2019-04-24

Multi-Robot Task Allocation and Scheduling with Spatio-Temporal and Energy Constraints

Dharini Dutia
Worcester Polytechnic Institute

Multi-Robot Task Allocation and Scheduling with Spatio-Temporal and Energy Constraints

by

Dharini Dutia

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Masters of Science

in

Robotics Engineering Department

May 2019

APPROVED:

Professor Carlo Pinciroli, Advisor

Professor Zhi Li

Professor Andrew Trapp

Acknowledgements

"If it is just us, seems like an awful waste of space."

— Carl Sagan

I would like to express immense gratitude to my advisor Prof. Carlo Pinciroli for providing me with this opportunity, and developing a research mindset which would help me greatly in my academic career. This wouldn't have been possible without his constant encouragement, unwavering support, and scintillating conversations. A special thanks to Prof. Zhi Li for being a great mentor and leading me to my first publication. And committee member Prof. Andrew Trapp for his insightful comments and straightforward advice.

I am grateful to my friends, peers, and colleagues, specifically team member Arsalan, my nocturnal friend Nishan, previous team member Heramb, my labmates Nathalie and Jayam for thorough discussions and enriching me with different perspectives on my work.

Last but not the least, I would like to thank my parents for believing me in at times when even I couldn't; for being my pillar of strength and supporting me in every possible way throughout my life.

Abstract

Autonomy in multi-robot systems is bounded by coordination among its agents. Coordination implies simultaneous task decomposition, task allocation, team formation, task scheduling and routing; collectively termed as *task planning*. In many real-world applications of multi-robot systems such as commercial cleaning, delivery systems, warehousing and inventory management: spatial & temporal constraints, variable execution time, and energy limitations need to be integrated into the planning module. Spatial constraints comprise of the location of the tasks, their reachability, and the structure of the environment; temporal constraints express task completion deadlines.

There has been significant research in multi-robot task allocation involving spatio-temporal constraints. However, limited attention has been paid to combine them with team formation and non-instantaneous task execution time. We achieve team formation by including *quota* constraints which ensure to schedule the number of robots required to perform the task. We introduce and integrate task activation (time) windows with the team effort of multiple robots in performing tasks for a given *duration*. Additionally, while visiting tasks in space, energy budget affects the robots operation time. We map *energy* depletion as a function of time to ensure long-term operation by periodically visiting recharging stations. Research on task planning approaches which combines all these conditions is still lacking.

In this thesis, we propose two variants of Team Orienteering Problem with task activation windows and limited energy budget to formulate the simultaneous task allocation and scheduling as an optimization problem. A complete mixed integer linear programming (MILP) formulation for both variants is presented in this work, implemented using Gurobi Optimizer and analyzed for scalability. This work compares the different objectives of the formulation like maximizing the number of tasks visited, minimizing the total distance travelled, and/or maximizing the reward, to suit various applications. Finally, analysis of optimal solutions discover trends in task selection based on the travel cost, task completion rewards, robot's energy level, and the time left to task inactivation.

Contents

List of Figures

Chapter 1

Introduction

In multi-robot systems [2], each robot plays a small yet significant role in achieving a global task. For example, in distribution centres [3], teams of robots transport the inventory from one part of a large warehouse to another. The robots operate in a shared environment and need to cooperate with each other. Suppose two robots are assigned to collect objects from the same location and do not coordinate, they might collide and block the path for other robots, sabotaging the entire operation. To ensure coordination in such tasks, actions of the robots can be sequenced by task planning. Task planning is a combination of task decomposition, task allocation and scheduling [4]. Task decomposition deals with how to break down the global task into subtasks which the robots can perform. The basic function of task allocation and scheduling algorithms is to determine which robot would do what and when. In other words, find a route which is a sequence of physical locations for the robots to visit. In this work, we are only concerned with task allocation and scheduling.

1.1 Motivation

Real-world tasks are spread out in space and have some kind of temporal and ordering constraints. For example, in delivery systems, the delivery locations are far from each other and a traveling cost is associated with moving from one location to another. Also, robots need to follow a strict timeline in order to accomplish urgent deliveries, which are within a certain time frame. Spatial constraints lay out rules depending on the structure of the environment (for example, walls and obstacles), which limit the mobility of the robots. Temporal constraints specify the time period in which the task must be performed, while task ordering specifies the ordering in which the tasks must be performed. Thus, incorporating these spatial and temporal ordering constraints in the task allocation process poses a challenge. This is because keeping track of schedules of multiple robots is intractable [5]. There is plethora of research in multi-robot task allocation with temporal and ordering constraints. However, limited attention has been paid to integrate team formation with complex temporal models.

It is necessary to integrate robot's energy limitations in the planning module. For instance, in surveillance, monitoring tasks spread over large areas have to be performed continuously for the entire day. A major consideration in deploying a robotic system for such scenarios is the total time it can operate for. It is important for robots to predict their energy level and reach back to base station for recharging. For collaborative tasks such as foraging, collective transport [6], exploration [7], where cooperation is the key to task completion, a robot low on energy can jeopardize entire assignment. Automating the maintenance and recharging process would allow for seamless operation of any commercial or industrial robotic system. Previous work emphasizes the need and complexity of predicting energy consumption for ensuring long-term operation. However, the literature still misses the fusion of time-critical planning, team formation, and on-the-fly recharging.

In this thesis, we propose an approach to incorporate team formation, time scheduling, and energy limitations in task allocation. A combination of all these features would produce highly applicable models to cover a larger domain of real-world problems than existing research.

1.2 Background

Gerkey and Mataric [8] devised a widely accepted taxonomy to categorize Multi-Robot Task Allocation (MRTA) problem according to three criteria. First, they classify robots according to their ability to perform *single-* or *multi-tasks* at a time. Second, they distinguished between tasks that require single robots to be performed, and tasks that require the coordinated effort of a team of robots. Third, considering the time needed to complete a task, they distinguished between *instantaneous* tasks and *time-extended* tasks. This work focuses on single-task robots, and time-extended allocation, with flexibility over single/multi-robots tasks.

A task allocation problem can be approached in two ways: centralized, and distributed methods. In centralized approach, there is only one decision making unit which is external to the robots. The robots just execute the route generated by this unit. In distributed methods, each robot is a decision making unit and communicates with other robots to coordinate their actions. Centralized allocation has the opportunity to produce optimal or near-optimal solutions, thanks to the fact that it is assumed that all the relevant information is gathered at the single decision point. In contrast, distributed approaches tend to produce approximate solutions because the individual decision makers typically work with only local information.

Task allocation is essentially an optimization problem, in which the objective is to find the best assignment of tasks to robots. One of the most common formalisms to capture the task allocation problem from a centralized standpoint is Mixed Integer Linear Programming (MILP) [5; 9]. In MILP, the decision variables (e.g., which tasks to perform and when) can take either integer or binary values, hence the terms "mixed integer". The objective function (e.g., the number of tasks performed) and the constraints (e.g., a task requires at least Q robots) are expressed as linear equations that involve the decision variables. In this thesis, we will use this formalism to solve task allocation problems.

The Team Orienteering Problem with Time Windows (TOPTW) [10] tackles similar problems as in Multi-Robot Task Allocation with Temporal Ordering Constraints (MRTA/TOC) [11]. In Team Orienteering, the goal is to find an optimal path to maximize the number of tasks visited within a time budget. This time budget is suitable to model the global time limit in many time critical problems. Also, each task in the Team Orienteering Problem has a reward associated to it. This makes it possible to encode priorities of tasks in the model. We use Team Orienteering Problem formulation as the basis of our MILP model.

In long-term missions, task planning should also consider the limited lifetime of any individual robot and enforce periodic recharging. Factors affecting energy depletion include distance covered, faulty hardware, unexpected maneuvers and/or uncertainty in execution. Majority of the previous work maps energy as a linear function of the distance covered. As distance covered is directly related to travel time, energy becomes a constraint on how long the robots can operate.

To visit task locations spread over large areas, robots might need to consider recharging along the way at some recharging stations. Automated recharging process can be accomplished in two ways.

- Static recharging stations, considered as a fixed node in the environment.

These nodes are added in the optimized route as per the energy consumption.

- Mobile charger robots, predicting the energy levels of worker robots and tracking their movements. The mobile charger robots meet the worker robots at their low energy state and recharge them either by docking or exchanging batteries.

We consider energy depletion to be linearly varying with distance travelled and the charging stations to be fixed in the environment.

1.3 Problem Statement

We focus on two problems which are extensions of simultaneous multi-robot task allocation and scheduling.

1. We first integrate task allocation, scheduling, team formation and task activation windows. The aim is to find optimal schedule of tasks having variable start times, deadlines and fixed number of agents required to accomplish those tasks. The type of tasks we focus on are spatially organizing tasks, translating to position specific constraints.

2. Further, we look into enabling the robots to operate for larger periods of time. The robots have energy budget limitations which might prevent them from visiting all task locations in space. Depending on the problem, it is desirable to instead visit as many tasks as possible within a given time budget. For long-term autonomous operation, the problem is to find a optimal solution which accounts for energy available in each agent of the team and predict at which point in the route would the robot need to recharge. According to the prediction, charging stations should be added in the route.

1.4 Contributions

We propose a variant of Team Orienteering Problem with Time Windows called as Team Orienteering Problem with Task Activation Windows (TOPTAW), which integrates quota requirements, task activation windows, and task duration in the original model. Additionally, we propose another variant of Team Orienteering Problem called as Team Orienteering Problem with Energy Constraints (TOPEC), to include energy limitations and recharging stations. These formulations are tested on a range of problem sizes, i.e. varying a number of parameters of the environment, to search for recurrent collective behaviours in the optimal solutions.

1.5 Overview of the Thesis

This document is structured as follows: Chapter 2 covers the necessary prerequisite topics like Team Orienteering Problem and comments on relevant literature, and Chapter 3 provide the mathematical formulation of the problem using Mixed Integer Linear Program (MILP). Finally, experimental results on computation time, solution quality and trends in the optimal solutions are summarized in Chapter 4. The conclusion and future scope is discussed in Chapter 5.

Chapter 2

Related Work

This chapter compiles the literature reviewed on the topics related to the task allocation and scheduling problem. In Section 2.1, we provide an introduction to standard problems in operations research domain and their current variants/applications. Section 2.2 covers Multi-Robot Task Allocation (MRTA) literature and its limitations. Finally, Section 2.3 illustrates different approaches for solving the problem of recharging mobile robots and teams of robots.

Summary

Overall, this work belongs to mathematical optimization and swarm robotics. The main keywords are linear programming, operations research, constrained optimization, multi-robot systems, multi-robot task allocation, task scheduling, routing with time windows, energy limitations, recharging robots, and persistent coverage. For a more thorough coverage of the state of the art, refer to [12; 13; 14; 15; 16; 17; 18].

2.1 Operations Research Background

2.1.1 Mixed Integer Linear Programming

In the context of mathematical optimization, the term “programming” refers to the concept of planning. When the equations involved in the optimization problem are linear, we speak of “linear programming”. The technique of linear programming was first invented by the Russian mathematician L. V. Kantorovich and developed later by George B. Dantzig. NEOS Guide [1] provides an optimization taxonomy, reported in Figure 2.1, focused mainly on the subfields of deterministic optimization with a single objective function.

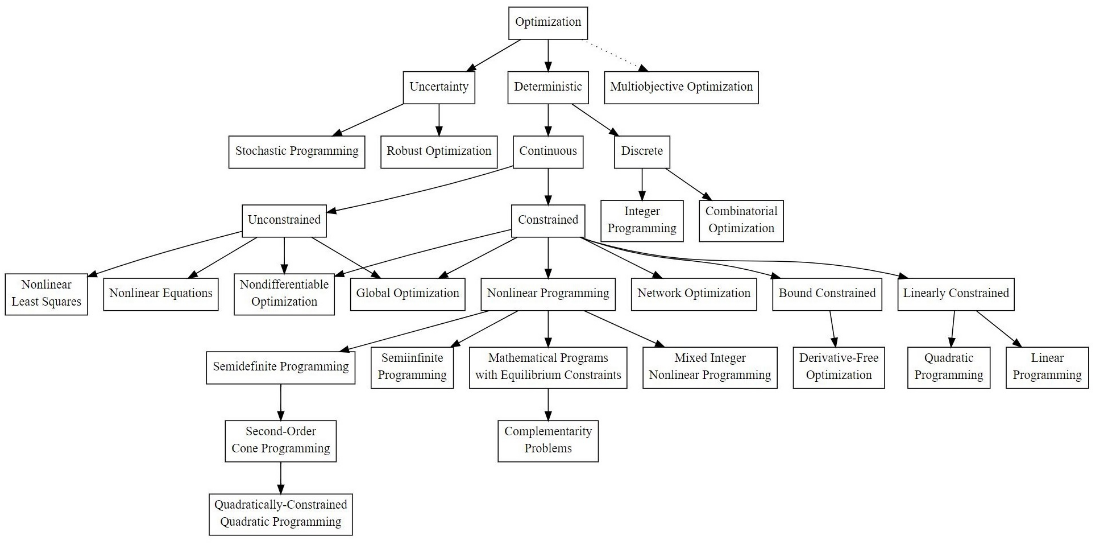


Figure 2.1: Optimization Taxonomy [1]

Linear programming deals with optimization problems that are deterministic, continuous and linearly constrained. A linear programming problem is one in which some function is either maximized or minimized relative to a given set of alternatives. The function to be minimized or maximized is called the *objective function* and the set of alternatives is called the feasible region determined by a system of linear inequalities (constraints). Mixed integer refers to the combination of integers and continuous decision variables. Below is an example of a MILP model.

$$\text{max or min} \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n \tag{2.1}$$

$$st.a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n(\leq, = or \geq)b_1 \tag{2.2}$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n(\leq, = or \geq)b_2 \tag{2.3}$$

$$\cdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n(\leq, = or \geq)b_m \tag{2.4}$$

$$lb <= xj <= ub, \quad \forall j = 1, \ldots, n \tag{2.5}$$

Equation 2.1 represents the objective function of the formulation, where c are the coefficients making the linear objective function and x is the decision variable (output of the solver). This objective is subject to a set of requirements which are enforced as *constraints*. As per these constraints, decisions are made which becomes the value of the *decision variable*. Equations 2.2 - 2.4 combines the equality and inequality constraints of the model. Finally, Equation 2.5 encode the upper and lower value bound of each decision variable.

Since all linear functions are convex, mixed integer linear programming problems are intrinsically easier to solve than non-linear problem types. The flexibility of MILP is what makes them the widely preferred method [19] in process scheduling problems. However, consider a model has n binary variables, there would be 2^n possible configurations to search from. There are several techniques to speed-up the generation of an optimal solution. One of them is the *Branch and Bound* technique. Initially, the integrality restrictions are removed and the problem is solved as a Linear Programming (LP) problem. This is known as *LP relaxation* of the original MILP. Usually, a perfect fit for the original problem is not found by simply relaxing the integer constraints. The next step is to select some variable (restricted as an integer), whose optimal value in the LP relaxation is fractional. This becomes the branching variable and we get two different branches, this process is continued till a solution is found which fits the integer bounds, which can be considered the best solution found so far known as *incumbent*. The generated search tree is explored for other such solutions having better values of the objective function. If they exist we have an *optimality gap*, otherwise we have found our optimal solution. The practitioner can also improve the computation runtime by providing integer bounds in the constraint set of the model despite defining them while the decision variable declaration. This helps in tightening the formulation by removing undesirable fractional solutions, termed as *cutting planes*. Some solvers use pre-existing knowledge of the defined problem and tighten the model to get solutions faster. Additionally, *heuristic* algorithms can be applied to sacrifice optimality and find a solution to the problem faster. This technique provides an initial feasible solution or incumbent to kick-start the search of optimal solution.

Several commercial and open source optimization solvers are available in the user can simply focus on formulating the model rather than dealing with details of actual solution algorithm. Notable softwares include IBM ILOG CPLEX Optimization Studio [20] and Gurobi Optimizer [21]. These have optimization IDE as well as support to model in other languages like C++, Python, MATLAB, R, etc.

Some useful guidelines to formulating a MILP problem, as summarized from [22] are:

1. *If-then statements* The idea is to force both “if” and “then” condition to depend upon another binary decision variable. When the “if” condition is satisfied the binary variable becomes 1 and switches on the “then” condi-

tion.

2. *Enforce at least k out of p constraints* Instead of one binary variable we have “p” binary decision variables. Each of them activates when their corresponding constraint is true. These binary variables can be summed over to equal the value “k”.
3. *Non-linear product terms* To deal with non-linear products, i.e. a multiplication of multiple decision variables, equate the entire non-linear term with a single variable. This variable is bounded by the lower and upper limits of the non-linear product term.

For elaborate description with examples, refer [22].

2.1.2 Team Orienteering Problem

The Orienteering Problem (OP) was introduced in 1987 by Golden, Levy and Vohra [23]. This problem differs from the Traveling Salesman Problem (TSP) by two major points: first the tasks have to be performed within a global deadline, termed as the given time budget T_{max}; second there is a reward associated with each task and the robots are supposed to maximize the rewards collected by visiting the tasks. Vansteenwegen et al. [10] presents an overview of benchmark instances of the Orienteering Problem (TOP). The practical applications, solution approaches and open problems of TOP are discussed. Gunawan, Lau and Vansteenwegen [12] surveys the research on all the variants of TOP and lists their best known solutions.

Van der Maerwe et al. [15] formulate a variant of the Team Orienteering Problem with Time Windows (TOPTW) called the Cooperative Orienteering with Time Windows (COPTW). This problem includes team formation by ensuring that the servicing tasks cannot start before all the required vehicles have arrived. Yu et al. [24] introduces the Correlated Orienteering Problem (COP) formulated as Mixed Integer Quadratic Programming (MIQP) model to find optimal tours for persistently monitoring a spatially correlated field. Spatial correlations observed are time-invariant, for example traffic congestion at one intersection would be correlated to the same situation at another intersection. The idea is to record and observe these correlations. A quadratic score function is added in the objective to capture these spatial correlations making the COP a non-linear extension of the Orienteering Problem.

2.2 Task Allocation and Scheduling Background

Gini et al. [5], and Nunes et al. [11] categorize the extensive research present in the multi-robot task allocation domain and help identifying possible solutions to this problem. This work emphasizes the importance of problems in which tasks are allocated according to time, distance and priority. The requirements change with the applications: search and rescue focuses on quick execution; deadlines matter in surveillance; in disaster response, proper ordering is necessary. All the variants of this problem consider a set of tasks mapped as a node on a acyclic graph, used to find an optimal route/schedule.

Proposed approaches in literature for task allocation are categorized into centralized, distributed or market-based approaches. In centralized approach, there is only one decision making unit which is external to the robots. In distributed methods, each robot is a decision making unit and communicates with other robots to coordinate their actions. Market-based approaches use auctions where the robots place bid on the tasks, and the lowest cost bid gets that task.

Multi-robot Task Allocation (MRTA) problem can be formulated as a Mixed Integer Linear Programming (MILP) model, and can be decomposed by different strategies, to improve scalability. The problem is viewed as a constraint optimization problem in Koes et al. [13]. The formulation attempts to simultaneously allocate tasks and determine a continuous time schedule to maximize team utility. The tasks considered are associated with a finite duration and specific abilities of the robots in a heterogeneous swarm. Task activation window is not considered in this model. But several real-valued timing decision variables for travel, starting execution, waiting and working are used. This model is the core of a heuristic solver, Anytime Scheduling, which allows re-planning under a fixed planning horizon.

Real-time tasks have a upper and lower time bounds for starting task execution. Gombolay et al. [14] considers these interval temporal constraints, along with spatial proximity restrictions on robots, to formulate a MILP model. The focus is on sequencing of tasks and not on preparing an exact time schedule at which the robots are supposed to perform a task. The main idea of this work is to blend real-time processor scheduling and MILP program to develop a fast task sequencer named Tercio. In this sequencer, MILP is solved by a third-party optimizer, which becomes input to a fast task sequencer. This hybrid approach is tested on KUKA Youbots [25] for assembling a mock airplane fuselage.

Prorok et al. [26] look into the problem of finding an optimal distribution of multi-task robots capabilities among the set of multi-robot tasks. A differentiable objective function minimized by gradient descent is used. The optimization returns transition rates with which a particular edge is chosen.

Market-based approaches are competitive schemes where each agent places a bid on the task as per its self-determined worth. Gerkey and Mataric [27] developed a distributed, auction-based, dynamic task allocation technique called MURDOCH. As tasks appear randomly, an agent broadcasts a message to the swarm with information regarding the attributes of the task. Robots place bids on this task and the auctioneer processes and assigns it. This whole process is dependent on distributed communication, however the method encompasses message drops and communication lag. The optimality of such assignments is not discussed. It is tested for simple single robot tasks as well as collective box-pushing, for which the constraints were customized. The whole system is independently allocating, executing and recharging. However, this approach claims to be dynamic, yet there is no provision for reintroducing tasks if the agents were busy the first time around. It is dynamic in the sense that it is reactive to changes in the environment like robot failures. Nunes and Gini [28] developed auction-based algorithm, TeSSI, for the type of tasks having temporal constraints, which experimentally performs better than other decentralized approaches. The bids placed to the central auctioneer, have individually calculated the travel cost, smallest of which would be allocated the task, thus satisfying the global objective.

Das et al. [29] present a Consensus-based Parallel Auction and Execution (CBPAE) algorithm for assisting the elderly by fall detection, food delivery, medication reminder, cleaning and surveillance. The bidding process for the next task is done simultaneously to the execution of the current task. Also, the task attributes and statuses are locally stored, hence it is possible that some robots might be bidding based on out-dated information. The algorithm recognizes this situation as deadlock and claim to solve it by consensus which increases the communication overhead.

MILP solutions cannot possibly account for uncertainties in execution and system dynamics. Probabilistic models can be developed including such uncertainties. Hanna [30] proposes to tackle the uncertainty in execution by a two-step process: task selection by Markov decision process (MDP) and allocation using auctions. The tasks considered here do not have any temporal or ordering conditions for simplicity. The execution probability distribution help estimate the possible amount of resources would be utilized to perform the task. An expected reward is calculated for all such possibilities and the decision is made to maximize this reward. After a local choice is made, bids are placed for task allocation which is not discussed in detail.

Another distributed strategy to allocate tasks as per task priorities is presented by Khaluf and Ramming [31]. Task priorities are determined from task deadlines. The objective is to select tasks using allocation probability matrix such that the tasks are executed by the deadline and the number of robot trials required to accomplish the task are reduced. By robot trials it is implied that a task require more than one trial to be completed. Each task is divided into parts or trial to be accomplished within a deadline. Hence the total number of robots required to finish the task is the product of k parts of the tasks and number of robots required for each trial. Using the success probability of the event, where n robots finish k parts of a task within a deadline, the total number of robots N required to complete the task, is determined. A major drawback of this work is that there is minimum interaction between robots.

2.3 Recharging Robots Background

Kanan et al. [32] first introduced the Autonomous Recharging Problem (ARP) for mobile robots to consider their energy level before scheduling tasks. The paper describes in detail the concepts associated with ARP. The robots should be aware of their energy level, and take it into account while deciding which tasks to perform. Recharging is provide using two ways: static recharging stations or mobile recharging stations. Static recharging stations are dispersed at fixed locations in the environment. Multiple robots coordinating the visit to static recharging stations is a subproblem of ARP. Mobile recharging approaches involve a moving recharger robot having different mechanisms for docking with the worker robot to recharge it. It is the responsibility of mobile recharger robots to predict and track the energy levels of worker robots.

Cheng et al. [33] estimate the energy consumption by approximating a non-linear energy function depending on a drone's payload. Other reviewed research map energy depletion as a linear function of distance travelled.

Sundar and Rathinam [34] considered the routing problem for a fuel-constrained Unmanned Aerial vehicle (UAV), where the environment also has refueling depots and the UAV can recharge at any of them before it runs out of fuel. A mixed integer Linear Programming (MILP) formulation as well as a heuristic is presented to solve the problem. In [35], Sundar et al. devise an approximation algorithm for the same problem, along with construction and improvement heuristics. Levy et al. [36] discuss a variable neighbourhood search based approach for the above problem. All of these approaches assume that all the task locations are accessible and can be visited without any time constraints. Hamann et al. [37] discuss the problem of efficiently positioning static charging stations in an warehouse.

Mitchell et al. [16] consider the problem of persistent coverage with multiple fuel constrained robots. A MILP formulation for the multi-robot version for fuel constrained robots is presented, along with a heuristic approach. To deal with the problem of tasks with unknown task costs, subsequent tasks in a cycle are dropped without considering the optimality of the path. Mitchell et al. [38] develop a greedy algorithm for deciding which targets to drop in a stochastic setting. A MILP model for stochastic task costs is formulated by introducing a chance constraint.

Mathew et al. [39] discuss rendezvous locations for charging robots with fuel-constrained robots, assuming that paths of the fuel-constrained robots are previously known. The recharging problem is formulated as a generalized travelling salesman problem (GTSP), and then transformed into an Asymmetric Travelling

Salesman Problem (ATSP) using Noon-Bean Transformation [40]. A multi-robot version of this problem and its applications for persistent long-term coverage is discussed as an extension in [17]. Maini et al. [41] find routes for the charger robots and fuel-constrained worker robots in a greedy algorithm.

Kamra et al. [18] formulate a Mixed Integer Quadratic Program (MIQP) for timed deliveries by delivery robots to fuel constrained worker robots. The delivery robots have to return to a control center if their fuel is depleting. A user-defined control parameter is used to decide late/early deliveries vs travel costs. To improve scalability, approximation algorithms for MIQP problem is developed in [42].

Chapter 3

Methodology

This chapter provides in-depth details of generating optimal solutions by mathematical modelling of the proposed variants of the Team Orienteering Problem. These include simultaneous task allocation and scheduling for task activation windows in the first variant and energy limitations in the second. Section 3.1 defines the problem mathematically, Section 3.2 and Section 3.3 present the core formulation. Section 3.4 adds task temporal deadlines and team formation, and Section 3.5 adds energy limitations, to the core formulation.

3.1 Problem Formulation

The Simultaneous Multi-Robot Task Allocation and Scheduling problem can be expressed as an optimization problem using Mixed-Integer Linear Program (MILP). Consider an environment $\mathcal{E} \in \mathbb{R}^2$ in a 2-D Euclidean space, and $T \in \mathbb{R}^+$ tasks spread around in $\mathcal{E}$. This environment is mapped on a graph $\mathcal{G}$ consisting of set of nodes $N = S \cup T \cup E$ representing the start position, task locations and robot end positions, respectively. The goal is to assign the tasks to K robots. To achieve that we need the following inputs:

- Start location S and desired end position E of robots
- Location of each task p_i in space
- Distance and time t_{ij} required to travel the edges of the fully-connected graph $\mathcal{G}$, assuming unit velocity of all the robots
- Total time budget for the system, T_{max}

The choice of decision variables changes with the application and structure of the environment, as a result changes the objective function. The core decision variables (outputs) of this problem are:

- $x_{ijk} \in \{0, 1\}$: Binary. 1 if, for robot k, vertex i is followed by a vertex j, otherwise 0.
- $y_i \in \{0, 1\}$: Binary. 1 if vertex i is visited by any robot k.

The underlying Team Orienteering Problem can thus be stated as, *Given a set of task locations T, robots start location S and end location E, find the optimal paths for each robot $r \in K$ such that the robots maximize the number of tasks visited in a given time budget T_{max}.*

Having stated this, we add time window and energy constraints. Additional input information and decision variables are explained in subsequent sections (Section 3.4 & 3.5). Note that as the Orienteering Problem is NP-Hard [23], the variants are NP-Hard too.

3.2 Objective Function

The simplest objective is to maximize the number of tasks performed, which maps to the binary decision variable $y_i, \forall i \in T$. This can be formalized as:

$$\max \sum_{i \in T} y_i \tag{3.1}$$

If each task is associated with a reward or score, $R_i, \forall i \in T$, the objective becomes maximizing the overall score. Here, y_i is 1 if task i is visited by any robot k. So, it does not matter how many robots visit task i, the reward would considered only once. Mathematically:

$$\max \sum_{i \in T} R_i y_i \tag{3.2}$$

If the application involves two objectives with unequal priorities, i.e. one of the objective is more important than the other, a scaling factor can be used. Here, we want to maximize the tasks visited while minimizing the total distance traveled. To get the total distance traveled, we can use the binary decision variable x_{ijk}. If an edge ij is present ($x_{ijk} = 1$) and the distance between the nodes i and j is represented by t_{ij}, the distance traveled while traversing that edge would become $t_{ij}x_{ijk}$. Summing this for all robots and edges we get the total distance traveled. This is scaled by a factor γ, and is always kept < 1. Otherwise, it would overpower the objective to visit tasks and end up making the distance traveled to be 0. It is then subtracted from the main objective, forming a multi-objective function, mathematically represented as:

$$\max \left(\sum_{i \in T} y_i - \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} \gamma t_{ij} x_{ijk} \right), \quad \gamma t_{ij} x_{ijk} < 1, i \neq j \tag{3.3}$$

Note these functions use binary decision variables. The problem might require a continuous variable to be maximized or minimized. For example, maximizing the time left to task inactivation after the robot's arrival. In other words, given the task end time a_i^{end}, and robot arrival time s_{ijk}, the time left to complete the task is $a_i^{end} - s_{ijk}$. Summing this over all robots and edges, we get:

$$\max \left(\sum_{k \in K} \sum_{i \in N} \sum_{j \in N} x_{ijk}(a_i^{end} - s_{jk}) \right), \quad i \neq j \tag{3.4}$$

This way single objective and multi-objective functions can be included in a model. The objective functions used for respective variants is further commented on in Section 4.1.

3.3 Core Constraints

For the given decision variables and problem statement described in Section 3.1, the following constraints complete the core formulation for this work, which is based on the Team Orienteering Problem formulation.

To ensure path integrity, all robots should start and end at the given start and end nodes. Equation 3.5 ensures the number of robots leaving the start node are equal to the total number of robots. Similarly, Equation 3.6 ensures the number of robots entering the end node are equal to the total number of robots.

$$\sum_{k \in K} \sum_{j \in T \cup E} x_{sjk} = |K|, \qquad \forall s \in S, k \in K \tag{3.5}$$

$$\sum_{k \in K} \sum_{i \in S \cup T} x_{iek} = |K|, \qquad \forall e \in E, k \in K \tag{3.6}$$

Equation 3.7 represents connectivity constraints. If a robot enters a node, it should leave it. Equations 3.8 and 3.9 maps the edge selection decision variable x_{ijk} to the task selection decision variable y_i to impose connectivity and update the selection.

$$\sum_{i \in S \cup T} x_{ihk} = \sum_{j \in T \cup E} x_{hjk}, \qquad \forall h \in T, \forall k \in K, \tag{3.7}$$

$$\sum_{i \in S \cup T} x_{ihk} \leq y_h, \qquad \forall h \in T, \forall k \in K \tag{3.8}$$

$$\sum_{j \in T \cup E} x_{hjk} \leq y_h, \qquad \forall h \in T, \forall k \in K \tag{3.9}$$

The task visitation is not a hard constraint, in other words it is not required to definitely visit all the tasks in T. Equation 3.10 allows the robots to drop some tasks by allowing the task selection decision variable y_i to be less than or equal to 1.

$$y_i \leq 1, \qquad \forall i \in T \tag{3.10}$$

This implies if there is not enough time to visit all tasks, visit whichever tasks are possible to complete. So the solver would still produce an optimal solution with fewer tasks, instead of no solution.

Time limit constraint characterizes a problem as a variant of the orienteering problem. Equation 3.11 implies that the total time (= distance as the robots have unit velocity) traveled for the entire assignment should be less than or equal to T_{max}.

$$\sum_{i \in N \setminus s} \sum_{j \in N \setminus s} t_{ij} x_{ijk} \leq T_{max} \quad \forall k \in K, i \neq j \tag{3.11}$$

This constraint ensures that each robot returns to the end location before the time T_{max} runs out.

3.4 Activation Windows

Consider each task $i \in T$ have to be performed within an *activation window*, consisting of task start time a_i^{start} and task end time a_i^{end}. Each task require *task duration*, T_{Di} time to finish the task. Also, each task has attributes related to *quota*, Q, which is a set of positive whole numbers indicating the number of robots required to perform the task.

The Team Orienteering Problem with Task Activation (Time) Windows (TOPT AW) is stated as, *Given a set of task locations T, robots start location S and end location E, quota requirements, task duration and task activation windows, find the optimal paths for each robot $r \in K$ such that the robots maximize the number of tasks performed within the task activation window $[a^{start}, a^{end}]$ by fulfilling the quota Q for the duration T_{Di}, in a given time T_{max}.*

In addition to the decision variables defined in Section 3.1, we need a decision variable which represents time to schedule the tasks. There are two possible ways to model that: discrete representation of time or continuous representation of time [19]. Discretizing time gives an approximate solution to the actual problem. To achieve a sufficient degree of accuracy, discretization of time must be done by keeping the intervals as small as possible. The computation time increases if the number of time intervals is large. Consequently, there is a tradeoff between solution accuracy and time taken to find that solution. On the other hand, continuous-time models allow to use any time instance on the scale, making it easier to introduce the concept of task start and end time.

Thus, we define: s_{ik} : Arrival time at vertex i for robot k (refer Figure 3.1).

Equation 3.12 ensures that the required number of robots visit the task by making the summation of edge selection decision variable x_{ijk} over all $k \in K$ to be equal to the quota for that task.

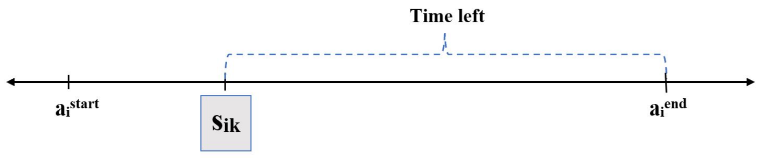


Figure 3.1: Time activation window of a single task i

$$\sum_{k\in K}\sum_{i\in S\cup T} x_{ijk} = q_j y_j, \qquad \forall j \in T \tag{3.12}$$

It is important for each of the robots in the team to reach the task at the same time. Equation 3.13 calculates the timeline for each robot. The arrival time at task j for robot k is calculated by taking the arrival time of previously visited task i by that robot, add the duration of task T_{Di} and time required to travel from task i to j, i.e. t_{ij}. The whole travel time is bounded by T_{max} in Equation 3.14, this replaces Equation 3.11 in the core constraints.

$$s_{ik} + t_{ij} + T_{Di} - s_{jk} \leq M(1 - x_{ijk}), \qquad \forall i, j \in T, \forall k \in K \tag{3.13}$$

$$\sum_{i\in S\cup T}\sum_{j\in T\cup E} (t_{ij} + T_{Di})x_{ijk} \leq T_{max}, \qquad \forall k \in K, i \neq j \tag{3.14}$$

where M is a large constant.

Equations 3.15 and 3.16 restrict the robot's arrival to the task activation window. The maximum arrival time at any task would be the task end time minus the task duration, so the robots would get enough time to accomplish the task even if they arrive at the last moment.

$$a_i^{start} \leq s_{ik}, \qquad \forall i \in T, \forall k \in K \tag{3.15}$$

$$s_{ik} \leq a_i^{end} - T_{Di}, \qquad \forall i \in T, \forall k \in K \tag{3.16}$$

3.5 Energy Limitations

Energy limitations are added by tracking energy levels of the robots. Initially, every robot has maximum *energy level* L. We assume the energy consumption to be distance-based, with each robot using one unit of energy per unit of distance covered. Thus, the robot can visit *recharging depots* D along the way to recharge themselves and visit locations which were otherwise infeasible.

The Team Orienteering Problem with Energy Constraints (TOPEC) is stated as, *Given a set of task locations T, depot locations D, robots start location S and end location E, find the optimal paths for each robot $r \in K$ with energy level L such that the robots maximize the number of tasks visited by recharging along the way, while minimizing the total distance traveled, in a given time T_{max}.*

We need two additional decision variables to represent the capacity and energy levels. Thus, we define:

- p_{ijk} to indicate the number of units held as robot k traverses the edge from i to j.
- $r_i \in [0, L]$: Amount of energy left in the robot when it visits task i.

These constraints are similar to the formulation in Mitchell et al. [16]. Note that the energy level decision variable r_i is not separate for each robot, because only one robot is visiting a task. So whichever robot visits the task i, the energy remaining in that robot is represented by r_i.

The capacity and flow constraints serve as subtour elimination constraints as they ensure that the set of tasks assigned to each robot comprises of a single closed tour.

$$\sum_{i \in N \setminus s} (p_{sik} - p_{isk}) = \sum_{i \in T \setminus s, j \in N} x_{ijk}, \qquad \forall k \in K \tag{3.17}$$

$$\sum_{j \in N \setminus \{i\}} (p_{jik} - p_{ijk}) = \sum_{j \in N} x_{ijk}, \qquad \forall i \in T \setminus s, \forall k \in K \tag{3.18}$$

$$\sum_{j \in N \setminus \{i\}} (p_{jik} - p_{ijk}) = 0, \qquad \forall i \in D \setminus \{s\}, \forall k \in K \tag{3.19}$$

$$0 \leq p_{ijk} \leq |T| x_{ijk}, \qquad \forall i, j \in N, \forall k \in K \tag{3.20}$$

Equation 3.17 captures the flow through the starting node. Here, the robot acquires $\sum_{i \in T \setminus s, j \in N} x_{ijk}$ units, which is equal to the number of targets assigned to the robot k. This capacity is then reduced by 1, as per Equation 3.18, if the corresponding target is contained in the robots assigned set. As the robot passes through recharging depots, this target capacity is prevented from changing, as given by Equation 3.19. This prevents recharging detours from disrupting the continuity of a robots tour. Equation 3.20 ensures that the target capacity for each robot does not exceed $|T|$.

Energy constraints ensure that the robot does not run out of energy as it traverses its route.

$$r_j - r_i + f_{ij} \leq M(1 - x_{ijk}), \qquad \forall i, j \in T, \forall k \in K, i \neq j \tag{3.21}$$

$$r_j - r_i + f_{ij} \geq -M(1 - x_{ijk}), \qquad \forall i, j \in T, \forall k \in K, i \neq j \tag{3.22}$$

$$r_j - L + f_{ij} \leq M(1 - x_{ijk}), \qquad \forall i \in D, \forall j \in T, \forall k \in K \tag{3.23}$$

$$r_j - L + f_{ij} \geq -M(1 - x_{ijk}), \qquad \forall i \in D, \forall j \in T, \forall k \in K \tag{3.24}$$

$$r_i - f_{ij} \geq -M(1 - x_{ijk}), \qquad \forall i \in T, \forall j \in D, \forall k \in K \tag{3.25}$$

$$f_{ij} x_{ijk} \leq L, \qquad \forall i, j \in D, \forall k \in K, i \neq j \tag{3.26}$$

Equations 3.21 and 3.22 map the energy consumption to the edge selection decision variable, in other words if a certain edge is included in the solution, only then the condition would be active. Precisely, $r_i - r_j = f_{ij}$ if $x_{ijk} = 1$, where f_{ij} is the energy required to travel the distance from task i to j. This pair of constraints ensures that the energy lost between two nodes is equal to the energy cost of travelling between them.

Equations 3.23 and 3.24 establish the condition that the energy level at a target visited after leaving a depot is equal to the energy capacity minus the energy cost of traversal. They depict the condition that $L - r_j = f_{ij}$.

Equation 3.25 restricts the energy lost in approaching a depot to being at most equal to the cost of travel from the preceding target. In other words, it represents the condition that $r_i \geq f_{ij}$ if x_{ijk} is active. Equation 3.26 restricts direct paths between recharging sites to exist only between sites atmost L distance away (assuming one unit of energy consumption per unit of distance covered).

Chapter 4

Experimental Evaluation

This chapter provides details of the implementation and summarizes the experimental results. Section 4.1 explains the experimental setup and implementation details. Section 4.2 determines the effect on model runtime with various parameters and Section 4.3 determines the effect on solution quality with change in various parameters. Finally, the Section 4.4 discusses the notable findings from the experiments.

Summary

All the experiments are run on a 100x100 unit sq. environment using the Python module of Gurobi Optimizer software, implementation of which can be found at [43]. Two performance parameters are used to quantify the results: *computation time*, the total runtime to generate optimal solution, and *solution quality*, the number of tasks visited. A general observation is the exponential increase in computation time with number of tasks/nodes. Quality tends to improve with increasing the total time budget T_{max}. A noteworthy trend in the optimal solutions is that the robots tend to arrive at the tasks at the very last moment.

4.1 Experimental Setup

The two variants of the Team Orienteering Problem are implemented and tested separately. Both of these implementations are done using the Python module of Gurobi Optimizer [21]. The implementation and usage information can be found at the github repository [43].

The formulations are tested on a 100x100 unit sq. environment. The task locations, depot locations, robot start and end locations, all are dispersed randomly within this environment. Details on other input data belonging to respective variant is discussed in the consecutive paragraphs.

The Team Orienteering Problem with Task Activation (Time) Windows, (TOP TAW), uses Equation 3.2 as its objective function and core constraints in Section 3.3 plus the time window constraints in Section 3.4. The quota requirements Q is chosen randomly in $[1, K-1]$, where K is the total number of robots. Task duration is selected randomly in the range $[1, 10]$ secs. The maximum start time a_i^{start} is set to $T_{max}/2$ and the interval between start and end time are generated randomly in $[1, 100]$ to produce solvable problems. Lastly, rewards are assigned randomly in the range $[1, 100]$.

The Team Orienteering Problem with Energy Constraints, (TOPEC), employs Equation 3.3 as its objective function and core constraints in Section 3.3 plus the energy constraints in Section 3.5. Maximum energy level L, and total time budget T_{max} is varied to infer trends in data.

More information of the environment are discussed in details in the subsequent sections.

4.2 Computation Time

Computation time corresponds to the total runtime of the Gurobi solver spent to find the optimal solution.

TOPTAW is executed for 5 sets of experiments with each set varying the problem size, generating in total 113 data points. The problem size consists of robots in range $(2, 3, 4, 5, 8, 10)$ and the tasks increase from 5 to 50. The aim is to study the relationship between computation time and problem size. Figure 4.1 compiles the plots for the selected range of robots. The x-axis represents the number of tasks, and the y-axis converts the computation time in $[0.003, 60.8]$ secs to log scale. The general trend is increase in computation time with number of tasks. As the number of robots increase, even lower number of tasks require more computational runtime.

TOPEC is implemented for a single set of experiments with each set varying the problem size, generating 40 data points on an average. The problem size consists of robots in range $(2, 3, 5)$ and the nodes in range $(5, 10, 15, 20)$. The number of nodes in the environment equates to the sum of number of tasks and number of depots. The aim is to study the effect of increasing number of nodes on the computation time. Figure 4.2 reports the plots for the selected range of robots. The x-axis represents the number of nodes, and the y-axis averages the computation time for each nodes. The time is mapped to log scale in the range $[0.01, 8745]$ secs. If Figures 4.2(a) and 4.2(b) are compared it can be observed that the computation time increases with the number of nodes. An important thing to note in Figure 4.2(b) is the problem instance with 5 robots, 7 tasks, and 13 depots, i.e. 20 nodes is not fully optimal, it reached an optimality gap of 0.03% in 8475 secs. This is because for higher number of robots it takes more time to compute an optimal solution which satisfy all the constraints.

Next, we want to determine the effect of maximum energy level L and total

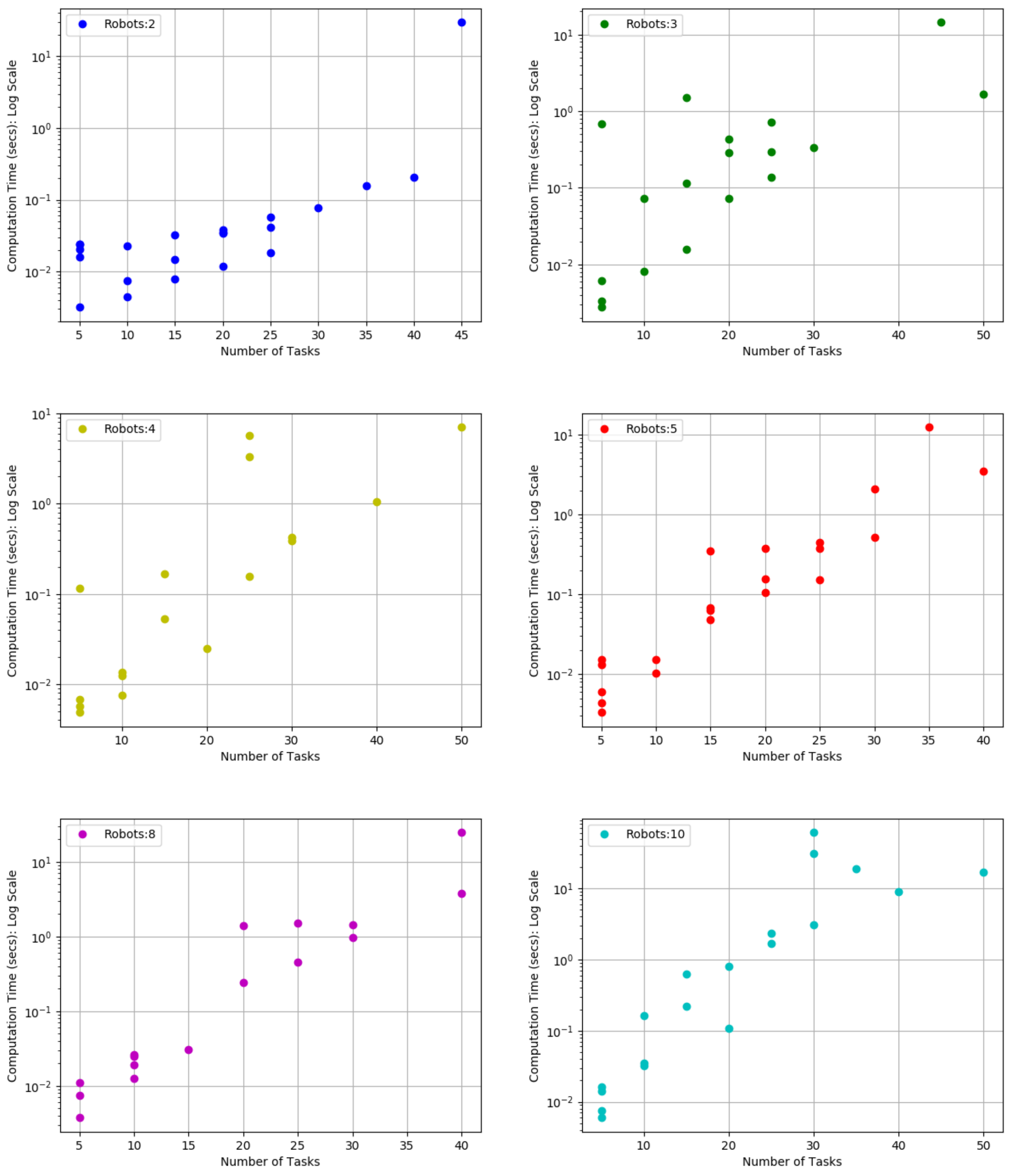


Figure 4.1: (TOPTAW) Effect on computation time with change in number of tasks, number of robots increase from top left to bottom right

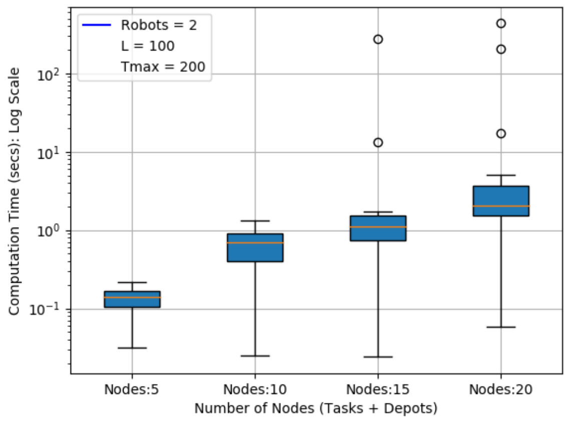
time budget T_{max} on the computation time. Four sets of experiments are performed, each with a unique combination values of L and T_{max}. Refer to Figures 4.2(c) and 4.2(d) to find that with constant $L = 100$, changing T_{max} from 100 to 200 reduces the computation time by a log factor of 10. This might be the case because in the grid environment with width equal to the maximum energy level, more distance can be traveled without recharging thus generating solution quickly. Conversely, in Figures 4.2(e) and 4.2(f) with constant $L = 75$, changing T_{max} from 100 to 200 increases the computation time by a factor of 10. The energy level being low forces more recharging diversions creating more constraints to satisfy, increasing the computation time. Note in Figure 4.2(f), the problem instance with 3 robots, 12 tasks, and 8 depots, i.e. 20 nodes isn't fully optimal, it reached an optimality gap of 0.04% in 7915 secs. This is because the problem size is large and requires more computation time to satisfy the constraints.

4.3 Solution Quality

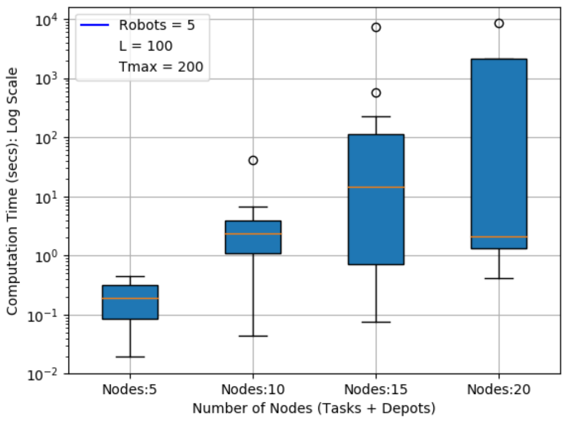
The quality of the solution is defined by the number of tasks visited. Experiments are performed to determine what factors affect the solution quality and why.

A sufficiently large problem size of 15 tasks and 5 robots is selected for TOPTAW to compare the relation between solution quality and T_{max}. The time budget is varied from 100 to 2000 secs, having 7 distinct data points. Figure 4.3 suggests the number of tasks visited heavily depends on the time given to the entire assignment (T_{max}). If sufficient time is allotted, then more tasks can be performed, otherwise tasks are dropped.

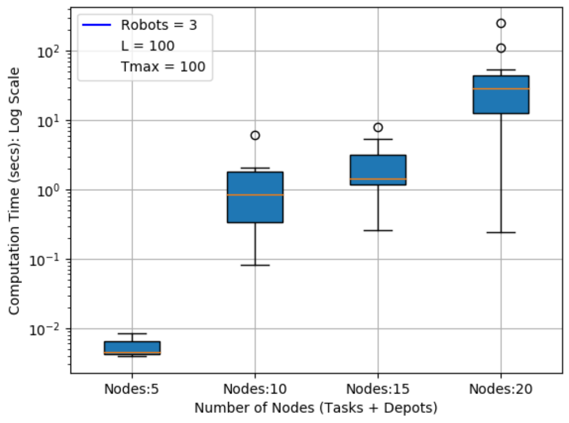
Similar analysis can be performed on the TOPEC formulation. The data used here is from the same four sets of experiment data obtained by varying L and T_{max} previously used to study computation time. Consider Figure 4.4(a), the



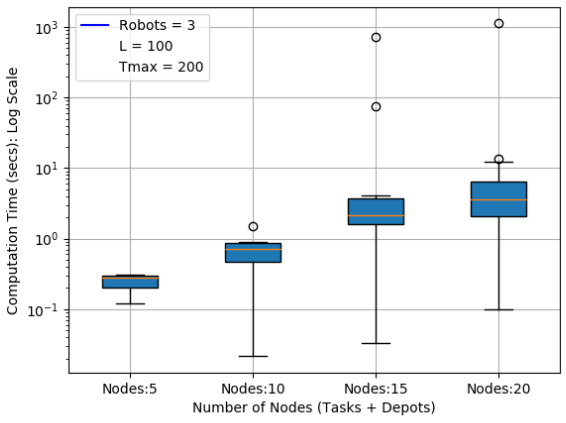
(a)



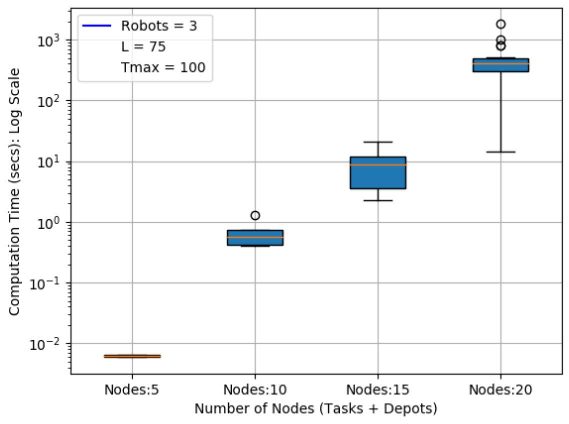
(b)



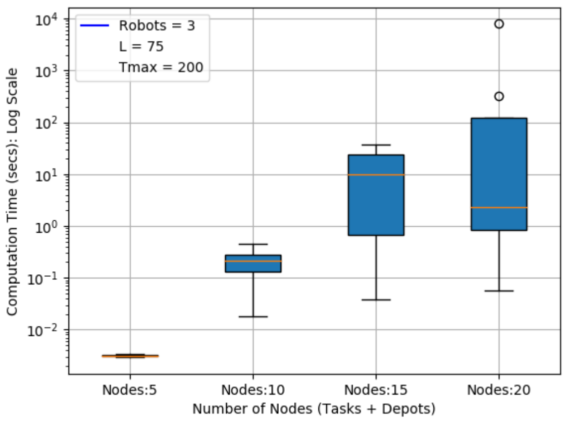
(c)



(d)



(e)



(f)

Figure 4.2: (TOPEC) Effect on computation time with increasing number of nodes (refer plots in top row); Effect on computation time with varying max energy level, L and time budget, Tmax (refer bottom two rows)

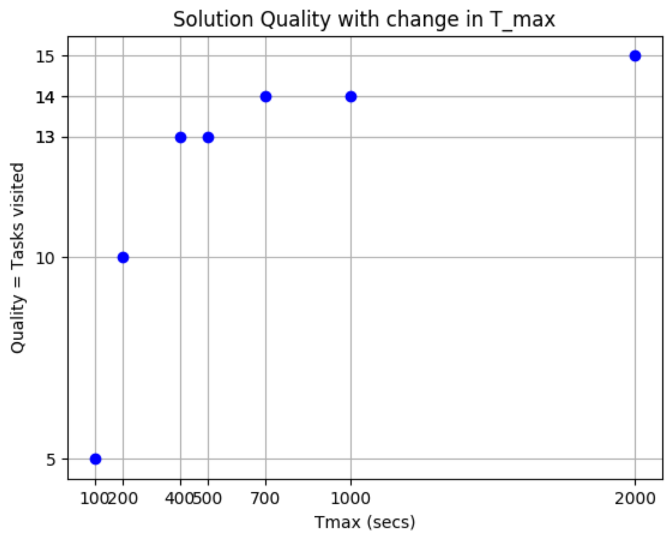
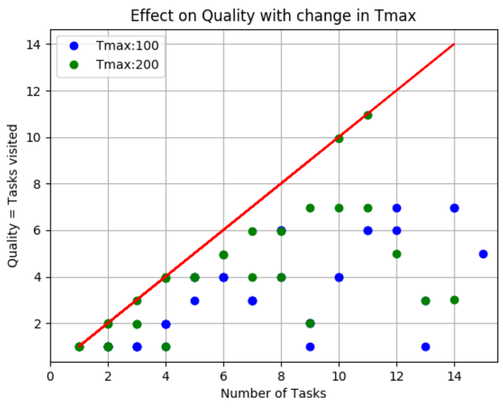
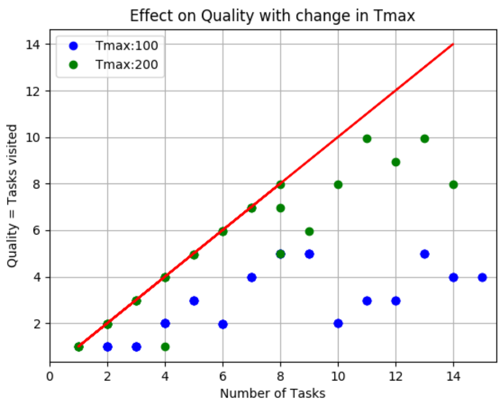


Figure 4.3: (TOPTAW) Effect on solution quality with varying time budget $T_{\max}$

solution quality for $T_{max} = 200$ is higher for each problem set than $T_{max} = 100$. As the energy level is higher in Figure 4.4(b), the formulation performs better than the previous case for same number of robots and tasks. The red line acts as a reference for ideal scenarios where no tasks are dropped.



(a) Maximum energy level, L=75



(b) Maximum energy level, L=100

Figure 4.4: (TOPEC) Effect on solution quality with varying $T_{\max}$ for given energy level L

4.4 Trends in Optimal Solutions

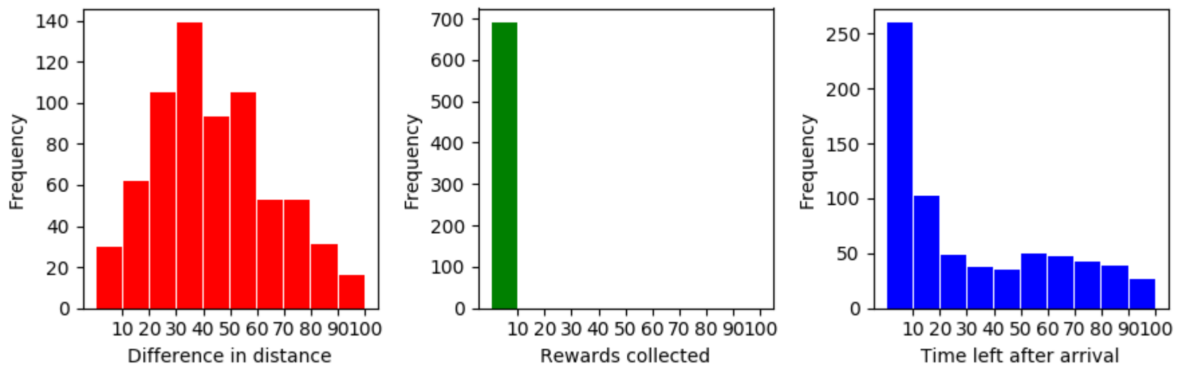
We now search for trends in optimal selection of tasks, in other words, given certain restrictions how task selection is affected. For that we gather all the collected data, and record the distance between each consecutive nodes of the optimal route generated. We then take this stream of data and calculate the frequency of instances for which a certain range of distances repeat. As the grid size is 100x100, the range is $[0, 100]$.

For TOPTAW (see Figure 4.5) we record additional information: the rewards associated to the visited nodes and the time left to complete the task after the robot's arrival. Figure 4.5(a) presents the three frequency plots generated from 690 data points. The distance graph shows the random distribution of tasks in space. A notable inference is that the robot's tend to arrive at the tasks at the very last moment, as the time left is < 10 secs for most of the tasks visited.

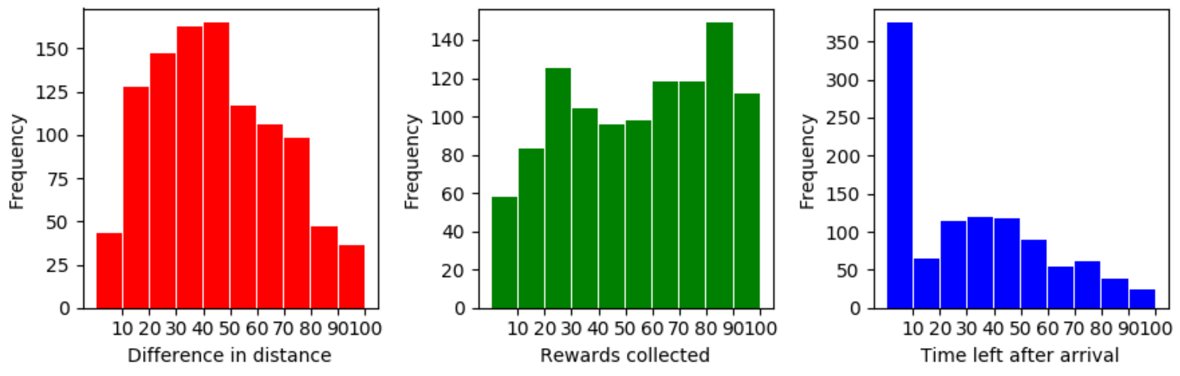
When rewards are introduced in another experiment generating 1061 data points. In Figure 4.5(b)), we can see that the robots try to visit tasks that are far away, and try to arrive earlier to collect those rewards.

To prevent the tasks from being performed at the last moment, we exploited the objective function in Equation 3.4 in this problem and collected the 381 data points. Figure 4.5(c) shows improvement in the time left after arrival.

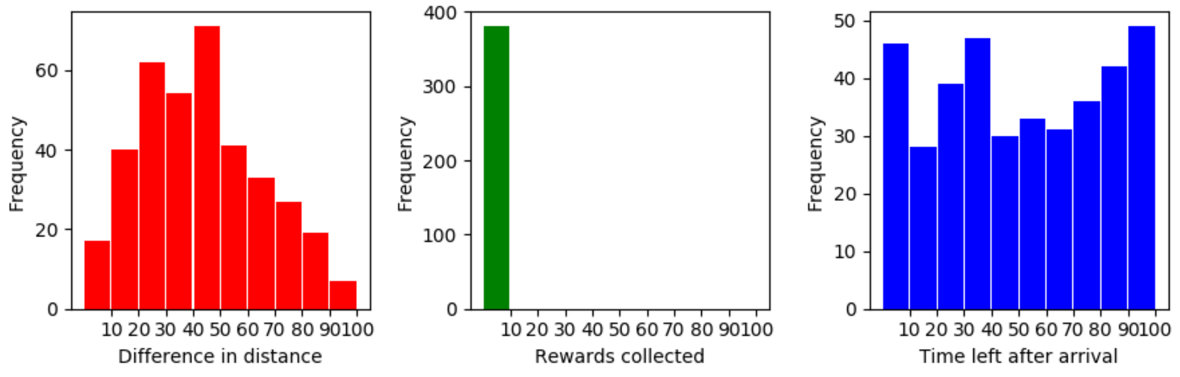
Additionally, the difference in distance data for TOPEC is shown in Figure 4.6. As the T_{max} increases in Figure 4.6(b), the robots get time to visit farther tasks as opposed to the situation in Figure 4.6(a). The same reasoning is applied to Figures 4.6(c) and 4.6(d).



(a) Experiment with unit rewards

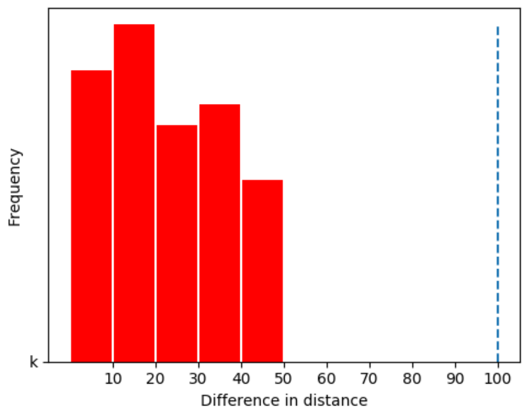


(b) Randomly assigned rewards to tasks

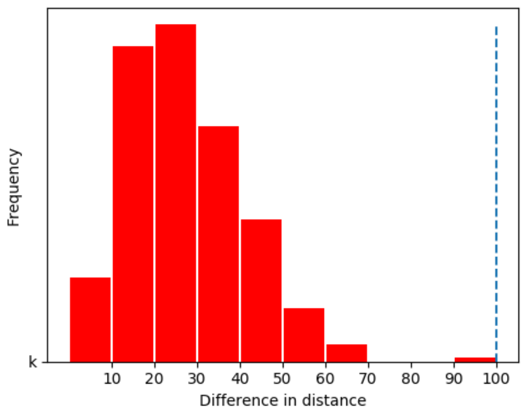


(c) Maximizing the time left after arrival as the objective function

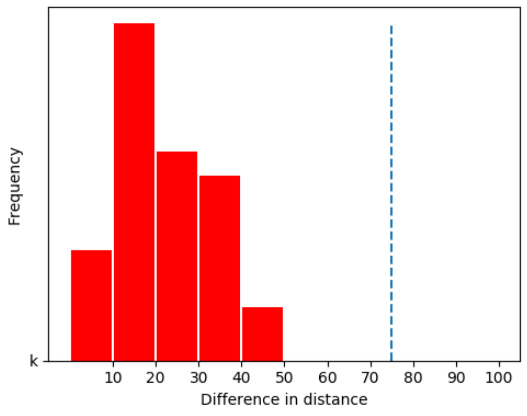
Figure 4.5: (TOPTAW) Trends in optimal solution for selecting tasks based of distance, rewards and deadline



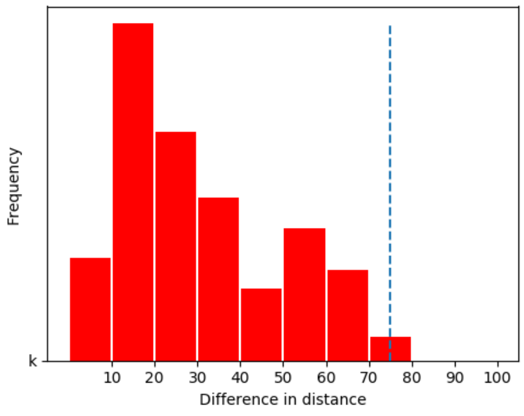
(a) Robots: 3, L: 100, Tmax: 100



(b) Robots: 3, L: 100, Tmax: 200



(c) Robots: 3, L: 75, Tmax: 100



(d) Robots: 3, L: 75, Tmax: 200

Figure 4.6: (TOPEC) Trends in optimal solution for selecting tasks based on distance and energy left

Chapter 5

Conclusions

This work focuses on the problem of simultaneous task allocation and scheduling for multi-robot systems. We formulate two variants of Team Orienteering Problem. The first, Team Orienteering Problem with Task Activation Windows (TOPTAW) consists of task activation windows, task duration and quota requirements. It is an approach to solve simultaneous task allocation and scheduling for cooperative tasks having temporal ordering constraints and team formation. The second, Team Orienteering Problem with Energy Constraints (TOPEC) involves recharging stations and energy limitations. This formulation integrates autonomous recharging of robots while allocating tasks and finding an optimal time schedule.

From the optimal solutions, we infer that computational runtime exponentially increases with the number of tasks. Optimal solutions for TOPTAW are generated as quick as 24 secs for 40 tasks and 8 robots. The optimal solutions tend to select task in which the robots perform the tasks at the very last moment. Optimal solutions for TOPEC, produced within 7429 secs for 7 tasks, 8 depots and 5 robots, are comparatively slower. In addition, the task selection heavily depends on the traveling distance when the energy is limited.

The next step is to combine simultaneous task allocation and scheduling with team formation and energy limitations in a single model. The temporal relationships among tasks in the combined is more complex due to the addition of energy limitations which maps to operational time limitations. The challenge is in finding valid inequalities to generate optimal solutions faster. Such formulation can model a larger domain of problems found in real-world applications like warehousing, delivery systems, search and rescue, and surveillance, than existing research. This work would be demonstrated in a multi-robot physics simulator ARGoS [44], along with an implementation on a physical multi-robot test-bed.

References

[1] "Morgridge Institute for Research and Wisconsin Institute for Discovery, NEOS Server,." `https://neos-guide.org/`. vii, 8

[2] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, Mar. 2013. 1

[3] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses," *AI Magazine*, vol. 29, p. 9, Mar. 2008. 1

[4] Z. Yan, N. Jouandeau, and A. A. Cherif, "A Survey and Analysis of Multi-Robot Coordination," *International Journal of Advanced Robotic Systems*, vol. 10, p. 399, Dec. 2013. 1

[5] M. Gini, "Multi-Robot Allocation of Tasks with Temporal and Ordering Constraints," in *Thirty-First AAAI Conference on Artificial Intelligence*, Feb. 2017. 2, 4, 12

[6] E. Tuci, M. H. M. Alkilabi, and O. Akanyeti, "Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art," *Frontiers in Robotics and AI*, vol. 5, 2018. 2

[7] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated

multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, pp. 376–386, June 2005. 2

[8] B. P. Gerkey and M. J. Matari, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, vol. 23, pp. 939–954, Sept. 2004. 3

[9] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, pp. 1495–1512, Oct. 2013. 4

[10] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011. 4, 11

[11] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, Apr. 2017. 4, 12

[12] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016. 7, 11

[13] M. Koes, I. Nourbakhsh, and K. Sycara, "Heterogeneous multirobot coordination with spatial and temporal constraints," in *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pp. 1292–1297, AAAI Press, June 2005. 7, 13

[14] M. Gombolay, R. Wilcox, and J. Shah, "Fast Scheduling of Multi-Robot Teams with Temporospatial Constraints," *MIT Web Domain*, June 2013. 7, 13

[15] M. Van der Merwe, J. Minas, M. Ozlen, and J. Hearne, "The cooperative orienteering problem with time windows." Optimization Online, 2014. 7, 11

[16] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1093–1099, May 2015. 7, 16, 26

[17] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot Rendezvous Planning for Recharging in Persistent Tasks," *IEEE Transactions on Robotics*, vol. 31, pp. 128–142, Feb. 2015. 7, 17

[18] N. Kamra and N. Ayanian, "A mixed integer programming model for timed deliveries in multirobot systems," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 612–617, Aug. 2015. 7, 17

[19] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications," *Annals of Operations Research*, vol. 139, no. 1, pp. 131–162, 2005. 9, 24

[20] IBM, "Cplex optimization studio." `https://www.ibm.com/analytics/cplex-optimizer`. 10

[21] L. Gurobi Optimization, "Gurobi optimizer reference manual." `http://www.gurobi.com`, 2018. 10, 30

[22] J. C. Smith and Z. C. Taskin, "A tutorial guide to mixed-integer programming models and solution techniques," *Optimization in Medicine and Biology*, pp. 521–548, 2008. 10, 11

[23] B. L. Golden, L. Levy, and R. Vohra, “The orienteering problem,” *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987. 11, 20

[24] J. Yu, M. Schwager, and D. Rus, “Correlated orienteering problem and its application to persistent monitoring tasks,” *IEEE Trans. Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016. 11

[25] KUKA, “Youbots.” `http://www.youbot-store.com/`. 13

[26] A. Prorok, M. A. Hsieh, and V. Kumar, “Fast Redistribution of a Swarm of Heterogeneous Robots,” *EAI Endorsed Transactions on Scalable Information Systems*, vol. ”3”, pp. 249–255, May 2016. 13

[27] B. P. Gerkey and M. J. Mataric, “Sold!: auction methods for multirobot coordination,” *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 758–768, Oct. 2002. 13

[28] E. Nunes and M. Gini, “Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints,” in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, Feb. 2015. 14

[29] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, “A Distributed Task Allocation Algorithm for a Multi-Robot System in Healthcare Facilities,” *Journal of Intelligent & Robotic Systems*, vol. 80, pp. 33–58, Oct. 2015. 14

[30] H. Hanna, “Decentralized approach for multi-robot task allocation problem with uncertain task execution,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 535–540, Aug. 2005. 14

[31] Y. Khaluf and F. Rammig, “Task Allocation Strategy for Time-Constrained Tasks in Robots Swarms,” *The 2018 Conference on Artificial Life: A Hybrid*

of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE), pp. 737–744, July 2013. 15

[32] B. Kannan, V. Marmol, J. Bourne, and M. B. Dias, “The Autonomous Recharging Problem: Formulation and a market-based solution,” in *2013 IEEE International Conference on Robotics and Automation*, pp. 3503–3510, May 2013. 15

[33] C. Cheng, Y. Adulyasak, and L.-M. Rousseau, *Formulations and Exact Algorithms for Drone Routing Problem.* July 2018. 16

[34] K. Sundar and S. Rathinam, “Route planning algorithms for unmanned aerial vehicles with refueling constraints,” in *2012 American Control Conference (ACC)*, pp. 3266–3271, IEEE, 2012. 16

[35] K. Sundar and S. Rathinam, “Algorithms for Routing an Unmanned Aerial Vehicle in the Presence of Refueling Depots,” *IEEE Transactions on Automation Science and Engineering*, vol. 11, pp. 287–294, Jan. 2014. 16

[36] K. S. David Levy and S. Rathinam, “Heuristics for routing heterogeneous unmanned vehicles with fuel constraints,” *Mathematical Problems in Engineering*, no. 131450, p. 12, 2014. 16

[37] H. Hamann, C. Markarian, F. M. a. d. Heide, and M. Wahby, “Pick, Pack, & Survive: Charging Robots in a Modern Warehouse based on Online Connected Dominating Sets,” in *9th International Conference on Fun with Algorithms (FUN 2018)* (H. Ito, S. Leonardi, L. Pagli, and G. Prencipe, eds.), vol. 100 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 22:1–22:13, Schloss DagstuhlLeibniz-Zentrum fuer Informatik, 2018. 16

[38] D. Mitchell, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot persistent coverage with stochastic task costs," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3401–3406, Sept 2015. 16

[39] N. Mathew, S. L. Smith, and S. L. Waslander, "A graph-based approach to multi-robot rendezvous for recharging in persistent tasks," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3497–3502, May 2013. 16

[40] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993. 17

[41] P. Maini and P. B. Sujit, "On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1370–1377, June 2015. 17

[42] N. Kamra, T. K. S. Kumar, and N. Ayanian, "Combinatorial Problems in Multirobot Battery Exchange Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 15, pp. 852–862, Apr. 2018. 17

[43] D. Dutia, "task planning, NESTLab." `https://github.com/NESTLab/task_planning`, 2018. 29, 30

[44] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012. 40