8-2018

Proactive and reactive resource/Task allocation for agent teams in uncertain environments

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

Proactive and Reactive Resource/Task Allocation for Agent Teams in Uncertain Environments

by

Pritee Agrawal

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Pradeep Varakantham (Supervisor/Chair)
Associate Professor, School of Information Systems
Singapore Management University

Akshat Kumar
Assistant Professor, School of Information Systems
Singapore Management University

Cheng Shih-Fen
Associate Professor, School of Information Systems
Singapore Management University

William Yeoh
Assistant Professor, CSE Department
Washington University in St. Louis

Singapore Management University
2018

Proactive and Reactive Resource/Task Allocation for Agent Teams in Uncertain Environments

Pritee Agrawal

Abstract

Synergistic interactions between task/resource allocation and multi-agent coordinated planning/assignment exist in many problem domains such as transportation and logistics, disaster rescue, security patrolling, sensor networks, power distribution networks, etc. These domains often feature dynamic environments where allocations of tasks/resources may have complex dependencies and agents may leave the team due to unforeseen conditions (e.g., emergency, accident or violation, damage to agent, reconfiguration of environment).

Existing research in exploiting these synergistic interactions between the two problems of task/resource allocation and multi-agent coordinated planning/ assignment have either only considered domains where tasks/resources are completely independent of each other or have focussed on approaches with limited scalability. In addition, task allocation has typically 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 non-dedicated teams (agents leave the team midway through the tasks being accomplished).

The overarching goal of this dissertation is to address the above mentioned limitations by developing computationally efficient and scalable mechanisms for solving task/resource constrained multi-agent planning/assignment with abilities to handle dependencies between tasks/resources, non dedication in agent teams and reconfiguration of the environment due to an external event. To that end, we develop generic models to handle task/resource constrained multi-agent planning/assignment for dedicated and non-dedicated agent teams. Given these models, we design scalable proactive and reactive algorithms that provide provable *quality-bounds*. The proactive approaches mainly exploit decomposability to solve independent agent planning/assignment problems and provide posterior quality guarantees while the reactive approaches are highly efficient and provide very quick solutions but without quality guarantees. Finally, we empirically show the high scalability and better solution performance of our approaches in comparison with existing work on the real-world and synthetic benchmarks from literature.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I owe a lot of gratitude to my advisor, Pradeep Varakantham. Without his support, patience, and guidance, this dissertation would not have been possible. I especially want to thank Pradeep for always inspiring me and encouraging me to think critically, and pushing my limits in my early Ph.d. days, which has helped me to overcome the challenges in my research. He has left an invaluable mark on my professional development. I truly believe Pradeep's top work priority is the growth of his students, and for that I offer my deepest thanks!

I have also been very fortunate to receive an excellent mentorship from my co-advisor, Akshat Kumar. He has always given me practical advice and feedback. I am very thankful to Shih-Fen Cheng for his valuable advice and insightful comments on my work. It was a great experience for me to explore the teaching aspect during my TA duties with him. I am also very grateful to Hoong Chuin Lau for his efforts in conducting group seminars and encouraging us to participate in discussions. William Yeoh has been an excellent collaborator and has provided valuable advice and insightful comments on my work.

I shared an amazing camraderie and friendships with my teammates. Thanks to all of you, and especially Supriyo, Rajiv, Meghna, Tanvi, Kapil, Murli and Asrar for adding life to my graduation years. I thoroughly enjoyed the time I spent with you all, having various technical and non-technical discussions. I also want to thank my friends, Nitya, Ranjan, Pratik and Saurabh for making my life in singapore much more enjoyable.

To my parents, I owe so much more than can be expressed on paper or in words. Mumma, Papa, thanks for everything! I cannot thank enough my sister, Swati who has always stood by me, no matter what. Finally, I thank my best friend and husband, Ankit, for his love and support. This journey would have been much difficult without him.

To
the almighty & my family

Chapter 1

Introduction

The synergistic interactions between task/resource allocation and multi-agent coordinated planning/assignment coexist in many real world environments ranging from urban transportation and logistics, UAV (Unmanned aerial vehicles) task assignment, traffic patrolling and disaster rescue to distributed planning and resource allocation, sensor networks and power distribution networks to name a few. In these environments, the process of planning/assignment is often complicated by the presence of multiple interdependent objectives for agents, whose achievement requires execution of action sequences which might have long-term, non-deterministic effects. Therefore, there exists a cyclic dependency between the task/resource allocation and the agent planning problem since the agent's value for a set of resources/tasks is defined by the solution to the planning problem but the agent requires information of the resources/tasks it will obtain to formulate it's planning problem. We refer to the agent teams dealing with the above mentioned cyclic dependency as **dedicated agent teams**. Agents of such a team compulsorily stay in the system until the end of the time horizon and execute the tasks assigned (or use the resources allocated) to them by the central planner.

However, most of the above mentioned domains experience dynamic environments where some agents may not be fully dedicated to the tasks assigned by the central planner and leave the system 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). 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**.

In all the above mentioned problem domains, the teams of agents operate in real world environments where uncertainty arises either due to transition uncertainty in the planning problems of individual agents or due to agents leaving the system or due to reconfiguration of the system after some event. Planning complexity may grow exponentially with increasing agents, tasks/resources, etc. Existing research has focussed on exact planning techniques that provide optimal solutions but are not scalable to very large real world problems. Decomposition techniques have been used to solve deterministic problems but not exploited extensively either with stochastic planning where the number of resources consumed (or tasks

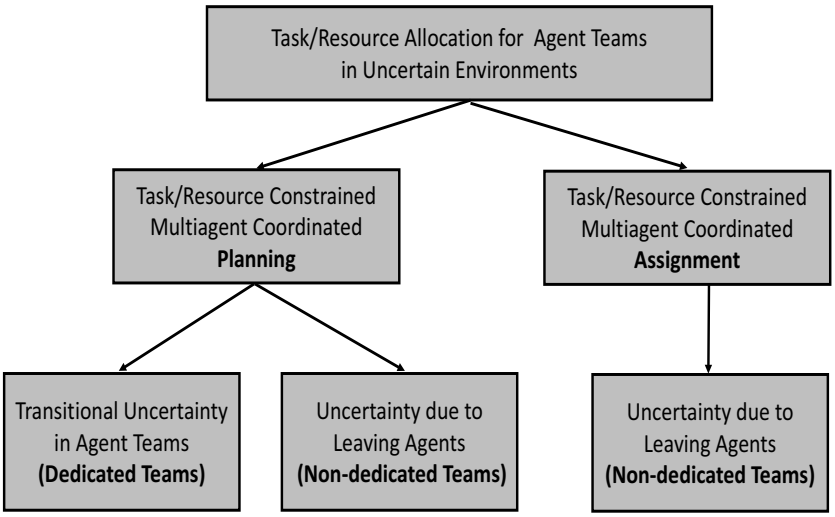


Figure 1.1: Contributions: Classification Based on Uncertainty

completed) is not easy to compute due to transition uncertainty or with multi-agent co-ordination problems. Furthermore, 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 are non-dedicated and leave the team after task allocation.

The focus of this dissertation is on providing generic models and scalable solution approaches with improved solution quality that can be easily applied to any problem domain involving task/resource allocation and multi-agent coordinated planning/assignment. The fundamental insight of the solution approaches in this work is that even though the interactions between task/resource allocation and the planning/assignment problem are synergistic, given the allocation of tasks/resources, the agent planning/assignment problems can be solved independent of each other. We exploit this key insight to solve the dedicated team problems by allocating tasks/resources to agents in one shot after which reallocation is not allowed during plan execution phase. However, in case of non-dedicated agent teams, reallocation or rearrangement of tasks/resources may be allowed, but only when there is a change in the system configuration (e.g., agents leave the team before the end of planning horizon, change in network structure due to an event). We describe a formal framework for dedicated and non-dedicated agent teams, analyse their properties, and develop computationally tractable solution algorithms in this dissertation.

1.1 Contributions

The work presented in this dissertation was done in collaboration with several colleagues. In all cases, I contributed substantially to the research.

The main contribution of this dissertation is a class of efficient and scalable solution approaches for the interdependent problems of task/resource allocation

and multi-agent coordinated planning/assignment for agent teams in uncertain environments. The key result of this work involves recognizing the decomposable structure in such problems, which is exploited to illustrate high scalability and improved solution performance of our approaches on large real world and synthetic benchmarks. In addition, we also exploit greedy approaches to provide quick and efficient solutions. More specifically, the major contributions of the work presented in this dissertation are depicted schematically in Figure 1.1. We classify the Task/Resource allocation for agent teams into two major categories based on the type of uncertainty handled by them.

(1) Task/Resource Constrained Multiagent Coordinated Planning : This class of problems deals with task/resource based interactions between the planning agents of a team. We handle uncertainty in these problems that arise either due to transition uncertainty or due to agents leaving the system midway through the accomplishment of tasks.

- **Transition Uncertainty in Agent Teams (Dedicated Teams) :** We classify agent teams dealing with only transition uncertainty as dedicated agent teams. Several generic models including weakly coupled MDPs and resource-parametrized MDPs have been proposed to represent the task/resource constrained multi-agent planning problems. However, these existing approaches are not scalable to large scale problems and do not consider dependencies between tasks/resources. Therefore, we propose a generic model for solving the task/resource constrained multi-agent coordinated planning problems that can handle task dependencies along with independent tasks. Our greedy based proactive solution approaches improve the scalability while providing high quality solutions.

- **Uncertainty due to Leaving Agents (Non-dedicated Teams) :** We extend the model for dedicated teams to represent the problems dealing with non-dedicated agent teams that handle transition uncertainty and uncertainty due to agents leaving the team midway before the tasks are accomplished. We provide multiple proactive and reactive approaches to generate policies for individual agents in the non-dedicated agent teams. For the centralized multi-agent planning problems, our proactive approaches based on sample average approximation and decomposition provide highly efficient offline one-stage and two-stage policies for the agents while greedy reactive approach provides instantaneous solutions when agents leave the team. In addition, for the decentralized multiagent planning problems, we exploit submodular rewards to provide greedy approaches that handle non-dedicated agent teams and provide excellent performance, almost at par with the benchmark approaches.

(2) Task/Resource Constrained Multiagent Coordinated Assignment : This class of problems deals with coordinated decision making by a team of agents along with allocation of tasks/resources.

- **Uncertainty due to Leaving Agents (Non-dedicated Teams) :** We classify task/resource constrained multi-agent coordinated assignment problems as non-dedicated agent teams due to the uncertainty introduced by the

agents leaving the system midway through the tasks. Existing research on task/resource constrained multi-agent coordinated assignment have used Resource Constrained DCOP (RC-DCOP) (Matsui, Matsuo, Silaghi, Hirayama, & Yokoo, 2008; Bowring, Tambe, & Yokoo, 2006; Kumar, Faltings, & Petcu, 2009) framework to handle resources for multiagent coordination problems. However, they suffer from exponential memory requirement problems and cannot scale up to large real world instances. Therefore, we provide a model to exploit decomposability in structured networks by dividing a global network into multiple small networks (each small network represents an agent). In case of availability of a centralized planner, the centralized planner broadcasts required information to different agents. However, in the absence of centralized planner, we use a simple distributed approach that uses local message passing for communication among different agents.

We briefly present the advancements obtained by our solution approaches.

- **Decomposability:** For all the problems mentioned in this thesis, agents are either weakly coupled or fully decentralized, and hence decomposition is naturally applicable. Therefore, we use Lagrangian dual decomposition (LDD) to exploit decomposability, which provides the desirable *anytime* property due to it's iterative nature and posterior guarantees on solution quality. Using these quality bounds, our solution approaches could provide near-optimal solutions on a number of large real-world and synthetic benchmarks.

- **Scalability:** Our approaches are able to generate solutions efficiently for multi-agent problems with hundreds of agents and thousands of tasks. We demonstrate superior performance to existing approaches, specifically on large scale problems. For example, the scalability of decentralised power network (discussed in details in chapter 7) improved by at least 30 fold over previous best multi-agent approaches by exploiting near decomposability amongst regions.

- **Solution Performance:** We obtain near optimal solutions in the real-world and synthetic benchmarks with significant speed-ups for the problems dealing with task/resource allocation and multiagent planning. In centralized settings, the solution quality obtained by greedy approaches varies between 60-70% of optimal while the solution quality of optimization approaches ranges between 70-90% of optimal. Furthermore, in decentralized settings, we were able to obtain strong quality guarantees of approximately 85% or more of the optimal, that is comparable to highly efficient centralized solver.

- **Homogeneity:** In many planning domains, agents are homogeneous, that is, they are exactly of the same type with identical agent models for all allocations of resources/tasks. For example, various robots in disaster rescue are mass produced with identical capabilities. Thus, we exploit homogeneity in agent models by ensuring evaluation for one agent of each type, thereby improving the scalability and decreasing the total runtime of our algorithms.

- **Parallelism:** Due to the exploitation of decomposability, parallel computation and communication among individual agent planning problems is easy. Hence, instead of solving the individual MDP models sequentially, they can be solved in parallel by each agent as they are independent of each other. Parallel execution of individual agent planning problems provides quick solutions by reducing the runtimes significantly due to the use of multiple processors and cheap communication.

1.2 Overview of the Thesis

The rest of the thesis is organised as follows:

Chapter 2 provides background on motivating domains, existing models and algorithms to solve the problem from literature.

Chapter 3 provides the models introduced by us to handle dedicated and non-dedicated agent teams for task/resource constrained multi-agent coordinated planning/assignment.

Chapter 4 deals with task/resource constrained multi-agent coordinated planning for dedicated agent teams with an ability to handle transition uncertainty. This chapter presents a generic model for task/resource constrained markov decision problems (TasC MDPs) with an ability to handle task/resource dependencies. It focusses on two proactive scalable greedy algorithms that provide high scalability and solution performance on existing benchmark problems from literature.

Chapter 5 extends the generic model presented in Chapter 4 for a non-dedicated team of agents with an ability to handle transition uncertainty and uncertainty due to agents leaving the team. Proactive and reactive benchmarking heuristics have been presented that bound the best performance achievable. This chapter provides a reactive approach that provides instantaneous policies for agents whenever agents leave the system and two proactive approaches that provide offline stage-based closed loop policies for any sample of leaving agents.

Chapter 6 deals with task/resource constrained multi-agent coordinated decentralized planning for non-dedicated agent teams. We extend the concept of non-dedication to transition independent Dec-MDP (Kumar, Varakantham, & Kumar, 2017) by formulating ND-TI-Dec-MDP model for a team of independently collaborating non-dedicated agents. We exploit joint submodular rewards for decentralised non-dedicated agent teams and provide strong quality guarantees (a priori, and posteriori guarantees). We extend the benchmark approaches introduced in Chapter 5 by providing a lazy greedy extension for decentralized settings. This chapter presents two greedy approaches (an offline one and an offline-online one) that are able to deal with agents leaving the team in an effective and efficient way by exploiting the submodularity property.

Chapter 7 presents a general approach for task/resource constrained multi-agent coordinated assignment for non-dedicated agent teams. This chapter presents a decentralized solution approach for resource constrained multi-agent problems in distributed network settings. In this chapter, this problem has been studied and examined on one such domain of distributed network, power supply restoration for smart grids which requires co-ordination of multiple operators managing different regions of the network. A proactive dual decomposition based approximate dynamic programming approach that uses local message passing between communicating agents has been proposed to provide scalable solutions for large real world and synthetic problem instances.

Chapter 8 presents related work for the dedicated teams and non dedicated teams to solve the problem of task/resource allocation in uncertain environments.

Chapter 9 finally presents a conclusion with future directions.

Chapter 2

Background

This chapter provides a brief background on the experimental domains, existing models and the existing algorithms to solve these models.

2.1 Motivating Domains

In this section, we describe a few concrete problems where dedicated and non-dedicated teams operate in uncertain environments.

2.1.1 Multi-agent Delivery Problem

A *Multi-Agent Delivery Problem* (MADP) (Dolgov & Durfee, 2006) deals with a team of dedicated agents, each agent operating in a different map specific to the agent and starting from a potentially different part of the map, need to deliver goods to their respective delivery locations. Each delivery task requires a subset of resources, and there are limited quantities of each type of resource. There are random delivery locations on the grid, and each location has a set of deliveries that it accepts. Each resource has some size requirements (capacity cost), and each delivery agent has bounded space to hold the resources (limited capacity). The agents are allocated resources by a centralized planner and the value of a resource depends on what other resources the agent acquires and what other deliveries it can make. Given a bundle of resources, a delivery agent can solve it's planning problem independently to find an optimal delivery plan.

Figure 2.1 shows an illustration of an example problem for one agent operating in a grid world 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 following actions: movement in any of the four directions and execution of any of the different delivery actions. The agents obtain a reward when they successfully make a delivery. The goal is to find a division of the resources to agents so that the overall reward is maximized.

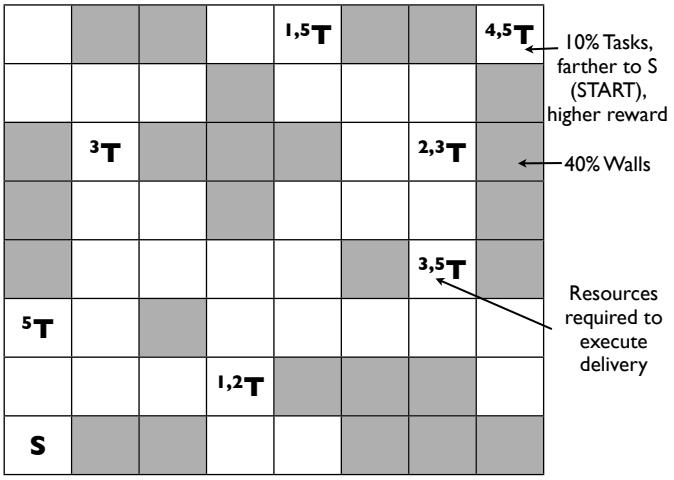


Figure 2.1: Example Delivery Problem for One Agent

2.1.2 Urban Consolidation Center

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, Nguyen, & Lau, 2014). Without a UCC, trucks from different companies have to move in the same parts of the city to deliver goods. UCC is an alliance concept for last mile deliveries where delivery orders of various participating companies are consolidated and sorted according to their destination addresses. This results in cost savings and higher truck utilization as fewer trucks are required. It is further assumed that the UCC operates its own vehicles to consolidate and deliver goods to the city centre by assigning a vehicle to every order for the last-mile deliveries. These vehicle 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 the vehicles have capabilities defined to handle the levels of tasks. Further, there can be temporal dependencies between tasks that may require the vehicles to complete some delivery task prior to another task due to time window restrictions. There are limited number of agents (i.e., tens) but thousands of tasks which increase the complexity of the problem.

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 dependencies between different delivery tasks(e.g., task A must be completed before task B , task B and C must be assigned to agent 1). Unlike the delivery problem, all agents operate on a single map where the agents can stay in a cell to perform more than one tasks available in the cell. Every task is independent of other tasks, irrespective of the cell it is present in. 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.

The above discussed details of UCC domain reflect the presence of a dedicated team of agents in uncertain environment where transitional uncertainty exists in the domain due to movement of truck agents on roads. Further, this domain can be extended to deal with non-dedicated teams where agents leave the team (either due to personal emergency or road accidents) before completing their assigned tasks.

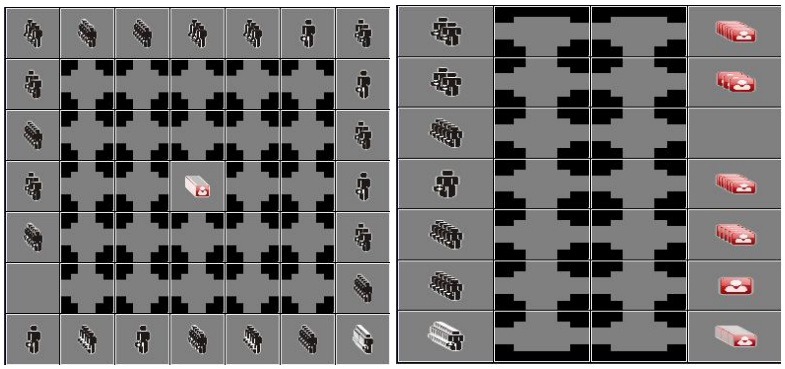
2.1.3 Power Supply Restoration in Smart Grids

A power distribution system is a network of electric lines connected by switching devices (SDs), and fed by circuit breakers (CBs) (Bertoli, Cimatti, Slanley, & Thiebaux, 2002). Both SDs and CBs have two device positions: closed, open. SDs are analogous to sinks (transformer stations) which consume some power and forward the rest on other lines if closed. Open SD stops power flow. Circuit breakers, which are analogous to power sources with finite available power, feed the network when closed. The positions of the devices are set such that the paths taken by the power of each CB forms a tree called *feeder tree*, and no sink is powered by more than one power line. In addition, Kirchhoff's law (or flow conservation requires that an edge must carry power for both, the directly connected sink and others which may receive power indirectly through it) must hold for all devices and the current load for any line must not exceed its capacity.

Uncertainty arises in power distribution systems due to faults in the feeders (power sources) which are responsible for supplying power. Further, faulty power lines may also isolate a feeder from supplying power to the assigned region. We consider the regions that were powered by these faulty feeders as agents leaving the system. This enforces the reconfiguration of power network to restore supply to the non-faulty areas of the regions with lost feeders. Therefore, restoration plans must be built to isolate these faulty elements by prescribing to open the switching devices surrounding them. It should be noted that we assume that faulty elements are identified in the network and isolated to prevent feeding power to a faulty line before the task of power supply restoration (PSR) is done. PSR reconfigures the network (sets positions of devices) such that power supply is restored to maximum possible affected sinks after one or more power lines or sources become faulty. Every sink is assigned a weight based on the priority (i.e., high priority is given to critical consumers like hospitals) so as to resupply power to them as soon as possible. Hence, power distribution networks deal with a non-dedicated team of feeder agents that handle coordinated assignment of power supply.

2.1.4 Movement of Goods at Large Warehouses

Inspired by the warehouse mechanics of Kiva systems, targeted at pick-pack-and-ship, large warehouses (e.g., Amazon, Walmart, etc.) employ several teams of robots to accomplish tasks of pickup and delivery (Kucera, 2012; D'andrea, Mansfield, Mountz, Polic, & Dingle, 2012). These robots are capable of lifting and carrying shelving units from storage locations to stations where workers can pick items off the shelves and put them into shipping cartons. A typical installation of such a system in a large warehouse will involve hundreds of robots. Although the overall system is cooperative, the Kiva robots handle their tasks independent of each other to complete an order. The resources are limited which include space and elements of the environment that are needed to accomplish the tasks. We are specifically interested in cases where there is a malfunction of the robots. In such cases, tasks have to be dynamically reallocated to the remaining robots so as to not effect efficiency



(a) Map 1 (b) Map 2

Figure 2.2: Disaster Rescue Maps

of the overall process.

2.1.5 Disaster/Emergency Response

In disaster rescue problems (Nair, Varakantham, Tambe, & Yokoo, 2005; Varakantham, young Kwak, Taylor, Marecki, Scerri, & Tambe, 2009), a team of heterogeneous robots have a goal of saving the victims trapped in a building with debris. A building is modelled as a grid with narrow corridors connecting neighbouring grid cells and debris in some grid cells. Narrow corridors allow for only one robot to pass through; when multiple robots try to pass through, a collision (modelled with negative rewards) occurs. In this problem, the resources are the rights to pass through narrow corridors and the agents are the robots. In the two disaster rescue maps shown in Figure 2.2, red icons represent victims (in the middle of Map 1 and in the right-most column of Map 2), black icons represent robot rescuers (along the perimeter of Map 1 and in the left-most column of Map 2), and the other icons represent narrow corridors. All the robot rescuers are identical in that they all have the same transitional uncertainties and the same reward functions. All robots must reason about uncertainty in their actual positions and slippages (action failures) when moving to locations. The goal of the robots is to save as many victims as possible within the time available.

Similarly, in emergency response problems (Saisubramanian, Varakantham, & Lau, 2015), a team of emergency response vehicles have to attend to multiple incidents. In these domains, there is uncertainty in transitions because of how the disaster/emergency evolves. We are also interested in cases where members of the team have to leave the current emergency and attend to other higher priority emergencies.

2.1.6 Infrastructure Security

Recent research has introduced security games to model the problem of preventing incidents from happening through patrols, checks and projecting presence where a set of defenders coordinate to secure a set of targets against an observing adversary. Specific instances of this problem that are of relevance to this work include the metro rail network problem (Shieh, Jiang, Yadav, Varakantham, & Tambe,

2014; Varakantham, Lau, & Yuan, 2013), traffic patrolling (Brown, Saisubramanian, Varakantham, & Tambe, 2014), airport security (Pita, Jain, Marecki, Ordóñez, Portway, Tambe, Western, Paruchuri, & Kraus, 2008) and coast guard protection (Shieh, An, Yang, Tambe, Baldwin, DiRenzo, Maule, & Meyer, 2012; An, Pita, Shieh, Tambe, Kiekintveld, & Marecki, 2011).

In the above mentioned prevention settings, the defender resources conduct patrols to secure their targets (e.g., stations in metro rail domain, streets in traffic domain, etc.) while the adversary conducts surveillance and may take an advantage of the defender's predictability to plan an attack or make violations. Furthermore, teamwork among the defenders for patrolling related incidents is complicated due to the requirement of coordination of their activities under transition uncertainty (e.g., delays may arise from unexpected situations leading to miscoordination, making the defenders unable to act simultaneously). In addition to patrolling responsibilities, defenders are also tasked with attending to other emergencies or violations that are not in the purview of the patrolling problem. Due to this additional responsibility, defenders can be forced to leave their assignment (patrols or check/screening schedule) to attend to an accident or incident (e.g., incursion, smuggling, road accident) or some other higher priority task before the end of their patrolling horizon. This provides the key reason for non-dedication of defenders which requires the non-leaving defenders to fill in the gaps that are created due to leaving agents. Furthermore, in many prevention settings, the effectiveness of patrols is submodular in the number of agents patrolling a target (i.e., the incremental improvement in effectiveness with an additional patrol team reduces with more agents). Providing policies for individual defenders given this non-dedicated nature of team members and the submodularity of rewards is an important problem of interest in this work.

2.1.7 Sensor Networks

Sensor networks (Nair et al., 2005; Kumar & Zilberstein, 2011) are useful for observing some spatial phenomenon (i.e., tracking of moving targets) where the coverage scope of every sensor is limited to its sensing range. More specifically, in this domain, the environment is modelled as a grid and the sensors are randomly placed at junctions of cells where every sensor can track four target cells surrounding the sensor. The reward function is submodular with n-ary interactions (any number of sensors can track a target). Intuitively, adding a sensor helps more if fewer sensors are tracking a target and helps less if the number of sensors is already high. This is formally the diminishing returns property of submodularity.

With a certain probability, deployed sensors can fail either due to wear and tear or due to unforeseen conditions. The goal is to maximize the area covered by the functioning sensors even after some sensors are spoilt. Due to closeness in sensing range, the neighbouring sensors can easily track the moving targets of the damaged sensors in order to maximize the reward. Therefore, reconfiguration of sensor policies after one or more sensors get spoilt is of high importance. In addition, the targets are assumed to move stochastically (according to some fixed distribution) in the grid and follow a path of fixed length for movement.

2.2 Background

In this section, we provide a background of the concepts that are fundamental for the models developed in this thesis. A brief introduction to markov decision processes (MDP) is provided in section 2.2.1 since we use MDPs to model transition uncertainty in dedicated and non-dedicated agent teams in upcoming chapters. We introduce the concepts of submodularity, matroids and submodular TI-Dec-MDP in sections 2.2.2 and 2.2.3 as they form the basis for ND-TI-Dec-MDP model introduced in chapter 6 to handle non-dedication in decentralized agent teams. Finally, distributed constraint optimization (DCOP) in section 2.2.4 have been used for handling structured resources in multiagent settings and provide a background for the assignment model introduced in chapter 7.

2.2.1 Markov Decision Process

Markov Decision Process (Puterman, 1994) provides a mathematical framework for modeling decision making in planning problems where the outcomes are partly random and partly under the control of a decision maker.

Definition 1. *A Markov Decision Problem (MDP) is defined by* $\boldsymbol{M} =< S, A, P, R, H, \alpha >$ *with state space* S*, action space* A*, time horizon* H*, transition function* $P : S \times A \times S \times H \rightarrow [0; 1]$ *and reward function* $R : S \times A \times H \rightarrow \mathbb{R}$*.* $\mathbb{R}$ *is the set of real numbers. The distribution of initial states is* α *and* $\alpha(s)$ *denotes the probability of starting in state* s*.*

For a given time horizon H, the objective of the MDP is to find a policy $\pi : S \times A \times H \rightarrow [0; 1]$ that maximizes the total expected reward within the time horizon H:

$$\max \sum_{s_0} \alpha(s_0) \cdot \sum_{t=0}^{H} P^t(s, a|s_0, \pi) \cdot R^t(s, a) \tag{2.1}$$

The optimal policy for an MDP can be obtained by solving the following linear program (Puterman, 1994):

$$\max \sum_{t=0}^{H} x^t(s, a) \cdot R^t(s, a) \quad \text{s.t.}$$

$$\sum_{a \in A} x^0(s, a) = \alpha(s) \qquad \forall s \in S \tag{2.2}$$

$$\sum_{a \in A} x^{t+1}(s', a) = \sum_{s \in S, a \in A} x^t(s, a) \cdot P^t(s, a, s') \qquad \forall s' \in S \tag{2.3}$$

in which $x^t(s, a)$ is the occupancy variable used to specify the expected number of times (s, a) visited at period t. Flow constraints (2.2), (2.3) enforce flow preservation, i.e., the total number of agents taking actions to move out of s' is equal to the total number of agents coming into s'. We can derive the policy π from solution of above discussed MDP as follows:

$$\pi^t(s, a) = \frac{x^t(s, a)}{\sum_{a \in A} x^t(s, a)} \tag{2.4}$$

2.2.2 Monotone Submodularity and Matroids

We now describe submodular functions and matroids.

Definition 1. *Given a finite set, Π, a **submodular** function is a set function, $g : 2^{\Pi} \rightarrow \mathbb{R}$, where 2^{Π} is the power set corresponding to Π. More importantly, $\forall X, Y \subseteq \Pi$ with $X \subseteq Y$ and for every $i \in \Pi \setminus Y$, we have:*

$$g(X \cup i) - g(X) \geq g(Y \cup i) - g(Y)$$

*A submodular function g is **monotone** if $g(Y) \geq g(X)$ for $X \subseteq Y$.*

Monotone submodular functions are interesting because maximizing a submodular function to pick a fixed number of elements (say k) from the finite set (Π) while difficult can be approximated efficiently with a strong quality guarantee. Specifically, a greedy algorithm that incrementally generates the solution set by maximizing marginal utility provides solutions that are at least 63% ($1 - \frac{1}{e}$) of the optimal solution.

If we have a submodular function under a specific constraint on the finite set (Π) and the elements that are picked, the constraint is specified using a partition matroid. In this dissertation, we are also interested in maximizing a submodular function, however, under a specific constraint on the finite set (Π) and the elements that are picked. Specifically, the constraint is specified using a partition matroid. We provide the formal definitions below:

Definition 2. *For a finite ground set, Π, let $\mathcal{P}$ be a non-empty collection of subsets of Π. The system $\Gamma = (\Pi, \mathcal{P})$ is a matroid if it satisfies the following two properties:*

- *The hereditary property: $\mathcal{P}_1 \in \mathcal{P} \wedge \mathcal{P}_2 \subset \mathcal{P}_1 \implies \mathcal{P}_2 \in \mathcal{P}$. In other words, all the subsets of $\mathcal{P}_1$ must be in $\mathcal{P}$.*
- *The exchange property: $\forall \mathcal{P}_1, \mathcal{P}_2 \in \mathcal{P} : |\mathcal{P}_1| < |\mathcal{P}_2| \implies \exists x \in \mathcal{P}_2 \setminus \mathcal{P}_1; \mathcal{P}_1 \cup x \in \mathcal{P}$.*

We are specifically interested in a ground set that is partitioned as $\Pi = \Pi_1 \cup \Pi_2 \cup \ldots \cup \Pi_k$. The family of subsets, $\mathcal{P} = \{P \subseteq \Pi : \forall i, |P \cap \Pi_i| \leq 1\}$ forms a matroid called a partition matroid. This family of subsets denotes that any solution can include at most one element from each ground set partition where the ground set partitions represent the policy space of each agent and exactly one policy must be picked for each agent.

2.2.3 Submodular TI-Dec-MDP

Submodular Transition Independent Decentralized Markov Decision Process (TI-Dec-MDP) model (Kumar et al., 2017) is characterized by the tuple:

$$\Big\langle \mathcal{A}g, S, A, \{P_i\}_{i \in \mathcal{A}g}, R, H, \alpha \Big\rangle$$

- $\mathcal{A}g$ is the set of agents.

- S is the factored joint state space. $S = S_1 \times S_2 \ldots S_{|\mathcal{A}g|}$, where S_i is the state space corresponding to each individual agent i. We can also have a global unaffected state feature, S_u.
- A is the joint action space. $A = \times_{i \in \mathcal{A}g} A_i$, where A_i is the action space corresponding to each individual agent i.
- P_i is the individual agent transition function. $P_i(s'_i|a_i, s'_i)$ indicates the transition probability of moving from s_i to s'_i on taking action a_i.
- R is the monotone submodular joint reward, with $R(s, a)$ representing the reward for taking joint action a in joint state s. In security domains (Shieh et al., 2014), reward is both monotonically increasing and submodular. It is defined as follows:

$$R(s, a) = \sum_b y_b \cdot f_b(\sigma(s, a, b)) \tag{2.5}$$

  y_b indicates value of target b and hence is a non-negative number. $f_b(.)$ is a monotone submodular function referred to as the effectiveness of patrolling a target b. Effectiveness of patrols at a target b depends on the number of agents patrolling the target. $\sigma(s, a, b)$ counts the number of agents at target b if the current joint state is s and joint action is a. The usual definition of $f(.)$ for effectiveness parameter ϵ $(0 < \epsilon \leq 1)$ is as follows:

$$f(k) = 1 - (1 - \epsilon)^k.$$

- H is the time horizon.
- α is the starting state distribution.

The goal is to obtain a joint policy $\pi^* = \langle \pi_1^*, \pi_2^*, \ldots, \rangle$ (with one policy, π_i^* for each agent i) that maximizes expected reward or value defined as follows:

$$V(\pi^*) = \sum_s \alpha(s) \cdot V^H(s, \pi^*) \tag{2.6}$$

$$V^t(s, \pi^*) = R\Big(s, \big\langle \pi_1^t(s_1), \ldots, \pi_{|\mathcal{A}g|}^t(s_{|\mathcal{A}g|}) \big\rangle\Big) + \sum_{s'} \Big[\prod_{i \in \mathcal{A}g} P_i\Big(s'_i | \pi_i^t(s_i), s_i\Big) \Big] \cdot V^{t-1}(s', \pi^*) \tag{2.7}$$

2.2.4 Distributed Constraint Optimization (DCOP) & Resource Constrained DCOP (RC-DCOP)

A distributed constraint optimization problem (DCOP) model (Modi, Shen, Tambe, & Yokoo, 2005; Petcu & Faltings, 2005; Gershman, Meisels, & Zivan, 2009; Yeoh & Yokoo, 2012) is represented as a tuple $\langle X, D, F \rangle$ where $X = \{X_1, ..., X_n\}$ is a set of variables, and $D = \{D_1, ..., D_n\}$ is a set of finite variable domains. $F = \{f_1, ..., f_m\}$ is a set of functions (also called constraints), where each f_i is a function with scope $(X_{i_1}, ..., X_{i_k})$. It is defined as $f_i : D_{i_1} \times ... \times D_{i_k} \rightarrow \mathcal{R}$, which denotes the cost assigned to each possible combination of values of the involved variables. In a DCOP, each variable and constraint is owned by an agent and the goal is to find a solution that minimizes the sum of constraint costs.

Variables: $\forall s, \sigma \in S; \forall a \in A; \forall o \in O; \forall c \in C; \forall m \in M;$

Maximize: $$\sum_m \sum_s \sum_a x^m(s,a) \cdot r^m(s,a) \tag{2.8}$$

Subject to:

$$\sum_a x^m(\sigma,a) - \gamma \cdot \sum_s \sum_a x^m(s,a) \cdot p^m(\sigma|s,a) = \alpha^m(\sigma), \forall \sigma, m \tag{2.9}$$

$$\sum_o \kappa(o,c) \cdot \delta^m(o) \leq \hat{\kappa}^m(c), \ \forall c, \forall m \tag{2.10}$$

$$\sum_m \delta^m(o) \ \leq \hat{\rho}(o), \ \forall o \tag{2.11}$$

$$\frac{1}{X} \sum_a \rho^m(a,o) \sum_s x^m(s,a) \leq \delta^m(o), \ \forall o, m \tag{2.12}$$

$$x^m(s,a) \geq 0, \delta^m(o) \in \{0,1\} \tag{2.13}$$

Table 2.1: Optimal MILP for Resource Parametrized MDP

Resource Constrained DCOP (RC-DCOP) (Matsui et al., 2008; Bowring et al., 2006) adds support for resources in the form of constraints to the DCOP framework explained above. The resource constraints are defined by a set of resources R with every resource $r_a \in R$ having a capacity $C(r_a) : R \rightarrow \mathcal{R}$. The set U defines the set of requirements or the quantity of resources required by agents i.e. $u_i(r_a, d_i) : R \times D_i \rightarrow \mathcal{R}$ defines the amount of the resource r_a required by agent i under the assignment d_i. The global resource constraint requires that the capacity of each resource must not be exceeded i.e. $\forall r \in \mathcal{R}, \sum_i u_i(r, d_i) \leq C(r)$ under the assignment X. An important generalization implies that each resource requirement u may take any arity leading to n-ary constraints among agents, leading to the increased complexity of solving an RC-DCOP compared to DCOP.

2.3 Existing Algorithms

This section presents the existing algorithms for solving multi-agent planning/assignment problems for dedicated and non-dedicated agent teams. We have compared our solution approaches with these existing benchmarks in the experimental sections of this document for performance comparison.

2.3.1 Resource Parameterized MDP

In the literature of multi-agent planning for dedicated agent teams, Dolgov and Durfee (2006, 2004) introduced Resource Parameterized MDPs where the action set of the MDP is paramaterized on the set of resources available to the agent. Every individual agent solves an optimization problem to choose a subset of available resources such that the best feasible policy under that bundle of resources yields highest utility.

Definition 2. *The model for agent's optimization problem is defined using the tuple* $\langle M, \langle S, A, p^m, r^m, \alpha^m, \rho^m, \hat{\kappa}^m \rangle, O, C, \kappa, \hat{\rho} \rangle$.

- *M is the set of agents.*
- $\langle S, A, p^m, r^m, \alpha^m, \rho^m, \hat{\kappa}^m \rangle$ *is the set of weakly coupled single agent MDPs.*
  - $\langle S, A, p^m, r^m \rangle$ *are the sets of states, actions, transition and reward functions, respectively for agent* m.
  - α^m *is the starting probability distribution for agent* i.
  - $\rho^m : A \times O \rightarrow \mathbb{R}_{\{0,1\}}$ *is a function that specifies the binary task/resource requirements of all actions.*
  - $\hat{\kappa}^m : C \rightarrow \mathbb{R}$ *corresponds to the upper bound on the capacities (e.g.,* $\hat{\kappa}^m(c)$ *gives the upper bound on capacity* $c \in C$ *) for agent* m.
- O *and* C *are the set of resources and capacities.*
- $\kappa : O \times C$ *represents the capacity cost of resources.*
- $\hat{\rho} : O \rightarrow \mathcal{R}$ *specifies the upper bound on the amounts of shared resources.*

Given the above model, the goal of resource parametrized MDP is allocation of resources to agents in a way that maximizes the social welfare of the agents. The resources in the model outlined above are non-consumable, i.e., actions require resources but do not consume them during execution. The multi-agent problem of selecting an optimal subset of resources that does not violate its capacity constraints and provides optimal policies for agents can be formulated as a mixed integer linear program (MILP) presented in table 2.1. The objective of the MILP is to maximize the total expected reward over all agents where $x^m(s, a)$ is the occupation measure of agent m for state action pair (s, a) and $r^m(s, a)$ is the reward for executing action a in state s. Equation 2.9 ensures outgoing flow from a state σ by taking any action is equal to the incoming flow into that state from other states. Equation 2.10 defines the budget constraint for every agent where the cost $\kappa(o, c)$ of obtaining a resource should not exceed the budget $\hat{\kappa}^m(c)$ for an agent. Equation 2.11 constrains the number of allocated resources for any given type, o to be less than the capacity of that resource $\hat{\rho}(o)$. A task can be executed only if there is a positive flow associated with the right state and action. This is enforced in Equation 2.12.

2.3.2 Greedy & Lazy Greedy for TI-Dec-MDP

Kumar *et al.* (2017) provided a greedy algorithm and its lazy greedy extension for TI-Dec-MDP. The greedy algorithm builds the solution set incrementally by adding policy for a different agent at each iteration. The solution starts with an empty set and computes policies for agents based on the marginal value of the policies in the current solution set. At every iteration, the policy with highest marginal value is added to the solution set. The highest marginal value policy is computed by constructing and solving an MDP, given the solution set. This process continues for $|Ag|$ iterations, when every agent is assigned exactly one policy.

Further, the above greedy algorithm is replaced by an efficient lazy greedy extension to improve the scalability of the algorithm with increasing agents.

The lazy greedy algorithm (Minoux, 1978) uses the submodularity property of value/objective function which guarantees that the marginal gain for an agent is always equal to or lower than the previous iteration. Therefore, the marginal gain computation is reduced by maintaining a sorted order of marginal values at each iteration. which is then computed according to the descending order in the next iteration.

2.3.3 RC-DCOP with Structured Resource Constraints

Resource constrained DCOPs (RC-DCOPs) (Bowring et al., 2006; Matsui et al., 2008) provide a general framework for coordinated decision making in distributed networks by modelling resources through the use of virtual variables. However, they cannot handle resources with additional structures associated with them in distributed networks. Thus, to utilize the structure associated with such resources, a dynamic programming optimization based approach or DPOP (explained in details below) was proposed by (Kumar et al., 2009) but with changes to its second phase (UTIL propagation phase) to handle the space of feeder trees. The approach exploits acyclicity and the flow conservation property of distribution networks to minimize the total cost of flow. The authors study the problem in the context of power distribution networks where the solution to the power restoration problem is represented as a collection of feeder trees. The feeder trees describe the power path from each power source that must be acyclic. Every power line must respect its capacity constraint. Finally, the best feeder tree configuration is searched in the space of all possible feeder trees to provide an optimal solution.

Distributed Pseudo-tree Optimization Procedure (or DPOP): A DPOP (Petcu & Faltings, 2005) is a complete, synchronous, inference-based algorithm that uses a depth first search (or DFS) pseudo-tree ordering of the agents. It involves three phases. In the first phase, the agents are ordered as a DFS pseudo-tree. In the second phase, called the UTIL propagation phase, each agent, starting from the leaves of the pseudo-tree, aggregates the costs in its subtree for each value combination of variables in its separator. The aggregated costs are encoded in a UTIL message, which is propagated from children to their parents, up to the root. In the third phase, called the VALUE propagation phase, each agent, starting from the root of the pseudo-tree, selects the optimal value for its variable. The optimal values are calculated based on the UTIL messages received from the agents children and the VALUE message received from its parent. The VALUE messages contain the optimal values of the agents and are propagated from parents to their children, down to the leaves of the pseudo-tree. A DPOP based algorithm is proposed but with changes to its bottom up util propagation phase to handle the space of feeder trees.

Chapter 3

Models

In this section, we introduce the models for dedicated and non-dedicated teams to handle coordinated planning/assignment problem and introduce our terminology and notation, that will be referenced in the upcoming chapters.

3.1 Planning Model for Dedicated Team: TasC-MDP

Task/Resource Constrained Markov Decision Process(TasC-MDP) presents a model to represent problems with dedicated teams (handles transition uncertainty) operating in uncertain environments. Specifically, we build on the TasC-MDP model introduced by Dolgov *et al.* (2006) and it is defined using the following tuple :

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

- $\mathcal{A}g$ 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 two types of dependencies in this work. One type of dependencies are temporal and a temporal dependency represented as $\tau_i \prec \tau_j$, entails that a predecessor-task τ_i should be executed before the successor-task τ_j. A second type of dependencies constrains allocations and 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 $\langle S_i, A_i, P_i, R_i, \rho_i, q_i, \hat{q}_i, \alpha_i^0 \rangle$.
  - 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.

- H is the time horizon for the decision problem.

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

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

$$\sum_{i \in \mathcal{A}g} |\boldsymbol{\delta}_i(\tau)| \leq |\mathcal{C}(\tau)| \qquad \forall \tau \in \Gamma \tag{3.1}$$

$$f(\pi_i, \tau) \leq \boldsymbol{\delta}_i(\tau) \qquad \forall i \in \mathcal{A}g, \forall \tau \in \Gamma \tag{3.2}$$

where π_i is the individual policy of agent i in the joint policy π; $\boldsymbol{\delta}_i$ is the set of tasks/resources allocated to agent i with $\boldsymbol{\delta}_i(\tau)$ 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 (3.1), 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 (3.2). 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.

3.2 Planning Model for Non-dedicated Team: ND-TasC-MDP

We now present a model to represent problems with non-dedicated teams that handles transition uncertainty and uncetainty due to agents leaving the team. Specifically, we build on the TasC-MDP model introduced in section 3.1. We refer to this model as Non Dedicated TasC-MDP (ND-TasC-MDP) and is characterised by the tuple:

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

The key distinction in ND-TasC-MDP with respect to TasC-MDP is the presence of a probability distribution for each agent, Δ_i. It denotes the non-dedication of each agent and is represented as a 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 $\sum_t \Delta_i^t = 1$. Further, with respect to task dependences D, we focus on problems with no task dependencies for purposes of easy exposition. However, all our approaches are easily extendable to cases with task dependencies.

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:

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

$$\sum_{i \in \mathcal{A}g} |\boldsymbol{\delta}_i(\tau)| \leq |\mathcal{C}(\tau)| \qquad \forall \tau \in \Gamma \tag{3.3}$$

$$f(\pi_i, \Delta_i, \tau) \leq \boldsymbol{\delta}_i(\tau) \qquad \forall i \in \mathcal{A}g, \forall \tau \in \Gamma \tag{3.4}$$

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 3.3, which ensures that number of tasks executed is less than the total number of tasks. The individual task accomplishments are modelled using the constraints (3.4). 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.

3.3 Planning Model for Decentralized Non-dedicated Team with Submodular Rewards: Submodular ND-TI-Dec-MDP

We now provide an extension to the Submodular TI-Dec-MDP model (explained in details in section 2.2.3) to consider non-dedicated teams. We refer to this model as Non Dedicated TI-Dec-MDP and it is characterised by the following tuple:

$$\langle \mathcal{A}g, \{\Delta_i\}_{i \in \mathcal{A}g}, S, A, \{P_i\}_{i \in \mathcal{A}g}, R, H, \alpha \rangle$$

The main change to the Submodular TI-Dec-MDP is Δ_i. Δ_i is the 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 $\sum_t \Delta_i^t = 1$. We use the global state S_u to represent the dead state (i.e., the state that agents enter when they move out of the system). The individual agent transition function $P_i^t(s_i'|s_i, a_i)$ is modified to $P_i^t(s_i'|s_i, a_i, \Delta_i)$ and is described as following:

$$P_i^t(s_i'|s_i, a_i, \Delta_i) = P_i^t(s_i'|s_i, a_i) \cdot (1 - \Delta_i^t) \tag{3.5}$$

$$P_i^t(S_u|s_i, a_i, \Delta_i) = \Delta_i^t \tag{3.6}$$

If $\Delta_i^t = 0$, it implies that the agent transitions to the expected state according to its transition probability $P_i^t(s_i'|s_i, a_i)$. Otherwise, if $\Delta_i^t \neq 0$, the transitions depend on the agent's probability of staying in the system (i.e., $1 - \Delta_i^t$). Furthermore, an agent transitions to the dead state from any other state with probability Δ_i^t. Note that once an agent transitions to the dead state S_u, it stays there until the end of horizon (i.e., $P_i^t(S_u|S_u, a_i, \Delta_i) = 1$) irrespective of the action taken. The joint reward function $R(s, a)$ however remains unchanged since the computation of reward

only requires the count of agents present in the joint state s. In addition, there is no reward associated with agents present in S_u and we simply have $R(S_u, a) = 0$. The goal of submodular ND-TI-Dec-MDP is to obtain a joint policy that maximizes the expected reward or value $V^t(s, \pi)$ over all agents with an additional constraint that the agents may leave the team. The joint policy π^* is obtained as

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_s V^H(\pi, \alpha(s), \Delta) \tag{3.7}$$

and the expected value over all agents for the joint state s at timestep t is given as :

$$V^t(s, \pi^*) = R\Big(s, \big\langle \pi_1^t(s_1), \ldots, \pi_{|\mathcal{A}g|}^t(s_{|\mathcal{A}g|}) \big\rangle, S_u\Big) + \sum_{s'} \left[\prod_{i \in \mathcal{A}g} P_i^t(s_i' | s_i, a_i, \Delta_i) \right] \cdot V^{t-1}(s', \pi^*) \tag{3.8}$$

3.4 Assignment Model for Non-dedicated Team

In this section, we present a model to represent multi-agent coordinated assignment problems dealing with non-dedicated teams in distributed networks. This model handles uncertainty due to agents leaving the team and utilizes the additional structure associated with resources to exploit acyclicity and the flow conservation property in these networks. We provide our model in the context of power distribution networks, however, it is extendible to any distributed network dealing with flow of commodity from source to sinks. We first describe the centralised power supply restoration (PSR) problem (for more details of the domain, refer to section 2.1.3) and then describe the multi-region decomposition that results in multi agent PSR.

We view a power distribution network as a graph $G = (V = \mathcal{P} \cup \mathcal{S}, E)$. Vertices represent power sources (CBs) $p_i \in \mathcal{P}$ and sinks (SDs) $s_i \in \mathcal{S}$. Each power source has a finite amount $|p_i|$ of power available. A sink s_i consumes $|s_i|$ units of power. A value v_i, called *sink weight*, is also associated with a sink s_i to denote the relative importance of the sink. Edges represent power lines connecting sinks and power sources. Let L denote the power capacity of a line. For ease of exposition, we assume it is the same for each line. There is a positive $\epsilon > 0$ line loss associated with power flow across a line.

We now describe the multiagent PSR problem. Due to the presence of underlying decomposition (due to de-regulation), we have a region-based decomposition of the underlying graph G. The vertices of the network G are partitioned into $\mathcal{R}$ regions: $V = \cup_{r \in \mathcal{R}} V^r$. Let E^r denote the edge set such that both its vertices lie in the region r. Intuitively, each region r is managed by a different entity. Therefore, each region r represents an *agent* in our multiagent PSR problem. The edges in set $E \setminus \cup_{r \in \mathcal{R}} E^r$ denote the *cut edges* that connect different regions. Power can flow from one region to another region via these cut edges. Analogously, only agents that share cut edges can communicate with each other along such edges, resulting in a *multiagent system* representation of a power network.

Relay Nodes: Our approach to solve the multiagent PSR via message-passing along the cut edges is to view each network region as a *separate* sub-network per agent.

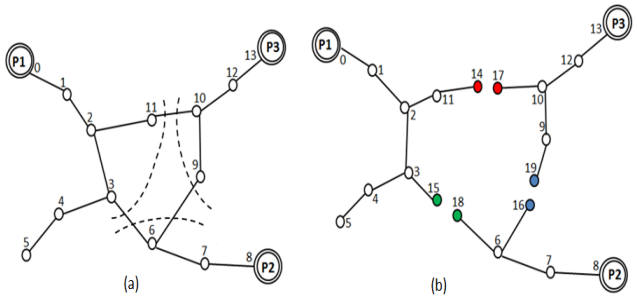


Figure 3.1: Multi-region decomposition of a power network using colored *relay nodes* on the right

Intuitively, cut edges are the *complicating* edges that connect two different regions. It is not clear whose agent's sub-network they belong to. To make separation among regions clear and exploitable by optimization algorithms later, we introduce the notion of *relay nodes*. A pair of relay nodes are defined for each cut edge $(u^r, v^{r'})$ connecting regions r and r'. One relay node c^r belongs to the region r and the second relay node $c^{r'}$ belongs to region r'. We then remove the cut edge $(u^r, v^{r'})$ from the graph G, and create two new *relay* edges (u^r, c^r) and $(c^{r'}, v^{r'})$. Figure 3.1 describes such a process of decomposing the original network G into multiple independent networks G^r, one for each region $r \in \mathcal{R}$. Relay nodes can consume any amount of power that comes in, and also act as power source with infinite supply. Even though we have partitioned the original network G into independent regions, flow conservation constraints for relay nodes still logically connect the whole network together. Intuitively, the total incoming power for a relay node c^r in region r must be equal to the total outgoing power for its paired relay node $c^{r'}$ in region r'.

Chapter 4

Task/Resource Constrained Planning for Dedicated Agent Teams

In delivery of services or goods (Dantzig & Ramser, 1959; Dolgov & Durfee, 2006), tasks have to be allocated to individual vehicles based on uncertain travel times to delivery locations. Also, in disaster rescue scenarios (Velagapudi, Varakantham, Scerri, & Sycara, 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, Wurman, & D'Andrea, 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 travelling 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 [1] 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. The main assumptions of a TasC-MDP problem include the following.

- **Dedication:** The agents of the team are fully dedicated and donot leave the team before the end of planning horizon.
- **Weak coupling:** The agents are weakly-coupled (Meuleau, Hauskrecht, Kim, Peshkin, Kaelbling, Dean, & Boutilier, 1998), i.e., they only interact through

[1] Examples of resources would be roads/paths or tools and examples of tasks would be target surveillance, delivery of items

Algorithm 1 GAPS($TasC-MDP$)

$\tilde{\boldsymbol{\delta}} \leftarrow \mathcal{C}$
$F \leftarrow \emptyset$
repeat
  for all $i \in \mathcal{A}g \setminus F$ **do**
    $\pi_i^* \leftarrow \max_{\pi_i} V_i(\pi_i, \alpha_i^0)$ **s.t.** $f(\pi_i) \leq \tilde{\boldsymbol{\delta}}$
  $\langle i^*, V_{i^*} \rangle \leftarrow \max_{i \in \mathcal{A}g \setminus F} V_i(\pi_i^*, \alpha_i^0) f(\pi_i) \leq \tilde{\boldsymbol{\delta}}$
  $\boldsymbol{\delta}_{i^*} \leftarrow$ GETCONSUMEDRESOURCE$(\pi_{i^*}^*)$
  $\tilde{\boldsymbol{\delta}} \leftarrow \tilde{\boldsymbol{\delta}} \setminus \boldsymbol{\delta}_{i^*}$
  $F \leftarrow F \cup \{i^*\}$
until $\tilde{\boldsymbol{\delta}} = \emptyset$ OR $V_{i^*} = 0$
return $\pi^* \leftarrow \{\pi_i^*\}_{i \in \mathcal{A}g}$

the shared tasks/resources, and once the tasks/resources are allocated, the agents' transitions and rewards are independent.

- **Initial central control over tasks/resources:** At the beginning of the task/resource allocation phase, the tasks/resources are controlled by a single authority and distributed once before the agents start executing their MDPs. There is no reallocation of tasks/resources during the MDP phase.

- **Task/Resource dependencies:** The tasks/resources can either be independent or have dependencies amongst themselves. In problems with *task/resource dependencies*, there can be dependencies such as "if task A is assigned to an agent, task B should also be assigned to the same agent" or "task A has to be done before task B". In *task/resource-independent problems*, all tasks/resources are independently available.

We make the following key contributions to solve TasC-MDPs in this chapter. 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.

4.1 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 $\tilde{\boldsymbol{\delta}}$ 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 $\tilde{\boldsymbol{\delta}}$ (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,[2] 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.

4.2 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 (Dolgov & Durfee, 2006). Table 4.1 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 (4.2) and (4.3) enforce that the total expected number of times state σ_i is exited (LHS of the constraints) equals the

[2] 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.

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: $$-\sum_i \sum_t \sum_{s_i} \sum_{a_i} x_i^t(s_i, a_i) \cdot R_i^t(s_i, a_i) \tag{4.1}$$

Subject to:

$$\sum_{a_i} x_i^{t+1}(\sigma_i, a_i) = \sum_{s_i} \sum_{a_i} x_i^t(s_i, a_i) \cdot P_i^t(s_i, a_i, \sigma_i), \forall \sigma_i, t, i \tag{4.2}$$

$$\sum_{a_i} x_i^0(s_i, a_i) = \alpha_i(s_i), \ \forall s_i, i \tag{4.3}$$

$$\sum_{\tau} \sum_{t} q_i(\tau, z) \cdot \delta_i^t(\tau) \leq \hat{q}_i(z), \ \forall z, \forall i \tag{4.4}$$

$$\sum_{t,i} \delta_i^t(\tau) \ \leq \mathcal{C}(\tau), \ \forall \tau \tag{4.5}$$

$$\frac{1}{X} \sum_{a_i} \rho_i(a_i, \tau) \sum_{s_i} x_i^t(s_i, a_i) \leq \delta_i^t(\tau), \ \forall \tau, t, i \tag{4.6}$$

$$\delta_i^{t+1}(\tau_k) - \sum_{t' \leq t} \delta_i^{t'}(\tau_j) \leq 0, \ \forall (\tau_j \prec \tau_k) \in D, t < H, i \tag{4.7}$$

$$\sum_{t} \delta_i^t(\tau_j) = \sum_{t} \delta_i^t(\tau_k), \ \forall (\tau_j \parallel \tau_k) \in D, t \tag{4.8}$$

$$\delta_i^t(\tau) \leq M \cdot \rho_i(a_i, \tau) \cdot x_i^t(s_i, a_i), \ \forall \tau \in D, s_i, t, i \tag{4.9}$$

$$x_i^t(s_i, a_i) \geq 0, \delta_t^i(\tau) \in \{0, 1\} \tag{4.10}$$

Table 4.1: Optimal MILP for TasC MDPs

expected number of times state σ_i is entered (RHS of the constraints).

- INDIVIDUAL CAPACITY LIMITS: Constraint (4.4) 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 chapter (i.e., the number of resources obtainable by an agent cannot exceed its capacity).
- GLOBAL CAPACITY LIMITS: Constraint (4.5) 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 (4.6) 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) that represents the maximum value of 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 (4.7) to (4.9) represent the resource dependencies. Constraint (4.7) represents the temporal resource dependencies $(\tau_j \prec \tau_k)$ and Constraint (4.8) represents the allocation constraining dependencies $(\tau_j \parallel \tau_k)$ for every agent i. Constraint (4.9) 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 4.1 is an optimal MILP as it provides an exact solution. Unfortunately, given the increase 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 (4.5)) 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 4.2.2), we use GAPS to extract a feasible primal solution from the dual solution.

4.2.1 Maximizing the Lagrangian Dual

We relax or *dualize* coupling Constraint (4.5) for all resource types to yield a significantly easier dual problem. For each resource type, we create dual variables λ_τ, $\forall \tau \in \Gamma$, 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 (4.5) is defined as follows:

$$\mathcal{L}(\{\boldsymbol{V}_i\}, \boldsymbol{\lambda}) = \sum_i -V_i(\pi_i, \alpha_i^0) + \sum_\tau \lambda_\tau \Bigg(\sum_{i,t} \delta_i^t(\tau) - \mathcal{C}(\tau) \Bigg) \tag{4.11}$$

$V_i(\pi_i, \alpha_i^0)$ 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:

$$\sum_i \min_{x_i, \delta_i} \left[-V_i(\pi_i, \alpha_i^0) + \sum_\tau \lambda_\tau \sum_t \delta_i^t(\tau) \right] - \sum_\tau \lambda_\tau \cdot \mathcal{C}(\tau) \tag{4.12}$$

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. (4.11), which is used to update the price variables for the next iteration $n+1$ as follows:

$$\lambda_\tau^{n+1} = \lambda_\tau^n + \gamma^{n+1} \left[\sum_{i,t} \delta_i^{t,n}(\tau) - \mathcal{C}(\tau) \right], \forall \tau \in \Gamma \tag{4.13}$$

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

$$\begin{aligned} \min_{\mathbf{x}} \quad & -\sum_t \sum_s \sum_a x_i^t(s,a) \cdot R_i^t(s,a) + \sum_\tau \lambda_\tau \sum_t \delta_i^t(\tau) \\ \text{s.t.} \quad & \text{Constraints (4.2)} - \text{(4.4)} \wedge \text{(4.6)} - \text{(4.10)} \end{aligned}$$

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:

$$\gamma^{n+1} = \frac{Primal^n - Dual^n}{||\nabla q^n||^2} \tag{4.14}$$

where the dual value $Dual^n$ can be easily obtained from Eq. (4.12) 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.

4.2.2 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.

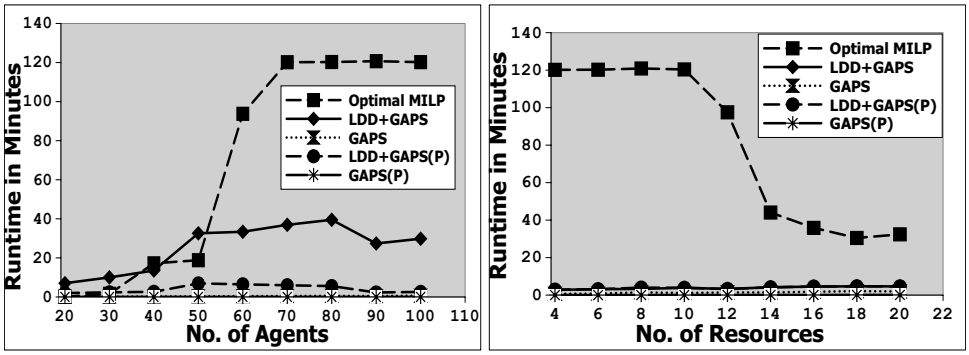
Finally, based on the current primal and dual values, we provide the error in solution quality obtained by Lagrangian dual decomposition to provide posteriori quality guarantees. We use these guarantees in our experiments to make solution quality comparisons.

4.3 Experiments

In this section, we empirically evaluate[3] 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 & 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, Handoko, & Lau, 2014). We compare both our approaches with an optimal MILP by Dolgov and Durfee (Dolgov & 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.

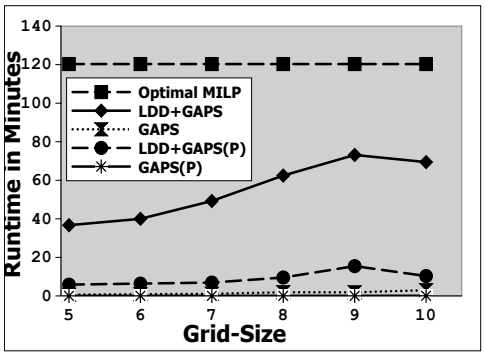
Experimental results are averaged over 15 randomly generated grid maps that randomly place the delivery locations and walls. For runtime comparisons, apart from providing standard runtime for GAPS and LDD+GAPS, we also provide the parallel runtimes for our approaches since they have components that can 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 Op-

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

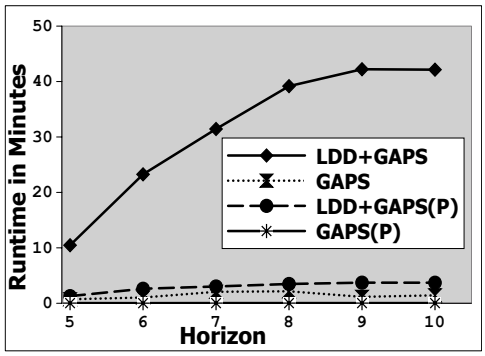


(a) $m = 5, H = 6, C(\tau) = r(5)$

(b) $m = 5, H = 6, |\mathcal{A}g| = 70$



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



(d) $m = 5, C(\tau) = r(5), |\mathcal{A}g| = 70$

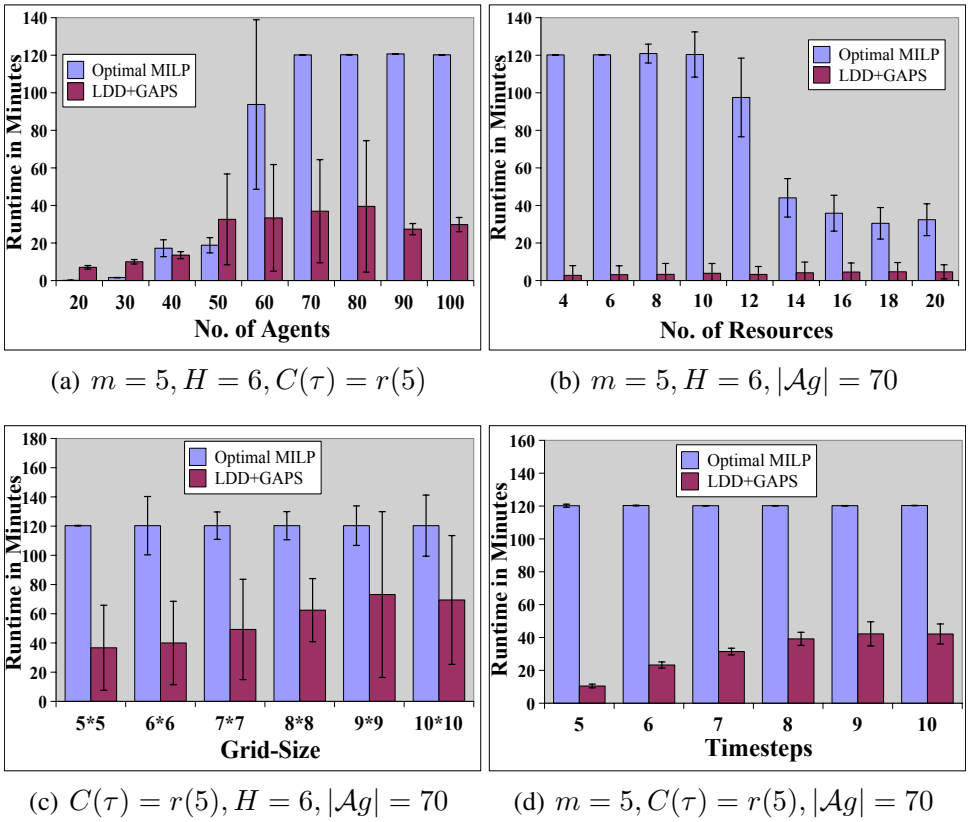
Figure 4.1: Runtime Comparison w.r.t. (a) Agents $|\mathcal{A}g|$; (b) Resources $C(\tau)$; (c) Grid-size $m \times m$; and (d) Horizon H for MADPs

timal 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.

4.3.1 Multi-Agent Delivery Problems (MADP)

We conduct experiments on Multi-Agent Delivery Problem (MADP) (refer section 2.1.1 for more details of domain) using the exact same domain representation as (Dolgov & Durfee, 2006). In our experiments, we use 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 4.1 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



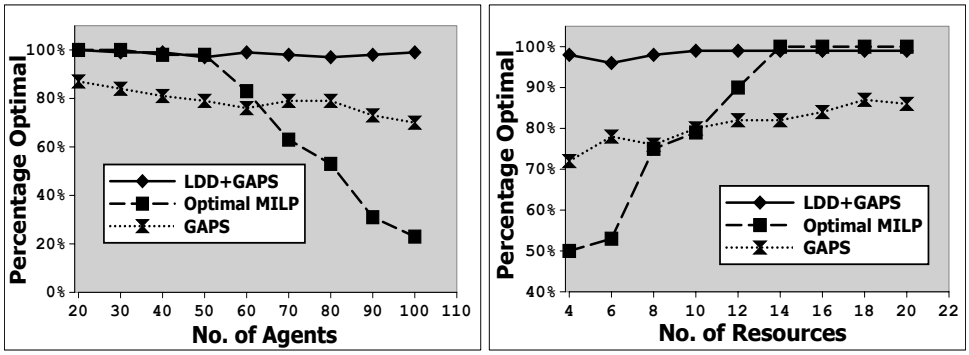
(a) $m = 5, H = 6, C(\tau) = r(5)$ (b) $m = 5, H = 6, |\mathcal{A}g| = 70$

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

Figure 4.2: Error in Runtime for Optimal MILP and LDD+GAPS with varying (a) Agents $|\mathcal{A}g|$; (b) Resources $C(\tau)$; (c) Grid-size m; and (d) Horizon H for MADPs

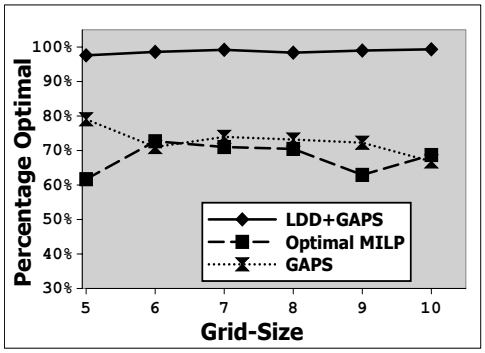
minute for all cases while LDD+GAPS(P) obtained solutions in 2 to 4 minutes. One important observation is that the runtime for Optimal MILP scales exponentially (in cases where it could not finish or provide a feasible solution, it was stopped at the time-limit) with increasing agents, grid-size, and horizon, and decreasing resources. Optimal MILP could not scale beyond 70 agents (and 5 resources) in Figure 4.1(a) but by increasing the number of resources beyond 12 for 70 agents in Figure 4.1(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.

Figure 4.2 shows the sequential runtimes along with the standard deviation for Optimal MILP and LDD+GAPS, having the same settings as Figure 4.1. In Figure 4.2(a), the amount of variation for 60 agents in Optimal MILP is highest because some of the randomly generated problem instances were difficult to solve while others could be solved quickly to optimality. For other problem instances with either less than or greater than 60 agents, the variability was quite less since all random instances for a particular problem setting were either solvable within the cut-off limits or all of them were difficult to solve. Similarly, in Figure 4.2(b) the variation for $C(\tau) = 12$ was high because only few problem instances could be solved to optimality. For LDD+GAPS, the variability increased with increasing

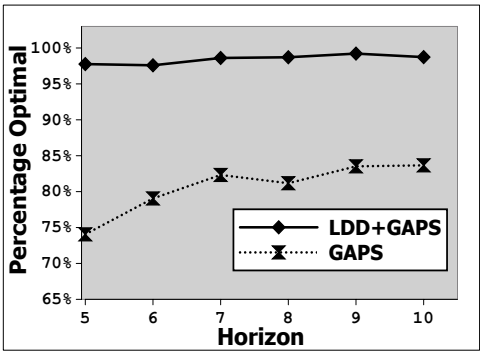


(a) $m = 5, H = 6, C(\tau) = r(5)$

(b) $m = 5, H = 6, |\mathcal{A}g| = 70$



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



(d) $m = 5, C(\tau) = r(5), |\mathcal{A}g| = 70$

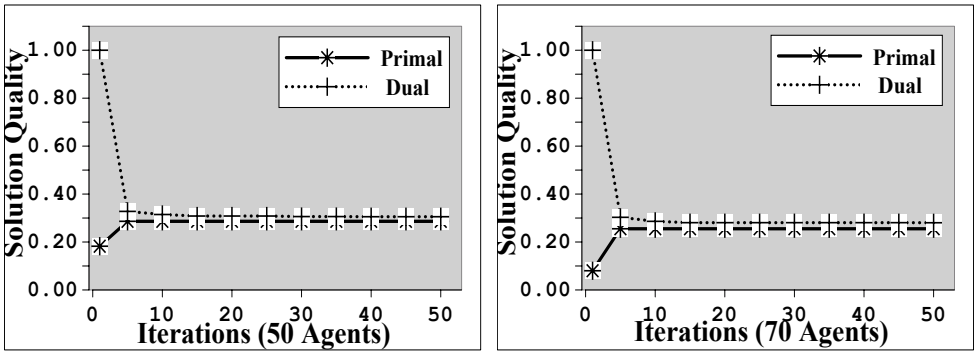
Figure 4.3: Quality Comparison w.r.t. (a) Agents $|\mathcal{A}g|$; (b) Resources $C(\tau)$; (c) Grid-size $m \times m$; and (d) Horizon H for MADPs

agents, grids and horizon while a steady runtime was observed for different values of resources.

Solution Quality: Figure 4.3 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. In addition, Figure 4.4 highlights the anytime performance of LDD+GAPS[4] where we plot the primal and dual values for different values of agents and the results clearly show that our approach is able to provide good primal solutions even in early iterations. 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.

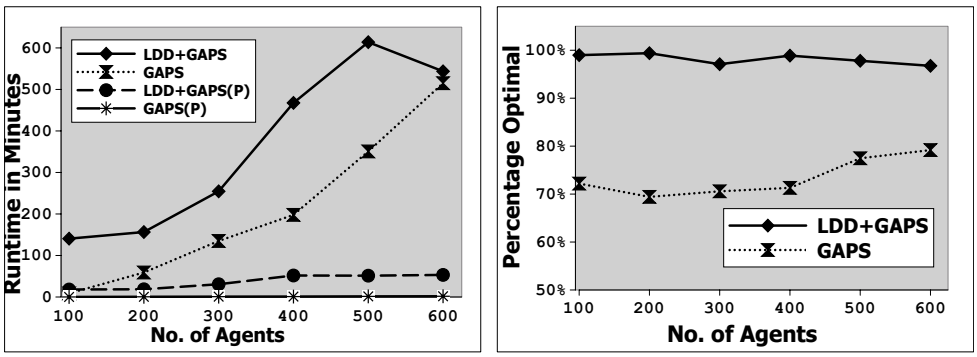
Scalability: To further experiment with scalability, we performed experiments with up to 600 agents on a grid with size $m = 10 \times 10$, horizon $H = 10$ and with resources $C(\tau) = 10\%$ of $|\mathcal{A}g|, \forall \tau \in \Gamma$. Figure 4.5 provides the runtime and quality

[4]We interpret the problem as a maximization problem with dual solution providing an upper bound



(a) $m=5, H=6, C(\tau)=r(5)$ (b) $m=5, H=6, C(\tau)=r(5)$

Figure 4.4: Comparison of Duality Gap in LDD+GAPS with Varying Agents



(a) Runtime Comparison (b) Quality Comparison

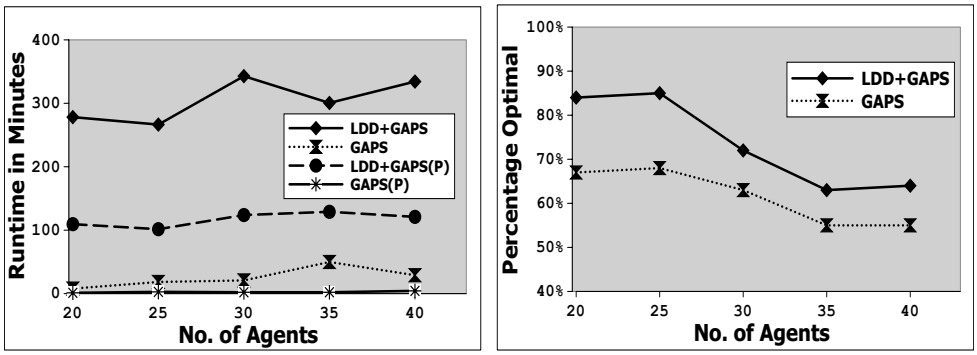
Figure 4.5: Scalability Test: $m=10, H=10, C(\tau)=r(\mathcal{A}g/10)$

comparisons for our approaches on large problems. Figure 4.5(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 4.5(b) shows the quality comparison where LDD+GAPS provided at least 96% of optimality while GAPS could only provide 70% of optimality.

4.3.2 Urban Consolidation Center Problems (UCC)

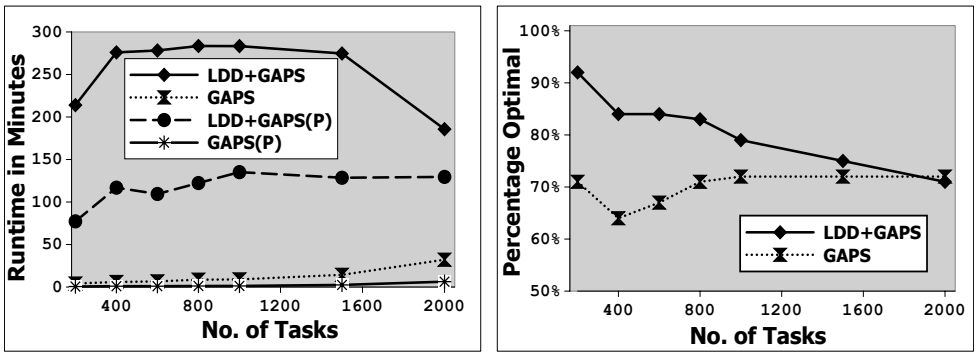
For UCC Problems (refer section 2.1.2 for more details), we use a grid world environment similar to the MADP domain, but with the difference that there can be multiple delivery tasks in a cell that do not require any resources to be completed and may have task dependencies. For ease of exposition, the 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).

Solution Quality and Runtime: Figures 4.6 and 4.7 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



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

Figure 4.6: Varying Agents in UCCs with Task Dependencies



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

Figure 4.7: Varying Tasks in UCCs with Task Dependencies

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.

With the above experimental observations in the MADP and UCC domains, we conclude that the performance of our optimization based decomposition approach, LDD+GAPS is not affected with increasing/decreasing capacity of each task/resource type ($C(\tau)$). However, with increasing count of agents ($|\mathcal{A}g|$) or task/resource types ($|\Gamma|$) after a certain extent, the complexity of TasC-MDP increases, making it difficult to solve the problem optimally and quickly. Hence, scalability of LDD+GAPS degrades with increasing agents and/or task/resource types, although gracefully. Similarly, for our greedy heuristic, GAPS, the solution quality is affected with increasing agents/resource types, but it still provides very quick solutions.

4.4 Summary

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

Chapter 5

Task/Resource Constrained Planning for Non-dedicated Agent Teams

Domains such as disaster rescue, security patrolling, warehouse management, 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 after task allocation. In these domains, a team of agents coordinate plans to achieve a goal while there is transition uncertainty in planning problems of individual agents and 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) in the above mentioned application problems, we refer to these agent teams as non-dedicated agent teams. These non-dedicated agent teams differ from the dedicated agent teams due to the requirement of reconfiguration of the system (i.e., reallocation of tasks/resources to the remaining team members) whenever agents leave the team. Furthermore, there is a need of a central planner throughout the horizon to monitor/control any changes in the state of the system (i.e., agents leaving the team). The agents, however, are still weakly coupled and execute their MDPs independent of each other after allocation (or reallocation) of tasks.

In this chapter, we first provide a general model for the study of non-dedicated agent teams that operate 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 while considering agent exits from the team. Another major contribution involves the formulation of a two stage approach where the state of the team at the end of first stage is input for the second stage to provide a two stage policy for the agents. Finally, we provide an exhaustive evaluation on the performance of our proactive and reactive approaches on benchmark problems.

5.1 Optimization Formulation for a Dedicated Team

For the case where no agent leaves the system (represented as $\Delta_i^H = 1$ for all agents), previous work (Dolgov & Durfee, 2006) and our work in chapter 4 has provided mixed integer formulations to address different types of resource and task interactions. We refer to the mixed integer linear program (MILP) in table 4.1 for the representation of a dedicated agent team. For ease of exposition, in this chapter, we do not focus on task dependencies and consider independent tasks, $D = \emptyset$.

5.2 Approaches

In non dedicated teams, the remaining team members must coordinate over the tasks being left by the agents leaving the team. As indicated earlier, the objective is to maximize the expected utility of a non-dedicated team of co-operative agents performing a set of tasks assigned to them. We provide both reactive (online) and proactive (offline) approaches to facilitate coordination among the remaining agents. In addition, we also provide heuristics that benchmark the performance of our approaches.

5.2.1 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 4.1 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 by solving the MILP of Table 4.1 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.

5.2.2 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 4.1. Specifically, we replace the actual flow with expected flow given "probability of staying back" in the flow preservation constraint of Equation 4.2. That is

Algorithm 2 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(π_{i^*}, δ_i)
4: $\boldsymbol{\delta}_{i^*}^- \leftarrow$ GETDISCARDEDTASKS(π_{i^*}, δ_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 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 4.1 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.

5.2.3 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 4.1. When one or more agents leave the team, the tasks of leaving agents must be assigned to the remaining agents in the system. In this approach, 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 2 provides the pseudocode for ReacT where inputs 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.

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

Minimize: $$-\frac{1}{K}\sum_{k}\sum_{i,s_i,a_i,t} x_i^{t,k}(s_i,a_i)\cdot R_i^{t,k}(s_i,a_i) \tag{5.1}$$

Subject to:

$$\sum_{a_i} x_i^{t+1,k}(\sigma_i,a_i) = \sum_{s_i}\sum_{a_i} x_i^{t,k}(s_i,a_i)\cdot P_i^{t,k}(s_i,a_i,\sigma_i), \forall \sigma_i,t,i,k \tag{5.2}$$

$$\sum_{a_i} x_i^{0,k}(s_i,a_i) = \alpha_i(s_i),\ \forall s_i,i,k \tag{5.3}$$

$$\sum_{t,i} \delta_i^{t,k}(\tau)\ \leq \mathcal{C}(\tau),\ \forall \tau,k \tag{5.4}$$

$$\frac{1}{X}\sum_{a_i}\rho_i(a_i,\tau)\sum_{s_i} x_i^{t,k}(s_i,a_i) \leq \delta_i^{t,k}(\tau),\ \forall \tau,t,i,k \tag{5.5}$$

$$x_i^{t,k}(s_i,a_i) \geq 0, \delta_i^{t,k}(\tau) \in \{0,1\} \tag{5.6}$$

Table 5.1: Optimal MILP for Non Dedicated Team

5.2.4 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 5.1(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:

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

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

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 5.1:

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

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 5.1 to provide an offline allocation of tasks that works well across all K samples using global task allocation variables, d :

$$\delta_i^{t,k}(\tau) = d_i^t(\tau), \quad \forall \tau, t, i, k \tag{5.7}$$

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 5.1 while considering the common allocation constraint of Equation 5.7 is given by:

$$\begin{aligned}\mathcal{L}(\mathbf{x},\boldsymbol{\delta},\mathbf{d}) &= \frac{-1}{K}\sum_{k=1}^{K} f^k(x^k) + \sum_{\tau,t,i,k} \lambda_i^{t,k}(\tau)\Big(d_i^t(\tau) - \delta_i^{t,k}(\tau)\Big) \\ &= \frac{-1}{K}\sum_{k=1}^{K}\Big(f^k(x^k) + \sum_{\tau,t,i} \lambda_i^{t,k}(\tau)\cdot\delta_i^{t,k}(\tau)\Big) + \sum_{\tau,t,i,k} \lambda_i^{t,k}(\tau)\cdot d_i^t(\tau) \end{aligned} \tag{5.8}$$

where $\boldsymbol{\lambda}$ is the dual vector associated with constraints in Equation 5.7. 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:

$$\begin{aligned}&\Lambda_i^t(\tau) = \{\lambda_i^{t,k}(\tau) | \sum_k \lambda_i^{t,k}(\tau) = 0\} \\ &\lambda_i^{t,k}(\tau) \in \Lambda_i^t(\tau), \ \forall \tau, t, i\end{aligned} \tag{5.9}$$

The above condition further simplifies the dual $G(\boldsymbol{\lambda})$, as the last term in Equation 5.8 vanishes leading to the below dual which is separable over K samples:

$$G(\boldsymbol{\lambda}) = \sum_k \min_{\mathbf{x},\boldsymbol{\delta}} \Big(f^k(x^k) + \sum_{\tau,t,i} \lambda_i^{t,k}(\tau)\cdot\delta_i^{t,k}(\tau)\Big) \tag{5.10}$$

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

$$\nabla G\left(\lambda_i^{t,k}(\tau)\right) = \delta_i^{t,k}(\tau) \tag{5.11}$$

where $\delta_i^{t,k}(\tau)$ denotes the solution of inner minimization problem of Eq.(5.10) for sample k which is used to update the price variables as follows:

$$\lambda_{i,n^*}^{t,k}(\tau) = \lambda_{i,n}^{t,k}(\tau) + \gamma_{n^*}\left[\delta_{i,n}^{t,k}(\tau) - \frac{\sum_{k'=1}^{K}\delta_{i,n}^{t,k'}(\tau)}{K}\right], \forall \tau, t, i, k \tag{5.12}$$

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.(5.9).

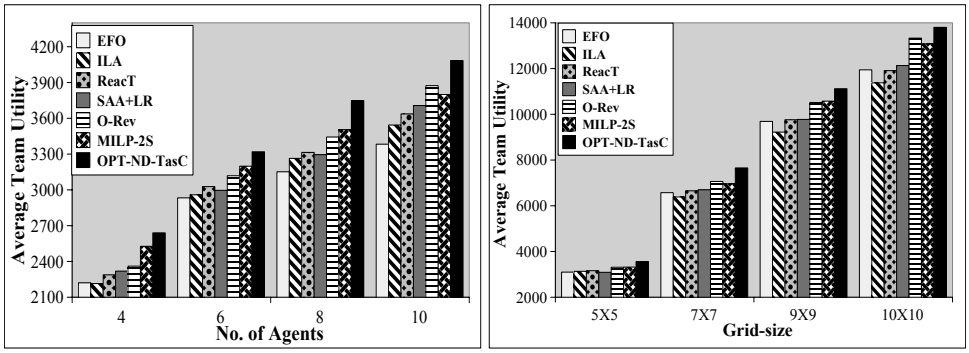
Extraction of Feasible Primal Solution

The dual solution obtained by solving the individual samples may be inconsistent due to violation of the common allocation constraint of Eq.(5.7) 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 5.1) 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 update of the step parameter $\gamma_{n^*} = \frac{Primal^n - Dual^n}{||\nabla g^k||^2}$ in Eq.(5.12) where $Dual^n$ is the dual value for iteration n.

5.2.5 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 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



(a) $m = 5, H = 10, C(\tau) = 20$ (b) $|\mathcal{A}g| = 6, H = 10, C(\tau) = 20$

Figure 5.1: Quality Comparison w.r.t. (a) Agents and (b) Grid-size

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 $x^{t,k}_{i,1}(s_i, a_i)$ and $x^{t,k}_{i,2}(s_i, a_i, o)$ denote the visitation frequency for agent i at time t for sample k with observation o in stages 1 and 2 respectively. $d^t_{i,n}$ and $y^t_{i,n}$ represent the common task allocation and policy assignment variables for the two stages $n \in \{1, 2\}$ over all samples.

The constraints 5.15 to 5.19 are the constraints for stage 1 where equations 5.15 and 5.16 are the flow constraints and equation 5.17 is the task assignment/execution constraint. The equations 5.18 and 5.19 are required to provide a unique policy and task allocation over all samples in the first stage. Similarly, constraints 5.20 to 5.24 are the second stage constraints. For every observation $o \in O$, equations 5.21 and 5.20 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 5.20. Equation 5.22 provides the task assignment constraint. Equations 5.23 and 5.24 provide a unique policy and unique task allocation over all samples for an observation. For every sample, the task assignment constraint 5.14 ensures that a task can be done only in either of the stages.

5.3 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.

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

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

Minimize: $$-\frac{1}{K}\sum_{k,i,s_i,a_i}\Big(\sum_{t\leq t'} x_{i,1}^{t,k}(s_i,a_i)\cdot R_{i,1}^{t,k}(s_i,a_i)+ \sum_{o}\sum_{t>t'} x_{i,2}^{t,k}(s_i,a_i,o)\cdot R_{i,2}^{t,k}(s_i,a_i)\Big) \tag{5.13}$$

Subject to:

$$\sum_i\left(\sum_{t\leq t'}\delta_{i,1}^{t,k}(\tau)+\sum_o\sum_{t>t'}\delta_{i,2}^{t,k}(\tau,o)\right)\leq \mathcal{C}(\tau),\forall\tau,k \tag{5.14}$$

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

$$\sum_{a_i} x_{i,1}^{0,k}(s_i,a_i)=\alpha_{i,1}(s_i),\forall s_i \tag{5.15}$$

$$\sum_{a_i} x_{i,1}^{t+1,k}(\sigma_i,a_i)=\sum_{s_i,a_i} x_{i,1}^{t,k}(s_i,a_i)\cdot P_i^{t,k}(s_i,a_i,\sigma_i),\forall\sigma_i \tag{5.16}$$

$$\frac{1}{X}\sum_{a_i}\rho_i(a_i,\tau)\sum_{s_i} x_{i,1}^{t,k}(s_i,a_i)\leq\delta_{i,1}^{t,k}(\tau),\forall\tau \tag{5.17}$$

$$\sum_{a} x_{i,1}^{t,k}(s_i,a_i)=\sum_{a} y_{i,1}^{t}(s_i,a_i),\forall s_i \tag{5.18}$$

$$\delta_{i,1}^{t,k}(\tau)=d_{i,1}^{t}(\tau),\forall\tau \tag{5.19}$$

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

$$\sum_{a_i} x_{i,2}^{t'+1,k}(s_i,a_i,o)=\sum_{s_i,a_i} x_{i,1}^{t',k}(s_i,a_i)\cdot P_i^{t,k}(s_i,a_i,\sigma_i) \tag{5.20}$$

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

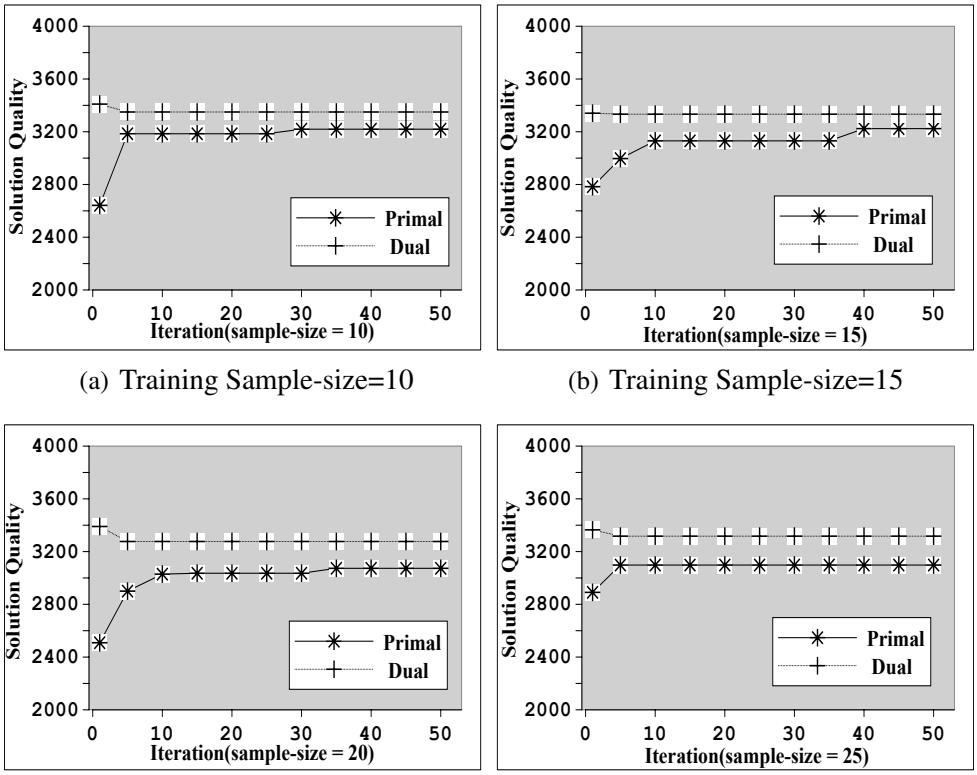
$$\sum_{a_i} x_{i,2}^{t+1,k}(\sigma_i,a_i,o)=\sum_{s_i,a_i} x_{i,2}^{t,k}(s_i,a_i,o)\cdot P_i^{t,k}(s_i,a_i,\sigma_i),\forall\sigma_i \tag{5.21}$$

$$\frac{1}{X}\sum_{a_i}\rho_i(a_i,\tau)\sum_{s_i} x_{i,2}^{t,k}(s_i,a_i,o)\leq\delta_{i,2}^{t,k}(\tau,o),\forall\tau \tag{5.22}$$

$$\sum_{a} x_{i,2}^{t,k}(s_i,a_i,o)=\sum_{a} y_{i,2}^{t}(s_i,a_i,o),\forall s_i \tag{5.23}$$

$$\delta_{i,2}^{t,k}(\tau,o)=d_{i,2}^{t}(\tau,o),\forall\tau \tag{5.24}$$

Table 5.2: Two-Stage MILP for Non Dedicated Team



(a) Training Sample-size=10

(b) Training Sample-size=15

(c) Training Sample-size=20

(d) Training Sample-size=25

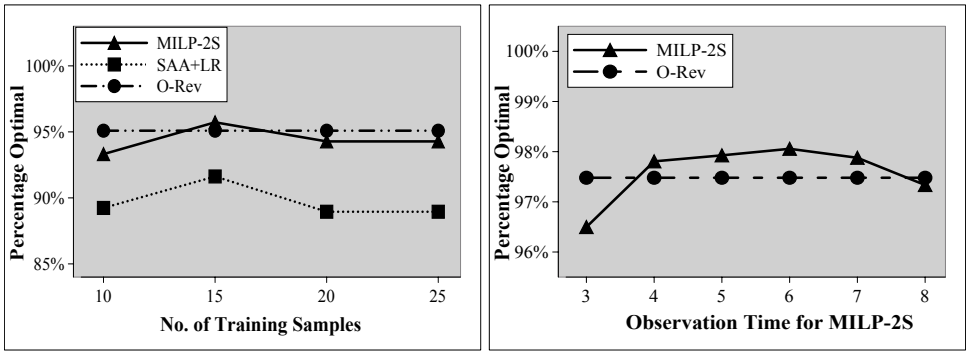
Figure 5.2: Quality Comparison w.r.t. Samples: $m = 5, |\mathcal{A}g| = 8, H = 10, C(\tau) = 20$

5.3.1 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 the agent availability samples from a probability distribution Δ_i for every agent. For comparison of all the approaches, we generate 1500 samples of agent availability, and divide it into training set of 1000 samples and a testing set of 500 samples. To obtain a fair comparison over all online and offline approaches, we compare the solution quality and runtime on the same test set. We simulate the policies obtained by different approaches on the test set.

We compare the following approaches in this section:



(a) $m = 5, |\mathcal{A}g| = 8, C(\tau) = 20$ (b) $m = 5, |\mathcal{A}g| = 6, C(\tau) = 20$

Figure 5.3: Optimality w.r.t. (a) Samples (b) Observation Time

- Ignore the leaving agent, ILA - Section 5.2.1.
- Online revamp, O-Rev - Section 5.2.1.
- Expected Flow Optimization, EFO - Section 5.2.2.
- Reactive assignment of tasks (ReacT) - Section 5.2.3.
- Sample Average Approximation based Lagrangian Relaxation (SAA+LR) - Section 5.2.4. Due to repetition in samples, we find the frequency of each generated sample in the training set and assign frequency-specific weights to each training sample and choose 10 best samples for calculating joint policy with SAA+LR. The number of training samples is fixed to 10 best samples unless mentioned specifically.
- OPT-ND-TasC - This corresponds to solving all the test samples offline. This is not really an approach that can be employed, but serves as a benchmark on the best possible performance achievable.
- Two Stage MILP for Non-Dedicated Team (MILP-2S) - Section 5.2.5. 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.

5.3.2 Results

Solution Quality: Figure 5.1 compares the solution quality for all the approaches discussed in section 6.2. We compare the average team utility in Figure 5.1(a) as 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 5.1(b), we compare for different grid-size m for a fixed number of agents $|\mathcal{A}g| = 6$, tasks $C(\tau) = 20$ and horizon $H = 10$. Here are the key observations:

- 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.
- 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.

- SAA+LR typically provides better performance than ReacT, but in some cases (e.g., 6 agents in figure 5.1(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.
- 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.

Solution Runtime: 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 approaches, we provide a maximum of three hours for training.

Quality of Bounds: Figure 5.2 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.

Optimality Comparison: Figure 5.3 compares the optimality of solution for any approach (computed as U(approach)$*100/$U(OPT-ND-TasC) where U(approach) represents the utility obtained using specified approach) on the test set. Figure 5.3(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 percentage of optimality improves to a certain extent with increasing training samples for both MILP-2S and SAA+LR after which the optimality may not improve significantly and approximately remains constant with increasing samples. 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 reducing the performance. Similarly, in figure 5.3(b), we vary the observation time for MILP-2S from $t' = 3$ to $t' = 8$ for a horizon $H = 10$ in training to obtain a two stage offline policy for every observation time t'. We simulate each of the policies on same test set and 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.

5.4 Summary

In this chapter, we addressed the problem of task allocation for a team of non dedicated agents operating in uncertain environments. The provided model is generic enough to be applied to different domains involving task allocation problem. Our proactive approaches provide efficient one stage and 2 stage strategies that work well across different feasible scenarios of agents leaving the team. Our extensive experiments on benchmark problems demonstrate that MILP-2S provides best performance, even for large problems.

Chapter 6

Decentralized Planning for Non-dedicated Agent Teams with Submodular Rewards

Decentralized planning under uncertainty for non-dedicated agent teams is important in a wide variety of problems such as disaster rescue, security patrolling, sensor networks, etc., where individual agents work in a decentralized, yet co-operative manner and are tied together through a global reward function. In problem domains 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. In target tracking by a team of sensors (Nair et al., 2005; Kumar & Zilberstein, 2011; Chapman & Varakantham, 2014), the sensors are used to monitor some spatial phenomenon. Also, in disaster rescue scenarios (Varakantham, Adulyasak, & Jaillet, 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, D'Andrea, & Mountz, 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). These domains have the following common characteristics: (a) A team of agents (sensors, ambulances, fire-trucks, etc.) coordinate plans to achieve a goal; (b) There is transition uncertainty in planning problems of individual agents, either due to travelling on roads (due to traffic) or due to physical constraints (sensors, robots, etc.) (c) The agent team is a non-dedicated agent team. For example, in the case of patrolling, (e.g., coast guard boats, traffic police) can be forced to leave their assignment to attend to an accident or incident (e.g., incursion, smuggling, road accident); and most importantly (d) The agents are independent and collaborate through a global reward (save victims, prevent attacks, etc.) defined as a submodular function. Furthermore, a partially centralized control over agents is required due to reallocation of tasks, and therefore, change in agent polices of the remaining agents after few agents have left (i.e., whenever the system configuration changes).

In this chapter, we are interested in application problems with above mentioned characteristics. To that end, we provide a general model to represent the class of

problems consisting of independently collaborating non-dedicated agents. A key contribution of our work lies in establishing connections between non-dedicated agent teams and submodularity. We show that with monotone submodular reward functions subject to the matroid constraint, greedy solutions computed at every decision epoch are still submodular with fewer number of agents and provide an a priori guarantee of at least 50% from the optimal and much better posterior guarantees. Another main contribution includes our two greedy approaches to efficiently deal with agent exits before the end of horizon. In our first approach, we exploit lazy greedy to obtain a unique offline policy for every agent irrespective of the agent exits from the team. The second approach is an offline-online approach where the offline phase creates a fixed number of joint policies to be used in the online phase. Finally, our experiments demonstrate the improved performance of our approaches on benchmark problems from literature.

6.1 Submodular ND-TI-Dec-MDP

The model for Submodular ND-TI-Dec-MDP is explained in details in chapter 3, section 3.3. In this section, we describe the important properties of ND-TI-Dec-MDPs with a joint reward function that is monotonically increasing and submodular. Let us first consider the case of a dedicated team where no agent leaves the system (represented as $\Delta_i^H = 1$ for all agents). In this case, the state of the system is fixed (i.e., no agents leaving) and already known to the decision maker, and hence, the policy of every agent can be determined in advance. However, in a non-dedicated agent team, agents may leave the team midway requiring reconfiguration of the remaining agent policies. The timestep at which an agent leaves the team is referred to as observation timestep, t' and the set of agents leaving the system at t' represent the observation ψ. All the agents that have left until t' constitute the observation set $\psi_{t'}$. The joint policy for a ND-TI-Dec-MDP is a concatenated policy which is formally defined for one observation timestep as following.

Definition 3. *Policy Concatenation: Let π_{ψ_0} be the joint policy over all agents until the first observation at time t' and $\pi_{\psi_{t'}}$ be the joint policy with observation set $\psi_{t'}$. The concatenated policy $\hat{\pi}$ is represented as:*

$$\hat{\pi} = \left[\pi_{\psi_0}\right]_{t=0}^{t<t'} + \left[\pi_{\psi_{t'}}\right]_{t=t'}^{t=H}$$

Proposition 1. *(Kumar et al., 2017): For a TI-Dec-MDP, $V^H(s,\pi)$ is monotonically increasing and submodular if the joint reward, R is monotonically increasing and submodular.*

At $t = 0$, ND-TI-Dec-MDP is similar to TI-Dec-MDP and is solved for $|\mathcal{A}g|$ agents and H timesteps. The value function, $V^H(s,\pi)$ is a monotone and submodular being the case of dedicated agent team. Similarly, for every observation timestep t', ND-TI-Dec-MDP is solved as a new TI-Dec-MDP problem with $\mathcal{A}g \setminus \psi_{t'}$ agents and $H - t'$ timesteps where $\psi_{t'}$ represents the set of agents that have left until t'. The value function, $V^{H-t'}(s,\pi)$ at t' is also monotonically increasing and submodular. Hence, for a single observation $\psi_{t'}$, the joint policy comprises of two components

(as per definition 3) where the second component is guaranteed to be submodular but not the first component. This is because submodularity of the value function $V^H(\pi)$ holds for $[t]_0^H$ but for ND-TI-Dec-MDP, we consider only timesteps $[t]_0^{t'}$ for the first component. Hence, the value function $V^H(\hat{\pi})$ for ND-TI-Dec-MDP is not guaranteed to be submodular for $\hat{\pi}$, however, it is submodular for every TI-Dec-MDP sub-problem.

The goal in ND-TI-Dec-MDPs is to maximize the expected value by obtaining a correct joint policy (i.e., exactly one policy per agent). Formally, the goal is to maximize $V^H(\pi)$ for every individual TI-Dec-MDP problem given the partition matroid $\Gamma = (\Pi, I)$ where $I = \{X \subseteq \Pi : |X \cap \Pi_i| = 1\}$. Intuitively, the partition matroid enforces that we can only have one policy for each agent.

Proposition 2. *(Fisher, Nemhauser, & Wolsey, 1978): Greedy algorithm for maximizing a monotone submodular function subject to a partition matroid yields solutions that are at least 50% of the optimal solution.*

For a non-dedicated agent team, the a priori bounds for every TI-Dec-MDP sub-problem at any t' is guaranteed to be at least 50% of optimal in the worst case. However, these bounds are quite loose since the solution provided by greedy is much better in most cases. Therefore, we compute online bounds by adding the marginal value of the best policy for every agent in the solution set to provide a tighter upper bound on the optimum. The online bound for a monotonically increasing and submodular value function is represented as below:

Proposition 3. *(Kumar et al., 2017): For any joint policy, π:*

$$V(\pi^*) \leq V(\pi) + \sum_{i \in Ag} \delta_i(\pi)$$

where $\quad \delta_i(\pi) = \max_{\pi_i \in \Pi_i} V(\pi \cup \pi_i) - V(\pi)$

Here, π^* is the optimal joint policy with optimal individual policies for every agent. For any joint policy π, we get an upper bound on the value of the optimal policy by adding the individual policies, π_i that yield best marginal values for each agent. In the context of ND-TI-Dec-MDP, at every observation timestep t', we solve a new TI-Dec-MDP problem with $\mathcal{A}g \setminus \psi_{t'}$ agents and $H - t'$ timesteps where any policy $\pi_{\psi_{t'}}$ provides an upper bound on the optimal policy $\pi^*_{\psi_{t'}}$. However, any concatenated policy $\hat{\pi}$ is not guaranteed to provide an upper bound on the optimal concatenated policy $\hat{\pi}^*$ since submodularity may not hold for π_{ψ_0}. We still compute the online bound for the concatenated policy as following.

$$V(\hat{\pi}^*) \leq \left[V(\pi_{\psi_0}) + \sum_{i \in \mathcal{A}g} \delta_i(\pi_{\psi_0}) \right]_{t=0}^{t<t'} + \left[V(\pi_{\psi_{t'}}) + \sum_{i \in \mathcal{A}g \setminus \psi_{t'}} \delta_i(\pi_{\psi_{t'}}) \right]_{t=t'}^{t=H} \tag{6.1}$$

where $\delta_i(\pi_{\psi_t}) = \max_{\pi_i \in \Pi_i} V(\pi_{\psi_t} \cup \pi_i) - V(\pi_{\psi_t}),\ t \in \{0, t'\}$

The expression in the first square bracket bounds the value of the optimal concatenated policy $V^H(\hat{\pi}^*)$ from $t = 0$ to $t \leq t'$ for the policy π_{ψ_0} (however, it is not a guaranteed online bound), while the second expression provides a guaranteed online bound on the value of the optimal concatenated policy from $t \geq t'$ to $t = H$. For our experiments, we compute online bounds for ND-TI-Dec-MDP using Equation 6.1.

Algorithm 3 ND-GREEDY($\mathcal{A}g, S, A, P, R, H - t', \alpha, \psi$)

1: $Z \leftarrow \emptyset$
2: $\pi_i^* \leftarrow \emptyset, \forall i \in \mathcal{A}g \setminus \psi$
3: **repeat**
4: **for all** $i \in \mathcal{A}g \setminus \{\psi \cup Z\}$ **do**
5: $\pi_i^* \leftarrow \max_{\pi_i} V_i(\pi_i, \alpha_i^{t'} | \pi_Z^*)$
6: $\langle i^*, V_{i^*} \rangle \leftarrow \max_{i \in \mathcal{A}g \setminus Z} V_i(\pi_i^*, \alpha_i^{t'} | \pi_Z^*)$
7: $Z \leftarrow Z \cup \{i^*\}$
8: **until** $\mathcal{A}g \setminus \{\psi \cup Z\} = \emptyset$
9: **return** $\pi^* \leftarrow \{\pi_i^*\}_{i \in \mathcal{A}g \setminus \psi}$

6.2 Approaches

In non dedicated teams, the remaining team members must coordinate to deal with agents leaving the team. As indicated earlier, the objective is to maximize the expected utility of a non-dedicated team of co-operative and decentralized agents performing a collective task assigned to them. We provide an offline and an offline-online approach for solving the decentralized planning problem for a non-dedicated agent team. Due to the inability of Mixed integer linear program (MILP) in computing the joint reward and joint policy for decentralized settings, we provide a lazy greedy extension for benchmark heuristics for non-dedicated agent teams provided in section 5.2 to provide good bounds on the solution quality of ND-TI-Dec-MDPs.

6.2.1 Lazy Greedy

For dedicated agent teams, greedy has been well explored in the context of Dec-MDPs (Shieh et al., 2014; Agrawal, Varakantham, & Yeoh, 2016; Kumar et al., 2017) while for non-dedicated agent teams, it has been explored only in centralized settings (Agrawal & Varakantham, 2017). Therefore, we extend the previous work by (Kumar et al., 2017) to provide a lazy greedy extension for non-dedicated teams in decentralized settings. Algorithm 3 provides the pseudocode for a non-dedicated greedy algorithm that is solved at every observation timestep, t' where $|\psi_{t'}|$ agents leave the team and $H - t'$ timesteps are remaining. The algorithm is initially invoked at the starting timestep (i.e., $t = t' = 0$ and $\psi_{t'=0} = \emptyset$) after which it is invoked only for timesteps where $\psi_{t'} \neq \emptyset$. It must be noted that at the starting timestep (i.e., t=0), ND-Greedy is exactly similar to the Greedy algorithm for dedicated team (Kumar et al., 2017) since $\psi_{t'=0} = \emptyset$ as all agents are available. ND-Greedy builds the solution set by incrementally adding a policy for every agent that has not been assigned a policy. Initially, we start with an empty solution set Z (line 1). At every iteration, for each agent in the set of remaining agents, $\mathcal{A}g \setminus \psi_{t'}$ that has not been assigned a policy (line 4), we compute a policy with the highest marginal value given the current solution set (line 5) by constructing and solving an MDP (similar to the TI-Dec-MDP. Among those highest marginal value policies, we choose the one with the highest value and add it to the solution set (lines 6-7). This process is repeated until all $\mathcal{A}g \setminus \psi_{t'}$ agents have been assigned a policy to collectively provide the joint policy π^* (Every agent is assigned exactly one policy with the help of partition ma-

troid constraint). Finally, the agents in Z are present in decreasing order of their marginal values. We refer this solution set Z as **selection order** of the agents.

ND-Greedy evaluates the marginal value for all the agents at every iteration, thereby affecting the scalability of the algorithm with increasing agents. Interestingly, submodularity of the value function $V^H()$ can be exploited to implement an accelerated version of classical greedy algorithm, otherwise known as Lazy Greedy (Minoux, 1978). Instead of computing the marginal gain for all agents, lazy greedy allows a lazy evaluation of marginal benefits by storing the upper bounds $\mu(i)$ on the marginal gain for all agents $i \in \mathcal{A}g$ sorted in descending order. This reduces the marginal gain computation as the submodularity of value/objective function guarantees that the marginal gain for an agent is always equal to or lower than the previous iteration. Intuitively, for each iteration, lazy greedy evaluates the agent on the top of the list, say i, and updates its upper bound, $\mu(i)$. If $\mu(i) \geq \mu(i'), \forall i' \neq i$, submodularity guarantees that agent i has the highest marginal gain. Therefore, lazy greedy leads to significant reduction in running times compared to the classical greedy.

Why is the new policy recomputation needed: The recomputation of the new joint policy over all agents is necessary at every observation timestep t' because the contribution of rewards by agents at every timestep may vary. This means that an agent may have higher rewards at earlier timesteps compared to later timesteps. In security games, if the agents continue with their initial policies, the coverage of important targets may be missed, making the system vulnerable to attacks. This creates an urgency for recomputation of the policy of agents. Therefore, lazy greedy is used to get a new selection order for agents by considering the reward contribution from the current timestep to the end of planning horizon.

For example, let $[A_2(555), A_3(545), A_1(500), A_4(490)]$ be the selection order of agents at $t = 0$ with their reward values specified alongside. Let a_1 leave the system at $t = 1$. The total value for agents at $t = 1$ could be $[A_2(500), A_3(505), A_4(490)]$ on recomputation of reward for the remaining agents which creates a change in order of selection of agents. This change in order of selection is because the contribution at $t = 0$ dominated the contribution over remaining timesteps for agents A_2 and A_3. Hence, the change in order contributes to the change in marginal gain, and therefore, agents must rearrange their policies to adapt to the change in system.

6.2.2 Benchmarking Heuristics

The existing benchmark heuristics for non dedicated agent teams in Chapter 5 are centralized approaches and incapable of computing joint policy and joint reward for the agent team. Hence, we provide a lazy greedy extension for the existing benchmarks to be able to solve ND-TI-Dec-MDPs and provide bounds on the solution quality.

Ignore the leaving agent, Dec-ILA: We start with our lazy greedy solution for dedicated team. Whenever agents leave the team, remaining agents still continue with their existing policies, but since the reward for the system is a joint reward over agents, we recompute the joint reward for the remaining agents. This provides

a good lower bound on solution quality that has to be achieved. For example, in security games domain, ignoring the targets covered by leaving defender agent is not the best choice since the leaving agent may be protecting a target of high importance. Hence, it is important for the remaining agents to modify their policies to provide an improved coverage to the targets that would become vulnerable to attacks. Similarly, in sensor domain, if a sensor is spoilt, the sensors in the vicinity should be be able to change their policy and sense the target locations assigned to the damaged sensor for better observation of any spatial phenomenon.

Offline Optimal, Dec-OPT: This heuristic assumes that the sample information (details of agents leaving the system) is received beforehand. A mixed integer program provides an optimal solution, but is not a suitable approach for finding the joint policy and the joint reward computation for a decentralized team of heterogeneous agents. Hence, we use lazy greedy for finding the agent policies where the agents are selected sequentially in the decreasing order of their values. Since the agents leaving the system have a shorter timespan compared to non-leaving agents, the marginal gain for such agents will be lowest. Hence, non-leaving agents are provided least preference in the selection process by greedy. Although not an exactly optimal approach, this heuristic provides a good upper bound on the solution quality.

Online Revamp, Dec-O-Rev: Similar to Dec-ILA, for this heuristic, we start with the initial lazy greedy solution until one or more agents leave the system. At the observation timestep t', the problem is solved again for the remaining agents $\mathcal{A}g \setminus \psi_{t'}$ and remaining timesteps $H - t'$. The starting distribution of the remaining agents is recomputed at t' and is input to the lazy greedy algorithm along with the information of leaving agents, $\psi_{t'}$. The new joint policy obtained for the remaining agents is executed by the agent team until there is a change in the system (i.e., an agent leaves the system). Dec-O-Rev provides a good upper bound on the desired performance for our proposed approaches but suffers from some limitations. Although the running time reduction due to lazy greedy is significant compared to classical greedy, the total number of function evaluations with lazy greedy cannot be predicted beforehand to provide the exact running cost. This makes the complete recomputation of selection order at observation timesteps time consuming and difficult to be evaluated on the fly. Secondly, if there is a requirement of recomputation at every timestep t, revamp would become infeasible since at least $\mathcal{A}g \setminus \psi_t$ rounds of sequential computation for agents will be required.

6.2.3 Offline-Greedy Approach

Offline-Greedy is a sampling-based approach that computes an offline selection order, O and a single joint policy π^* over multiple scenarios of agent availability. Since it is impossible to consider all the samples of agent availability on larger problems, we choose a smaller training set for the joint policy computation. The sample set is represented as ξ and has $|K|$ samples. Due to repetition of samples, we assign frequency-specific weights $W^k, \forall k \in K$ and select 20 best samples in decreasing order of weights. Every sample of agent availability, ξ^k is generated by

Algorithm 4 OFFLINE-GREEDY $(\xi, \mathcal{A}g, W)$

1: $Z \leftarrow \emptyset, O \leftarrow \emptyset$
2: $V_i \leftarrow 0, \forall i \in \mathcal{A}g$
3: **for all** $\xi^k \in \xi$ **do**
4: $\quad V^k \leftarrow$ Dec-OPT$(\mathcal{A}g, S, A, P, R, H, \alpha, \xi^k)$
5: $\quad V_i \leftarrow V_i + W^k \cdot V_i^k$
6: **for all** $i \in \mathcal{A}g$ **do**
7: $\quad V_{i^*} \leftarrow \max_{i \in \mathcal{A}g \setminus O} V_i$
8: $\quad O \leftarrow O \cup i^*$
9: **for all** $o \in O$ **do**
10: $\quad \langle \pi_o^*, V_o^* \rangle \leftarrow V_o(\pi_o^*, \alpha_o^0 | \pi_Z^*)$
11: $\quad Z \leftarrow Z \cup \{o\}$
12: $\pi^* \leftarrow \{\pi_o^*\}_{o \in O}$
13: **return** $\langle \pi^*, O \rangle$

sampling from a biased coin with probability p_i independently for every agent i. At every timestep t, the coin is tossed to decide whether agent i leaves or stays in the team depending on the value of associated probability in Δ_i. Hence, for every sample ξ^k, we know the available horizon $\xi^k(i)$ for every agent i.

Algorithm 4 provides the pseudocode for Offline-Greedy with the training set ξ, the agent set $\mathcal{A}g$ and the vector of frequency weights over all samples W as inputs. The agent selection set, Z and the selection order O are initialized as empty sets and the total value of every agent over all samples V_i is set to 0 (line 1-2). For every sample ξ^k in the training set, the available horizon of every agent is already known, and therefore, we use Dec-OPT heuristic to obtain the total value, V^k for every $\xi^k \in \xi$ (line 4). The total value for every agent V_i is computed as the weighted sum of values over the sample set (i.e., $W^k \cdot V_i^k$) (line 5). The selection order O is computed by sorting the agents in decreasing order of their values V_i (line 6-9) such that the agents with higher probability of staying in the system are added before the agents with higher probability of leaving. For all the agents in the selection order, highest marginal value policy for an agent given the current solution set (line 10) is computed by constructing and solving an MDP (similar to TI-Dec-MDP) and the computed agent is then added to the solution set (line 11). Finally, we return the best selection order O and the offline joint policy π^* over all agents and all training samples.

For every test sample, the agents are assigned their individual policies from the offline joint policy π^*. However, irrespective of the observations obtained at different observation timesteps, the agents continue with their pre-assigned policies while the joint reward is recomputed for the remaining agents. This approach saves the online recomputation of policy at observation timesteps but with a compromise in the solution quality.

6.2.4 Offline-Online Approach

In this section, we present our Offline-Online algorithm which is a randomized greedy algorithm with an offline and an online phase. The offline phase focuses on

Algorithm 5 OFFLINE-ONLINE ($\mathcal{A}g, N$)

for all $n \in N$ **do**
  $Z \leftarrow \emptyset$
  $\pi_i^n \leftarrow \emptyset, \forall i \in \mathcal{A}g$
  repeat
    $r_i \leftarrow Random(\mathcal{A}g \setminus Z)$
    $\pi_{r_i}^n \leftarrow V_{r_i}^n(\pi_{r_i}, \alpha_{r_i}^0 | \pi_Z^n)$
    $Z \leftarrow Z \cup \{r_i\}$
  until $\mathcal{A}g \setminus Z = \emptyset$
  $\pi^n \leftarrow \{\pi_i^n\}_{i \in \mathcal{A}g}$
$\Pi \leftarrow \Pi \cup \{\pi^n\}$
return Π

the generation of multiple agent(s) selection orders to handle the different possibilities of scenarios, while the online phase focuses on choosing the best selection order for remaining agents depending on the current observation (availability of agents). We note that having multiple selection orders is better than having one fixed selection order for all scenarios (as present in Offline-Greedy) because the total value of a selection order can change at different observation timesteps due to the dominance of rewards in previous timesteps (explained in details in section 6.2.1). We generate a fixed number of selection orders for the agent set since the total number of orderings possible with $|\mathcal{A}g|$ agents is $|\mathcal{A}g|!$ orders which is difficult to maintain with increasing agents. At every decision stage, we choose the best/closest selection order such that the position of the leaving agent is towards the end of the selection order, thereby, avoiding the recomputations. The time complexity of the offline phase is linear in the number of agents (or $O(|\mathcal{A}g|)$) while it takes constant time for the online phase. The main difference with respect to lazy greedy (used in all the above approaches) is that instead of choosing the agent with highest marginal gain at every iteration, we randomly pick an agent and add it to the selection set. However, due to the joint reward computation and the presence of submodular rewards, the total utility always improves with addition of agents iteratively.

Algorithm 5 shows the offline phase of Offline-Online algorithm where the input to the algorithm is the agent count $|\mathcal{A}g|$, and the number of selection orders to be generated (N). For computing every order n, we start with an empty agent selection set Z and add one agent at a time by randomly selecting agents from the set of remaining agents $\mathcal{A}g \setminus Z$. The policy and value of every agent is obtained by solving an MDP and is stored in π^n. Finally, we return Π that represents the set of policies for all the N selection orders. The online phase of our algorithm does not require any computation and only reacts to a situation by choosing the best order from the set of offline orders for the remaining agents and providing a new policy for every agent from the observation timestep t' until the end of horizon. The selection criteria for choosing the best order for the defender team whenever any agent leaves the team depends on the number of exact matching and closest matching selection orders. For example, let us assume that there are 4 agents in the system $\{a_1, a_2, a_3, a_4\}$ and the available set of selection order contains three orders, $O_1 = \{3, 2, 1, 4\}$, $O_2 = \{4, 2, 3, 1\}$ and $O_3 = \{3, 2, 4, 1\}$ with total utility of

$\{200, 150, 100\}$ for the orders respectively. Let us consider two case studies:

- **Agent a_1 leaves the system:** In this case, O_2 and O_3 are the best suitable orders since they require no re-evaluation but the order with highest utility is given preference and hence, O_2 is chosen.

- **Agent a_3 leaves the system:** In this case, none of the matches are exact and therefore, we find the closest match. We choose O_2 to assign policies to the remaining agents since it requires minimal updates to agent policies. At the observation timestep, the previous policy of a_1 is replaced by the existing policy of a_3, but after considering the change in state distribution of the agents since a_1 and a_3 are not guaranteed to be in the same state at the considered timestep. However, due to the replacement of agent policies, a_1 would now become the third agent in the system, assuming the presence of two agents. Policy recomputation is not required because the offline joint policy (of every selection order) computes the $V^t(s, \pi)$ values for all states at all intermediate timesteps (i.e., joint value after selection of every agent in the selection order). Furthermore, no reward recomputation is required since the joint reward considers only the count of agents (and not the identity of agents) at any state due to the monotone submodular reward structure for the joint reward.

In this manner, the online phase improves the value of solution roughly the same as Dec-O-Rev, but very quickly.

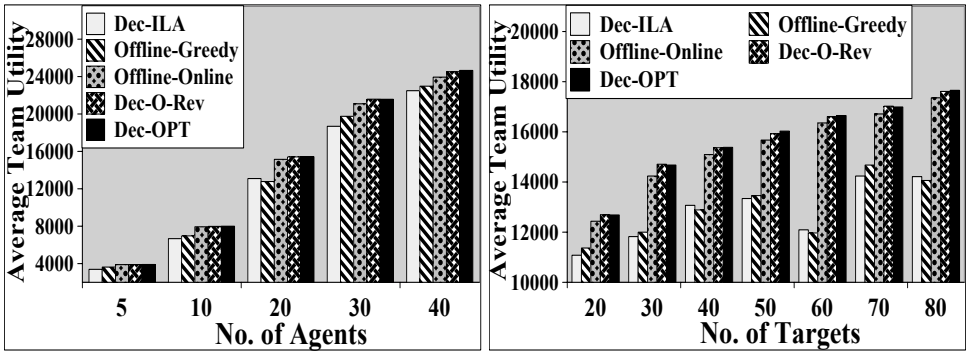
6.3 Experiments

We evaluate[1] the performance of our greedy approaches and compare them with the benchmark approaches for non dedicated teams introduced in section 5.2 for the security games domain provided by Shieh et al. (2014) and the sensor network domain provided by Kumar et al. (2017) in decentralized settings. We implement the benchmark approaches mentioned in the section 6.2 to provide a comparative study of the results and evaluate the performance of all the approaches on various metrics: (a) solution quality; (b) runtime; (c) quality of online bounds. We generate 1500 samples of agent availability (defenders in security domain and sensors in sensor domain) and divide it into training and testing sets of 1000 and 500 samples, respectively. To obtain a fair comparison over all approaches, we compare the solution quality and runtime on the same test set.

6.3.1 Security Games Domain

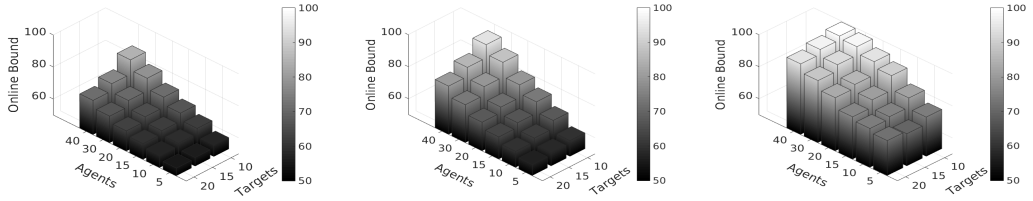
The security games domain is explained in details in section 2.1.6. We perform experiments on the metro rail network with a set of targets (train stations) defended by a team of decentralized (yet cooperative) defenders in the presence of transition uncertainty. The metro graphs were formed by connecting stations together in lines of length 5 and then randomly adding $|b|/2$ edges between targets, to resemble train systems in the real world with complex loops. The reward is a joint reward which

[1] All our optimization problems are run on CPLEX v12.7



(a) $\tau = 40, H = 20, \epsilon = 0.7$ (b) $|\mathcal{A}g| = 20, H = 20, \epsilon = 0.7$

Figure 6.1: Quality Comparison w.r.t. (a) Agents and (b) Targets



(a) $H = 10, \epsilon = 0.5$ (b) $H = 10, \epsilon = 0.7$ (c) $H = 10, \epsilon = 0.9$

Figure 6.2: Comparison of Online Bound w.r.t. Effectiveness

is a function of the number of active defenders and the targets for a joint state s. We describe the submodularity of reward function in section 2.2.3. The test results were averaged over 15 randomly generated metro-based graph networks and the rewards were generated randomly in the range of [0,100]. We run the scenarios with a probability delay of .2 and a maximum of 5 agents (varies from 10% to 25% across scenarios) with an ability to leave the system, defined by probability vector Δ. The defender agents are homogeneous (due to same reward and transition function) but differ from each other in their starting states (generated randomly for every agent) and their capability to leave the system.

Solution Quality: We compare different approaches with respect to average team utility in Figure 5.1(a) as the number of defenders $|\mathcal{A}g|$ is increased. Specifically, we consider a metro network with targets $b = 40$, horizon $H = 20$ and the effectiveness $\epsilon = 0.7$ for every defender resource. Similarly, in Figure 5.1(b), we compare different targets (i.e. stations), b for a fixed number of agents $|\mathcal{A}g| = 20$, horizon $H = 10$ and effectiveness $\epsilon = 0.7$. The key observations are summarized as following:

(1) The average team utility increases as the number of defender agents increase for a fixed number of targets and planning horizon due to the submodular reward structure. Similarly, the team utility increases with increasing targets due to increased number of choices for obtaining better rewards.

(2) Dec-ILA provides low team utility solutions since the remaining agents continue with existing policies even after agents leave. This impacts the scope of improvement in rewards and is a cause of serious concern in security domain since it allows easy access to an adversary to plan an attack in unprotected areas.

(3) Offline-Greedy provides similar or better solutions than Dec-ILA. An important observation from the training data is that the agents with higher probability of staying in the system are added before the agents with high leaving probability in the selection set. Further, we see that with fewer agents, targets and smaller planning horizon, Offline-Greedy performs almost at par with Dec-O-Rev as agent exits are given due importance during the offline policy design but the performance degrades quickly with increasing count of agents and the planning horizon. In the worst case, the performance was seen to be even lower than Dec-ILA.

(4) Offline-Online provides a steady performance almost at par with the upper bound benchmarks (O-Rev and Offline optimal) even with increasing problem sizes. Due to the random selection of agents at different observation times and the presence of submodular reward function, in the best case, Offline-Online could provide better team utility than Dec-O-Rev. However, on an average, Offline-Online achieves solution quality close to O-Rev in almost all cases.

(5) Offline optimal provides a good upper bound but is not always guaranteed to provide better utility compared to revamp due to the dominance of rewards in earlier timesteps explained in the section 6.2.1. However, on an average, Offline optimal provides almost similar or slightly better solution quality compared to Dec-O-Rev even after having the knowledge of samples before-hand.

Solution Runtime: With respect to runtime, we compare only online runtime since the offline runtimes do not matter. Due to decentralized planning of agents, individual agent planning time varies from 100 ms to 5000 ms from the smallest problem instance (20 targets and 10 timesteps) to the largest instance (40 targets and 20 timesteps). For Dec-O-Rev, due to the use of lazy greedy approach at every timestep of revamp, the revamp time varies from 15 seconds to 1700 seconds depending on the number of defenders and the problem size per defender. Further, there can be multiple revamps for one planning scenario making Dec-O-Rev infeasible for providing new policies quickly. However, our Offline-Online approach uses a proactive offline planning which reduces the online execution time to milliseconds, even in the worst case (although it requires offline training time). Similarly for Dec-ILA, offline-optimal and offline-greedy, online runtime is minimal.

Online Bound Comparison: For the online bound comparisons, we use a consistent reward structure for every randomly generated metro network. For every metro network, we generate various scenarios of agents availabilities for different number of defenders and varying effectiveness of defenders. We compute online bound for every scenario of non-dedicated agent teams using Equation 6.1 and average the online bounds over all test samples and all randomly generated graphs and

Grid-Size	Sensors,Targets Global States	ϵ		
		.3	.5	.7
2×2	4, 1, 4	57.3	63	68.8
3×3	4, 2, 6*6	57.9	62.9	67.4
5×5	5, 1, 10	58.3	64.5	71.5
5×5	5, 2, 6*6	58.6	65	71.5
10×5	6, 3, 14*10*10	57.4	61.2	64.2
10×5	6, 4, 5*5*5*5	57.3	61.6	63.7
10×5	6, 5, 6*5*5*5*5	55.7	61.3	65.3
10×5	10, 3, 14*10*10	58.5	63.5	69
10×5	10, 4, 5*5*5*5	55.5	58.5	61.7
10×5	10, 5, 6*5*5*5*5	55.9	61.5	67.7
10×10	10, 4, 6*5*5*5	58.7	64.5	71.2
10×10	15, 4, 6*5*5*5	58.5	63.8	72.2
10×10	20, 4, 6*5*5*5	58.9	65.5	72.7
10×10	10, 5, 5*5*5*5*5	55.5	61.2	67.3

Table 6.1: Online Bound Comparison for Sensor Domain

infer that the online guarantees are significantly better than the a priori guarantees (of 50% from optimal), with the best case of atleast 90% from optimal for varying effectiveness.

Figure 6.2 compares the online (or posterior) guarantees obtained for the solutions generated by O-Rev for different values of agents, targets and ϵ. It shows that the online guarantees improve with increasing agents and decreasing targets over varying effectiveness with highest guarantee being reported for 10 targets and 40 agents. Further, with increasing effectiveness of agents, the optimality increases with highest guarantees (up to 99%) observed for $\epsilon = 0.9$. To avoid clutter, we do not plot the online guarantees provided by policies generated using Offline-Online in the same graph. However, Offline-Online fared slightly lower than Dec-O-Rev in terms of guarantees and provided a guarantee that was 0.7% lower than Dec-O-Rev in the best case, while in the worst case, it was 2 % lower than Dec-O-Rev.

6.3.2 Sensor Network Domain

For the sensor network domain, we model the environment as a grid world and use similar settings as (Kumar et al., 2017).

Online Bound Comparison: Table 6.1 shows the online guarantees obtained for Offline-Online by varying the grid-size, number of sensors and their effectiveness, number of targets and the number of global states. We vary the effectiveness parameter from 0.3 to 0.7 and observe that the online bounds vary from 55% to 73% for Offline-Online, while the guarantees provided by Dec-O-Rev was seen to be 4% and 1.8% better than offline-online in the worst case and best case respectively. An important observation is that with increasing targets, the number of global states increases exponentially, leading to memory issues. We note that 5 targets for a 10×10 grid with every target having a path-length of 5 was very

difficult instance to solve with 5^5 or 3125 global states. However, increasing the number of sensors with a fixed number of targets was comparatively easier to solve since every sensor agent problem was solved independent of other agents due to the decentralized settings.

Solution Runtime: With respect to runtime, the time taken by any sensor agent for individual planning varies from few milliseconds to 10 seconds with increasing number of targets and the global states. Due to lazy greedy evaluations for Dec-O-Rev, every revamp may take time ranging from less than a minute to 40 minutes depending on the complexity of the problem being solved. This makes Dec-O-Rev infeasible to use in online settings. Further, there can be multiple revamps for every scenario to make the situation worse. Similar to the security domain, the performance of Offline-Greedy is very similar to Dec-ILA while Dec-OPT provides results similar to Dec-O-Rev. More interestingly, our Offline-Online approach continues to perform gracefully with increasing number of sensors, targets and the grid size and takes minimal time (in milliseconds) even for the largest problem, making it the best choice considering the trade-off of running time and compromise of quality.

6.4 Summary

In this chapter, we focussed on cooperative decentralized stochastic planning for non-dedicated agent teams. We provided a general model for decentralized non dedicated agent teams. Our offline greedy based approach provided good results in small instances while our Offline-Online approach provided the best results even in large instances in an effective manner. Finally, our extensive experiments on benchmark problems demonstrate that our Offline-Online approach provides the best solutions that are on par with benchmarks that provide an upper bound on the performance while taking negligible online runtime making it effective even for taking decisions at every step.

Chapter 7

Task/Resource Constrained Assignment for Non-dedicated Agent Teams

In many large scale multiagent distribution network based applications such as power distribution networks (Kumar et al., 2009), distributed meeting scheduling (Maheswaran, Tambe, Bowring, Pearce, & Varakantham, 2004), distributed planning and resource allocation, target tracking in sensor networks (Modi et al., 2005), multiple agents coordinate to optimize a single objective function. In these problems, agents consume resources which have limited capacity that must not be violated. For example, in the distributed meeting scheduling problem, agents may have a travel budget which can not be exceeded and the meeting rooms may have a limited capacity on the number of attendees. Similarly, the important complexity in modeling power networks are the resource constraints (capacity of edges). Furthermore, agents may leave the system midway through the tasks which may require reconfiguration of the system. For example, power distribution networks require local co-ordination among energized regions to reconfigure the network after some regions are de-energized due to faults in power sources or power lines. Similarly, due to fault in some resources or unexpectedly higher number of attendees in a meeting, meeting scheduler may need rescheduling of the facilities.

In this chapter, we address the important problem of multiagent coordinated assignment along with task/resource allocation and examine the domain of power distribution networks. In this domain, we limit the scope of our study to the problem of power supply restoration (PSR) (refer section 2.1.3 for more details of the domain), where a power grid has to be reconfigured after multiple line failures subject to constraints such as acyclic power flow and line capacities (Thiebaux & Cordier, 2001; Bertoli et al., 2002; Thiébaux, Coffrin, Hijazi, & Slaney, 2013). In particular, we focus on the *multiagent* PSR problem where different sub-regions of the grid are controlled by different agents (Kumar et al., 2009). The power supply needs to be restored to de-energized areas by local coordination among agents without the oversight of a central authority. Such multiagent PSR problems have been previously addressed using the framework of distributed constraint optimization (Kumar et al., 2009; Matsui & Matsuo, 2012). However, the scalability of such previous approaches is limited due to the NP-Hard (Hadžic, Wasowski, & Andersen, 2007)

complexity of solving PSR problems.

The main contribution in this chapter is the development of a decentralised Lagrangian Relaxation mechanism to solve the PSR problem. A key component that enables decentralisation of LR is the extraction of optimal feasible PSR solution at each iteration of LR in a decentralised manner. Extracting optimal feasible PSR solution even for a tree-structured network is NP-Hard. Fortunately, it is an *easy* NP-Hard problem and we develop a decentralized fully polynomial time approximation scheme (FPTAS) to find a near-optimal feasible PSR solution by exploiting its connections with the knapsack problem. Our decentralised LR mechanism has multiple necessary and important properties, especially in the context of solving the multi agent PSR problem:

- We only use *local* message-passing among agents (each representing a different region) to extract near-optimal feasible PSR solution, thereby enabling decentralised control.
- We are able to provide provable guarantees on solution quality.
- Due to iterative nature of LR, our approach has the desirable *anytime* property.
- Our approach is significantly more scalable than previous multi-agent approaches and can solve existing real-world benchmarks near optimally with significant speedups and with significantly low message-passing overhead.

We test our decentralised LR approach on real world and large synthetic benchmarks (Hadžic et al., 2007; Kumar et al., 2009) and were able to show that our approach is significantly more scalable than previous multi-agent approaches while obtaining near optimal solutions. To show that our approach is competitive with centralized solvers, we also compare it against a highly efficient centralized math programming solver CPLEX. We show that our approach can achieve near optimal solutions, close to 90% optimality, within comparable runtimes as CPLEX.

In the next section, we provide a mixed-integer linear program (MILP) to solve PSR for a network decomposed into multiple regions using relay nodes. Naturally, this approach is not a decentralized approach. However, using the technique of Lagrangian relaxation (LR) (Bertsekas, 1999), we relax the flow conservation constraints for relay nodes. We then show that the resulting dual problem can solved *independently* for each region r via local message-passing along the cut edges. Thus, introducing the idea of relay nodes in a global network divided into multiple regions leads to a decentralized algorithm within the LR framework.

7.1 Dual Decomposition for Multiagent PSR

We first develop a MILP to solve the PSR problem for the decomposed global network. We then show how to relax the complicating constraints within this MILP such that the resulting dual problem can be solved in a decentralized manner. Before we introduce the MILP, we provide the notation employed in Table 7.1. Let $\mathcal{C}^r$ denote the set of all relay nodes and $\mathcal{E}^r$ denote the set of all relay edges belonging to a region r. Each region r is then described using the graph $G^r = (V^r \cup \mathcal{C}^r, E^r \cup \mathcal{E}^r)$. The node set V^r consists of sink set $\mathcal{S}^r$ and power source set $\mathcal{P}^r$ for the region r.

Table 7.2 shows the math program for finding the optimal solution to the PSR problem. The objective of this program is to maximize the total weight of nodes that

Variable	Definition
x_i^r	$x_i^r = 1$ indicates that sink s_i in region r consumes $\|s_i\|$ units of power
x_{ij}^r	x_{ij}^r = 1 indicates that power flows from node i to j in region r
d_{ij}^r	Power flow between nodes i and j in region r
$\mathcal{S}^r$	Set of sinks in region r
$\mathcal{P}^r$	Set of power sources in region r
$\mathcal{C}^r$	Set of relay nodes in region r
$\mathcal{E}^r$	Set of relay edges in region r
δ_{ij}^r	Intermediate variable employed to linearize $x_{ij}^r \cdot d_{ij}^r$

Table 7.1: Notation

are supplied power. Binary decision variables x_i are created for each sink and relay node for each region (superscripts denote region). If $x_i = 1$, then sink s_i consumes $|s_i|$ units of power. Binary variables x_{ij} are created for each edge in the network for a region r. If $x_{ij} = 1$, it means that power flows from node i to j. The continuous variable d_{ij} denotes the amount of power flow along the directed edge (i, j). The constraints in table 7.2 denote the following:

- Constraint 7.2 denotes that a sink is always switched off if there is no incoming power flow, otherwise, it can be switched on.
- Constraint 7.3 denotes that a sink can receive power from at most one of its neighbors. This is also a necessary constraint for PSR. This also helps maintain the acyclicity of power flow.
- Constraint 7.4 is the capacity constraint for a line.
- Constraint 7.5 denotes flow conservation for sink nodes. For every sink s_j, total power inflow must be equal to the sum of power consumed by it, the line loss ϵ for incoming power, and the total power forwarded to other nodes.
- Constraint 7.6 denotes flow conservation for relay nodes. As relay nodes are created in *pairs*, we use the terminology that for every relay node $c \in \mathcal{C}^r$ in region r, its pair is denoted as $c' \in \mathcal{C}^{r'}$ in region r'[1]This flow constraint denotes that the total *incoming* power into relay node c in region r should be equal to the total outgoing power from c' in region r'.
- Constraint 7.7 is the capacity constraint for all the power sources across all regions $p_i \in \mathcal{P}$.The total outgoing power from a power source should not exceed the power available to it.

Notice that the math program in table 7.2 is nonlinear due to quadratic terms, such as $x_{ij}^r \cdot d_{ij}^r$ in constraints. However, this constraint can be linearized as the variables x_{ij}^r are binary and the power flow d_{ij}^r is bounded. To linearize such quadratic terms, we replace $x_{ij}^r \cdot d_{ij}^r$ by a new variable δ_{ij}^r. The constraints (7.5), (7.6) and (7.7) in the previous formulation are thus replaced with following constraints (7.9), (7.10) and (7.11). To complete the linearization, we also add a new constraint (7.12). The

[1]For ease of exposition, for every region r, we assume that there is at most one connecting region r'. In general, there can be multiple regions r' that share a cut edge with r and we can use multiple flow conservation constraints (7.6) to represent such connectivity.

Variables: $x_i^r \ \forall s_i^r \in \mathcal{S}^r \cup \mathcal{C}^r; x_{ij}^r, d_{ij}^r \ \forall (i,j) \in E^r \cup \mathcal{E}^r \ \forall r \in \mathcal{R}$

Minimize: $$-\sum_{r \in \mathcal{R}} \sum_{i \in \mathcal{S}^r} x_i^r \cdot v_i^r \tag{7.1}$$

Subject to:

$$x_j^r \leq \sum_{i \in V^r \cup \mathcal{C}^r} x_{ij}^r \ , \forall j \in \mathcal{S}^r \cup \mathcal{C}^r, \forall r \tag{7.2}$$

$$\sum_{i \in V^r \cup \mathcal{C}^r} x_{ij}^r \leq 1 \ , \forall j \in \mathcal{S}^r \cup \mathcal{C}^r, \forall r \tag{7.3}$$

$$d_{ij}^r \leq L \ \ \forall (i,j) \in E^r \cup \mathcal{E}^r, \forall r \tag{7.4}$$

$$\sum_{i \in V^r \cup \mathcal{C}^r} (d_{ij}^r - \epsilon) \cdot x_{ij}^r = \sum_{l \in \mathcal{S}^n \cup \mathcal{C}^r} d_{jl}^r \cdot x_{jl}^r + x_j^r \cdot |s_j^r|, \forall j \in \mathcal{S}^r, \forall r \tag{7.5}$$

$$\sum_{i \in V^r} (d_{ic}^r - \epsilon) \cdot x_{ic}^r = \sum_{j \in \mathcal{S}^{r\prime} \cup \mathcal{C}^{r\prime}} d_{c'j}^{r'} \cdot x_{c'j}^{r'}, \forall c \in \mathcal{C}^r, \forall r \tag{7.6}$$

$$\sum_{j \in \mathcal{S}^n \cup \mathcal{C}^r} d_{pj}^r \cdot x_{pj}^r \leq |p| \ , \forall p \in \mathcal{P}^r, \forall r \tag{7.7}$$

$$d_{ij}^r \geq 0 \ \ , x_{ij}^r \in \{0,1\} \ \ , x_i^r \in \{0,1\} \tag{7.8}$$

Table 7.2: Nonlinear Mathematical Program for PSR

constant M is a large number. Such constraints are also known as 'big M' constraints in the OR literature.

$$\sum_{i \in V^r \cup \mathcal{C}^r} \delta_{ij}^r - \epsilon \cdot x_{ij}^r = \sum_{l \in \mathcal{S}^n \cup \mathcal{C}^r} \delta_{jl}^r + x_j^r \cdot |s_j^r|, \forall j \in \mathcal{S}^r, \forall r \tag{7.9}$$

$$\sum_{i \in V^r} \delta_{ic}^r - \epsilon \cdot x_{ic}^r = \sum_{j \in \mathcal{S}^{r\prime} \cup \mathcal{C}^{r\prime}} \delta_{c'j}^{r'}, \forall c \in \mathcal{C}^r, \forall r \tag{7.10}$$

$$\sum_{j \in \mathcal{S}^n \cup \mathcal{C}^r} \delta_{pj}^r \leq |p| \ , \forall p \in \mathcal{P}^r, \forall r \tag{7.11}$$

$$\delta_{ij}^r \begin{cases} \leq d_{ij}^r, & \forall (i,j) \in E^r \cup \mathcal{E}^r, \forall r \\ \leq M \cdot x_{ij}^r & \forall (i,j) \in E^r \cup \mathcal{E}^r, \forall r \\ \geq d_{ij}^r + (x_{ij}^r - 1) \cdot M, & \forall (i,j) \in E^r \cup \mathcal{E}^r, \forall r \end{cases} \tag{7.12}$$

We refer to the program of table 7.2 with the above linearized constraints as the *global MILP* for PSR.

Model Extensions The basic model we presented in table 7.2 can be extended in multiple ways to account for varying preferences of power grid operators. For example, one can include linear terms in the objective function that penalize switching of devices from their previous positions to minimize the number of device switching operations in the network. We used a simple approximation of the line loss. In networks, where accurate modeling of line loss is required, a linear approximation

of the line loss can be added to flow equations (Coffrin, Van Hentenryck, & Bent, 2011; Thiébaux et al., 2013).

7.1.1 Relaxing Flow Conservation For Relay Nodes

Notice that every constraint and variable in the global MIP is separable for each region r, except for the flow conservation constraint for relay nodes (7.10). Therefore, if we are able to principally *relax* this constraint, then our global MIP would decompose into independent parts, one per region, which can be solved independently of each other. This is indeed achievable by using the technique of Lagrangian relaxation (LR) or dual decomposition (Bertsekas, 1999).

We first provide a brief overview of the LR approach, more details can be found in (Bertsekas, 1999, Chapter 6). Consider the optimization problem:

$$\begin{aligned} &\min f(x) \\ &\text{s.t} \quad x \in X, \;\; g_j(x) \leq 0, \; j = 1, \ldots, p \end{aligned}$$

The dual function $q(\cdot)$ and the Lagrangian $L(\cdot)$ of the above problem after dualizing all the constraints g are given as:

$$q(\mu) = \inf_{x \in X} L(x, \mu) = \inf_{x \in X} \left\{ f(x) + \mu \cdot g(x) \right\} \tag{7.13}$$

The dual solution $q(\mu)$ is a lower bound of optimal $f^\star(x)$ for every value of dual variables μ. The advantage while working with the dual formulation of the original problem is that the structure of the dual is often much simpler leading to computational gains. Furthermore, the dual solution also provides a lower bound on the original problem. In addition, the dual optimization problem, $\max_{\mu : q(\mu) > -\infty} q(\mu)$, is always concave, and can be solved optimally using the projected subgradient method even in the case of non-differentiable objective function (Bertsekas, 1999).

We therefore relax or *dualize* the complicating flow conservation constraint (7.10) for all the relay nodes in each region. For each complicating constraint, we create a dual variable $\lambda_c^r \; \forall c \in \mathcal{C}^r, \; \forall r$. This dual variable can be thought of as the cost of violating the flow conservation constraint. The dual decomposition technique will try to find the 'right' cost such that violations of the dualized constraints are minimal. The Lagrangian function, $L(\{\boldsymbol{x}^r, \boldsymbol{\delta}^r, \boldsymbol{d}^r\}, \boldsymbol{\lambda})$, (ignoring the line loss ϵ for ease of exposition) is given as:

$$\sum_{r \in \mathcal{R}, i \in \mathcal{S}^r} -x_i^r v_i^r + \sum_{r \in \mathcal{R}, c \in \mathcal{C}^r} \lambda_c^r \left(\sum_j \delta_{c'j}^{r'} - \sum_i \delta_{ic}^r \right) \tag{7.14}$$

Upon rearranging the terms to highlight the separable structure of Lagrangian further, we simplify the above to get:

$$\sum_{r \in \mathcal{R}} \left\{ \sum_{i \in \mathcal{S}^r} -x_i^r v_i^r + \sum_{c \in \mathcal{C}^r} \left(\sum_j \delta_{cj}^r \cdot \lambda_{c'}^{r'} - \sum_i \delta_{ic}^r \cdot \lambda_c^r \right) \right\} \tag{7.15}$$

Notice that in the above equation, nodes c and c' are paired relay nodes in region r and r'. Using the above equation, we get our dual $q(\boldsymbol{\lambda})$ as below:

$$\sum_{r\in\mathcal{R}} \min_{\boldsymbol{x}^r,\boldsymbol{\delta}^r,\boldsymbol{d}^r} \left\{ \sum_{i\in\mathcal{S}^r} -x_i^r v_i^r + \sum_{c\in\mathcal{C}^r} \left(\sum_j \delta_{cj}^r \cdot \lambda_{c'}^{r'} - \sum_i \delta_{ic}^r \cdot \lambda_c^r \right) \right\} \tag{7.16}$$

Notice that the above dual function can be evaluated by solving the inner optimization problem *independently* for each region r controlled by the corresponding agent. Thus, evaluating the dual is substantially simplified, and follows the region-based decomposition of the global network G.

Practical Considerations The performance of the LR approach w.r.t. the solution quality is affected by the number of constraints that are being relaxed or dualized. In general, the optimal dual solution $q^\star$ may not be equal to the optimal primal solution $p^\star$. The gap between these solutions ($p^\star - q^\star$), also called the duality gap, can increase with higher number of relaxed constraints. Nonetheless, we show empirically that for several large instances, the LR approach is able to provide good solution quality despite large number of dualized constraints. The degradation in the performance of LR with increasing number of relaxed constraints is graceful.

7.1.2 Maximizing the Dual Function

As the dual function is a lower bound on the optimal primal optimal solution for every value of $\boldsymbol{\lambda}$, we now address the problem of optimizing dual: $\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda})$. We maximize the dual iteratively by using the projected sub-gradient ascent technique (Bertsekas, 1999). Using the sub-gradient information, the updated value of the dual variable $\lambda_c^{r\star}$ for the next iteration is:

$$\lambda_c^{r\star} = \lambda_c^r + \alpha \left[\sum_{j\in\mathcal{S}^{r'}\cup\mathcal{C}^{r'}} \bar{\delta}_{c'j}^{r'} - \sum_{i\in V^r} \bar{\delta}_{ic}^r \right] \forall c \in \mathcal{C}^r, \forall r \tag{7.17}$$

where $\bar{\delta}_{c'j}^{r'}$ and $\bar{\delta}_{ic}^r$ are the variable values obtained while solving the corresponding minimization problem in (7.16) for regions r' and r respectively; c and c' are paired relay nodes in regions r and r'. The parameter α is the step parameter. Notice that such an update of dual variables requires exchange of variable values only across the cut edges of the global network among neighboring agents. Thus, this step can be carried out in a distributed fashion.

The step size α is crucial to fast and accurate convergence of the LR approach. There are a number of recommendations for setting the step size in sub-gradient method. We use the following rule which has theoretical justifications in (Bertsekas, 1999). The step size α^{i+1} for the iteration $i+1$ is set based on quantities computed in iteration i as follows:

$$\alpha^{i+1} = \frac{Primal^i - Dual^i}{||\nabla q^i||^2} \tag{7.18}$$

where $Primal^i$ denotes the primal solution quality (or the solution quality corresponding to a feasible solution satisfying all the power network constraints) and $Dual^i$ denotes the dual value $q(\boldsymbol{\lambda}^i)$ for the current iteration i, and ∇q^i denotes the sub-gradient of the dual function q. The sub-gradient w.r.t. a variable λ_c^r is essentially the quantity in square brackets in Eq. (7.17). Each quantity in Eq. (7.18) is readily available except the primal solution corresponding to the current iteration's dual solution. Therefore, we need to extract a good primal solution at each iteration, which also the imparts the desirable anytime property to the LR approach. Thus, at each iteration, our approach provides a feasible reconfiguration plan corresponding to the primal solution and the quality bounds using the dual solution $q(\boldsymbol{\lambda})$.

7.1.3 Extraction of Feasible Primal Solution

At each iteration of our LR approach, we need to extract a good quality feasible primal solution that satisfies all power network constraints from the current dual solution. It is challenging to extract a feasible solution from the dual solution as flow conservation will be violated for relay nodes. Our strategy is to first extract one feeder tree for each power source in the global network. This can be done in a distributed manner by locally inspecting the direction and amount of power flow (x_{ij}, δ_{ij}) for relay edges within each region. Once we have such feeder trees extracted from the current dual solution with a power source as root node, we attempt to determine which nodes should be switched on or off such that total sink weight of switched on nodes is maximized. Unfortunately, we show below that even this problem is challenging.

Proposition 4. *Solving optimally the PSR problem is NP-Hard even for tree-structured power networks.*

Proof. We reduce the well known 0/1 knapsack problem to a tree-structured power network. Consider a generic 0/1 knapsack problem with a set of m items, and two m tuples of positive integers, corresponding to values: $\langle v_1, v_2, \cdots, v_m \rangle$, and weights of items: $\langle w_1, w_2, \cdots, w_m \rangle$. The goal is to identify the set of items whose total weight is less than the given capacity W and it yields the highest value.

For this 0/1 knapsack problem, we have an equivalent PSR problem with graph $G = (\mathcal{P} \cup \mathcal{S}, E)$ where.

- $\mathcal{P} = \{p_1\}$, where p_1 has a finite supply W.
- $\mathcal{S} = \{s_1, s_2, \cdots, s_m\}$, where s_i consumes w_i units of power and has a sink weight of v_i.
- $E = \{(p_1, s_1), (s_1, s_2), (s_2, s_3), \cdots, (s_{m-1}, s_m)\}$ denotes a chain network

We set the line capacity of each edge as W. Clearly, the above reduction is a polynomial time. We have the following correspondence between the optimal PSR solution for this chain and the knapsack. If sink s_i consumes power (i.e., $x_i = 1$), then the item i is included in the knapsack, otherwise not. Given that line and power source capacity is W, total switched-on sink consumption will always be less than W. As we maximize $\sum_i x_i v_i$ for the power network, this implies finding the best set of items to be included in the knapsack. One can show that such an optimal PSR solution is also optimal for knapsack, and vice-versa. Thus, PSR problem is NP-Hard even for tree-structured networks. □

Despite the above negative result, we exploit the fact that the knapsack problem is an *easy* NP-Hard problem and admits a fully polynomial time approximation scheme (FPTAS). Thus, our approach is to derive an FPTAS for optimal PSR in a tree-structured problem. Furthermore, we also show that such a scheme can be implemented using message-passing along the edges of the power network making all the steps of our approach—evaluating the dual, iteratively maximizing the dual and extracting a feasible solution—distributed in nature.

Feeder Tree Extraction

We provide a sketch of the procedure to extract feeder trees from the current dual solution. The first step for extracting a feeder tree is to determine the direction of power flow for the cut edges. Notice that for every edge (i, j) for any region that is not a cut edge, the binary variable x_{ij} denotes the power flow direction. However, while decomposing a power network into regions, a cut edge (u, v) that connects two different regions r and r' is being decomposed into two relay edges (u, c) in region r and (c', v) in region r'.

We now outline a simple procedure that determines the power flow direction for a cut edge (u, v) based on inspecting the dual solution (the δ variable) for the relay edges. Figure 7.1 visually shows this procedure. Based on the dual solution for the relay edges (u, c) and (c', v), there are three cases possible. In the first case, positive power flows from node u to node c, and from node c' to node v. In this case, we make the direction of flow from the node u to node v for the cut edge. Notice that we are only inspecting the direction of power flow. The amount of power may be inconsistent in the dual solution. For e.g., one relay node may assume that it is forwarding 10 units of power to its paired node, and its paired node may assume that it is receiving 20 units, leading to violation of flow conservation constraints. Correcting such flow conservation violation is exactly the main task for primal extraction.

For the other two cases (case 2 and case 3 in Figure 7.1), there is no flow across the cut edge and its corresponding devices are set to the open position. Once we have determined the direction of power flow for each edge in the global power network, the only remaining task is to extract a feeder tree corresponding to each power source. This can be done easily as the direction of power flow provides the parent-child relationship required for the tree construction. Furthermore, given the constraint that a node cannot receive power from multiple incoming edges, there are going to be no cycles while extracting such trees. Notice also that we discard any tree and its corresponding nodes where a relay node is the root. For e.g., case 3 in figure 7.1 denotes one such setting. Once we have such trees extracted, we develop a tree-based dynamic programming algorithm that determines which sinks are switched on and off to maximize the sink weight for each feeder tree.

Tree Based Dynamic Programming (TBDP)

Our scheme, called TBDP, provides an FPTAS for a tree-structured PSR. It follows the high level architecture of (Wu, Sheldon, & Zilberstein, 2014) for stochastic network design in ecology. Our approach is different on multiple fronts due to

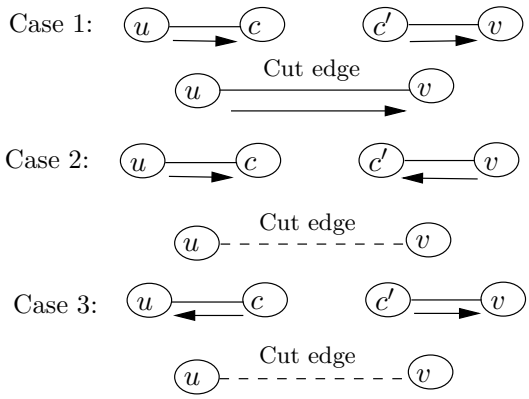


Figure 7.1: Determining power flow for the cut edge (u, v) from the dual solution for relay edges (u, c) in region r and (c', v) in paired region r'. Arrows denote power flow direction. A dotted line denotes no power flow with corresponding devices being open.

different recurrence relationships than (Wu et al., 2014), which are essential for deriving the FPTAS for PSR.

We are given a tree $\mathcal{A} = (V, E)$. The root of this tree is a power source P with maximum supply p. We denote other nodes (total of m) using i and js. Their sink consumption is s_i, value is v_i. The capacity of each line is L. We denote using π the policy that specifies which node should be switched on and thus, consume some power, and forward the rest downstream. If a node is switched off, it does not consume any power and forwards all power to its children. The binary decision variable is $\pi(i)$ indicating whether a sink node is on or off. The *total power* consumed by the entire subtree rooted at (and including) node i be $t(i) = \sum_{j \in \mathcal{T}_i} \pi(j) s_j$. For a given policy π, the optimization problem to solve is:

$$\max_{\pi} \sum_{i} \pi(i) v_i \quad \text{s.t.} \tag{7.19}$$

$$t(i) \leq L \; \forall i \in V, \; t(P) \leq p \tag{7.20}$$

Let $z_i(\pi)$ denote the total utility of the subtree $\mathcal{T}_i$:

$$z_i(\pi) = \sum_{j \in \mathcal{T}_i} \pi(j) v_j$$

We first highlight a simple, yet important recurrence relation for quantities $z(\cdot)$ that forms the basis of the FPTAS for PSR:

$$z_i(\pi) = \pi(i) v_i + \sum_{j \in \text{Ch}(i)} z_j(\pi), \qquad \forall i \tag{7.21}$$

where $\text{Ch}(i)$ is the set of children of i. Another key recurrence relationship is for the *minimum* power supply $C_i(z)$ needed to produce a utility of exactly z from the subtree $\mathcal{T}_i$ rooted at node i. Without loss of generality, we assume that each node has at most two children, otherwise, any tree with more than 2 children per node can be reconfigured to a two-children tree by creating dummy nodes as in (Wu et al., 2014).

$$C_i(z) = \min_{\pi(i)\in\{0,1\}} C_{i,\pi(i)}(z) \tag{7.22}$$

$$C_{i,\pi(i)}(z) = \min_{0\le d\le z'_i} \pi(i)s_i + C_j(d) + C_k(z'_i - d) \tag{7.23}$$

where $z'_i = z - \pi(i)v_i$ encodes the total value that the two children j and k are required to contribute in order to get a value z for the parent node i. The expression in (7.23) simplifies to $C_{i,\pi(i)}(z) = \pi(i)s_i + C_j(z'_i)$ when node i has only one child j. Similarly, for the base case when i is a leaf node, $C_{i,\pi(i)}(z) = \pi(i)s_i$ if $\pi(i)s_i = z$, and $C_{i,\pi(i)}(z) = +\infty$, otherwise.

The recurrence relationships (7.21) and (7.23) for the PSR problem form the basis of key differences in our work from that of (Wu et al., 2014). The above recurrences can provide a pseudo-polynomial time dynamic programming approach for optimal PSR similar to the knapsack problem. However, to develop an FPTAS, we need to *discretize* all the possible achievable utilities $z(\pi)$. We write a discretized version of Eq. (7.21) as:

$$\hat{z}_i(\pi) = K_i \Big\lfloor \frac{\pi(i)v_i + \sum_{j\in \text{Ch}(i)} \hat{z}_j(\pi)}{K_i} \Big\rfloor \ \forall i \tag{7.24}$$

where K_i is a constant provided as input. Using theoretical analysis similar to (Wu et al., 2014), we can show that $K_i = K = 0.5 \cdot \beta \cdot (\min_{j\in V} v_j)$ provides an FPTAS, where β is an input parameter that determines the optimally guarantee for the FPTAS. Once we have discretized the set of all possible utility values using constant K, all that is remaining is to perform a bottom-up dynamic programming to solve the problem (7.22) for each node in the feeder tree, and a final policy determining top-down phase on the feeder tree. These computations can be performed by using message-passing and is again distributed. We omit the details of such bottom-up and top down pass of TBDP as it is similar to (Wu et al., 2014). While doing the bottom-up pass, we make sure that if $C_i(z) > L$, then we make it infeasible as $C_i(z) = \infty$ to respect line capacity constraint. Once the $C(z)$ values are computed at the root node, we choose the highest utility z such that $C(z) \le p$, the power capacity of the source. Using similar theoretical analysis provided in (Wu et al., 2014), our approach can provably provide a solution within β fraction of the optimal with a worst-case runtime guarantee of $O(m^2/\beta^2)$ given m nodes in the tree. In practice, TBDP was much faster than the worst-case time.

7.1.4 D^{2}ADP Approach

D^{2}ADP or dual decomposition based approximate dynamic programming uses lagrangian dual decomposition along with dynamic programming in a decentralized manner. The algorithm has four major phases. The first phase involves the SolveRegion() function that computes dual solution over all regions by solving each region independently as seen in Eq. (7.16). The second phase involves the FeederTreeExtract() function that extracts the feeder trees $\mathcal{A}$ to determine the parent-child relationship between relay node pairs. We extract one feeder tree for

every power source, regardless of the regions, as described in the subsection 7.1.3. All trees are disjoint (at most one incoming power line for a node) and may contain nodes from different regions (via exchange of messages for relay node pairs). The third phase employs the TBDP() function to compute primal solution $\mathcal{T}_{r,p}$ for every feeder tree using a bottom up followed by a top down message passing scheme. The bottom up message passing stage finds the best solution where every node sends its reward information to it's parent node. The top down stage of TBDP is responsible for assignment of resources where every power source chooses the best assignment configuration for all nodes that passes down the tree. Finally, the price update phase updates the price variables, λ using the price update rule Eq. (7.17). This process is repeated for every iteration it until convergence, which occurs when the duality gap is less than a very small number, ϵ or when a specified number of iterations is reached.

Algorithm 6 D^{2}ADP ALGORITHM

Initialization : $\boldsymbol{\lambda^0} \leftarrow 0; it \leftarrow 0$
repeat
  $D^r, \boldsymbol{x^r}, \boldsymbol{\delta^r} \leftarrow SolveRegion(\boldsymbol{\lambda^{it,r}}, Region\ r),\ \forall r \in R$
  $\mathcal{A}_{r,p} \leftarrow FeederTreeExtract(\boldsymbol{x^r}, \boldsymbol{\delta^r}),\ \forall p \in \mathcal{P}^r, \forall r$
  $\mathcal{T}_{r,p} \leftarrow TBDP(\mathcal{A}_{r,p}), \forall p \in \mathcal{P}^r, \forall r$
  $\lambda_c^{r,\star} = \lambda_c^r + \alpha \left[\sum_j \bar{\delta}_{c'j}^{r'} - \sum_i \bar{\delta}_{ic}^{r}\right], \forall c \in \mathcal{C}^r, \forall r$
  $it \leftarrow it + 1$
until Convergence
return P, x, δ

Space Complexity Analysis We provide the space complexity for D^{2}ADP since exchange of messages requires storage space. We show that the space requirement of D^{2}ADP is very small, even in the worst case where the count and size of messages exchanged increase proportionally with the increase in number of cut edges and total nodes respectively.

Proposition 5. *Total number of messages exchanged over cut edges is $O(E_c)$, where E_c is total number of cut edges.*

Proof. All inter-region message exchanges take place only over cut edges. Every relay node obtained from cut edges requires $O(1)$ message exchange for dual variables, $O(1)$ for determining parent-child relationship in feeder tree and $O(2)$ for top-down/bottom-up messages over feeder tree. Hence, total messages exchanged is $O(E_c)$. □

Proposition 6. *The maximum size of message exchanged over cut edges is $O(N/\beta)$, where N denotes total number of nodes and β is the FPTAS parameter.*

Proof. The maximum size of any message in D^{2}ADP is dominated by dynamic programming for primal extraction which requires top-down and bottom-up message exchanges for feeder trees. Let us consider the worst case where every node is a region and a single power source with infinite power is available. Let the power lines

have infinite line capacity. Here, all nodes will be the part of a single feeder tree where the discretized utilities for root node can be obtained as $\{0K_i, 1K_i, ..., l_iK_i\}$ from Eq. (7.24). l_i provides the upper bound for message size and it is an integer value given by $\left\lceil \frac{z_i(\pi)}{K_i} \right\rceil$. We further assume that $m \leq v_j \leq M,\ \forall j \in \mathcal{T}_i$ where m and M are universal constants. Then, $l_i = \left\lceil \frac{z_i(\pi)}{K_i} \right\rceil \leq \left\lceil \frac{NM}{K_i} \right\rceil = O\left(\frac{N}{\beta}\right)$ by using the fact that $K_i = M \cdot \beta$ □

7.2 Experiments

In this section, we perform experiments[2] to compare D^{2}ADP with the dynamic programming based decentralized approach for power supply restoration in the literature (Kumar et al., 2009), referred to as PSR-DPOP or DPOP for short. We also compare against the centralized global MILP solved using CPLEX. Our goal in these experiments is the following. We demonstrate that our approach, despite being approximate in nature, can provide provably near-optimal solutions for a range of problems. We also show the scalability of our approach on large synthetic benchmarks while providing good quality guarantees, close to 90% optimality. We further highlight the anytime nature of our approach to provide good solutions quickly.

Real World Benchmarks: We test on real world benchmarks representing real world configurations of NESA, a power distribution company in Denmark (Hadžic et al., 2007; Kumar et al., 2009). We specifically take the two largest configurations and decompose them into multiple regions. The smaller of the two, 'Large' network, has 56 sinks, 66 lines and 2 power sources. We decompose this network into 7, 10 and 15 underlying regions to test the scalability of our approach with increasing regions. The PSR-DPOP approach of (Kumar et al., 2009) cannot exploit the region based decomposition of the power network, and considers each node in the network to be controlled by a different agent. Therefore, for fair comparisons against PSR-DPOP, we also show results on the 'Large' benchmark with 56 regions (denoted as 'Large(56R)'). The largest instance 'Complex' has 119 sinks, 146 lines and 3 power sources. We consider 10, 15 and 20 regions for this case along with the extreme case in which every network node is a separate agent. We run our approach, D^{2}ADP for 100 iterations. The PSR-DPOP is not an anytime approach, so we run it with the maximum memory limited to 4GB.

Table 7.3 provides the comparisons against PSR-DPOP on real world configurations, with respect to runtime, space requirements (total and maximum message size), and solution quality. PSR-DPOP was able to terminate and provide a solution only on the 'Large' configuration. As PSR-DPOP is agnostic to region based decomposition of the network, the results for PSR-DPOP are exactly the same for varying number of regions. As PSR-DPOP is an optimal approach, it provided the optimal solution upon termination for 'Large(56R)'. We can clearly see that our

[2]All our experiments were performed on a 3.2GHz CPU with 4GB RAM. All our optimization problems are run on CPLEX v12.5

Configuration	Algorithm	Time(s)	Total(Kb)	Max(Kb)	Optimality
Large(56R)	DPOP	54.5	36000.0	3900.0	100%
Large(7R)	D^2ADP	8.7	119.7	0.3	97.3%
Large(10R)	D^2ADP	6.0	149.4	0.3	92.9%
Large(15R)	D^2ADP	8.6	178.3	0.2	89.1%
Large(56R)	D^2ADP	17.8	639.7	0.3	87.8%
Complex(119R)	DPOP	–	–	–	–
Complex(10R)	D^2ADP	15.0	219.5	0.5	91.7%
Complex(15R)	D^2ADP	14.8	239.3	0.4	88.4%
Complex(20R)	D^2ADP	15.5	272.9	0.4	78.3%
Complex(119R)	D^2ADP	41.0	1086.3	0.5	70.1%

Table 7.3: Real World Configurations. The quantities in $(\cdot)$ denote total number of regions for the instance. 'Total' denotes the total size of all the messages exchanged, 'Max' denotes the maximum message size between any two agents.

approach provides provably near-optimal solutions without the large message overhead of the PSR-DPOP approach. The total message size and the maximum size is significantly smaller for D^2ADP.

For the largest 'Complex' instance, PSR-DPOP was unable to provide any solution as its memory requirements are exponential in the tree-width of the underlying network, which was about 40 for the 'Complex' instance. In contrast, our approach scales well for this largest instance with varying number of regions. As highlighted earlier, the performance of the LR approach is adversely affected by the increasing number of regions in the network as this causes several constraints to be relaxed. Nonetheless, D^2ADP's performance w.r.t. solution quality varied gracefully while increasing the number of regions from 10 to 119. Notice that the instance 'Complex(119R)' represents the worst possible scenario for our approach as each network node is an agent, thereby relaxing the flow conservation constraint for each edge. Despite this, our approach is able to get a solution provably within 70% of the optimal. We expect that in real networks, such an extreme case is unlikely to occur as each region in a power network is typically composed of multiple nodes.

Large Synthetic Benchmarks: To further experiment with the scalability, we created synthetic configurations based on the 'Complex' real-world configuration. We refer to them as 'L-Complex(<Number of regions>R)'. For example, a L-Complex (2R) configuration refers to two layers of the complex configuration connected through multiple randomly selected nodes. Each layer forms a region controlled by an agent. The network within each layer is the same as 'Complex'. Layers are arranged in a 3D fashion with each layer connected via inter-layer edges to the layer above and below. Agents can only communicate via inter-layer edges. The number of inter-layer edges was about 10% of total edges (=14) in the 'Complex' network. Such a layered model of construction helps conserve the network structure of real power networks while increasing the scale with increasing number of layers. We consider instances with up to 30 layers. We consider each region to be one layer in the synthetic configurations. For instance, we use L-Complex (10R) to mean 10

Configuration	Algorithm	Time(s)	Total(Kb)	Max(Kb)	Optimality
L-Complex(1R)	DPOP	–	–	–	–
L-Complex(10R)	D^2ADP	247.2	910.7	5.4	88.0%
L-Complex(15R)	D^2ADP	389.4	1579.8	6.1	89.0%
L-Complex(20R)	D^2ADP	516.6	2231.1	7.0	86.7%
L-Complex(25R)	D^2ADP	540.0	2823.1	7.4	86.5%
L-Complex(30R)	D^2ADP	786.0	16858.9	8.9	86.4%

Table 7.4: Solution quality, runtime and message size results for synthetic configurations

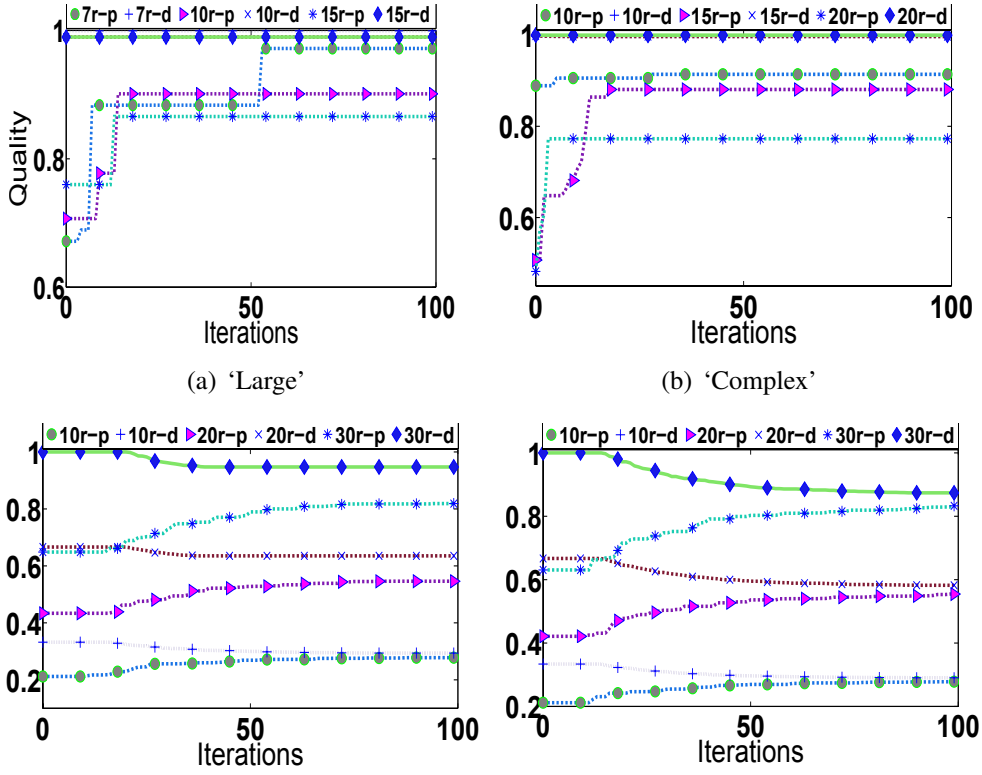
layered 'Complex' configuration with 10 regions or 10 agents.

We have generated 10 random instances, where sink-weights are varied, for the synthetic configurations to ensure there is no specific dependence on sink weights. The capacity of power sources was set such that it was necessary for power to flow from one layer to another; one layer was severely deficient in total power, another had excess power. Table 7.4 shows the results on such synthetic instances. Even on such large and intricate instances, our decentralized approach is able to get good solution quality. Furthermore, the size of each individual message in our approach increases quite moderately w.r.t. increasing number of regions. This is because the messages only contain local information about the dual variables and the sub-gradient information for each cut edge. Thus, our approach scales well with the network size and is able to provide good quality solutions with limited message passing overhead.

Anytime Performance: We next show solution quality results obtained using D^2ADP on the 'Large' and 'Complex' configurations for different regions for each iteration. We show primal and dual values for each instance with increasing iterations. Figure 7.2(a) provides the results on the 'Large' configuration. '7r-p' refers to the primal solution with 7 regions, '7r-d' refers to the dual solution with 7 regions, '15r-p' is primal for 15 regions and so on. We interpret the problem as a maximization problem with dual always providing an upper bound. We can clearly see from these results that our approach is able to provide good primal solutions even in early iterations. Figure 7.2(b) provides the results on the 'Complex' configuration with 10, 15 and 20 regions. Again, we get good solution quality for 10 and 15 regions. As the number of regions increased to 20, our approach had to relax and dualize many constraints. That resulted in performance hit. We do note that with 20 regions, each region has about 6 nodes which is very harsh partitioning of the network. Still our approach provided a decent solution quality.

The results for different L-Complex instances and configurations are presented in Figures 7.2(c) and 7.2(d). 10r-p in this graph refers to the primal quality for a 10 layered L-Complex configuration with 10 regions. We observe that as number of layers are increased, the duality gap increases. However, even on multiple instances of different L-Complex configurations, we obtain strong quality guarantees of around 85% or more of the optimal as shown in Figures 7.2(c) and 7.2(d).

Comparison with Centralized Solver: As shown in table 7.4, existing decentralised algorithms are unable to scale to L-Complex instances. Hence, we



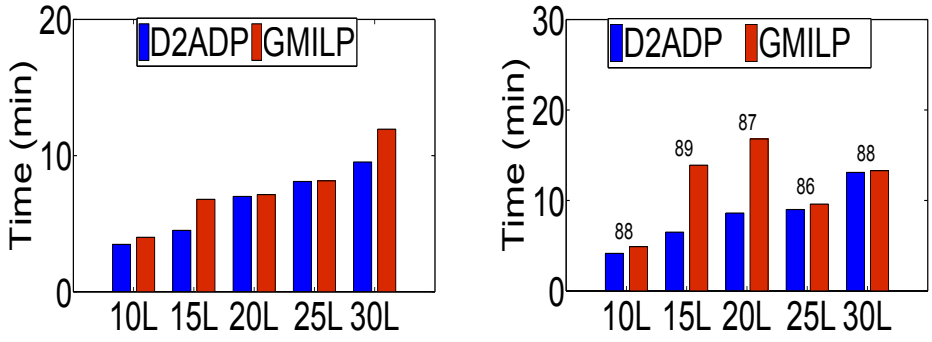
(a) 'Large' (b) 'Complex'

(c) 'L-Complex' instance:1 (d) 'L-Complex' instance:2

Figure 7.2: Duality gap for real datasets (figures a & b) & synthetic datasets (figures c & d) with varying regions:#r-p denotes #regions-primal & #r-d denotes #regions-dual. All values are normalzied to 1

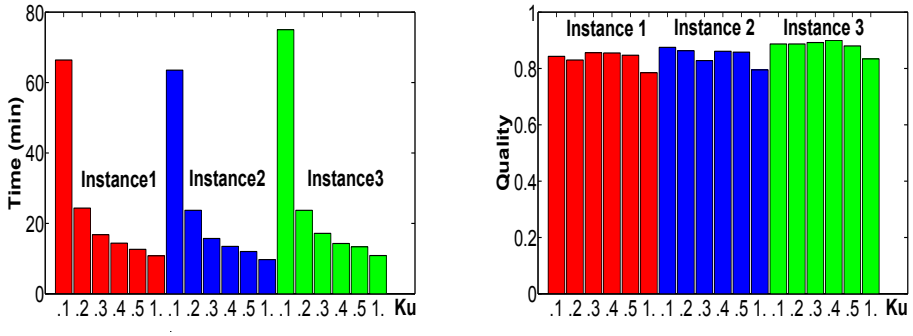
benchmark the run-time performance of D^{2}ADP with the Global MILP (GMILP) of Table 7.2. This is not a fair comparison for our decentralized approach D^{2}ADP which solves the PSR problem using message passing, whereas the centralized solver has complete knowledge of the problem. Nonetheless, these results shed light on the effectiveness of our approach. Figure 7.3 provides the time taken by D^{2}ADP and GMILP for a given quality bound for different multilayered networks. We experiment with 10 random settings of sink-weights and compare the time taken for an average of result. Despite being a decentralized approach, D^{2}ADP runtime is highly competitive with respect to GMILP for a 85% quality bound. The best solution quality achieved (in percentage of optimal) for the respective L-Complex configuration by D^{2}ADP is mentioned on top of bars in figure 7.3(b). We show in figure 7.3(b) the time required by the GMILP solver (CPLEX) to achieve the same quality bound as provided by D^{2}ADP. This table further highlights that our approach is highly competitive to a strong centralized baseline. Thus, our key message from these results is that our approach can provide similar quality guarantees as a highly efficient centralized solver in a *decentralized* setting. In contrast, MILP solvers such as CPLEX are unable to work in a decentralized setting.

Primal Extraction: Finally, we experiment with different values of constant K to obtain the right setting for TBDP. A key practical issue with choosing K according



(a) 85% Quality (b) Best Quality of D^{2}ADP

Figure 7.3: Time Comparision w.r.t. Quality with CPLEX



(a) Time change w.r.t. K (b) Quality change w.r.t. K

Figure 7.4: Time and Quality Variation w.r.t.K

to the formula $0.5\beta\cdot(\min_j v_j)$ for a feeder tree is that such a value of K that provides the theoretical guarantee for the FPTAS may be small, and unnecessarily lead to large running time for primal extraction. As also highlighted in (Wu et al., 2014), larger values of K can provide similar empirical performance. Notice that in our case, the dual solution provides the upper bound, which is not affected by the primal extraction part. The theoretical guarantee for primal extraction does not affect the overall optimality gap for our approach.

We experiment on K values as small as .01 and as large as 1. Figure 7.4(a) provides the time taken and Figure 7.4(b) provides the solution quality obtained for different values of K on 3 different problem instances. We can see that for smaller value of K, runtime is significantly higher than for the larger K values. However, figure 7.4(b) shows that the solution quality remains almost the same for different K values. Therefore, we chose $K = 0.4$ that provided the right tradeoff between time taken and optimality.

7.3 Summary

In this chapter, we addressed the problem of power grid reconfiguration after multiple line failures. We presented a number of advances for the multiagent version

of the PSR problem. We developed a novel and iterative dual decomposition based approach that effectively utilized the underlying multi-region structure of the power grid. We also addressed the challenging problem of extracting a feasible solution providing anytime nature to our approach by developing a provable approximation technique. Our approach only requires local message-passing among different grid regions, resulting in a distributed approach. Using the quality bounds provided by our approach, we showed that it can achieve near-optimal solutions on a number of large real-world and synthetic benchmarks. Our approach is faster and significantly more scalable than the previous best multiagent approach for the PSR problem. We also showed empirically that our approach was highly competitive both in runtime and solution quality against a strong centralized baseline CPLEX, while retaining all the benefits of a decentralized approach.

Chapter 8

Related Work

8.1 Related Work for TasC-MDP

Task/resource constrained stochastic planning problems can be either competitive or cooperative. In *competitive problems*, agents, or subgroups of agents, are selfish and hence optimize their own objectives. The solution concept is usually to find an equilibrium solution given all possible resource allocations. These problems can be solved using game-theoretic methods (Korzhyk, Yin, Kiekintveld, Conitzer, & Tambe, 2011; Yin & Tambe, 2012) and market-based methods like auctions (Wellman, Walsh, Wurman, & MacKie-Mason, 2001). In *cooperative problems*, all agents are part of a team and they coordinate to find a resource/task allocation and a joint plan that results in the maximum utility for a central authority. We are specifically focussed on cooperative problems.

Researchers have modelled TasC-MDP problems in a variety of ways including cooperative auctions (Koenig, Keskinocak, & Tovey, 2010), resource-constrained MDPs (Dolgov & Durfee, 2006; Guestrin & Gordon, 2002), decentralized MDPs (Dec-MDPs) (Bernstein, Givan, Immerman, & Zilberstein, 2002), and Dec-MDP variants that exploit the sparsity of agent interactions (Nair et al., 2005; Kumar & Zilberstein, 2009a; Velagapudi et al., 2011; Varakantham et al., 2014; Witwicki & 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 & Durfee, 2006; Wu & Durfee, 2010) and weakly-coupled MDPs (Guestrin & Gordon, 2002; Gocgun & Ghate, 2012). 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. 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 deterministic routing for multiple agents (Christofides, Mingozzi, & Toth, 1981; Campbell, Clarke, Kleywegt, & Savelsbergh, 1998), where the routing problems for individual agents are dependent on the tasks/resources allocated to them. Owing to the intrinsic difficulty of this

problem class and to deal with practical size problems, decomposition techniques such as *Column Generation* (CG) (Desrochers, Desrosiers, & Solomon, 1992), *Lagrangian Relaxation* (LR) (Kohl & Madsen, 1997; Chien, Balakrishnan, & Wong, 1989) and Benders Decomposition (Federgruen & 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.

8.2 Related Work for ND-TasC-MDP

The notion of non-dedicated teams was introduced and studied extensively in the context of Belief Desire Intention (BDI) frameworks (Cohen & Levesque, 1991; Grosz & Kraus, 1996; Tambe, 1997) for multi-agent planning. However, these works relied on existence of plan libraries and considered primarily deterministic outcomes to actions. However, we consider domains dealing with probabilistic outcomes to actions and automatic generation of plans for agents.

Another closely related thread of research is on adhoc teams (Barrett, Rosenfeld, Kraus, & Stone, 2017), where the focus is on a newly added team member that cooperates with team-mates coming from a variety of sources without directly altering the behaviour of team mates. We focus on non-dedicated teams where the configuration of the team is altered to accommodate the leaving of a team member. More recently, 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, they provide an exhaustive offline approach that is not scalable.

8.3 Related Work for ND-TI-Dec-MDP

Decentralized stochastic planning for a team of agents is required in a wide variety of problems such as target tracking by a team of sensors (Nair et al., 2005; Kumar & Zilberstein, 2011), securing targets from unknown attackers using a team of defenders (Shieh et al., 2014), rescuing of victims by a team of robots during disaster (Melo & Veloso, 2011; Varakantham et al., 2009) and analysing underwater samples using a team of underwater vehicles (Yin & Tambe, 2011). Existing literature has focussed on Decentralized Markov Decision Processes (Dec-MDPs) which precludes solving problems with more agents. While approximate approaches have been proposed to solve multiple agent problems (Kumar & Zilberstein, 2011; Velagapudi et al., 2011; Varakantham et al., 2009), there is little or no research in approximation methods that provide strong guarantees on solution quality in such decentralized settings.

Non-dedication in agent teams has been explored by (Agrawal & Varakantham, 2017) for centralized planning. Further, (Shieh et al., 2014) considered non-dedicated teams in decentralized settings, but they provide exhaustive offline approach that is not scalable. Our contributions differ from this line of work in providing quick solutions for decentralized planning such that remaining agents can

reconfigure their policies to attend to tasks of leaving agents.

Existing literature has considered submodularity in the context of sequential decision making for multi-agent problems. Kumar *et al.* (2009b) introduced the idea of partition matroid based planning in the context of Multi-agent MDPs (MMDPs) and considered submodular rewards across time steps for MMDPs in weather tracking satellites. Detecting storms early is of key interest and this reward function for detecting storms is submodular over time steps. Satsangi *et al.* (2015) considered submodularity in multi-agent sensor selection problem modelled as a POMDP with submodular belief based reward function. Above mentioned works focus on centralized planning model while decentralized multiagent sequential planning has been exploited by Kumar *et al.* (2017). Our contributions differ from this line of work in considering non dedicated agent teams with multiple agent exits from the team while still considering joint submodular reward functions for decentralized cooperative multiagent sequential planning.

Another closely related thread of research is on adaptive submodularity (Golovin & Krause, 2011), where a sequence of decisions are taken by accounting for the observations of past decisions. Our work differs from this thread in the sense that the problem of non-dedicated teams is a multi-stage submodular problem, where at every stage of decision (i.e., when agents leave the team), the current state of system serves as the observation for a new problem with reduced count of agents and horizon to provide a new joint policy for the remaining agents in the system.

8.4 Related Work for PSR

Due to increased world-wide incentive for cleaner electricity generation (Miller, Ramchurn, & Rogers, 2012; Ramchurn, Vytelingum, Rogers, & Jennings, 2012; Kok, Scheepers, & Kamphuis, 2010), next generation of smart grids would feature co-generation from intermittent renewable power sources. In addition, the deregulation of power markets has enabled the presence of multiple operators (Griffin & Puller, 2005; Kumar et al., 2009), which marks a shift away from highly regulated monopolies of power grids. Such heterogeneous structure of future smart grids where multiple operators control different sub-regions of the grid presents a unique opportunity for agent-based *decentralized control* of smart grids. Decentralized control of power grids entails high operational readiness, increased robustness and faster response time after disasters. In fact, the importance of such decentralised control has been already recognized in power systems and multiagent systems community (Nagata & Sasaki, 2002; Kumar et al., 2009; Miller et al., 2012; Matsui & Matsuo, 2012).

We use the framework of *Lagrangian relaxation* or dual decomposition (Bertsekas, 1999), to solve the multiagent PSR problem. Lagrangian relaxation (LR) has a rich history in power networks community (Kim & Baldick, 1997, 2000; Nogales, Prieto, & Conejo, 2003). These previous approaches use the LR approach to solve the problem of optimal power flow (OPF), also known as the optimal dispatch problem. Our application of LR to the PSR problem is significantly different than OPF, as power restoration is a *discrete optimization* problem where we are changing the underlying structure of active power lines in the network. The OPF problem

is concerned with deciding how much power each generation unit in the grid must generate to ensure that all the demand is satisfied (Nogales et al., 2003). Thus, OPF problem is a continuous, albeit, non-convex problem. Such *discrete* versus *continuous* nature between PSR and OPF also gets magnified when we develop strategies to extract a feasible PSR solution during each iteration of LR.

Our decentralised LR approach for the multi agent PSR problem bears some similarity to distributed constraint optimization (DCOP) (Modi et al., 2005; Petcu & Faltings, 2005; Gershman et al., 2009; Yeoh & Yokoo, 2012) approaches that partially centralize the optimization problem, such as the asynchronous partial overlay (APO) algorithm (Mailler & Lesser, 2004, 2006). A key difference in our work is that the sub-regions of a power network remain fixed with the LR approach not requiring any additional centralization, whereas the APO approach dynamically changes the (partial) centralization while solving the underlying DCOP problem. Another LR based DCOP approach, decomposition with quadratic encoding to decentralize (DeQED) (Hatano & Hirayama, 2013) is based on the divide-and-coordinate (DaC) framework and it differs from our work in using quadratic encoding to solve their problem where an inter-agent cost function is encoded into the quadratic programming problem while we formulate our problem as a MILP. Furthermore, as highlighted in (Kumar et al., 2009), a straightforward conversion of the PSR problem to a DCOP presents several challenges. For example, the resulting DCOP has high arity constraints to represent the flow conservation and the line capacity requirement of the PSR problem. In addition, the discretization of the line capacity is also required to allow standard DCOP approaches to solve PSR problems. In contrast, the LR technique we develop does not require any discretization, and results in a simple distributed approach that works by passing messages among different connected sub-regions of the smart grid.

Chapter 9

Conclusions

This thesis presents techniques for the problem of resource/task allocation for multiagent systems operating in real world uncertain environments where the planning/assignment problem of individual agents is dependent on the allocation of resources/tasks. In addition, the task/resource allocation to the individual agents is further complicated by the non-dedication of agents that arises due to unforeseen conditions (e.g., high priority task, emergency/incident or damage to an agent) which may force the team members to leave their tasks before the end of planning horizon. While Markov Decision Processes and Distributed MDPs have been used to address uncertainties in real-world domains, they are not scalable with increasing agents and resources/tasks. Further, there is little or no literature that considers situations where agents leave the team after task allocation. Towards addressing the above mentioned challenges, the main contributions of this dissertation can be summarised as mentioned below.

- For a dedicated agent team, we handled the interdependent problems of task/ resource allocation and multiagent planning by providing a generic TasC-MDP framework with an ability to handle task dependencies for agents. We provided a highly efficient and scalable greedy approach, GAPS that provided best runtime performance (i.e., less than 5 minutes for upto 600 agents) but without quality guarantees. We also provided an optimization based approach called LDD+GAPS that exploited the decomposable structure in TasC-MDPs and provided a good trade-off between GAPS and the optimal MILP by providing solutions with quality guarantees within 5% of optimal even in very large problems.

- For the case of non-dedicated agent team, we extended the TasC-MDP model to provide ND-TasC-MDP that is capable of handling transition uncertainty and uncetainty due to agents leaving the team. We provided multiple proactive and reactive approaches to facilitate coordination among the remaining team members over the tasks left undone by agents leaving the team. We also developed heuristics that benchmark the performance of our approaches by providing good upper and lower bounds on the solution quality. We provided an online approach, ReacT that performed reactive updates on the current solution to generate a new allocation in less than a minute. Our proactive sample average approximation based Lagrangian relaxation approach, SAA+LR

computed a unique policy for agents irrespective of the agent exits from the team. In addition, our two stage MILP (MILP-2S) provided the flexibility of a two stage policy that further improved the team utility compared to a single stage allocation and the performance was almost at par with the benchmark heuristics that provided upper bound. Our extensive experiments on benchmark problems demonstrated that by reconfiguration of the system (rearrangement of the left over tasks/resources to remaining agents) after one or more agents have left is crucial in order to respond immediately and efficiently in critical situations to avoid mishaps.

- We extended the concept of non-dedication to transition independent Dec-MDP (TI-Dec-MDP) by formulating the ND-TI-Dec-MDP model for a team of independently collaborating non-dedicated agents. We established connections between submodularity and non-dedicated agent teams in decentralized settings by showing that with monotone submodular reward functions, greedy solution provides an a priori guarantee of at least 50% from optimal. We also exploited online bounds to compute the posteriori guarantees and conclude that the online guarantees improve with increasing agents and decreasing targets with the best case of atleast 90% from optimal for varying effectiveness. For the solution quality comparison in decentralized settings, we extended the benchmark approaches from ND-TasC-MDP with the help of lazy greedy. Our offline-greedy and the offline-online greedy approaches provided comparable solutions with respect to the benchmark approaches on multiple benchmark problems dealing with cooperative decentralized non-dedicated agent teams.

- For problems dealing with task/resource constrained assignment, we studied multiagent coordinated assignment along with resource allocation in large distribution networks. In particular, we examined the power supply restoration (PSR) problem and proposed a dual decomposition based approximate dynamic programming approach (D^2ADP) that provided quick and efficient reconfiguration of the network in a decentralized manner. For the primal extraction from the dual solution, we developed a decentralized fully polynomial time approximation scheme (FPTAS) to find a near-optimal feasible PSR solution by exploiting its connections with the knapsack problem. Our approach, D^2ADP significantly improved the scalability and solved existing real-world and synthetic benchmarks near optimally with significant speedups and with a very low message-passing overhead.

In this thesis, we provided general models that broadly fit various problem domains. However, every problem domain may have specific environment settings that differ from other domains. For example, agents leaving the team midway before the end of horizon is of more relevance to infrastructure security domain where patrolling officers are required to leave their patrolling tasks (no physical item involved) midway to attend to accidents/incidents compared to an urban consolidation center where every vehicle agent is dedicated, and allocated a fixed set of tasks before it leaves the center. It involves physical items to be delivered to city center and agents are assigned tasks in specific locations, and therefore, if an agent

leaves midway either due to breakdown/accident, etc., it is very difficult for remaining agents to complete the tasks of leaving agent since the vehicles may have space/time/location constraints. Further, there may be a penalty for late deliveries and a central planner may not be available after the initial task allocation. Therefore, the relevance of a particular model may be limited (but still applicable) to be applied to all the problem domains discussed in this thesis. Similarly, the problem of power supply restoration (PSR) has specific requirements such as network structure, inter-agent communication, etc. which mandates the requirement of a different approach to solve the non-dedication problem in PSR and distributed network problems. Therefore, we use different kinds of problems for different models and solution approaches.

9.1 Future Directions

This thesis explores new directions for multiagent teams operating in uncertain environments, that need further examination and analysis. Below, we briefly outline some of the questions that remain open in the area explored by this dissertation and provide the possible directions of future work.

- The decision version of a TasC-MDP problem is NP-complete since it is an extended version of the resource-constrained MDP (shown as NP-complete by Dolgov and Durfee (2006)). These problems remain NP-complete with just the presence of global constraint on the number of shared resources, which can be shown with a straightforward reduction from KNAPSACK. Even though the decision problem of knapsack is NP-complete and the optimization problem is NP-hard, it admits a fully polynomial time approximation scheme (FPTAS). For a TasC-MDP, every single agent sub-problem can be solved independently to compute the resource requirements of every agent. Hence, it should be possible to develop a FPTAS (similar to the PSR problem in chapter 7) to find a feasible and near-optimal solution for TasC-MDP by exploiting its connections with the knapsack problem. However, the recurrence relations to obtain the total reward over all agents and the discretization of rewards to obtain FPTAS are still challenging.

- For the dedicated and non-dedicated agent teams, we have used MDPs with discrete and finite state and action spaces for modelling the task/resource constrained multiagent coordinated planning problems. In many realistic cases (e.g., UAV assignment problems), however, the continuous model parameters such as continuous states and action spaces are available to the agents. Existing literature on MDPs has exploited continuous state and action spaces for factored MDPs and constrained MDPs, but not for task/resource constrained multiagent coordinated planning. Function approximation techniques have also been widely used to solve continuous MDPs quickly and efficiently. The MDP-based task/resource allocation models and the approaches presented in this thesis appear compatible and complementary for solving continuous MDPs and hence, extending our MDP-based task/resource allocation mechanisms to continuous MDPs appears to be a promising direction.

- For non-dedicated teams, our proposed heuristic approaches only consider agents leaving the system, and do not account for the situations where a leaving agent returns back to the system or there exists some probability of additional forces entering the system. The addition of agents to the system is equally interesting and appears to be a viable and fruitful future direction. Our broad framework for non dedicated agents with two stage observations can still be useful in sampling of incoming/outgoing agents. Furthermore, the two-stage MILP approach can be extended to multi-stage MILP to provide an online and robust solution approach, considering the unpredictability in the number of available agents.

- For the task/resource constrained multiagent coordinated assignment problems, we addressed only the resource assignment problem for power supply restoration in power distribution networks. MDPs and distributed MDPs can be useful in extending these problems to consider stochasticity in the distributed power generation. In addition, parallel fault restoration can be studied and validated considering the availability of a mixture of different renewable distributed generations (e.g., solar photovoltaics, wind turbines) as well as onsite diesel generators.

To conclude, this thesis has provided general models and scalable solution approaches for solving different integrated problems of task/resource allocation and multiagent planning/assignment. We exploited the decomposable structure in these problems to significantly lower their computational complexity. We further provided algorithmic advances in non-dedicated agent teams dealing with these integrated problems which demonstrate that the reconfiguration of any system after some team members have left is crucial to avoid any mishappenings. Finally, we believe that the advancements developed in this dissertation significantly further the applicability and general usefulness of both task/resource allocation and multiagent planning/assignment in practical settings.

Bibliography

Agrawal, P., & Varakantham, P. (2017). Proactive and reactive coordination of non-dedicated agent teams operating in uncertain environments. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 28–34.

Agrawal, P., Varakantham, P., & Yeoh, W. (2016). Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning. In *IJCAI'16*.

An, B., Pita, J., Shieh, E., Tambe, M., Kiekintveld, C., & Marecki, J. (2011). Guards and protect: Next generation applications of security games. *ACM SIGecom Exchanges*, *10*(1), 31–34.

Barrett, S., Rosenfeld, A., Kraus, S., & Stone, P. (2017). Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, *242*, 132–171.

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

Bertoli, P., Cimatti, A., Slanley, J., & Thiebaux, S. (2002). Solving power supply restoration problems with planning via symbolic model checking. In *European Conference on Artificial Intelligence*, pp. 576–580, Lyon, France.

Bertsekas, D. P. (1999). *Nonlinear Programming* (2nd edition). Athena Scientific.

Bowring, E., Tambe, M., & Yokoo, M. (2006). Multiply-constrained distributed constraint optimization. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 1413–1420.

Brown, M., Saisubramanian, S., Varakantham, P., & Tambe, M. (2014). STREETS: game-theoretic traffic patrolling with exploration and exploitation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 2966–2971.

Campbell, A., Clarke, L., Kleywegt, A., & Savelsbergh, M. (1998). The inventory routing problem. In *Fleet management and logistics*, pp. 95–113. Springer.

Chapman, A. C., & Varakantham, P. (2014). Marginal contribution stochastic games for dynamic resource allocation. In *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 333–340. Springer.

Chien, T. W., Balakrishnan, A., & Wong, R. T. (1989). An integrated inventory allocation and vehicle routing problem. *Transportation Science*, *23*(2), 67–76.

Christofides, N., Mingozzi, A., & Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, *20*(1), 255–282.

Coffrin, C., Van Hentenryck, P., & Bent, R. (2011). Approximating line losses and apparent power in ac power flow linearizations. In *IEEE Power and Energy Society General Meeting*, pp. 1–8.

Cohen, P. R., & Levesque, H. J. (1991). Teamwork. *Nous*, *25*(4), 487–512.

D'andrea, R., Mansfield, P. K., Mountz, M. C., Polic, D., & Dingle, P. R. (2012). Method and system for transporting inventory items.. US Patent 8,170,711.

Dantzig, B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*, 80–91.

Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, *40*(2), 342–354.

Dolgov, D., & Durfee, E. (2004). Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes. In *Proceedings of the International Conference on Planning and Scheduling (ICAPS)*, pp. 315–324.

Dolgov, D., & Durfee, E. (2006). Resource allocation among agents with MDP-induced preferences. *Journal of Artificial Intelligence Research*, *27*, 505–549.

Federgruen, A., & Zipkin, P. (1984). A combined vehicle routing and inventory allocation problem. *Operations Research*, *32*(5), 1019–1037.

Fisher, M. L., Nemhauser, G. L., & Wolsey, L. A. (1978). An analysis of approximations for maximizing submodular set functions- ii. In *Polyhedral combinatorics*, pp. 73–87. Springer.

Gershman, A., Meisels, A., & Zivan, R. (2009). Asynchronous forward bounding for distributed cops. *Journal of Artificial Intelligence Research*, *34*, 61–88.

Gocgun, Y., & Ghate, A. (2012). Lagrangian relaxation and constraint generation for allocation and advanced scheduling. *Computers and Operations Research*, *39*(10), 2323–2336.

Golovin, D., & Krause, A. (2011). Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, *42*, 427–486.

Griffin, J., & Puller, S. (2005). *Electricity deregulation: choices and challenges*. University of Chicago Press, Chicago.

Grosz, B. J., & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, *86*(2), 269–357.

Guestrin, C., & Gordon, G. (2002). Distributed planning in hierarchical factored MDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 197–206.

Hadžic, T., Wasowski, A., & Andersen, H. R. (2007). Techniques for efficient interactive configuration of distribution networks. In *International joint conference on Artificial Intelligence*, pp. 100–105.

Handoko, S. D., Nguyen, D. T., & Lau, H. C. (2014). An auction mechanism for the last-mile deliveries via urban consolidation centre. In *Proceedings of the International Conference on Automation Science and Engineering (CASE)*, pp. 607–612.

Hatano, D., & Hirayama, K. (2013). Deqed: An efficient divide-and-coordinate algorithm for dcop. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1325–1326. International Foundation for Autonomous Agents and Multiagent Systems.

Hazard, C., Wurman, P., & D'Andrea, R. (2006). Alphabet soup: A testbed for studying resource allocation in multi-vehicle systems. In *Proceedings of the AAAI Workshop on Auction Mechanisms for Robot Coordination*, pp. 23–30.

Kim, B. H., & Baldick, R. (1997). Coarse-grained distributed optimal power flow. *IEEE Transactions on Power Systems*, *12*(2), 932–939.

Kim, B. H., & Baldick, R. (2000). A comparison of distributed optimal power flow algorithms. *Power Systems, IEEE Transactions on*, *15*(2), 599–604.

Koenig, S., Keskinocak, P., & Tovey, C. (2010). Progress on agent coordination with cooperative auctions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1713–1717.

Kohl, N., & Madsen, O. B. (1997). An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research*, *45*(3), 395–406.

Kok, J., Scheepers, M., & Kamphuis, I. (2010). Intelligence in electricity networks for embedding renewables and distributed generation. *Intelligent Infrastructures*, *42*, 179–209.

Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., & Tambe, M. (2011). Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, *41*, 297–327.

Kucera, D. (2012). Amazon acquires kiva systems in second-biggest takeover. *Availabl e at http://bloom. bg/Gzo6GU*.

Kumar, A., Faltings, B., & Petcu, A. (2009). Distributed constraint optimization with structured resource constraints. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 923–930.

Kumar, A., & Zilberstein, S. (2009a). Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 561–568.

Kumar, A., & Zilberstein, S. (2009b). Event-detecting multi-agent mdps: Complexity and constant-factor approximation. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Kumar, A., & Zilberstein, S. (2011). Message-passing algorithms for large structured decentralized POMDPs. In *Proceedings of the Tenth International*

Conference on Autonomous Agents and Multiagent Systems, pp. 1087–1088, Taipei, Taiwan.

Kumar, R. R., Varakantham, P., & Kumar, A. (2017). Decentralized planning in stochastic environments with submodular rewards.. In *AAAI*, pp. 3021–3028.

Maheswaran, R. T., Tambe, M., Bowring, E., Pearce, J. P., & Varakantham, P. (2004). Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 310–317. IEEE Computer Society.

Mailler, R., & Lesser, V. (2004). Solving distributed constraint optimization problems using cooperative mediation. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 438–445.

Mailler, R., & Lesser, V. R. (2006). Asynchronous partial overlay: A new algorithm for solving distributed constraint satisfaction problems. *Journal of Artificial Intelligence Research*, *25*, 529–576.

Matsui, T., & Matsuo, H. (2012). Considering equality on distributed constraint optimization problem for resource supply network. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 02*, pp. 25–32.

Matsui, T., Matsuo, H., Silaghi, M., Hirayama, K., & Yokoo, M. (2008). Resource constrained distributed constraint optimization with virtual variables.. In *AAAI*, pp. 120–125.

Melo, F. S., & Veloso, M. (2011). Decentralized mdps with sparse interactions. *Artificial Intelligence*, *175*(11), 1757–1789.

Meuleau, N., Hauskrecht, M., Kim, K.-E., Peshkin, L., Kaelbling, L. P., Dean, T. L., & Boutilier, C. (1998). Solving very large weakly coupled markov decision processes. In *AAAI/IAAI*, pp. 165–172.

Miller, S., Ramchurn, S. D., & Rogers, A. (2012). Optimal decentralised dispatch of embedded generation in the smart grid. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pp. 281–288.

Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pp. 234–243. Springer.

Modi, P. J., Shen, W.-M., Tambe, M., & Yokoo, M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, *161*(1), 149–180.

Nagata, T., & Sasaki, H. (2002). A multi-agent approach to power system restoration. *Power Systems, IEEE Transactions on*, *17*(2), 457–462.

Nair, R., Varakantham, P., Tambe, M., & Yokoo, M. (2005). Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *AAAI*, Vol. 5, pp. 133–139.

Nogales, F. J., Prieto, F. J., & Conejo, A. J. (2003). A decomposition methodology applied to the multi-area optimal power flow problem. *Annals of operations research*, *120*(1-4), 99–116.

Petcu, A., & Faltings, B. (2005). DPOP: A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence*, pp. 266–271.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2008). Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pp. 125–132. International Foundation for Autonomous Agents and Multiagent Systems.

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

Ramchurn, S. D., Vytelingum, P., Rogers, A., & Jennings, N. R. (2012). Putting the 'smarts' into the smart grid: A grand challenge for artificial intelligence. *Commun. ACM*, *55*(4), 86–97.

Saisubramanian, S., Varakantham, P., & Lau, H. C. (2015). Risk based optimization for improving emergency medical systems.. In *AAAI*, pp. 702–708.

Satsangi, Y., Whiteson, S., Oliehoek, F. A., et al. (2015). Exploiting submodular value functions for faster dynamic sensor selection.. In *AAAI*, pp. 3356–3363.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012). Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 13–20. International Foundation for Autonomous Agents and Multiagent Systems.

Shieh, E. A., Jiang, A. X., Yadav, A., Varakantham, P., & Tambe, M. (2014). Unleashing dec-mdps in security games: Enabling effective defender teamwork. In *ECAI 2014 - 21st European Conference on Artificial Intelligence*, pp. 819–824.

Tambe, M. (1997). Towards flexible teamwork. *Journal of artificial intelligence research*, *7*, 83–124.

Thiebaux, S., & Cordier, M. (2001). Supply restoration in power distribution system - a benchmark for planning under uncertainty. In *ECP*, pp. 85–96.

Thiébaux, S., Coffrin, C., Hijazi, H., & Slaney, J. K. (2013). Planning with MIP for supply restoration in power distribution systems. In *International Joint Conference on Artificial Intelligence*.

Varakantham, P., Adulyasak, Y., & Jaillet, P. (2014). Decentralized stochastic planning with anonymity in interactions. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 2505–2512.

Varakantham, P., Lau, H. C., & Yuan, Z. (2013). Scalable randomized patrolling for securing rapid transit networks. In *Proceedings of the Twenty-Fifth Innovative Applications of Artificial Intelligence Conference, IAAI*.

Varakantham, P., young Kwak, J., Taylor, M. E., Marecki, J., Scerri, P., & Tambe, M. (2009). Exploiting coordination locales in distributed POMDPs via social model shaping. In *International Conference on Automated Planning and Scheduling*.

Velagapudi, P., Varakantham, P., Scerri, P., & Sycara, K. (2011). 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)*, pp. 955–962.

Wang, C., Handoko, S. D., & Lau, H. C. (2014). An auction with rolling horizon for urban consolidation centre. In *Proceedings of the International Conference on Service Operations and Logistics and Informatics (SOLI)*, pp. 438–443.

Wellman, M., Walsh, W., Wurman, P., & MacKie-Mason, J. (2001). Auction protocols for decentralized scheduling. *Games and Economic Behavior*, *35*(1–2), 271–303.

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

Wu, J., & Durfee, E. (2010). Resource-driven mission-phasing techniques for constrained agents in stochastic environments. *Journal of Artificial Intelligence Research*, *38*, 415–473.

Wu, X., Sheldon, D., & Zilberstein, S. (2014). Rounded dynamic programming for tree-structured stochastic network design. In *AAAI Conference on Artificial Intelligence*, pp. 479–485.

Wurman, P. R., D'Andrea, R., & Mountz, M. (2007). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pp. 1752–1760.

Yeoh, W., & Yokoo, M. (2012). Distributed problem solving. *AI Magazine*, *33*(3), 53.

Yin, Z., & Tambe, M. (2011). Continuous time planning for multiagent teams with temporal constraints. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, Vol. 22, p. 465.

Yin, Z., & Tambe, M. (2012). A unified method for handling discrete and continuous uncertainty in Bayesian Stackelberg games. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 855–862.