task specific actions.
  - α_i^0 is the starting probability distribution for agent i.
- Δ_i: Vector of probabilities for agent i leaving the system at different times. Specifically, Δ_i^t represents the probability of agent i leaving the team at time t and $\sum_t \Delta_i^t = 1$.
- H is the time horizon for the decision problem.

The key distinction in ND-TasC-MDP is the presence of a probability distribution for each agent, Δ_i that represents the non-dedication of each agent. The goal is to compute a joint policy π^* that has the highest expected reward among all joint policies given that agents can leave the team according to probability distribution Δ:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_i V_i(\pi_i, \alpha_i^0) \quad \text{s.t.}$$

$$\sum_{i \in \mathcal{A}g} |\boldsymbol{\delta}_i(\tau)| \leq |\mathcal{C}(\tau)| \qquad \forall \tau \in \Gamma \qquad (1)$$

$$f(\pi_i, \Delta_i, \tau) \leq \boldsymbol{\delta}_i(\tau) \qquad \forall i \in \mathcal{A}g, \forall \tau \in \Gamma \qquad (2)$$

Variables: $\forall s_i, \sigma_i \in S_i; \forall a_i \in A_i; \forall \tau \in \Gamma; \forall i \in \mathcal{A}g; \forall t \in H$

Maximize: $$\sum_{i,t,s_i,a_i} x_i^t(s_i, a_i) \cdot R_i^t(s_i, a_i) \qquad (3)$$

Subject to:

$$\sum_{a_i} x_i^{t+1}(\sigma_i, a_i) = \sum_{s_i, a_i} x_i^t(s_i, a_i) P_i^t(s_i, a_i, \sigma_i), \forall \sigma_i, t, i \quad (4)$$

$$\sum_{a_i} x_i^0(s_i, a_i) = \alpha_i(s_i), \ \forall s_i, i \qquad (5)$$

$$\sum_{t,i} \delta_i^t(\tau) \ \leq \mathcal{C}(\tau), \ \forall \tau \qquad (6)$$

$$\frac{1}{X} \sum_{a_i} \rho_i(a_i, \tau) \sum_{s_i} x_i^t(s_i, a_i) \leq \delta_i^t(\tau), \ \forall \tau, t, i \qquad (7)$$

$$x_i^t(s_i, a_i) \geq 0, \delta_t^i(\tau) \in \{0, 1\} \qquad (8)$$

Table 1: Optimal MILP for Dedicated Team

where π_i is the individual policy of agent i in the joint policy π, $\boldsymbol{\delta}_i$ is the set of tasks allocated to agent i with $\boldsymbol{\delta}_i(\tau)$ indicating the number of tasks of type τ. $V_i(\pi_i, \alpha_i^0)$ is the expected value for the individual policy, π_i on model $\mathbf{M}_i$. The task-based interactions are explicitly modelled in Equation 1, which ensures that number of tasks executed is less than the total number of tasks. The individual task accomplishments are modelled using the constraints (2). The function f is used to compute the number of tasks type τ completed by using a policy π_i for agent i given the non-dedication parameter, Δ.

2.1 Formulation for a Dedicated Team

For the case where no agent leaves the system (represented as Δ_i^H = 1 for all agents), previous work [Dolgov and Durfee, 2006; Agrawal *et al.*, 2016] has provided mixed integer formulations to address different types of resource and task interactions. We provide this mixed integer linear program (MILP) for a dedicated agent team in table 1. Objective maximizes the sum of expected rewards over all agents [Puterman, 1994] where $x_i^t(s_i, a_i)$ represents the occupation measure of agent i for state action pair (s, a). Equation 4 ensures outgoing flow from a state σ_i by taking any action is equal to the incoming flow into that state from other states. For the initial state, Equation 5 ensures the flow at the beginning for an agent is equal to its starting distribution. Equation 6 constrains the number of executed tasks for any given type, τ to be less than the number of tasks of that type. A task can be executed only if there is a positive flow associated with the right state and action. This is enforced in Equation 7.

3 Approaches

We provide both reactive (online) and proactive (offline) approaches to facilitate coordination among the remaining agents over the tasks left undone by the agents leaving the team. In addition, we also provide heuristics that benchmark the performance of our approaches.

3.1 Benchmarking Heuristics

We first outline two benchmarking heuristics that will be employed to benchmark the performance of our approaches introduced in later sections.

Ignore the leaving agent, ILA: In this heuristic, we solve the MILP of Table 1 and obtain solution assuming agents will not leave the system. When some agents leave the system, other agents ignore their departure and execute the policies computed by solving the MILP. This provides a good lower bound on solution quality that has to be achieved by any new proposed approach.

Online Revamp, O-Rev: In this heuristic, the agents execute their policies obtained from solving the MILP of Table 1 until one or more agents leave the system. At the decision epoch t where at least one of the agents leaves, the problem is solved again for the remaining agents and time steps. The information of leaving agents and the starting probability distribution over states for agents at the leaving time is input to the MILP. The new policy obtained from re-solving is executed by the agents until there is a change in the system (i.e. some agent leaves the system). Even though this approach is not feasible for online decision making (due to the computational complexity of solving the MILP), this revamp approach provides a good upper bound on the desired performance for our proposed approaches.

3.2 Proactive Expected Flow Optimization

Expected flow optimization (EFO) is a proactive approach that given the probability distribution for agents leaving the system, i.e., Δ, we update the formulation of Table 1. Specifically, we replace the actual flow with expected flow given "probability of staying back" in the flow preservation constraint of Equation 4. That is to say,

$$\sum_{a_i} x_i^{t+1}(\sigma_i, a_i) = \sum_{s_i}\sum_{a_i} x_i^t(s_i, a_i) \cdot P_i^t(s_i, a_i, \sigma_i) \cdot (1 - \Delta_i^t)$$

All other constraints in the optimization problem of Table 1 remain exactly the same. As will be noted in our experiments, such an approach though easy to implement performs poorly in comparison with other approaches.

3.3 Reactive Assignment of Tasks

We now describe *Reactive Assignment of Tasks* (ReacT) that performs reactive updates to the current solution as agents leave the system. Initially, we start with the joint policy obtained by solving the MILP of Table 1. When one or more agents leave the team, the tasks of leaving agents must be assigned to the remaining agents. Each remaining agent evaluates the value of changing the current allocation (i.e., taking on some of the newly available tasks and discarding some of the currently assigned tasks) given the newly available tasks. The agent which obtains the highest value by changing its allocation is first assigned tasks from the newly available list and tasks discarded by that agent are added to the newly available task list. This process is repeated with the remaining agents who evaluate the value of changing their current allocation until the new task list is exhausted or all agents have changed their allocation once. Evaluating value of taking on additional tasks at the cost of discarding some of the current tasks is performed by solving an MDP over the remaining horizon. The starting distribution of the agents are updated to consider the distribution of states at the current time step. Algorithm 1 provides the pseudocode for ReacT where inputs

Algorithm 1 REACTIVE TASK ASSIGNMENT($F, \tilde{\boldsymbol{\delta}}, t$)

```
1: repeat
2:     $\langle i^*, V_{i^*}\rangle \leftarrow \max_{i \in \mathcal{A}g \setminus F} V_i(\pi_i^*, \alpha_i^t, \delta_i \cup \tilde{\boldsymbol{\delta}})$
3:     $\boldsymbol{\delta}_{i^*}^+ \leftarrow$ GETNEWLYALLOCATEDTASKS$(\pi_{i^*}, \delta_i)$
4:     $\boldsymbol{\delta}_{i^*}^- \leftarrow$ GETDISCARDEDTASKS$(\pi_{i^*}, \delta_i)$
5:     $\tilde{\boldsymbol{\delta}} \leftarrow (\tilde{\boldsymbol{\delta}} \setminus \boldsymbol{\delta}_{i^*}^+) \cup \boldsymbol{\delta}_{i^*}^-$
6:     $F \leftarrow F \cup \{i^*\}$
7: until $\tilde{\boldsymbol{\delta}} = \emptyset$ OR $V_{i^*} = 0$
```

to the algorithm are the set of agents leaving the system, F, the newly available tasks, $\tilde{\boldsymbol{\delta}}$ and the current time step, t. The algorithm finds the agent with largest expected reward (line 2) where each of the remaining agents are allowed to consider all tasks from the list $\tilde{\boldsymbol{\delta}} \cup \delta_i$. The newly available tasks assigned to the agent with highest reward are determined in line 3 and the discarded tasks are determined in line 4. The list of newly available tasks $\tilde{\boldsymbol{\delta}}$ is then updated to remove the assigned tasks and add the tasks discarded by agent i^* on line 5. F is updated to include i^*. This process is continued until either of the termination conditions are met.

3.4 Proactive Planning through SAA+LR

We now describe a sample average approximation based Lagrangian relaxation (SAA+LR) approach that computes a policy for each of the agents for multiple scenarios of agent availability over the entire time horizon. Since it is impossible to consider all the samples of agent availability on larger problems, this is primarily an approximate approach that optimizes the expected value given the probability distributions of agents leaving.

A sample of agent availability is generated by sampling from a biased coin with probability p_i independently for every agent i. At every time step t, the coin is tossed to decide if the agent i either leaves or stays in the team depending on the value of associated probability in Δ_i. Therefore, in every sample of agent availability ξ^k, we know the availability horizon $\xi^k(i)$ of every agent i. The MILP in table 2 (referred to as OPT-ND-TasC) provides an optimal offline solution for the optimization problem of a non dedicated team over K samples with a different solution (allocation of tasks and policy) for each sample.

For transition function, we have $P_i^{t,k}$ instead of just P_i. Similarly, for reward, $R_i^{t,k}$ instead of R_i. They are defined as:

$$R_i^{t,k}(s_i, a_i) = \begin{cases} R_i(s_i, a_i) & \textbf{if } \xi^k(i) > t \\ 0 & \textbf{otherwise} \end{cases}$$

$$P_i^{t,k}(s_i, a_i, \sigma_i) = \begin{cases} P_i(s_i, a_i, \sigma_i) & \textbf{if } \xi^k(i) > t - 1 \\ 0 & \textbf{otherwise} \end{cases}$$

Intuitively, these indicate that once the agent leaves the system (at $\xi^k(i)$) reward and transitions are set to 0. To ensure conciseness in expressions, we will use the following short form for the objective employed in Table 2:

$$\sum_{k=1}^{K} f^k(x^k) = \sum_k \sum_{i, s_i, a_i, t} x_i^{t,k}(s_i, a_i) \cdot R_i^{t,k}(s_i, a_i)$$

Variables: $\forall s_i, \sigma_i \in S_i; \forall a_i \in A_i; \forall i \in \mathcal{A}g; \forall t \in \xi^k(i)$

$$\text{Minimize: } -\frac{1}{K}\sum_k \sum_{i,s_i,a_i,t} x_i^{t,k}(s_i,a_i)\cdot R_i^{t,k}(s_i,a_i) \quad (9)$$

Subject to:

$$\sum_{a_i} x_i^{t+1,k}(\sigma_i,a_i) = \sum_{s_i}\sum_{a_i} x_i^{t,k}(s_i,a_i)\cdot P_i^{t,k}(s_i,a_i,\sigma_i), \forall \sigma_i,t,i,k \quad (10)$$

$$\sum_{a_i} x_i^{0,k}(s_i,a_i) = \alpha_i(s_i),\ \forall s_i,i,k \quad (11)$$

$$\sum_{t,i} \delta_i^{t,k}(\tau) \le \mathcal{C}(\tau),\ \forall \tau,k \quad (12)$$

$$\frac{1}{X}\sum_{a_i}\rho_i(a_i,\tau)\sum_{s_i} x_i^{t,k}(s_i,a_i) \le \delta_i^{t,k}(\tau),\ \forall \tau,t,i,k \quad (13)$$

$$x_i^{t,k}(s_i,a_i) \ge 0, \delta_i^{t,k}(\tau) \in \{0,1\} \quad (14)$$

Table 2: Optimal MILP for Non Dedicated Team

The MILP for non dedicated team (OPT-ND-TasC) is separable over the number of samples. However, it can provide a usable solution only if the agent availability sample is known beforehand. But due to the dynamic nature of the problem, agent availability information is not available offline. Therefore, we modify the optimization problem in table 2 to provide an offline allocation of tasks that works well across all K samples using global task allocation variables, d :

$$\delta_i^{t,k}(\tau) = d_i^t(\tau), \quad \forall \tau,t,i,k \quad (15)$$

That is to say, task allocations for all samples should agree on the same solution. This constraint links the optimization of allocations across all samples. Lagrangian dual for the optimization problem of Table 2 while considering the common allocation constraint of Equation 15 is given by:

$$\mathcal{L}(\mathbf{x},\boldsymbol{\delta},\mathbf{d}) = \frac{-1}{K}\sum_{k=1}^{K} f^k(x^k) + \sum_{\tau,t,i,k}\lambda_i^{t,k}(\tau)\left(d_i^t(\tau) - \delta_i^{t,k}(\tau)\right)$$
$$= \frac{-1}{K}\sum_{k=1}^{K}\left(f^k(x^k) + \sum_{\tau,t,i}\lambda_i^{t,k}(\tau)\cdot\delta_i^{t,k}(\tau)\right) + \sum_{\tau,t,i,k}\lambda_i^{t,k}(\tau)\cdot d_i^t(\tau) \quad (16)$$

where $\boldsymbol{\lambda}$ is the dual vector associated with constraints in Equation 15. A solution with respect to a given vector $\boldsymbol{\lambda}$ is given by $G(\boldsymbol{\lambda}) = \min_{\mathbf{x},\boldsymbol{\delta}} \mathcal{L}(\mathbf{x},\boldsymbol{\delta},\mathbf{d})$. The variable $d_i^t(\tau)$ is unconstrained, which may lead to an unbounded dual. Therefore, to avoid unboundedness, the price variables must satisfy the following constraints:

$$\Lambda_i^t(\tau) = \{\lambda_i^{t,k}(\tau) | \sum_k \lambda_i^{t,k}(\tau) = 0\} \quad (17)$$
$$\lambda_i^{t,k}(\tau) \in \Lambda_i^t(\tau),\ \forall \tau,t,i$$

The above condition further simplifies the dual $G(\boldsymbol{\lambda})$, as the last term in Equation 16 vanishes leading to the below dual which is separable over K samples:

$$G(\boldsymbol{\lambda}) = \sum_k \min_{\mathbf{x},\boldsymbol{\delta}}\left(f^k(x^k) + \sum_{\tau,t,i}\lambda_i^{t,k}(\tau)\cdot\delta_i^{t,k}(\tau)\right) \quad (18)$$

Maximizing the Dual Function

We now address the master problem of maximizing the Lagrangian lower bound over the price variables $\boldsymbol{\lambda}$, which can be solved by using projected sub-gradient ascent[Bertsekas, 1999]. The gradient w.r.t. a variable $\lambda_i^k(\tau)$ is given by $\nabla G(\lambda_i^{t,k}(\tau)) = \delta_i^{t,k}(\tau)$ where $\delta_i^{t,k}(\tau)$ denotes the solution of inner minimization problem of Eq.(18) for sample k which is used to update the price variables as follows:

$$\lambda_{i,n^*}^{t,k}(\tau) = \lambda_{i,n}^{t,k}(\tau) + \gamma_{n^*}\left[\delta_{i,n}^{t,k}(\tau) - \frac{\sum_{k'=1}^{K}\delta_{i,n}^{t,k'}(\tau)}{K}\right], \forall \tau,t,i,k \quad (19)$$

Here, n and n^* represent the current and next iteration, respectively. The second term is the projection into the feasible that ensures that $\sum_k \lambda_i^{t,k}(\tau) = 0$ as indicated in Eq.(17).

Extraction of Feasible Primal Solution

The dual solution obtained by solving the individual samples may be inconsistent due to violation of the common allocation constraint of Eq.(15) among different copies of the task allocation variable δ, resulting in $\delta_i^{t,k}(\tau) \neq \delta_i^{t,k'}(\tau)$ for any two samples k and k'. To obtain a consistent task allocation to agents over K samples, we find the best-agent i^* for every task $\tau \in \Gamma$. The best-agent is nominated by choosing the agent with highest number of assignments of task τ over all samples. We obtain the best-agent for all tasks and formulate a reduced version of the OPT-ND-TasC MILP (table 2) by using the unique task allocation obtained over all samples. This reduced MILP is easy to solve and provides a consistent policy assignment for every agent of the non dedicated team irrespective of the sample. The solution obtained is the primal solution $Primal^n$ which is employed in the updation of step parameter $\gamma_{n^*} = \frac{Primal^n - Dual^n}{||\nabla g^k||^2}$ in Eq.(19) where $Dual^n$ is the dual value for iteration n.

3.5 Two Stage MILP for Non-Dedicated Team

The SAA+LR computes one allocation of tasks for the entire duration. In this section, we extend the underlying MILP formulation of SAA+LR to compute an initial allocation and then change the allocation once before the time horizon based on the current state of the system (agents available). Since, we consider the state before changing the allocation, this formulation improves team utility when compared to a single stage allocation. It should be noted that the dual decomposition approach described in the previous section is also directly applicable for this extended formulation. Due to the similarity, we do not describe the dual decomposition approach here and furthermore, we refer to two-stage MILP and Lagrangian relaxation of two-stage MILP synonymously.

State of the system (of relevance to task allocation) is the set of agents leaving the system. In this extended MILP, we compute an initial allocation and then at an observation time, t' compute a new allocation based on the observation, $o \in O$ of the state of the system. The samples of agent availability $< \xi^1, \xi^2, ..., \xi^k >$ provide the set of possible observations at t' that marks the beginning of second stage. An observation belief $< b^1, b^2, ..., b^k >$ is maintained over the sample set ξ for every observation o at the observation time t'. It must be

$$Variables: x_{i,n}^{t,k} \geq 0,\ \delta_{i,n}^{t,k} \in \{0,1\},\ d_{i,n}^{t} \in \{0,1\}$$
$$y_{i,n}^{t} \geq 0, n \in \{1,2\}$$

$$\max \frac{1}{K} \sum_{k,i,s_i,a_i} \Big(\sum_{t \leq t'} x_{i,1}^{t,k}(s_i,a_i) \cdot R_{i,1}^{t,k}(s_i,a_i) + + \sum_{o} \sum_{t > t'} x_{i,2}^{t,k}(s_i,a_i,o) \cdot R_{i,2}^{t,k}(s_i,a_i) \Big) \tag{20}$$

$$\sum_{i} \left(\sum_{t \leq t'} \delta_{i,1}^{t,k}(\tau) + \sum_{o} \sum_{t > t'} \delta_{i,2}^{t,k}(\tau,o) \right) \leq \mathcal{C}(\tau), \forall \tau, k \tag{21}$$

$\forall \mathbf{t} \leq \mathbf{t}', \mathbf{i}, \mathbf{k}:$

$$\sum_{a_i} x_{i,1}^{0,k}(s_i,a_i) = \alpha_{i,1}(s_i) \tag{22}$$

$$\sum_{a_i} x_{i,1}^{t+1,k}(\sigma_i,a_i) = \sum_{s_i,a_i} x_{i,1}^{t,k}(s_i,a_i) P_i^{t,k}(s_i,a_i,\sigma_i), \forall \sigma_i \tag{23}$$

$$\frac{1}{X} \sum_{a_i} \rho_i(a_i,\tau) \sum_{s_i} x_{i,1}^{t,k}(s_i,a_i) \leq \delta_{i,1}^{t,k}(\tau), \forall \tau \tag{24}$$

$$\sum_{a} x_{i,1}^{t,k}(s_i,a_i) = \sum_{a} y_{i,1}^{t}(s_i,a_i), \forall s_i \tag{25}$$

$$\delta_{i,1}^{t,k}(\tau) = d_{i,1}^{t}(\tau), \forall \tau \tag{26}$$

$\forall \mathbf{s_i}, \mathbf{i}, \mathbf{o}, \mathbf{k}:$

$$\sum_{a_i} x_{i,2}^{t'+1,k}(s_i,a_i,o) = \sum_{s_i,a_i} x_{i,1}^{t',k}(s_i,a_i) P_i^{t,k}(s_i,a_i,\sigma_i) \tag{27}$$

$\forall \mathbf{t} > \mathbf{t}', \mathbf{i}, \mathbf{o}, \mathbf{k}:$

$$\sum_{a_i} x_{i,2}^{t+1,k}(\sigma_i,a_i,o) = \sum_{s_i,a_i} x_{i,2}^{t,k}(s_i,a_i,o) P_i^{t,k}(s_i,a_i,\sigma_i), \forall \sigma_i \tag{28}$$

$$\frac{1}{X} \sum_{a_i} \rho_i(a_i,\tau) \sum_{s_i} x_{i,2}^{t,k}(s_i,a_i,o) \leq \delta_{i,2}^{t,k}(\tau,o), \forall \tau \tag{29}$$

$$\sum_{a} x_{i,2}^{t,k}(s_i,a_i,o) = \sum_{a} y_{i,2}^{t}(s_i,a_i,o), \forall s_i \tag{30}$$

$$\delta_{i,2}^{t,k}(\tau,o) = d_{i,2}^{t}(\tau,o), \forall \tau \tag{31}$$

Table 3: Two-Stage MILP for Non Dedicated Team

noted that any sample $\xi^k, k \in K$ will have a non-zero belief for at most one observation and zero for the remaining set of observations. This reduces the number of samples to be solved at observation level, thereby, reducing the complexity of the problem at second stage to solving maximum K samples instead of $K \times |O|$.

The objective is to maximize the total value over all samples ξ where the first and second term denote the objectives for the two stages, respectively. The variables $x_{i,1}^{t,k}(s_i,a_i)$ and $x_{i,2}^{t,k}(s_i,a_i,o)$ denote the visitation frequency for agent i at time t for sample k with observation o in stages 1 and 2 respectively. $d_{i,n}^{t}$ and $y_{i,n}^{t}$ represent the common task allocation and policy assignment variables for the two stages $n \in \{1,2\}$ over all samples.

The constraints 22 to 26 are the constraints for stage 1 where equations 22 and 23 are the flow constraints and equation 24 is the task assignment/execution constraint. The equations 25 and 26 are required to provide a unique policy and task allocation over all samples in the first stage. Similarly, constraints 27 to 31 are the second stage constraints. For every observation $o \in O$, equations 27 and 28 are the flow constraints. The initial flow for every observation in the second stage at $t' + 1$ is the outflow from the last time-step t' of the first stage as shown in 27. Equation 29 provides the task assignment constraint. Equations 30 and 31 provide a unique policy and unique task allocation over all samples for an observation. For every sample, equation 21 is the task assignment constraint that ensures that a task can be done only in either of the stages.

4 Experiments

We evaluate[1] the performance of our reactive and proactive approaches. While there are no benchmark problems to study performance of non-dedicated teams, we rely on the benchmark problems available for multi-agent coordination in uncertain domains and augment them with probability distributions that represent the non-dedicative nature of agents.

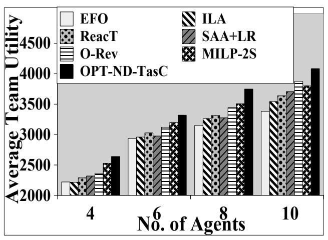
4.1 Experimental Setup

We employ a generic setting that can be easily adapted to the domains described in introduction. Specifically, we employ the Urban Consolidation Center problems that deal with allocation of tasks and have to consider transitional uncertainty [Handoko *et al.*, 2014]. We evaluate the performance of all the approaches on various metrics: (a) solution quality; (b) runtime; (c) quality of bounds provided by dual solution; and (d) percentage of optimality with increasing training samples and varying observation time.

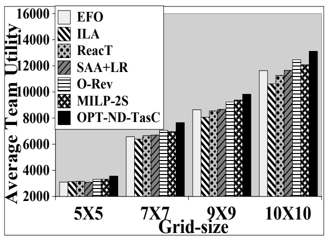
Experimental results are averaged over 15 randomly generated grid maps that randomly place the delivery locations (tasks) and walls. Grids are described using a single parameter m that represents number of rows and columns (i.e., m x m grid). The actions are classified into movement actions (stay, left, right, up and down) and task actions. Rewards are generated only for task actions using a pseudo-random function dependent on the task location and task id. We generate 1500 samples of agent availability and divide it into training and testing sets of 1000 and 500 samples, respectively. To obtain a fair comparison over all online and offline approaches, we compare the solution quality and runtime on the same test set for the following approaches.

(**1**) Ignore the leaving agent, ILA - Section 3.1.
(**2**) Online revamp, O-Rev - Section 3.1.
(**3**) Expected Flow Optimization, EFO - Section 3.2.
(**4**) Reactive assignment of tasks (ReacT) - Section 3.3.
(**5**) Sample Average Approximation based Lagrangian Relaxation (SAA + LR) - Section 3.4. Due to repetition of samples, we assign frequency-specific weights to distinct training samples and select the 10 best samples for computing the joint policy with SAA+LR. The number of training samples is fixed to 10 best samples unless mentioned specifically.
(**6**) OPT-ND-TasC - This corresponds to solving all the chosen test samples offline. This is not really an approach that can be employed, but serves as a benchmark on the best possible performance achievable.

[1]All our optimization problems are run on CPLEX v12.5

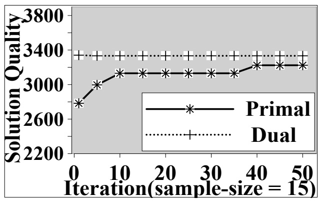


(a) $m = 5, H = 10, C(\tau) = 20$

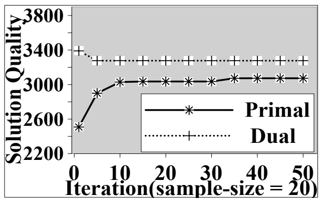


(b) $|\mathcal{A}g| = 6, H = 10, C(\tau) = 20$

Figure 1: Quality comparison w.r.t. (a) Agents and (b) Grid-size



(a) $m = 5, |\mathcal{A}g| = 8, C(\tau) = 20$



(b) $m = 5, |\mathcal{A}g| = 8, C(\tau) = 20$

Figure 2: Quality Comparison with increasing samples

(**7**) Two Stage MILP for Non-Dedicated Team (MILP-2S) - Section 3.5. The training set, obtained similar to SAA+LR, is used to find the set of possible observations O beforehand for the second stage in MILP-2S. For any observation outside O in the testing set, we employ ReacT to obtain the solution of second stage for the samples. The observation time is fixed to $t' = 6$ for a horizon $H = 10$ unless specifically mentioned.

4.2 Results

We compare the different approaches with respect to average team utility in Figure 1(a) as the number of agents $|\mathcal{A}g|$ is increased. Specifically, we consider grids with $m = 5$, tasks $C(\tau) = 20$ and horizon $H = 10$. Similarly, in Figure 1(b), we compare for different grid-size m for a fixed number of agents $|\mathcal{A}g| = 6$, tasks $C(\tau) = 20$ and horizon $H = 10$. The key observations are summarized as following:

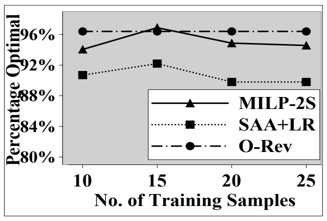
(1) EFO provides low team utility solutions. In the best case, EFO performs similar to ReacT but in the worst case, the performance is even lower than ILA.

(2) Since ILA ignores the leaving agents which impacts the scope of improvement in the rewards, ReacT performs better. Moreover, ReacT allocates high valued tasks of leaving agents to improve its utility.

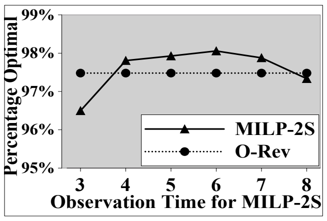
(3) SAA+LR typically provides better performance than ReacT, but in some cases (e.g., 6 agents in Figure 1(a)), ReacT performs slightly better. O-Rev provides better utility than ReacT, ILA and SAA+LR but requires a lot of ”online” cycles to solve the MILP at every stage of leaving agents.

(4) Finally, we observe that OPT-ND-TasC provides the best utility but it requires the knowledge of samples before-hand. MILP-2S provides comparable or better performance than O-Rev. O-Rev does not provide the optimal solution, because it does not reason about future scenarios of agents leaving while planning at a time step. Overall, amongst all the relevant and reasonable approaches, MILP-2S provides the best performance with respect to solution quality.

With respect to runtime, there are offline and online runtimes. Since EFO, ILA, SAA+LR and MILP-2S are offline approaches, online runtime is minimal (milliseconds). ReacT is an online approach and takes less than 30 seconds to generate a new allocation. O-Rev is an online approach but takes a long time (10-30 minutes) to generate results and hence is not applicable. For all offline approaches, we provide a maximum of three hours for training.



(a) $m = 5, |\mathcal{A}g| = 8, C(\tau) = 20$



(b) $m = 5, |\mathcal{A}g| = 6, C(\tau) = 20$

Figure 3: Optimality w.r.t. (a) Samples (b) Observation Time

Figure 2 shows the convergence graph for the training samples of SAA+LR where the primal and dual evolve with increasing iterations and the number of training samples. A key observation is that SAA+LR provides near optimal solution very early and converges quickly even with increasing samples. The solution quality of the primal (calculated as $Primal * 100/Dual$) is atleast 90% of the optimal.

Figure 3 shows the comparison of optimality (calculated as U(approach)$*$100/U(OPT-ND-TasC) where U(approach) represents the utility obtained using specified approach) on the test set with different training samples and observation times in the training phase. Figure 3(a) shows the comparison of MILP-2S, SAA+LR and O-Rev with increasing training samples for MILP-2S and SAA+LR. Notice that O-Rev requires no training being online and it's performance remains unchanged. We observe that the performance of MILP-2S and SAA+LR improves to a certain extent with increasing training samples after which the performance becomes approximately constant. This is because for higher sample sizes (ordered in decreasing frequency weights), few samples with lower weights would be given preference in training which may not even occur in the test set at all, thereby impacting the performance. Similarly, in Figure 3(b), we vary the observation time for MILP-2S to obtain a two-stage offline policy for the different observation times and simulate the policies on the same test set. We observe that MILP-2S performs approximately as good as O-Rev (benchmark heuristic not dependent on observation time) with different observation times, but performs extremely well towards the centre.

5 Conclusion

In this work, we focussed on the problem of task allocation for a non-dedicated agent team. We provided both proactive (SAA+LR, MILP-2S) and reactive (ReacT) approaches for handling agents leaving the team in an effective manner. Our extensive experiments on benchmark problems demonstrate that the offline approach, MILP-2S provides the best solutions that are on par with benchmarks that compute upper bound on performance. Furthermore, our two stage approach (MILP-2S) can be easily extended to multiple stages to handle both leaving and returning agents simultaneously given the distribution of leaving/returning of agents.

References

[Agrawal *et al.*, 2016] Pritee Agrawal, Pradeep Varakantham, and William Yeoh. Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI*, pages 10–16, 2016.

[Barrett *et al.*, 2017] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, 242:132–171, 2017.

[Bertsekas, 1999] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[Brown *et al.*, 2014] Matthew Brown, Sandhya Saisubramanian, Pradeep Varakantham, and Milind Tambe. STREETS: game-theoretic traffic patrolling with exploration and exploitation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2966–2971, 2014.

[Cohen and Levesque, 1991] Philip R Cohen and Hector J Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[Dantzig and Ramser, 1959] B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[Dolgov and Durfee, 2006] Dmitri Dolgov and Edmund Durfee. Resource allocation among agents with MDP-induced preferences. *Journal of Artificial Intelligence Research*, 27:505–549, 2006.

[Grosz and Kraus, 1996] Barbara J Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[Handoko *et al.*, 2014] Stephanus Daniel Handoko, Duc Thien Nguyen, and Hoong Chuin Lau. An auction mechanism for the last-mile deliveries via urban consolidation centre. In *Proceedings of the International Conference on Automation Science and Engineering (CASE)*, pages 607–612, 2014.

[Hazard *et al.*, 2006] Christopher Hazard, Peter Wurman, and Raffaello D'Andrea. Alphabet soup: A testbed for studying resource allocation in multi-vehicle systems. In *Proceedings of the AAAI Workshop on Auction Mechanisms for Robot Coordination*, pages 23–30, 2006.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[Shieh *et al.*, 2014] Eric Anyung Shieh, Albert Xin Jiang, Amulya Yadav, Pradeep Varakantham, and Milind Tambe. Unleashing dec-mdps in security games: Enabling effective defender teamwork. In *ECAI 2014 - 21st European Conference on Artificial Intelligence*, pages 819–824, 2014.

[Tambe, 1997] Milind Tambe. Towards flexible teamwork. *Journal of artificial intelligence research*, 7:83–124, 1997.

[Varakantham *et al.*, 2009] Pradeep Varakantham, Jun-Young Kwak, Matthew Taylor, Janusz Marecki, Paul Scerri, and Milind Tambe. Exploiting coordination locales in distributed POMDPs via social model shaping. In *Proceedings of the International Conference on Planning and Scheduling (ICAPS)*, pages 313–320, 2009.

[Varakantham *et al.*, 2013] Pradeep Varakantham, Hoong Chuin Lau, and Zhi Yuan. Scalable randomized patrolling for securing rapid transit networks. In *Proceedings of the Twenty-Fifth Innovative Applications of Artificial Intelligence Conference, IAAI*, 2013.

[Varakantham *et al.*, 2014] Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Decentralized stochastic planning with anonymity in interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 2505–2512, 2014.

[Velagapudi *et al.*, 2011] Prasanna Velagapudi, Pradeep Varakantham, Paul Scerri, and Katia Sycara. Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 955–962, 2011.

[Wurman *et al.*, 2007] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1752–1760, 2007.