Scalable Greedy Algorithms For Task/Resource Constrained Multi-Agent Stochastic Planning

Pritee Agrawal
Singapore Management University
Singapore 188065
priteea.2013@phdis.smu.edu.sg

Pradeep Varakantham
Singapore Management Univ.
Singapore 188065
pradeepv@smu.edu.sg

William Yeoh
New Mexico State Univ.
Las Cruces, NM 88003 USA
wyeoh@cs.nmsu.edu

Abstract

Synergistic interactions between task/resource allocation and stochastic planning exist in many environments such as transportation and logistics, UAV task assignment and disaster rescue. Existing research in exploiting these synergistic interactions between the two problems have either only considered domains where tasks/resources are completely independent of each other or have focussed on approaches with limited scalability. In this paper, we address these two limitations by introducing a generic model for task/resource constrained multi-agent stochastic planning, referred to as TasC-MDPs. We provide two scalable greedy algorithms, one of which provides posterior quality guarantees. Finally, we illustrate the high scalability and solution performance of our approaches in comparison with existing work on two benchmark problems from the literature.

1 Introduction

In delivery of services or goods [Dantzig and Ramser, 1959; Dolgov and Durfee, 2006], tasks have to be allocated to individual vehicles based on uncertain travel times to delivery locations. Also, in disaster rescue scenarios [Velagapudi *et al.*, 2011; Varakantham *et al.*, 2009], victims have to be allocated to robots while considering the uncertainty in traveling through disaster prone areas. Furthermore, in large warehouses [Hazard *et al.*, 2006] 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). These domains have the following common characteristics: (a) Multiple 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 traveling on roads (due to traffic) or uncertain demand (online orders) or physical constraints (e.g., robots); and (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.

We can view these domains as having a synergistic combination of two interdependent challenges, namely task/resource allocation and planning under transition uncertainty for multiple agents. While the task allocation determines the plans for the individual agents, the feedback from the plans can help improve the allocation (as not all allocated tasks/resources can be executed/utilized due to uncertainty). We refer to these problems as *Task/Resource Constrained Markov Decision Problems* (TasC-MDPs) and we are specifically focussed on cooperative TasC-MDPs.

Researchers have modeled task/resource constrained stochastic planning problems in a variety of ways including cooperative auctions [Koenig *et al.*, 2010], resource-constrained MDPs [Dolgov and Durfee, 2006; Guestrin and Gordon, 2002], decentralized MDPs (Dec-MDPs) [Bernstein *et al.*, 2002], and Dec-MDP variants that exploit the sparsity of agent interactions [Nair *et al.*, 2005; Kumar and Zilberstein, 2009; Velagapudi *et al.*, 2011; Varakantham *et al.*, 2014; Witwicki and Durfee, 2011]. While Dec-MDPs are a rich model, they do not represent tasks explicitly and, because of this, all agents would be represented as being dependent on each other. This significantly impacts the scalability of solution approaches. The most relevant models for this work are the resource-parametrized MDPs [Dolgov and Durfee, 2006; Wu and Durfee, 2010] and weakly-coupled MDPs [Guestrin and Gordon, 2002]. While these works are relevant, they are not scalable (i.e., to tens/hundreds of agents and hundreds of tasks/resources) and do not consider dependencies between tasks/resources. While there are many other relevant papers, the following aspects differentiate our work from existing work in this thread: (i) We consider problems where there exist dependencies (temporal and allocation constraining) between tasks/resources. (ii) Our unique mechanism of employing a greedy heuristic method in the context of dual decomposition for improved scalability and quality bounds.

Another thread of existing research considers multi-agent deterministic routing [Christofides *et al.*, 1981; Campbell *et al.*, 1998], where the individual routing problems are dependent on the tasks/resources allocated to them. Decomposition techniques such as *Column Generation* (CG) [Desrochers *et al.*, 1992], *Lagrangian Relaxation* (LR) [Kohl and Madsen, 1997; Chien *et al.*, 1989] and Benders Decomposition [Federgruen and Zipkin, 1984] have been used to improve scalability. The key difference and enhancement over this line of existing work is due to consideration of transition uncertainty. Unlike deterministic routing problems, in an MDP, the number of resources consumed or tasks completed is not easy to compute due to the transition uncertainty.

We make the following key contributions to solve TasC-MDPs in this paper. First, we provide a generic model for TasC-MDPs with an ability to handle task dependencies, specifically temporal task dependencies and task allocation dependencies for agents. Second, we provide a greedy approach referred to as GAPS to greedily allocate tasks/resources to agents based on their marginal value contribution. Third, we provide a unique method of employing GAPS in the context of dual decomposition to improve scalability and provide quality bounds. Finally, on two benchmark problems from the literature, we show that our approach based on dual decomposition provides a good trade-off between the GAPS approach and optimal MILP.

2 Task/Resource Constrained MDP

A *Task/Resource Constrained Markov Decision Process* (TasC-MDP) is defined using the tuple $\left\langle \mathcal{A}g, \Gamma, \mathcal{C}, D, Z, \langle \mathbf{M}_i \rangle_{i \in \mathcal{A}g} \right\rangle$.

- $\mathcal{A}g$ is the set of agents.
- Γ is the set of the different types of tasks/resources.
- $\mathcal{C} = \bigcup_{\tau \in \Gamma} \mathcal{C}(\tau)$ corresponds to the set of all tasks/resources, where $\mathcal{C}(\tau)$ is the set of tasks/resources of type τ; $\mathcal{P}(\mathcal{C})$ is the power set of $\mathcal{C}$, that is, it is the set of all possible task/resource allocations; $|\mathcal{C}(\tau)|$ is the global capacity bound for tasks/resources of type τ.
- $D = \{\tau_i \prec \tau_j, \tau_k \parallel \tau_l, \ldots\}$ is the set of dependencies for tasks/resources. While there are potentially other types of dependencies, we restrict ourselves to temporal and allocation constraining dependencies in this paper. A temporal dependency is represented as $\tau_i \prec \tau_j$, indicating that a predecessor-task τ_i should be executed before the successor-task τ_j. An allocation constraining dependency is represented as $\tau_i \parallel \tau_j$, indicating that both tasks τ_i and τ_j should be allocated to same agent.
- Z is the finite set of capacities for an agent, where $z \in Z$ represents a capacity type (e.g., weight, money, etc.).
- $\mathbf{M}_i$ is the MDP model for agent i along with the task/resource associations of actions. It is defined as the tuple $\left\langle S_i, A_i, P_i, R_i, \rho_i, q_i, \hat{q}_i, \alpha_i^0 \right\rangle$.
  - S_i, A_i, P_i, R_i are the sets of states, actions, transition and reward functions, respectively.
  - $\rho_i : A_i \times \Gamma \rightarrow \mathbb{R}_{\{0,1\}}$ is a function that specifies the binary task/resource requirements of all actions.
  - $q_i : \Gamma \times Z \rightarrow \mathbb{R}$ is a function that specifies the capacity costs of tasks/resources (e.g., $q_i(\tau, z)$ defines how much capacity $z \in Z$ is required for task/resource type $\tau \in \Gamma$).
  - $\hat{q}_i : \Gamma \rightarrow \mathbb{R}$ corresponds to the upper bound on the capacities (e.g., $\hat{q}_i(z)$ gives the upper bound on capacity $z \in Z$) for agent i.
  - α_i^0 is the starting probability distribution for agent i.

The goal is to compute a joint policy π^* that has the highest expected reward among all joint policies:

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

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

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

where π_i is the individual policy of agent i in the joint policy π; $\mathbf{r}_i$ is the set of tasks/resources allocated to agent i with $\mathbf{r}_i(\tau)$ indicating the number of tasks/resources of type τ; and $V_i(\pi_i, \alpha_i^0)$ is the expected value for the individual policy π_i on model $\mathbf{M}_i$. The task/resource-based interactions are explicitly modelled in Constraint (1), which ensures that the number of resources used (or tasks executed) is less than the capacity. The individual resource requirements or task completions are modelled using Constraint (2). The function f is used to compute the number of resources required or tasks completed of type τ by using a policy π_i for agent i.

Algorithm 1 GAPS($TasC - MDP$)

```
1: $\tilde{\mathbf{r}} \leftarrow \mathcal{C}$
2: $F \leftarrow \emptyset$
3: repeat
4:     for all $i \in \mathcal{A}g \setminus F$ do
5:         $\pi_i^* \leftarrow \max_{\pi_i} V_i(\pi_i, \alpha_i^0)$ s.t. $f(\pi_i) \leq \tilde{\mathbf{r}}$
6:     $\langle i^*, V_{i^*} \rangle \leftarrow \max_{i \in \mathcal{A}g \setminus F} V_i(\pi_i^*, \alpha_i^0) f(\pi_i) \leq \tilde{\mathbf{r}}$
7:     $\mathbf{r}_{i^*} \leftarrow$ GETCONSUMEDRESOURCE($\pi_{i^*}^*$)
8:     $\tilde{\mathbf{r}} \leftarrow \tilde{\mathbf{r}} \setminus \mathbf{r}_{i^*}$
9:     $F \leftarrow F \cup \{i^*\}$
10: until $\tilde{\mathbf{r}} = \emptyset$ OR $V_{i^*} = 0$
11: return $\pi^* \leftarrow \{\pi_i^*\}_{i \in \mathcal{A}g}$
```

Overall, TasC-MDPs represent task/resource-based interactions between stochastic planning agents, i.e., given the allocation of tasks/resources, the agent planning problems are independent of each other. This is one of the key insights that we exploit in a formal way with our greedy-based dual decomposition approach. It should be noted that since TasC-MDPs generalize the 0/1 knapsack problem, it is trivial to show that optimally solving TasC-MDPs is at least in NP. Finally, as our approach can be applied on both resource and task allocation problems, we will use "resource" and "task" interchangeably in the rest of the paper.

3 GAPS Algorithm

We now introduce the *Greedy Agent-based Prioritized Shaping* (GAPS) algorithm to solve TasC-MDPs. GAPS greedily allocates resources to the agent that yields the highest increase in expected value. Initially, it computes individual best policy for all agents given all the resources. Once the agent with the highest value is identified, excess resources that were not utilized are determined. It fixes the policy for the highest-value agent and repeats the process with the excess resources for the remaining agents until there are no more resources available or the value of adding an agent is 0.

Algorithm 1 shows the pseudocode. The algorithm uses $\tilde{\mathbf{r}}$ to represent the set of unallocated resources and F to represent the set of agents with allocated resources. They are initialized to the set of all resources $\mathcal{C}$ (line 1) and the empty set (line 2), respectively. GAPS then iterates over the set of agents without allocated resources (line 4), and for each agent

i in this set, it solves the individual agent model assuming that the agent is allocated all remaining unallocated resources $\tilde{\mathbf{r}}$ (line 5). Among these agents, GAPS chooses the agent with the largest expected reward (line 6), removes the resources that it consumed from the set of available resources (lines 7-8) and adds that agent to the set of agents with allocated resources (line 9). GAPS repeats this process until there are no more unallocated resources or the unallocated resources are not useful to any agent (lines 3 and 10) and returns the joint policy of all agents before terminating (line 11).

GAPS is an easily parallelizable algorithm. Instead of iterating through the agents and solving the individual MDP models sequentially (lines 4-5), they can be solved in parallel by each agent as they are independent of each other. Once the individual MDP models are solved, they will then need to communicate with each other to identify the agent with the largest expected reward and the resources that it consumed(or tasks completed) (lines 6-8) in each iteration. By employing this model of parallel computation and communication, GAPS can be made even more scalable when there are multiple processors and cheap communication.

In each iteration, GAPS computes policies for all the remaining agents. Thus, the runtime complexity of GAPS is $O(|\mathcal{A}g|^2\times$ Complexity of solving an MDP). However, in many planning problems like disaster rescue,[1] if the number of agent types (sets of homogeneous agents) is k ($\leq \mathcal{A}g$), then the number of policy computations is at most k and hence the runtime complexity is $O(k \cdot |\mathcal{A}g| \times$ Complexity of solving an MDP), which is linear in the number of agents. Thus, we exploit the property of homogeneous agents to improve the scalability of GAPS.

4 Greedy-Based Dual Decomposition

While GAPS is highly efficient, it provides no guarantee on the solution quality. We thus describe an optimization-based approach that provides posteriori guarantees on solution quality. We first provide a *Mixed Integer Linear Program* (MILP) formulation to solve TasC-MDPs that extends the formulation by Dolgov and Durfee [2006]. Table 1 shows the MILP, where variable $x_i^t(s,a)$ denotes the occupation-measure of agent i for state action pair (s,a). The binary decision variable $\delta_i(\tau)$ denotes the allocation of a resource of type τ to agent i. The objective is to maximize the sum of expected rewards over all agents, while ensuring that their policies (individually and jointly) satisfy the following constraints:

- Flow conservation: Constraints (4) and (5) enforce that the total expected number of times state σ_i is exited (LHS of the constraints) equals the expected number of times state σ_i is entered (RHS of the constraints).
- Individual capacity limits: Constraint (6) is a capacity bound constraint for an agent on all capacity types $z \in Z$. The total cost for obtaining all shared resources $\tau \in \Gamma$ must be less than the capacity bound of agent $\hat{q}_i(z)$. For simplification, we use $|Z| = 1$ and $q(\tau, z) = 1$ throughout the paper (i.e., the number of resources obtainable by an agent cannot exceed its capacity).

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

Minimize: $$-\sum_i \sum_t \sum_{s_i} \sum_{a_i} x_i^t(s_i,a_i) \cdot R_i^t(s_i,a_i) \tag{3}$$

Subject to:

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

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

$$\sum_{\tau} \sum_{k} q_i(\tau,z) \cdot \delta_i^t(\tau) \leq \hat{q}_i(z),\ \forall z, \forall i \tag{6}$$

$$\sum_{t,i} \delta_i^t(\tau)\ \leq \mathcal{C}(\tau),\ \forall \tau \tag{7}$$

$$\frac{1}{X} \sum_{a_i} \rho_i(a_i,\tau) \sum_{s_i} x_i^t(s_i,a_i) \leq \delta_i^t(\tau),\ \forall \tau, t, i \tag{8}$$

$$\delta_i^{t+1}(\tau_k) - \sum_{t' \leq t} \delta_i^{t'}(\tau_j) \leq 0,\ \forall (\tau_j \prec \tau_k) \in D, t < H, i \tag{9}$$

$$\sum_t \delta_i^t(\tau_j) = \sum_t \delta_i^t(\tau_k),\ \forall (\tau_j \parallel \tau_k) \in D, t \tag{10}$$

$$\delta_i^t(\tau) \leq M \cdot \rho_i(a_i,\tau) \cdot x_i^t(s_i,a_i),\ \forall \tau \in D, s_i, t, i \tag{11}$$

$$x_i^t(s_i,a_i) \geq 0, \delta_t^i(\tau) \in \{0,1\} \tag{12}$$

Table 1: Optimal MILP

- Global capacity limits: Constraint (7) prevents violation of global capacity limitations for all resource types (i.e., total resources assigned over all agents $i \in \mathcal{A}g$ for a given type τ should not exceed the available resources of that type $C(\tau)$).
- Resource requirements of policy: Constraint (8) computes the resource requirement of each type τ for a policy at each time step t. Intuitively, this constraint ensures that if occupation measure $x_i^t(s_i,a_i)$ for (s_i,a_i) is a positive number and resource τ is required for executing action a_i (i.e., $\rho_i(a_i,\tau) = 1$), then 1 unit of resource type τ is required. Here, X is a normalization constant (calculated offline) representing the maximum flow for an agent: $X \geq \max_{\tau,i} \sum_{a_i} \rho^i(a,\tau) \sum_{s_i} \sum_t x_i^t(s_i,a_i)$.
- Task dependencies: Constraints (9) to (11) represent the resource dependencies. Constraint (9) represents the temporal resource dependencies ($\tau_j \prec \tau_k$) and Constraint (10) represents the allocation constraining dependencies ($\tau_j \parallel \tau_k$) for every agent i. Constraint (11) is helping constraints ensure that any resource τ is executed in state s_i by enforcing the occupation measure $x_i^t(s_i,a_i)$ for (s_i,a_i) to be a positive number and resource τ is required for executing action a_i (i.e., $\rho_i(a_i,\tau) = 1$). M is a large positive number. It should be noted that for independent resources, $D = \emptyset$.

Note that the MILP in Table 1 is an optimal MILP as it provides an exact solution. Unfortunately, given the in-

[1] In disaster rescue, while all the robots have the same model, we can assume only those agents starting from same state and having to rescue a victim from the same cell as homogeneous agents.

crease in the number of binary integer variables and the constraints that contain them, this optimal MILP is not scalable with increasing problem complexity (increasing agents, tasks/resources, states, etc.). Therefore, we propose the use of *Lagrangian dual decomposition* (LDD) [Bertsekas, 1999] along with GAPS (referred as LDD+GAPS) to efficiently solve TasC-MDPs. LDD+GAPS is an iterative method that contains two stages at each iteration. In the first stage, we relax the global capacity constraint (Constraint (7)) to obtain a dual problem that can be solved independently for each agent. Each individual agent solves an unconstrained MDP (with dual variables in its objective) to get the best possible reward value and resource allocation. However, the relaxation can, in many cases, lead to violation of the global capacity constraint. Therefore, in the second stage (discussed in detail in Section 4.2), we use GAPS to extract a feasible primal solution from the dual solution.

4.1 Maximizing the Lagrangian Dual

We relax or *dualize* coupling Constraint (7) for all resource types to yield a significantly easier dual problem. For each resource type, we create dual variables $\lambda_\tau, \forall \tau \in \Gamma$, that represent the price of violating the global capacity constraint. Over the iterations, our approach will try to find the 'right' penalty in order to minimize the number of violations.

The Lagrangian dual $\mathcal{L}(\{\boldsymbol{V}_i\}, \boldsymbol{\lambda})$ corresponding to a relaxation of Constraint (7) is defined as follows:

$$\mathcal{L}(\{\boldsymbol{V}_i\}, \boldsymbol{\lambda}) = \sum_i -V_i(\pi_i, \alpha_i^0) + \sum_\tau \lambda_\tau \left(\sum_{i,t} \delta_i^t(\tau) - \mathcal{C}(\tau) \right) \quad (13)$$

$V_i(\pi_i, \alpha_i^0)$ represents the single agent contribution $\sum_t \sum_s \sum_a x_i^t(s,a) \cdot R_i^t(s,a)$ in the Optimal MILP. On rearranging the above equation, the separable structure of Lagrangian over the agents can be obtained as:

$$\sum_i \min_{x_i, \delta_i} \left[-V_i(\pi_i, \alpha_i^0) + \sum_\tau \lambda_\tau \sum_t \delta_i^t(\tau) \right] - \sum_\tau \lambda_\tau \cdot \mathcal{C}(\tau) \quad (14)$$

For a given λ, we note that the extra term $\sum_\tau \lambda_\tau \cdot \mathcal{C}(\tau)$ is a constant that can be accounted for later in the master problem. Thus, given a $\boldsymbol{\lambda}$, the above Lagrangian dual $\mathcal{L}(\{\boldsymbol{V}_i\}, \boldsymbol{\lambda})$ can be minimized for each agent separately as the objective and constraints are clearly delineated.

We now address the master problem of maximizing the Lagrangian lower bound over the price variables λ, which can be solved by using projected sub-gradient ascent [Bertsekas, 1999]. The sub-gradient w.r.t. a variable λ_τ is essentially the quantity in parentheses in Eq. (13), which is used to update the price variables for the next iteration $n+1$ as follows:

$$\lambda_\tau^{n+1} = \lambda_\tau^n + \gamma^{n+1} \left[\sum_{i,t} \delta_i^{t,n}(\tau) - \mathcal{C}(\tau) \right], \forall \tau \in \Gamma \quad (15)$$

where $\delta_i^{t,n}(\tau)$ represents the solution values obtained by solving the slave planning problem for agent i at iteration n:

$$\min_{\mathbf{x}} \; - \sum_t \sum_s \sum_a x_i^t(s,a) \cdot R_i^t(s,a) + \sum_\tau \lambda_\tau \sum_t \delta_i^t(\tau)$$
$$\text{s.t. Constraints } (4) - (6) \wedge (8) - (12)$$

and γ^{n+1} is the step parameter for iteration $n+1$ that is set based on the values computed in iteration n as below:

$$\gamma^{n+1} = \frac{Primal^n - Dual^n}{||\nabla q^n||^2} \quad (16)$$

where the dual value $Dual^n$ can be easily obtained from Eq. (14) and the primal value $Primal^n$ is obtained as discussed in the next section. ∇q^n denotes the total sub-gradient of the dual function.

4.2 Extraction of Feasible Primal Solution

The dual solution obtained in the first stage may not be a feasible primal solution. Therefore, we use the GAPS algorithm to obtain a feasible primal solution from the dual solution. This primal solution is crucial to set the step parameter γ that helps improve the convergence rate of LDD+GAPS, and imparts the desirable *anytime* property to our approach.

The input to the GAPS algorithm is the resource requirement $\{\delta_t^i(\tau), \forall\tau\}$ and the dual values $\{V_i(\pi_i, \alpha_i^0)\}$ for every agent $i \in \mathcal{A}g$. Among these agents, GAPS chooses the agent with highest expected reward (line 6), removes the resources used by the agent, i.e., $\sum_t \delta_t^i(k), \forall k$ (line 8) and adds the agent into the set of agents with allocated resources (line 9). Notice that all agents in stage 1 solve unconstrained MDP models that would violate the global capacity for each resource type. GAPS resolves this issue by ignoring the requests that cannot be served. Thus, the agent i^* obtains its new resource allocation that is globally feasible and solves the individual agent model with that allocation to obtain a solution to its planning problem. This process is repeated by GAPS until all resources are allocated or the unallocated resources are of no use (lines 3 and 10) before returning the joint policy and joint reward of all agents and terminating. The joint reward of all agents gives the total primal value.

Finally, based on the current primal and dual values, we provide the error in solution quality obtained by Lagrangian dual decomposition to provide posteriori quality guarantees. We use these guarantees in our experiments to make solution quality comparisons.

5 Experiments

In this section, we empirically evaluate[2] our GAPS and LDD+GAPS algorithms on two benchmark problems from the literature. The first domain is *Multi-Agent Delivery Problems* (MADPs) that have transitional uncertainty but without resource dependencies [Dolgov and Durfee, 2006]. The second domain is *Urban Consolidation Center* (UCC) problems that deal with allocation of tasks and has transitional uncertainty along with task dependencies [Handoko *et al.*, 2014; Wang *et al.*, 2014]. We compare both our approaches with an optimal MILP by Dolgov and Durfee [2006] (referred to as *Optimal MILP*), solved using CPLEX, which is a state-of-the-art algorithm for multi-agent coordination. We evaluate the performance and scalability of our approaches in reference to the optimal MILP in MADP and UCC domains.

Experimental results are averaged over 15 randomly generated grid maps that randomly place the delivery locations

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

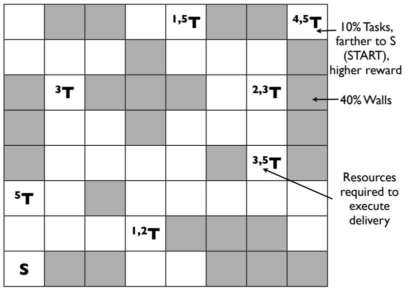

1,5T
4,5T
10% Tasks, farther to S (START), higher reward
3T
2,3T
40% Walls
3,5T
Resources required to execute delivery
5T
1,2T
S

Figure 1: Example Delivery Problem for One Agent

and walls. For runtime comparisons, apart from providing standard runtime for GAPS and LDD+GAPS, we also provide the parallel runtimes for our approaches since they have components that can be run in parallel. We compute the parallel runtime (denoted by GAPS(P) and LDD+GAPS(P)) by considering the maximum time taken by any agent when the agents are solved in parallel. A time cut-off limit of two hours was used for Optimal MILP in cases where CPLEX was unable to terminate and provide a solution. Further, for quality comparison, we compare the percentage of optimality for the three approaches (Optimal MILP, GAPS, and LDD+GAPS). For Optimal MILP, the percentage of optimality is computed using the optimality gap at cut-off time (provided by CPLEX). For GAPS, it is $Dual^*$ and, for LDD+GAPS, it is $Primal^* \cdot 100/Dual^*$, where $Primal^*$ and $Dual^*$ are the Lagrangian primal and dual values.

5.1 Multi-Agent Delivery Problems

A *Multi-Agent Delivery Problem* (MADP) deals with a team of agents, each starting from a potentially different part of the map, need to deliver goods to their respective delivery locations. We model the environment as a grid. Figure 1 shows an illustration of an example problem for one agent where 40% of the cells are untraversable (marked grey) and remaining cells are traversable (marked white). The cell with the letter "S" is the starting cell of the agent. 10% of the traversable cells are delivery locations marked with the letter "T" placed randomly throughout the grid. Each delivery task requires a set of (limited) resources, which are shown in numbers in the delivery location cells. Each agent has $4 + |\Gamma|$ actions: movement in any of the four directions and execute any of the $|\Gamma|$ delivery actions. The agents obtain reward when they successfully make a delivery. The goal is to find the division of resources to agents so that the overall reward is maximized.

We use the exact same domain representation as Dolgov and Durfee [2006]. In our experiments, we have 10 resource types $|\Gamma|$, where total resources $C(\tau)$ for each resource type $\tau \in \Gamma$ is bounded by a randomly generated number between 1 and a maximum resource limit max given as $C(\tau) = r(max)$. Each agent i has a fixed limited budget $\hat{q}_i$ of 6 resources to perform its tasks.

Solution Runtime: Figure 2 shows the runtime comparison of the three algorithms with increasing agents $|\mathcal{A}g|$, resources $|\Gamma|$, grid-size m, and horizon H. The most significant result is that GAPS and GAPS(P) obtained results in less than a minute for all cases while LDD+GAPS(P) obtained solutions in 2 to 4 minutes. One important observation is that the runtime for Optimal MILP scales exponentially (in cases where it could not finish or provide a feasible solution, it was stopped at the time-limit) with increasing agents, grid-size, and horizon, and decreasing resources. Optimal MILP could not scale beyond 70 agents (and 5 resources) in Figure 2(a) but by increasing the number of resources beyond 12 for 70 agents in Figure 2(b), the problem was less constrained and easier for Optimal MILP to solve. Further, by increasing the grid-size and horizon, we observed that Optimal MILP could not scale beyond horizon of 5 on a 6×6 grid. Overall, with respect to runtime, GAPS provides the best performance, but LDD+GAPS(P) was not far behind and both our approaches clearly outperformed (as expected) Optimal MILP.

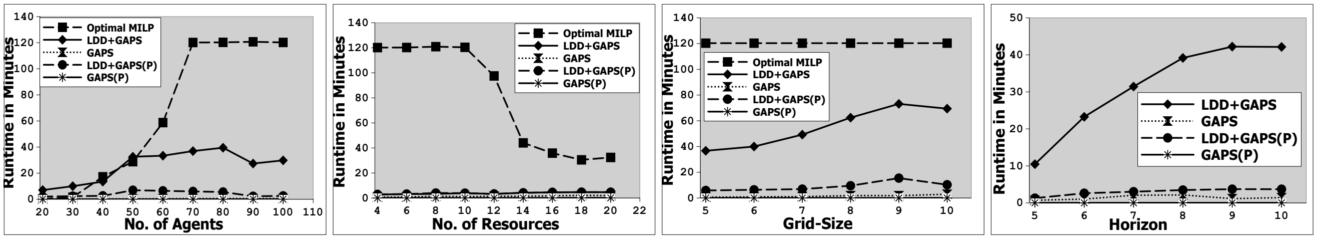
Solution Quality: Figure 3 shows the quality comparison for the three approaches where we observe that GAPS provides solutions with at least 70% of the optimal quality while LDD+GAPS performs gracefully by providing at least 98% of optimality. Optimal MILP provided 100% optimal solutions for instances with up to 50 agents, resources greater than 12, grid sizes of up to 6×6, and horizon of 6, after which the solution quality degraded significantly. Based on the above observations, we conclude that LDD+GAPS provides an excellent tradeoff between GAPS, which finds decent solutions but very quickly, and the MILP, which finds optimal solutions but is computationally expensive.

Scalability: To further experiment with scalability, we performed experiments with up to 600 agents on a grid with size $m = 10 \times 10$, horizon $H = 10$ and with resources $C(\tau) = 10\%$ of $|\mathcal{A}g|, \forall \tau \in \Gamma$. Figure 4 provides the runtime and quality comparisons for our approaches on large problems. Figure 4(a) shows the runtime comparison where the runtimes of GAPS and LDD+GAPS increase exponentially with increasing agents. However, the parallel runtime for both approaches (i.e., GAPS(P) and LDD+GAPS(P)) still remains unaffected. Figure 4(b) shows the quality comparison where LDD+GAPS provided at least 96% of optimality while GAPS could only provide 70% of optimality.

5.2 Urban Consolidation Center Problems

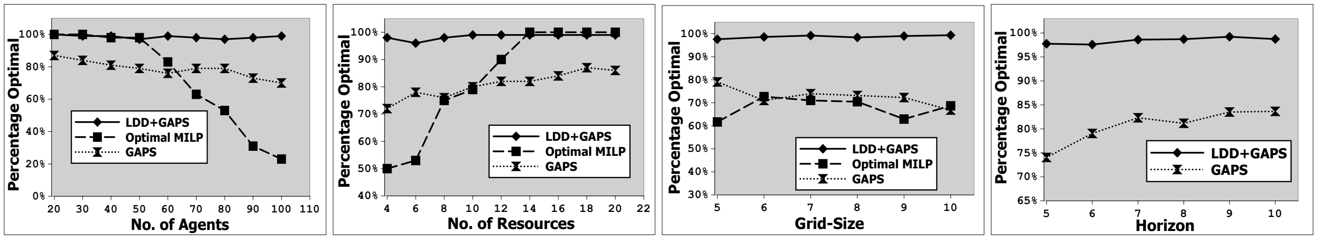
To reduce traffic and pollution due to large trucks delivering goods to distribution centers from ports, many cities have started constructing *Urban Consolidation Centers* (UCCs) [Handoko *et al.*, 2014]. Without a UCC, trucks from different companies have to move in the same parts of the city to deliver good. A UCC consolidates goods for different companies and delivers them using a team of truck agents. These truck agents have specific capabilities that help them in delivery tasks. Specifically, the tasks have difficulty levels associated with them ranging from low and medium to high while agents have capabilities defined to handle the levels of tasks. Further, there can be temporal dependencies between tasks that may require the agent to complete some delivery task prior to another task due to time window restrictions.

We use a grid world environment similar to the MADP domain, but with the difference that there can be multiple delivery tasks in a cell that do not require any resources to be completed and may have task dependencies. Unlike the deliv-



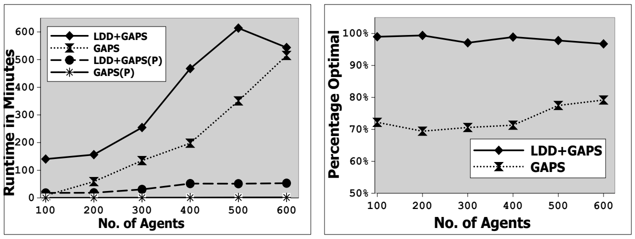
(a) $m = 5, H = 6, C(\tau) = r(5)$ (b) $m = 5, H = 6, |\mathcal{A}g| = 70$ (c) $C(\tau) = r(5), H = 6, |\mathcal{A}g| = 70$ (d) $m = 5, C(\tau) = r(5), |\mathcal{A}g| = 70$

Figure 2: Runtime comparison w.r.t. (a) Agents $|\mathcal{A}g|$; (b) Resources $C(\tau)$; (c) Grid-size $m \times m$; and (d) Horizon H for MADPs



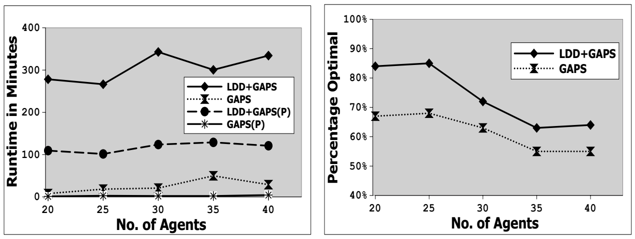
(a) $m = 5, H = 6, C(\tau) = r(5)$ (b) $m = 5, H = 6, |\mathcal{A}g| = 70$ (c) $C(\tau) = r(5), H = 6, |\mathcal{A}g| = 70$ (d) $m = 5, C(\tau) = r(5), |\mathcal{A}g| = 70$

Figure 3: Quality comparison w.r.t. (a) Agents $|\mathcal{A}g|$; (b) Resources $C(\tau)$; (c) Grid-size $m \times m$; and (d) Horizon H for MADPs



(a) Runtime Comparison (b) Quality Comparison

Figure 4: Scalability Test: $m = 10, H = 10, C(\tau) = r(\mathcal{A}g/10)$



(a) $m = 10, H = 10, |\Gamma| = 600$ (b) $m = 10, H = 10, |\Gamma| = 600$

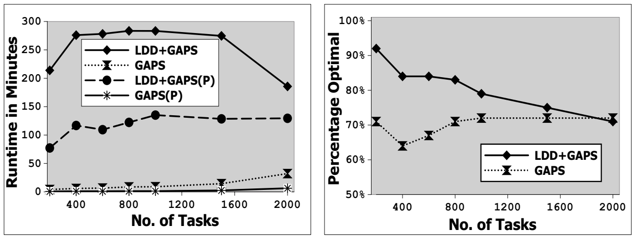
Figure 5: Varying Agents in UCCs with Task Dependencies

ery problem, the agents can stay in the same cell to perform more than one tasks. The task capacity over all agents is set to 50% of the available tasks with equal capacity for every agent. Each task in a cell represents a different task-type. Thus, the number of actions a depend on the number of tasks $\tau \in \Gamma$ present in every state (or cell). The agents are penalized for late deliveries and rewarded for on-time deliveries. The goal is to find the division of tasks to agents so that the overall reward is maximized.

Solution Quality and Runtime: Figures 5 and 6 show the runtime and quality comparison between GAPS and LDD+GAPS for UCC with task dependencies. We do not compare with Optimal MILP as it was not scalable beyond 600 tasks and 20 agents where it could solve only 30% of the random grid sets. The runtime for LDD+GAPS and LDD+GAPS(P) is approximately 5 hours and 2 hours, respectively, while for GAPS and GAPS(P), it is still between 5 to 30 minutes and 2 to 6 minutes, respectively. However, the solution quality obtained by GAPS is between 60-70% of optimality while the solution quality of LDD+GAPS ranges between 70-90% of optimality with at least 10% more optimal than GAPS. Thus, even in these domains, LDD+GAPS provides the best performance with GAPS providing a good alternative in comparison to Optimal MILP.

6 Conclusions

Motivated by stochastic planning problems in delivery of goods and services; disaster rescue; and large warehouses, we have introduced a generic model called TasC-MDP for cooperative task/resource constrained multi-agent planning problems. This model not only captures independent tasks and resources, but also temporal and allocation constraining dependencies between tasks and resources. We first provided an approach called GAPS that incrementally allocates resources in a greedy manner and then developed an optimization based approach called LDD+GAPS that exploits the decomposable structure in TasC-MDPs and uses GAPS as a subroutine. Our extensive experiments on benchmark problems demonstrate that LDD+GAPS is very scalable and provides highly optimal



(a) $m = 10, H = 10, |\mathcal{A}g| = 20$ (b) $m = 10, H = 10, |\mathcal{A}g| = 20$

Figure 6: Varying Tasks in UCCs with Task Dependencies

solutions (within 5% of optimal) even in very large problems.

Acknowledgements

This research is funded by the National Research Foundation (NRF) Singapore under its Corp Lab@University scheme.

References

[Bernstein *et al.*, 2002] Daniel Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

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

[Campbell *et al.*, 1998] Ann Campbell, Lloyd Clarke, Anton Kleywegt, and Martin Savelsbergh. The inventory routing problem. In *Fleet management and logistics*, pages 95–113. Springer, 1998.

[Chien *et al.*, 1989] T William Chien, Anantaram Balakrishnan, and Richard T Wong. An integrated inventory allocation and vehicle routing problem. *Transportation Science*, 23(2):67–76, 1989.

[Christofides *et al.*, 1981] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282, 1981.

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

[Desrochers *et al.*, 1992] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[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.

[Federgruen and Zipkin, 1984] Awi Federgruen and Paul Zipkin. A combined vehicle routing and inventory allocation problem. *Operations Research*, 32(5):1019–1037, 1984.

[Guestrin and Gordon, 2002] Carlos Guestrin and Geoffrey Gordon. Distributed planning in hierarchical factored MDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 197–206, 2002.

[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.

[Koenig *et al.*, 2010] Sven Koenig, Pinar Keskinocak, and Craig Tovey. Progress on agent coordination with cooperative auctions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1713–1717, 2010.

[Kohl and Madsen, 1997] Niklas Kohl and Oli BG Madsen. An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research*, 45(3):395–406, 1997.

[Kumar and Zilberstein, 2009] Akshat Kumar and Shlomo Zilberstein. Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 561–568, 2009.

[Nair *et al.*, 2005] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 133–139, 2005.

[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.*, 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.

[Wang *et al.*, 2014] Chen Wang, Stephanus Daniel Handoko, and Hoong Chuin Lau. An auction with rolling horizon for urban consolidation centre. In *Proceedings of the International Conference on Service Operations and Logistics and Informatics (SOLI)*, pages 438–443, 2014.

[Witwicki and Durfee, 2011] Stefan Witwicki and Edmund Durfee. Towards a unifying characterization for quantifying weak coupling in Dec-POMDPs. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 29–36, 2011.

[Wu and Durfee, 2010] Jianhui Wu and Edmund Durfee. Resource-driven mission-phasing techniques for constrained agents in stochastic environments. *Journal of Artificial Intelligence Research*, 38:415–473, 2010.