Coalition Formation and Scheduling for Heterogeneous Robot Swarms with Ant Colony Optimization

A Major Qualifying Project
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science in

Computer Science

By:

William Babincsak

Project Advisors:

Prof. Carlo Pinciroli

Ashay Aswale

Date: April 2024

Contents

6 Conclusions

Abstract

In this work, we tackle the problem of task scheduling in heterogeneous multi-robot systems. In our setting, the tasks require diverse skills to be fulfilled; however, the robots offer some, but not all, of the required skills. Thus, the robots must construct individual schedules that allow *coalitions*, i.e., dedicated teams, to be formed and disbanded dynamically. This results in cross-schedule dependencies that make generating high-quality solutions difficult, especially as the number of robots, skills, and tasks grows. First, we explore the multi-robot scheduling problem without coalition constraints. This is equivalent to a multiple traveling salesman problem (MTSP). We present a novel ant colony optimization (ACO) solution to the MTSP we call *Territorial Ant Colony Optimization (TACO)* that outperforms other state-of-the-art metaheuristics on the MTSP. We then extend this method to the MTSP with coalition constraints, which we term the *Collab-MTSP*. We call the extended method *Deadlock Reversal TACO (DR-TACO)*. In addition to DR-TACO, we present an alternative ACO-based method for the Collab-MTSP we call *Swarm Ant System (SAS)*. We compare both DR-TACO and SAS to baseline methods: *(i)* an optimal, but not scalable, formulation based on mixed-integer linear programming, and *(ii)* a scalable, but suboptimal, greedy algorithm. Our experiments show that our algorithms can produce solutions with costs as low as 0.5x those of the greedy algorithm at scales that are intractable to solve with the MILP baseline. Each of these methods represents a step forward in quickly solving difficult, time-extend task allocation problems with cross-schedule dependencies.

Chapter 1

Introduction

Robotic systems are capable of solving real-world problems at a large scale. They have already proven themselves in structured environments such as in warehouses and product assembly; however, solving larger, dynamic problems, such as fighting forest fires, is currently out of the scope of our robotic systems. In these scenarios, using teams of robots is essential for handling the scale and complexity of tasks involved in a mission. In these missions, allocating tasks to the members of the team is difficult.

Consider a team of robots putting out a large forest fire. Many jobs need to be done to complete the mission, such as fire monitoring, digging fire lines, and fire extinguishing. Ideally, any of these tasks could be completed by any robot in the team. However, having one robot design that has all the skills required to complete the mission is unrealistic; instead, the team would consist of heterogeneous robots, each with different skills, that collaborate to complete the mission. When robots meet to complete a task, this is called forming a *coalition*. A heterogeneous team of firefighting robots is depicted in Figure 1.1.

Creating time-efficient schedules for each robot while considering coalition formation is very difficult, especially as the number of robots, tasks, and skills increases. This is because the schedules of each robot must be such that, when robots need to collaborate, all collaborators are present at a task without being too late and stalling the mission. This is a combinatorial optimization problem where we want to minimize the overall cost of the mission. Exact solution methods, such as Mixed-Integer Linear Programming (MILP), to these types of problems exist and can generate high-quality solutions [1]. However, as the problem gets larger, these methods take a long time to generate the solution, which makes them impractical for real-world applications [1]. This makes approximate solution methods attractive. These methods do not guarantee an optimal solution, but can return good solutions quickly. One such approximation method is called *ant colony optimization* (ACO) [2]. ACO is a swarm-intelligence inspired approach that is based on the foraging behaviour of ants. ACO is state-of-the-art in solving other combinatorial optimization problems such as the classical travelling salesman problem (TSP) [2], and performs well on the related multiple travelling salesman problem (MTSP) [3], which is similar to our skilled task allocation problem. In the TSP, one salesman must visit n cities while minimizing the travel cost between cities. The MTSP is similar, except there are m salesmen that must collectively visit all cities. We formulate our combined coalition formation and scheduling problem as a more complex variant of the MTSP, which we call the *Collab-MTSP*. To our best knowldge, no ACO

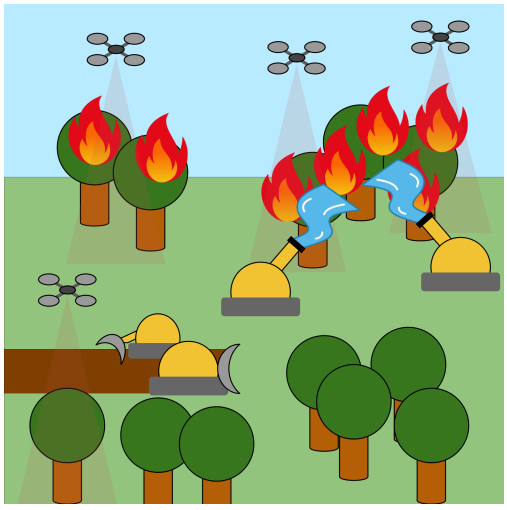

Figure 1.1: A swarm of firefighting robots forming coalitions.The diagram shows fire extinguishing (top right), fire line digging (middle), and fire monitoring (top left and bottom left). Fire extinguishing requires multiple tanker robots and fire monitoring drones to detect and transmit the fire state. Creating fire lines requires digging robots with different tools. Fire monitoring requires at least one monitoring drone.

based-methods to solve the Collab-MTSP currently exist in the
literature. As a result, this work focuses on developing and ACO-based approach to the Collab-MTSP.

We began this research by developing a new approach, which we call *Territorial Ant Colony Optimization* (TACO), for the MTSP. This method forms coalitions and creates schedules simultaneously. We show that this method outperforms other state-of-the-art metaheuristic approaches to the MTSP. We then extend this method to solve the Collab-MTSP and we call the new method *Deadlock-Reversal TACO* (DR-TACO). This algorithm is the first ACO-based method that can generate solutions to the Collab-MTSP and can generate near-optimal solutions to small-scale Collab-MTSP problems. This approach, however, struggles to effectively explore the solutions space and suffers from the possibility of deadlocks while constructing solutions. In an attempt to overcome these issues, we propose another ACO-based method that uses a novel ant formulation to overcome the deadlock issue, which we call *Swarm Ant System* (SAS). We show that this method is generally able to outperform DR-TACO in generating efficient schedules. These techniques both represent a step forward in quickly solving hard, cross-schedule planning problems using ant colony optimization.

Chapter 2

Background

In this section, we discuss the relevant background on heterogeneous multi-robot task allocation and ant colony optimization, in particular, the formulation using the multiple traveling salesman problem. We end the discussion with the works that explore MTSP variants, particularly those that are most similar to the Collab-MTSP.

2.1 Heterogeneous Multi-Robot Systems

Single-robot systems are not sufficient to approach many large-scale real-world problems like wildfire management and disaster recovery. These large-scale, spatially distributed problems need large teams of robots that can work together to complete the mission. In particular, different types of robots with different capabilities, or *skills*, need to efficiently collaborate to complete missions. Two of the core problems in coordinating multi-robot systems that need to be further investigated are coalition formation and task allocation [4]. These two problems are interdependent on each other and pose a difficult combinatorial problem when considered together.

2.1.1 Coalition Formation

Large-scale, distributed missions can often be split into smaller, actionable tasks (for example, digging fire lines or suppressing fire at a specific location) that can be completed by a small team of robots. (The process of segmenting the bigger mission into atomic tasks is called *task decomposition*, and the specifics of how this is done are not considered in this work.) For missions requiring heterogeneous swarms, these tasks can require robots with different skills to work together to complete the task. Coalition formation is the process of forming sub-teams (*coalitions*) from the entire swarm that will work together to adequately complete the given tasks.

2.1.2 Task Allocation

Task allocation is the process of assigning tasks to members of a robot swarm. In this work, we consider the case where tasks are all preassigned to the robots before the mission begins and these schedules do not change during the mission. In this case, task allocation involves generating a schedule for each robot that lays out when it should be present at each task. In our formulation, the time to travel between tasks is not trivial, thus the tasks can be thought of as points distributed in Cartesian spaces that form a fully-connected graph well call the *task space*.

2.1.3 Time-Extended Coalition Formation and Task Allocation

In this work, we consider the problem where robots in the swarm each will complete many tasks and participate in many coalitions across time. This is a single-task robot (ST), multi-robot task (MR), time-extended assignment (TA) problem as classificed based on the influential taxonomy by Gerkey and Matarič

[5]. Some previous work has studied similar problems, though none touch on it exactly. Oh *et al.* approached a similar ST-MR-TA problem for UAV scheduling; however, they do not consider differing skills across robots in the swarm [6]. Mourdian *et al.* consider varying capabilities, but they only consider a ST-MR-IA (instantaneous assignment) problem, which means that robots do not participate in many coalitions across time [7]. Arif *et al.* is one of the few works that considers a ST-MR-TA problem with skills, however, they assume that every robot has every skill but can only contribute one at a time to a task [8]. We assume that robots have a subset of the available skills but can contribute all of their skills towards a task.

2.2 Ant Colony Optimization

Ant colony optimization (ACO) was first proposed by Dorigo *et.al.* in 1994 [9]. The algorithm was inspired by the real-life behavior of ants. As ants search for food, they leave a chemical called a pheromone along their path. They follow the pheromone trails left by other ants probabilistically, meaning they usually follow paths with stronger pheromones but sometimes explore other paths. The algorithm uses simulated 'ants' that build candidate solutions based on a pheromone matrix that encodes estimated solution quality.

Ant colony algorithms generally have three main phases: initialization, solution construction, and pheromone updating. The initialization phase runs once at the beginning of the algorithm and the solution construction and pheromone update steps are looped until a termination condition is met. This is usually a specified number of iterations.

In the initialization phase essential data structures are initialized with sensible values. In the solution construction phase, simulated ants incrementally construct many candidate solutions. They do this using a lookup table that contains information about the quality of potential solutions called a pheromone matrix. In the pheromone update phase, the values in the pheromone matrix are updated based on the quality of the most recent candidate solutions.

2.2.1 The Base ACO Algorithm: Ant System

Ant system (AS) was the first ACO algorithm proposed to solve the traveling salesman problem (TSP) [9]. In the TSP, a salesman needs to visit n cities, visiting each city once and starting and returning to the same city. This algorithm provides the base from which most other ACO algorithms can be understood. The high-level structure of the algorithm follows Algorithm 1. The initialization, solution construction, and pheromone updating steps will now be described in detail.

Algorithm 1: High Level Structure of the Ant System

```
Initialization();
iter = 0;
while iter < threshold do
    Solution_construction();
    Pheromone_updating();
    iter = iter + 1;
end
```

Initialization

In AS, each edge in the graph of all cities is associated with a numerical pheromone value. These values are stored in a pheromone matrix. To initialize all of the edge pheromones to a sensible value t_0, a sample tour is constructed using a reasonable method (for example, a greedy approach), and the length of the tour is calculated as C^{greedy}. If we use m ants in the solution construction phase, the $t_0 = m/C^{\text{greedy}}$. Initializing pheromones is the only action taken during initialization.

Solution Construction

In the solution construction phase, m ants (where m is a parameter of the algorithm) each independently build tours guided by the pheromones. Each of the m ants starts at a random city and chooses the next city to visit probabilistically based on the following probability:

$$P(\text{Ant } k \text{ visits city } j \text{ from city } i) = p_{ij}^k = \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{l \in N_i^k} \tau_{il}^\alpha \eta_{il}^\beta} \text{ where } j \in N_i^k$$

N_i^k is the set of tasks that still need to be visited by ant k while it is at task i, τ_{ij} is the pheromone on the edge connecting task i to task j, η_{ij} is the pheromone on the edge connecting task i to task j (in AS, the heuristic is simply the reciprocal of the distance between the tasks, $\eta_{ij} = 1/d_{ij}$), and α and β are parameters of AS to weight the relative importance of heuristic and pheromone information.

Pheromone Updating

Once all ants construct their solutions, the pheromone matrix is updated based on the qualities of the generated solutions. First, all pheromones are evaporated, meaning they are all multiplied by some number less than but close to 1, like 0.95. This allows old information to decay away as the algorithm progresses. Then each ant k updates the pheromones for the edges it travels along using the following update equation:

$$\tau_{ij} = \tau_{ij} + \frac{Q}{L_k} \text{ where } i, j \text{ in ant } k\text{'s tour}$$

The value Q is a constant and L_k is the length of ant k's tour. Applying this for each ant, the final value of τ_{ij} is

$$\tau_{ij} = \tau_{ij} + \sum_{a \in A_{ij}} \frac{Q}{L_a}$$

where A is the set of ants that travel on edge i, j.

Improvements to Ant System

Later, more ACO algorithms were proposed such as MIN-MAX Ant System, Ant Colony System, and Rank-Based Ant System [10]. These algorithms implement changes to improve solution quality and convergence properties of Ant System. For example, MIN-MAX Ant System sets a minimum and maximum value for the pheromone values to maintain exploration of potential solutions and only updates the pheromones based on the best-performing ants, rather than all of them as in AS. Ant Colony System makes an interesting change where as soon as an ant moves along an edge i, j the pheromone on that edge is decreased. This means that pheromone values are modified during solution construction and the paths of the different ants are no longer independent. Later as ACO solutions to the MTSP are explored, this concept of making ants dependent on each other becomes more relevant.

2.3 The Multiple Travelling Salesman Problem (MTSP)

The MTSP is a generalized version of the TSP. The optimization goal of the TSP is clear: minimize the path cost for traveling between the cities. The MTSP is generalized to include m salesmen who collaborate to visit all the cities. In introducing multiple salesmen, there ceases to be one clear optimization goal. One can minimize the summed path cost of all agents or the maximum cost of a single agent. To account for this, modern solutions to the MTSP treat it as a Multi-Objective Optimization Problem (MOOP). This means that algorithms simultaneously optimize for more than one objective and return a Pareto set of all non-dominated solutions instead of a single solution. Many variants of the MTSP also exist that modify the problem formulation but keep the same core structure.

The general solution methods for the MTSP and its variants in the literature include constraint programming [11, 12, 13], market-based methods [14], and metaheuristic optimization [15, 16, 17]. This section

will briefly summarize the broad metaheuristic methods before deeply discussing the multi-objective ant colony optimization literature for the MTSP. It will conclude with a discussion of the current direction of the literature, identifying the gap that this work addresses.

Metaheuristic approaches to the MTSP are prominent in the literature [3]. Shuai *et al.* [18] proposed novel crossover and mutation operators for the NSGA-ii genetic algorithm that performs very well on the MTSP and is often used as a baseline for other metaheuristic approaches [19, 17]. Particle swarm optimization (PSO) is another swarm-intelligence-inspired optimization technique that has been used for the MTSP [17, 7, 6].

ACO has been widely used in solving the MTSP and its variants [3]. As a result, ACO approaches to the MTSP often use a multi-objective ant colony optimization (MOACO) algorithm. These MOACO algorithms often contain different multi-dimensional pheromone and heuristic matrices, solutions construction, and pheromone update methods to effectively explore Pareto optimal solutions, often using multiple ant colonies [20].

MOACO algorithms have been developed widely for many multi-objective optimization problems. Alaya *et al.* proposed a MOACO algorithm called m-ACO [21]. This algorithm uses multiple colonies, the exact number of which is an algorithm parameter. Ants in each colony develop solutions based on heuristics and pheromones unique to the colony. They propose four variants of m-ACO that slightly change the structure and demonstrated the performance of these general MOACO algorithms on the multiple knapsack problem.

Ke *et al.* added to the concept of using multiple colonies to explore different sub-problems. They define neighborhoods of colonies that are likely to produce similar solutions and update each colony's solutions based on information from its neighbors' solutions [22].

Deng *et al.* used different classes of ants within the same colony and a pheromone diffusion technique to improve convergence properties of ACO on scheduling problems like the TSP [23].

As mentioned, using local search to improve ant-generated solutions is often critical to the performance of MOACO algorithms. Chen *et al.* developed a MOACO algorithm specifically for the MTSP that uses a sequential variable neighborhood descent local search method to improve performance.

In most previous work with multiple colonies, the colonies either construct solutions separately or share information about solutions between the solution construction phase of the algorithm [20]. Wang *et al.* developed a novel solution construction technique where ants are divided into groups that co-construct a solution, and multiple ant groups construct solutions during one iteration of the algorithm [15]. They use one set of pheromones per objective in the MOOP. Bao *et al.* proposed a similar solution method around the same time, where teams of ants co-construct solutions and use a shortest distance biased dispatch scheme to resolve conflicts [24]. This technique of using teams of ants that co-construct solutions are newly emerging for ACO and deserve to be further explored.

2.3.1 Extending the MTSP

Work has been done on MRTA variants that significantly differ from the standard. The variants that are most similar to our problem, the Collab-MTSP, are discussed here. These variants include the Colored MTSP (CTSP), Vehicle Routing Problem (VRP) with time windows, tightly coupled human-swarm collaboration, and the law enforcement problem.

Colored MTSP

One relevant variant is the Colored MTSP proposed by Li *et al.* [25]. In this variant, each salesman j has a single color c_j, and each city i is associated with a set of colors C_i, which contains either one color or every color. A salesman can only visit city i if the salesman's color $c_j \in C_i$. This is one of the first variants of the MTSP that explicitly considers that some cities need to be visited by multiple salesmen [3]. The original authors proposed a genetic algorithm enhanced with simulated annealing to generate solutions to the CTSP. Han *et al.* then presented an improved approach to the CTSP that combines ant colony optimization with ITÒ processes to improve convergence properties and solution quality [26].

While this variant is closer to the Collab-MTSP than the standard MTSP is, it still does not consider simultaneous collaboration on tasks that can require arbitrary sets of skills. Rather, tasks in the CTSP can only have a single color or all colors and salesman do not have to meet.

Vehicle Routing Problem with Time Windows

The Vehicle Routing Problem (VRP) is a problem prominent in logistics where a fleet of vehicles needs to visit different locations for pickup and delivery of goods [27]. The connections to the MTSP are obvious, and the VRP is often modeled as an MTSP much like MRTA. A common constraint for package delivery is time windows, where vehicles need to arrive at a task within a given period of time. Since this is an important constraint for real-life customer satisfaction, variants of the VRP with time windows have been widely studied. While the Collab-MTSP does not have time windows as usually defined in the literature, it does exhibit a type of coordinated time window in its simultaneity constraint. The time that a robot must arrive at the task should coincide as closely as possible with its collaborators on the task, so the robots need to define their own coordinated arrival time, which is similar enough to time window constraints to consider the literature surrounding them.

Wang *el at.* used an ACO approach to solve a variant of the VRP with time windows, among other constraints. Zhang *et al.* [27] developed a MOACO approach to the VRP with flexible time windows. Flexible time windows allow agents to violate the time windows constraints with a penalty. This type of constraint is more similar to our coordinated arrival times than a hard time window constraint is. The ACO approaches used to solve the time windows problems usually do not significantly differ in structure from ACO approaches to the standard MTSP. An exception to this is the time-based pheromone used by Yildrim *et al.* [28]. They proposed a MOACO solution that divides the pheromone matrix into Z time windows, where Z is an algorithm parameter. This embeds knowledge of how good it is to arrive at particular tasks at certain times into solution construction, improving algorithm results. Works that significantly extend the structure of the MOACO metaheuristic to take advantage of problem-specific insights like this are uncommon in our literature review.

Human-Swarm Collaboration

Much of the literature that considers both skill and simultaneity constraints is in the context of Human-Swarm Interaction (HSI), where humans and robots must work together to complete complex tasks. Shannon *et al.* proposed a polynomial-time planning algorithm to quickly allocate tightly coupled tasks to a team of humans and robots [29]. This is akin to a multi-agent system with different agents having different skills (humans and robots). This algorithm is fast enough to adapt to system changes in real time, although as a result, it produces sub-optimal solutions. Emam *et al.* [12] extend this work to handle cases where the efficacy of robots at completing tasks is unknown or changes with time.

Constraint programming solutions have also been considered for this HSI problem. Lippi and Marino proposed a mixed integer linear programming (MILP) formulation of the collaboration problem with both skills (humans and different types of robots) and simultaneity (tightly coupled tasks where humans and robots must work together). They proposed both a MILP formulation to solve the task allocation problem offline as well as an adaptive system to monitor and alter the allocations in real-time as needed.

This HSI collaboration is the closest we have found to the Collab-MTSP in the literature. Is it, however, not formulated as an MTSP, and therefore approaches common to MTSPs, like metaheuristics, have not been rigorously explored on this variant to our best knowledge.

2.4 Our Work

To further explore ACO techniques for the MRTA problem, we adopt a multi-dimensional pheromone matrix that gives a unique pheromone matrix to each ant in the construction group. We then adopt a novel solution-construction technique that uses the information about the distribution of pheromones throughout the matrix to guide the solution-construction process. Accordingly, we call this new algorithm Territorial Ant Colony Optimization (TACO). Since each robot is represented by a single ant in TACO, we call this an ant-as-robot approach. We extend this method to the Collab-MTSP with a method we call Deadlock-Reversal TACO (DR-TACO). We also present another solution method for the Collab-MTSP called Swarm Ant System (SAS) that uses a swarm-as-ant approach, meaning that a single ant represents the entire swarm, rather than an individual robot. We show that this formulation has better scaling properties and performance than the ant-as-robot technique.

Chapter 3

Territorial Ant Colony Optimization (TACO) for the MTSP

Territorial Ant Colony Optimization (TACO) expands on the ACO metaheuristic to address the MTSP by adding *territorial pheromones* and a *willingness* function. This section describes the initialization, solution construction, and pheromone update steps of the TACO algorithm in detail. For the purposes of the MTSP, we consider the salesmen as robots and the cities as tasks.

3.1 MTSP Problem Formulation

We represent the set of all tasks as a complete graph $G(V, E)$ and will refer to this as a *task space*. Each node represents a task. We define $N = |V|$ as the total number of tasks. For simplicity, a task is simply referred to by its number i, where $0 \leq i < |V|$. An edge between nodes r and s represents the transition between going from task r to task s. A weight c_{rs} is associated with each edge $e_{rs} \in E$ that represents the cost to transition from task r to task s. We assume symmetry, so $c_{rs} = c_{sr}$. In all of our experiments, it is assumed that the cost to complete a task is negligible compared to the cost associated with transitioning between tasks. Accordingly, the cost associated with completing a task r is 0. Task 0 is designated as a common start and end point for all robots. In applications, consider this the docking point that all robots are deployed from and must return to. This makes our problem a single-depot MTSP because all salesmen start and end at the same task. Each of the remaining tasks must be visited exactly once by exactly one salesman. The number of robots in the swarm is designated by k.

A pair containing a task space and a swarm size, $\langle G, k \rangle$, denotes a single problem. A solution P for a problem is a set of k paths. The path of a robot i is denoted by P_i where $0 \leq i < k$. The cost of a single path P_i is the sum of all edge costs for each edge contained in the path.

$$\text{Cost}(P_i) = \sum_{e_{rs} \in E} \mathbf{I}[e_{rs} \in P_i] c_{rs} \tag{3.1}$$

In this case, $\mathbf{I}[x]$ is the indicator function that returns 1 when x is true and 0 otherwise.

We take a multi-objective optimization approach to this problem considering the objective

$$\text{minimize } \mathbf{F} = \langle f_1(P), f_2(P) \rangle \tag{3.2}$$

$$f_1(P) = \sum_{i=0}^{k-1} \text{Cost}(P_i) \tag{3.3}$$

$$f_2(P) = \max_{0 \leq i < k}(\text{Cost}(P_i)) \tag{3.4}$$

subject to the constraints

Algorithm 2: Initialization

Result: An initial Pareto set **P**
Result: An initial pheromone matrix τ
$current\_ant \leftarrow \text{rand}[0, k)$;
$C \leftarrow \{\}$;
$P_i \leftarrow [0], \forall\ i \in [0, k)$;
while $|C| \leq N$ **do**
  $r \leftarrow$ current location of $current\_ant$;
  $closest \leftarrow \underset{s}{\text{argmin}}(c_{rs})$;
  Append $closest$ to $P_{current\_ant}$;
  $C \leftarrow C + \{closest\}$;
  $current\_ant \leftarrow \underset{i}{\text{argmin}}(\text{length}(P_i))$;
end
Append 0 to the end of all paths P_i;
Calculate τ_0 using equation (3.8);
$\tau_{rs}^{(i)} \leftarrow \tau_0\ \forall$ edges ;
$\mathbf{P} = \{P\}$;

$$\bigcup_{i=0}^{k-1} P_i = V \tag{3.5}$$

$$P_i \cap P_j = \{0\} \ {}^{\forall[i,k)}_{\forall[j,k)} \tag{3.6}$$

$$P_i[0] = P_i[\text{length}(P_i) - 1] = 0\ \forall[i, k) \tag{3.7}$$

The objective is to simultaneously minimize the total cost of all robots in the mission and the maximum cost of any individual robot. These are shown in equations 3.3 and 3.4, respectively.

Constraint 3.5 enforces that all tasks are visited.Constraint 3.6 enforces that all paths are disjoint except for the depot. Constraint 3.7 enforces that all agents start and end at the depot.

3.2 The TACO Algorithm

This section breaks down the three main phases of the TACO algorithm: initialization, solution construction, and pheromone updating.

3.2.1 Initialization

Pseudo-code of the initialization phase is in Algorithm 2. A single initial solution is generated using a greedy construction strategy. A team of ants with one ant per robot all start at the starting task. On ant i's turn to choose a task, it acts greedily and chooses the task closest to its current location as its next task. The next ant to select a node is the ant with the shortest path length so far. Ants select tasks until there are none left to be claimed. All ants then return to task 0 to complete all of their paths. From this initial solution P, the initial pheromone value τ_0 is calculated as

$$\tau_0 = 1/(f_1(P) + kf_2(P)) \tag{3.8}$$

Where, recall, $f_1(P)$ is the total cost of the mission and $f_2(P)$ is the maximum single robot path cost. The Pareto set is initialized with P as the only element.

Algorithm 3: Solution Construction

Result: N_g solutions to the MTSP
$S \leftarrow empty$;
for $i = 1$ *to* N_g **do**
  $current\_ant \leftarrow \text{rand}[0, k)$;
  $C \leftarrow \{\}$;
  $P_i \leftarrow [0], \forall\ i \in \mathbf{N} \mid i < k$;
  $W \leftarrow \{\}$;
  while $|C| < N$ **do**
    Choose task *observed* according to (3.11);
    Calculate willingness w according to (3.13);
    if $random(0, 1) < w$ **then**
      Append *observed* to $P_{current\_ant}$;
      $C \leftarrow C + \{observed\}$;
      $W \leftarrow \{\}$
    else
      $W \leftarrow W + current\_ant$;
      if $|W| = k$ **then**
        $current\_ant \leftarrow \underset{i}{\text{argmax}}(\text{willingness}(i,\ observed))$;
        Append *observed* to $P_{current\_ant}$;
        $C \leftarrow C + \{observed\}$;
        $W \leftarrow \{\}$
      end
    end
    $current\_ant \leftarrow \underset{i \in [0,\ldots,k-1]-W}{\text{argmin}}(\text{Cost}(P_i))$;
  end
  Append 0 to the end of all paths P_i
end
return S

3.2.2 Solution construction

Pseudo-code of the solution construction phase is in Algorithm 3. During each solution construction phase, N_g solutions are generated independently by a group of ants that co-construct the solution. N_g is a parameter of the TACO algorithm.

In constructing a single solution, ants alternate in adding tasks to their path similar to the initialization phase. On ant a's turn, it selects a task to observe. For ant a currently at task r, a weight for an unassigned task s is calculated using the equation

$$w_{rs}^{(a)} = {\tau_{rs}^{(a)}}^{\alpha} \eta_{rs}^{\beta} \tag{3.9}$$

$\tau_{rs}^{(a)}$ is the pheromone information on the edge from r to s for ant a and η_{rs} is heuristic information for the edge from r to s. The heuristic for an edge is the reciprocal of the cost between r and s ($\eta_{rs} = 1/c_{rs}$) so that higher heuristics are associated with shorter paths. α and β are parameters of the algorithm that weigh the influence of the pheromone and heuristic on the weight, respectively. With probability p_0, the task with the largest weight is chosen next. With probability $1 - p_0$, the next task is chosen with a probability proportional to its weight. So, the probability of ant a selecting task s from task r is

$$q = \begin{cases} \frac{w_{rs}^{(a)}}{\sum_{r \in R} w_{r\hat{r}}^{(a)}} & \hat{r} \in R \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where R is the set of all remaining tasks. In all, the task transition rule is as follows:

$$observed = \begin{cases} \underset{\hat{r} \in R}{\text{argmax}}(w_{r\hat{r}}^{(a)}) & p < p_0 \\ X & \text{otherwise} \end{cases} \tag{3.11}$$

X is a random variable that returns elements from R probabilistically according to probabilities from (3.10). Once an ant has selected a task v to observe, it will claim the task and add it to its path if it is willing. Intuitively, ants should be less willing to take a task that is incredibly out of its way or is better taken by another ant instead. To

The willingness of an ant a to select a task s while currently at task r is a function that returns $x \in \mathbf{R}_{(0,1)}$. Two quantities essential to calculating willingness are

$$T_r^a = \max_{0 \leq s < |V|}(\tau_{rs})$$

and

$$\hat{T}_r^a = \max_{n \in [0 \ldots k-1] - \{a\}}(T_r^n).$$

T_r^a is the maximum pheromone value for ant a that goes into task r and $\hat{T}_r^a$ is the maximum pheromone value for any ant other than a that goes into task i. Using these, we define an intermediate function $s(a, r)$ as:

$$s(a, r) = \frac{\hat{T}_r^a}{T_i^a} \frac{c_{cv} + c_{v0} - c_{c0}}{Cost(P_a)} \frac{|V|}{|R|} \tag{3.12}$$

There are three main components to $s(a, r)$: $\frac{\hat{T}_r^a}{T_i^a}$, $\frac{c_{cv}+c_{v0}-c_{c0}}{Cost(P_a)}$, and $\frac{|V|}{|R|}$.

$\frac{\hat{T}_r^a}{T_r^a}$ represents the ratio of other ants' pheromone to ant a's pheromone. Ant a should be less willing to claim task v if it is also a good choice for other ants, indicted by high levels of pheromone for other ants and by a higher value of $\frac{\hat{T}_r^a}{T_r^a}$.

$\frac{c_{cv}+c_{v0}-c_{c0}}{Cost(P_a)}$ represents the extra cost that traveling to task v would have over just returning to the starting depot and ending the tour. $c_{cv} + c_{v0} - c_{c0}$ is the raw extra cost of travelling to task v and it is normalized by $Cost(P_a)$. Ants should be less willing to take a task that takes them far away from where they will need to end, indicated by a higher value of $\frac{c_{cv}+c_{v0}-c_{c0}}{Cost(P_a)}$.

$\frac{|V|}{|R|}$ represents how well the task space has been covered by all of the ants. Ants should act pickier when more of the task space has been covered to decrease exploration as the algorithm progresses and to allow the chance for other ants that are otherwise more willing to claim node v. Ants should be less likely to pick task v when $\frac{|V|}{|R|}$ is higher.

By multiplying all of these together we get that ants should be less willing to claim v as $s(a, v)$ increases. To map this to a value bounded between 0 and 1, we compute

$$\text{willingness}(a, r) = e^{-(\gamma s(a,r)^2)} \tag{3.13}$$

Higher values of willingness(a, v) correspond to ant a being more willing to claim node v and γ is a scalar parameter of the TACO algorithm. When ant a observes v, it claims v with probability willingness(a, v).

If an ant is not willing to claim a task, it loses the ability to observe a task until another ant claims one. If all ants are unwilling to claim a task, then the most recently observed task will be given to the most willing ant.

3.2.3 Pheromone update

Pseudo-code of the solution construction phase is in Algorithm 4. First, all pheromones are uniformly evaporated by a scalar factor of $1 - \rho$, where $0 \leq \rho \leq 1$. Then, the Pareto set $\mathbf{P}$ is updated to include the non-dominated solutions from the most recent set of candidate solutions S and newly dominated old solutions are removed from $\mathbf{P}$. For each solution $P^{(x)}$ in $\mathbf{P}$ where $0 \leq x < |\mathbf{P}|$, pheromone is updated via the following equation:

Algorithm 4: Pheromone Update

Data: $\tau_{ab}^{(i)}$ Pheromone matrices for ant i
Data: S solutions generated by previous solution construction phase
Data: $\mathbf{P}$ the Pareto set of non-dominated solutions
Evaporate all edges $\tau_{ab}^{(i)} \leftarrow \tau_{ab}^{(i)} * (1 - \rho)$;
Updated $\mathbf{P}$ with solutions from S;
for $j = 1$ *to* $|\mathbf{P}|$ **do**
  Update all edges from solution $\mathbf{P}_j$ according to Equation (3.14);
end
return S

$$\tau_{ij}^{(a)} = \begin{cases} \frac{1}{2}\tau_{ij}^{(a)} + \frac{1}{2}\frac{1}{f_1(P_a^{(x)}) + k f_2(P_a^{(x)})} & e_{ij} \in P_a^{(x)} \\ \tau_{ij}^{(a)} & \text{otherwise} \end{cases} \tag{3.14}$$

Put simply, all of the pheromones for the edges along a path $P_a^{(x)}$ for ant a are updated by averaging the current value of $\tau_{ij}^{(a)}$ with $\frac{1}{f_1(P_a^{(x)}) + k f_2(P_a^{(x)})}$.

3.3 Evaluation

Evaluating the performance of a multi-objective optimization algorithm is non-trivial as there is no single best metric for solution analysis. This motivates looking at multiple metrics of algorithm quality. Two metrics prevalent in the MTSP and MOOP literature are the hypervolume indicator and inverted generational distance plus [15, 20, 17, 30]. We look at both of these in evaluating our algorithm.

TACO is compared against two state-of-the-art algorithms: MOACS [15] and NSGA-ii [18]. The parameter settings used for each algorithm can be seen in Tables 3.1, 3.2, and 3.3.

Benchmark task spaces from the TSPLIB dataset repository are used for evaluation [31]. Datasets kroA100, kroA150, kroA200, kroB100, kroB150, and kroB200 are studied. Swarm sizes ranging from 3 to 8 are tested for all datasets. Since there are 6 task spaces with 6 swarm sizes considered for each, a total of 36 problems were studied in our experiments.

Because all of these algorithms are non-deterministic, each of them was run 30 times per benchmark. The average value and standard deviation of the IGD+ for each problem are shown in Table 3.5. The average value and standard deviation of the hypervolume indicator for each problem are shown in Table 3.4. For IGD+ smaller values are better. Larger values are better for the hypervolume. A visualization of the hypervolume results for the kroA200 dataset is shown in figure 3.1. The best average values for each problem are denoted in bold in the tables. To test for statistical significance, we use the Wilcoxon Rank Sign test [30, 27]. This is a non-parametric, paired test for significance. For each problem, the Wilcoxon test was used to compare the best-performing algorithm and the next best-performing algorithm. Of the bolded best performers, those that are significantly better than the next best are denoted with a (*). An α of 0.05 was used for determining significance. We can clearly see that TACO significantly outperforms the next-best algorithm for most problems for both IGD+ and hypervolume.

3.4 Conclusions

We have presented a multi-objective ant colony optimization-based algorithm to solve the single-depot multiple traveling salesmen problem called TACO. Our algorithm contains a novel territorial pheromone matrix and willingness-based solution construction technique. Our experimental results show that our algorithm is able to outperform other state-of-the-art algorithms from the literature in generating Pareto sets of non-dominated solutions. Since this method shows promise, we next present a method of extending it to the full Collab-MTSP with both skill and simultaneous collaboration constraints.

Table 3.1: Experiment parameters for TACO.

Symbol	Description	Value
N_i	Number of iterations	1000
N_g	Number of ant groups per iteration	100
γ	Willingness function parameter	1/30
p_0	Probability to select minimum cost task	0.9
α	Pheromone exponent	1
β	Heuristic exponent	2

Table 3.2: Experiment parameters for MOACS. For more detail on each parameter, refer to the original paper [15].

Symbol	Description	Value
N_i	Number of iterations	1000
N_g	Number of ant groups per iteration	100
q_0	Probability of selection minimum partial-cost-ant	0.90
q_1	Probability of selection minimum partial-cost-ant	0.05
α_1	Pheromone 1 exponent	1
α_2	Pheromone 2 exponent	1
β	Heuristic exponent	2
ρ	Relative importance of the newly added and historic pheromone information	0.5

Table 3.3: Experiment parameters for Hybrid NSGA-ii. For more detail on each parameter, refer to the original paper [18]

Symbol	Description	Value
N_i	Number of iterations	2000
N_p	Population size	100
Symbol	Mutation rate	0.05

Dataset	k	TACO		NSGA-II		MOACS	
		Mean	STD	Mean	STD	Mean	STD
kroA100	3	**0.81***	0.13	0.6	0.09	0.36	0.14
kroA150	3	**0.85***	0.09	0.5	0.09	0.4	0.09
kroA200	3	**0.8***	0.06	0.48	0.08	0.4	0.09
kroB100	3	**0.77***	0.08	0.53	0.07	0.43	0.12
kroB150	3	**0.89***	0.04	0.48	0.08	0.56	0.06
kroB200	3	**0.85***	0.07	0.51	0.08	0.48	0.11
kroA100	4	**0.83***	0.08	0.61	0.09	0.4	0.16
kroA150	4	**0.85***	0.07	0.56	0.06	0.51	0.1
kroA200	4	**0.87***	0.07	0.56	0.08	0.49	0.09
kroB100	4	**0.75***	0.07	0.62	0.06	0.42	0.1
kroB150	4	**0.89***	0.05	0.47	0.07	0.54	0.09
kroB200	4	**0.84***	0.08	0.6	0.05	0.46	0.09
kroA100	5	**0.81***	0.05	0.67	0.07	0.41	0.09
kroA150	5	**0.89***	0.07	0.63	0.06	0.55	0.11
kroA200	5	**0.81***	0.07	0.54	0.06	0.47	0.06
kroB100	5	**0.81***	0.08	0.68	0.05	0.38	0.12
kroB150	5	**0.92***	0.04	0.5	0.05	0.56	0.08
kroB200	5	**0.77***	0.08	0.62	0.05	0.44	0.11
kroA100	6	**0.79***	0.08	0.7	0.05	0.32	0.11
kroA150	6	**0.8***	0.08	0.6	0.06	0.39	0.11
kroA200	6	**0.92***	0.04	0.67	0.07	0.55	0.08
kroB100	6	**0.82***	0.07	0.75	0.04	0.36	0.08
kroB150	6	**0.91***	0.04	0.53	0.06	0.51	0.07
kroB200	6	**0.72**	0.1	0.69	0.06	0.36	0.09
kroA100	7	**0.78**	0.06	0.77	0.05	0.38	0.09
kroA150	7	**0.83***	0.05	0.68	0.05	0.47	0.12
kroA200	7	**0.85***	0.05	0.57	0.07	0.42	0.07
kroB100	7	0.74	0.07	**0.77***	0.04	0.28	0.09
kroB150	7	**0.92***	0.04	0.55	0.05	0.5	0.08
kroB200	7	0.71	0.11	**0.71**	0.05	0.27	0.11
kroA100	8	0.74	0.06	**0.77***	0.05	0.3	0.08
kroA150	8	**0.85***	0.08	0.69	0.06	0.38	0.09
kroA200	8	**0.86***	0.04	0.67	0.07	0.48	0.04
kroB100	8	0.73	0.07	**0.82***	0.03	0.27	0.08
kroB150	8	**0.95***	0.03	0.62	0.05	0.57	0.08
kroB200	8	0.68	0.1	**0.75***	0.05	0.27	0.09

Table 3.4: Experimental Results for Hypervolume Indicator

Dataset	k	TACO		NSGA-II		MOACS	
		Mean	STD	Mean	STD	Mean	STD
kroA100	3	**1126.67***	891.68	4308.67	765.02	4377.5	1123.81
kroA150	3	**1753.43***	1023.48	7920.97	1064.8	7355.96	1269.25
kroA200	3	**1966.0***	756.73	7754.61	1292.19	7563.76	1333.97
kroB100	3	**1752.5***	776.55	6153.99	523.46	5380.94	1331.05
kroB150	3	**1554.02***	544.3	8322.05	1534.38	5984.83	900.07
kroB200	3	**1959.5***	889.2	8360.35	1266.35	6608.42	1562.57
kroA100	4	**1724.67***	759.54	4277.9	777.31	6040.13	1668.84
kroA150	4	**1806.09***	1014.16	7723.52	1256.15	7298.38	1820.4
kroA200	4	**1844.42***	1047.25	9071.11	1368.63	7536.4	1527.24
kroB100	4	**2073.0***	720.11	4905.47	753.66	6562.68	1461.93
kroB150	4	**1260.76***	529.11	8967.66	1398.83	5956.16	1575.37
kroB200	4	**1900.87***	1085.54	7953.92	837.38	7611.38	1431.09
kroA100	5	**1017.14***	383.51	3862.86	710.54	6440.31	1473.4
kroA150	5	**1731.46***	1005.94	6913.93	1205.08	7044.76	1813.46
kroA200	5	**2944.8***	987.72	9361.96	1127.95	8364.92	1269.31
kroB100	5	**1997.8***	1005.71	4893.85	580.65	8616.12	2125.26
kroB150	5	**1031.42***	363.02	8862.26	1094.14	6106.06	1463.15
kroB200	5	**2617.17***	1153.94	7568.63	1238.94	8358.38	2137.72
kroA100	6	**1600.19***	939.12	3813.2	620.75	8242.32	1698.11
kroA150	6	**1215.37***	664.44	5769.93	835.64	7216.66	1860.37
kroA200	6	**1258.78***	671.09	6895.74	1435.31	7880.05	1638.07
kroB100	6	**1324.89***	769.7	3484.0	469.53	8516.65	1653.34
kroB150	6	**1117.27***	560.87	9083.52	1126.72	7997.36	1504.77
kroB200	6	**3614.87***	1838.19	5600.28	1132.15	10631.7	1973.29
kroA100	7	3340.68	1109.8	**3256.16**	699.4	11837.3	1983.21
kroA150	7	**1825.53***	829.91	5401.59	853.99	8365.25	2377.93
kroA200	7	**1708.59***	780.57	8595.85	1367.06	9969.16	1518.47
kroB100	7	**2048.5***	866.53	2924.93	517.38	10398.5	2144.04
kroB150	7	**685.95***	383.24	7978.75	1234.54	7488.69	1636.7
kroB200	7	**3609.24***	1753.88	4749.98	711.8	12365.5	2371.09
kroA100	8	**2511.1**	895.56	2747.08	532.95	11777.3	2090.56
kroA150	8	**1325.76***	742.32	5088.86	1003.32	8575.25	1728.25
kroA200	8	**1469.85***	555.68	6855.34	1317.97	9529.3	1109.36
kroB100	8	2529.03	947.25	**2512.89**	382.87	13232.2	2210.82
kroB150	8	**603.46***	376.19	8007.9	1120.88	8377.07	1984.92
kroB200	8	4791.21	2024.84	**4391.38**	815.86	14628.7	2506.32

Table 3.5: Experimental Results for Inverted Generational Distance Plus

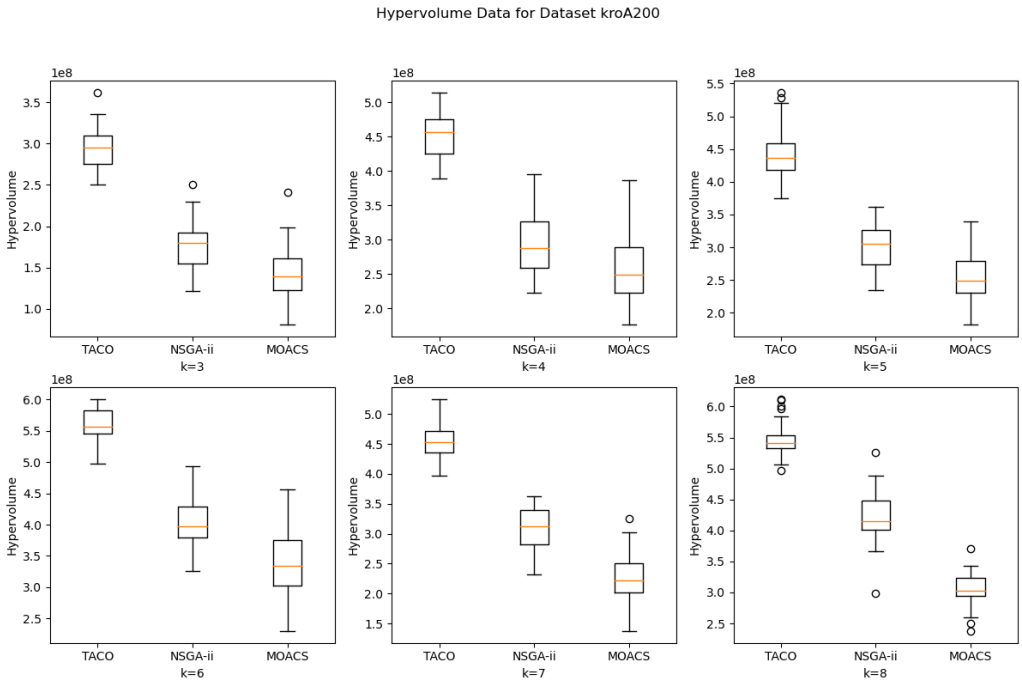

Figure 3.1: Hypervolume statistics for the kroA200 dataset. Each plot represents a different swarm size, and each box plot summarizes the 30 samples taken for a single algorithm on kroA200.

Chapter 4

Deadlock-Reversal TACO (DR-TACO) for the Collab-MTSP

Given TACO's state-of-the-art performance on the MTSP, it is natural to try and directly extend it to solve the Collab-MTSP. Handling skills and time-extended coalition formation present problems to the original TACO, since *deadlocks* can occur that TACO cannot handle. At the core of this solution is a deadlock reversal step to prevent circular dependencies in schedules, so the resulting algorithm is called DR-TACO. The specifics of the DR-TACO algorithm and experimental performance are presented in this section.

4.1 Collab-MTSP Problem Formulation

The Collab-MTSP is formulated similarly to the MTSP with a complete graph $G(V, E)$ as the task space with each node representing a task. The edge between nodes r and s represents the transition between going from task r to task s and weight c_{rs} represents the transition cost. Associated with each task n is also a completion cost b_n and a *required skillset*, denoted by RS_n. We assume that the total number of skills S for the problem is fixed. Using notation similar to Arif *el at.* [8], the skillset contains skills λ_x where $x \in [0, S)$. This denotes the skills that task n requires to be simultaneously present in order to be completed. For example, a task i that requires skills 0, 3, and 4 has the required skillset $RS_i = \{\lambda_0, \lambda_3, \lambda_4\}$. Under this formulation, a task cannot require more than one skill of the same type, so $\{\lambda_0, \lambda_0, \lambda_3, \lambda_4\}$ is not a valid skillset.

The *swarm* in a Collab-MTSP problem is a set of R robots where each robot R_i is fully defined by its skillset SS_i. Many robots can come together to form a *coalition* to complete a task. The skillset of a coalition is the union of the skillsets of all of the robots in the coalition.

Consider an example with a task s that has $SS_s = \{\lambda_0, \lambda_3, \lambda_4\}$ and two robots R_0 and R_1 with skillsets $\{\lambda_0, \lambda_3\}$ and $\{\lambda_3, \lambda_4\}$ respectively. Neither robot R_0 nor robot R_1 has the skills to complete task s independently, but they could form a coalition to complete the task together. To do this, they must both be present at the task at the same time and for the entire time, the task is being completed (cost b_s). Notice that since robots R_0 and R_1 will likely arrive at task s at different times, one of the robots will need to wait until the other arrives. The wait for robot i at task s is denoted by w_{is}. This wait time is added to the path cost for robot i. All robots start and return to the same location in the task space and each task must be completed by at least one robot.

A tuple containing a task space and a swarm, $\langle G, SW \rangle$, denotes a single problem. A solution P for a problem is a set of R paths. The path of a robot R_i is denoted by P_i where $0 \leq i < R$. The cost of a single path P_i is the sum of all edge costs, wait times, and completion costs for each edge contained in the path.

$$\text{Cost}(P_i) = \sum_{e_{rs} \in E} \mathbf{I}[e_{rs} \in P_i](c_{rs} + w_{is} + b_s) \tag{4.1}$$

In this case, $\mathbf{I}[x]$ is the indicator function that returns 1 when x is true and 0 otherwise.

Algorithm 5: General Structure for DR-TACO Solution Construction

```
C ← {};
P_i ← [0], ∀ i ∈ [0, R);
L_i ← {}, ∀ i ∈ [0, T);
Q = [0, R);
while |C| ≤ N do
    if Q is empty then
        Handle deadlock as in Algorithm 6
    end
    a ← select next ant from Q;
    next ← next task to be claimed
    if Ant a is willing to claim next then
        L_next = L_next + a;
        if skillset(next) ⊆ ⋃_{i∈L_next} skillset(R_i) then
            Append next to P_i ∀ i ∈ L_next;
            Q = Q + L_next;
            L_next = {};
            C ← C + {next};
        end
    end
end
```

We take a multi-objective optimization approach to this problem exactly the same as for the MTSP, by attempting to simultaneously minimize the total cost of all robots in the mission and the maximum cost of any individual robot.

4.2 Approach

The core structure of the TACO algorithm remains the same: a greedy solution method is used to seed the iterative process, ants representing robots alternate claiming tasks guided by pheromones in the solution construction phase, and pheromones are updated from solutions in the Pareto set between iterations. The additions to TACO are discussed below and are:

- A list of waiting ants for each node and a deadlock reversal step
- A new heuristic for guiding the search
- A pheromone approximation method to make pheromone a function of task arrival time

4.2.1 Handling Skills and Deadlocks

For both constructing the greedy solution and performing solution construction, the methods for handling skills and deadlocks are the same. A team of ants with one ant per robot all start at the starting task. On ant i's turn to choose a task, it chooses a task to claim and will claim it if it is willing. Ants only select from the subset of tasks that they have the skills to contribute to. If an ant is able to complete a task on its own, it will complete the task as in TACO. However, if the ant alone cannot complete the selected task, it will *wait* at the task for another ant with the appropriate skills to arrive and form a coalition. The set of ants waiting at task s is denoted by L_s. While waiting, ants cannot claim any other tasks.

If all ants are waiting on another ant to come and form a coalition, then a *deadlock* has occurred. This is the core difficulty with taking a one-ant-per-robot approach to this problem. To handle this, the DR-TACO algorithm performs a *deadlock reversal* step to free an ant to continue building a solution. This is a curative approach to handling deadlocks, where we allow them to occur but then fix them we they do.

Algorithm 6: Deadlock Reversal

Data: W_i is the set of all waiting ants with skill i
Data: L is the set of all waiting lists of ants
Data: C is the set of completed tasks
Data: Q is the set of ants not waiting at a task
Selected random node n from all nodes with ants;
$S \leftarrow \text{skillset}(n) - \bigcup_{i \in L_n} \text{skillset}(R_i)$;
while S != {} **do**
    $s \leftarrow$ random elements from S;
    $R_i \leftarrow$ random robot currently waiting with such that $s \in \text{skillset}(R_i)$;
    $p \leftarrow$ node that R_i is currently waiting at
    $L_p = L_p - R_i$;
    $L_n = L_n + R_i$;
    $SS \leftarrow \text{skillset}(R_i)$;
    $S \leftarrow S - SS$;
end
Append n to $P_i \; \forall \; i \in L_n$;
$Q = Q + L_n$;
$L_n = \{\}$;
$C \leftarrow C + \{n\}$;

To undo the deadlock, a node with waiting ants is picked randomly. For each missing skill, pick a random, waiting ant that has that skill and release it from waiting. This effectively undoes that ant's initial choice and instead assigns them to the new node. This process is repeated until the selected node can be completed and the deadlock is undone and is detailed in Algorithm 6.

4.2.2 A Heuristic Based on Waiting Ants

We present three potential heuristics for guiding the search for DR-TACO. The first is the same heuristic as used in TACO, which is the reciprocal of the cost between tasks r and s ($\eta_{rs}^{\text{distance}} = 1/c_{rs}$). Another is based on the number of required skills currently present at the task. We can express the required skills already at the task s as $\text{skillset}(n) \cap \bigcup_{i \in L_n} \text{skillset}(R_i)$. To encourage ants to go to tasks that already have ants waiting, another heuristic can be defined as

$$\eta_{rs}^{\text{completion}} = \frac{|\text{skillset}(n) \cap \bigcup_{i \in L_n} \text{skillset}(R_i)|}{|\text{skillset}(n)|} \tag{4.2}$$

Thirdly, we can multiply these two heuristics to create a new, combined heuristic as

$$\eta_{rs}^{\text{combined}} = \eta_{rs}^{\text{distance}} \eta_{rs}^{\text{completion}} \tag{4.3}$$

4.2.3 Pheromone as a Function of Arrival Time

As a method to improve the solution quality of the DR-TACO algorithm, a means of making the pheromone matrix a function of arrival time is introduced. We now define the edge pheromone as

$$\tau_{rs}^{(a)}(t) = \tau_{rs}^{(a)} \nu_s(t) \tag{4.4}$$

with $\tau_{rs}^{(a)}$ the same edge pheromone from DR-TACO and $\nu_s(t)$ a new pheromone function approximator for each *node*, rather than each *edge*. The intuition behind this is to try and coordinate the arrival times of each agent, regardless of what edge they arrive from. This is also why $\nu_s(t)$ is not a function of ant a either - arrival time should be coordinated among all of the ants.

To approximate $\nu_s(t)$, the average arrival times at node s from each group g in the previous round of solution construction is stored as $\bar{t}_g$ with the corresponding solution quality q_g. For the purposes of keeping

the approximation function bounded, the arrival time is normalized by the max path cost of the greedy solution used to seed DR-TACO. $\nu_s(t)$ is calculated as the following sum:

$$\nu_s(t) = 1 + \frac{\sum_{(q_g, \bar{t}_g)} q_g e^{-\frac{(t-\bar{t}_g)^2}{c}}}{\sum_{q_i} q_i} \tag{4.5}$$

$$c = 2(\frac{R}{N})^2 \tag{4.6}$$

4.3 Experimental Evaluation

We ran three experiments to assess the performance of the DR-TACO algorithm, these are:

1. Experiments on small problems (4 robots, 8 tasks, 2 skills) where we compared our results to the optimal solutions using a mixed-integer linear programming approach.
2. Experiments on a dataset of problems with increasing scales comparing the DR-TACO algorithm to a greedy approach
3. Experiments on the same dataset evaluating the impacts of the pheromone function approximator and new completion heuristics on DR-TACO

There is no equivalent to TSPLIB for the Collab-MTSP, which is why we made our own. Since the problem space of the Collab-MTSP is more complex than that of the MTSP, we also present a brief classification of core components of a Collab-MTSP problem and have created our benchmarks to span these classes. Our benchmark problems for experiments 2 and 3 were created to represent different classes for Collab-MTSP problems.

4.4 Classification of the Problem Space of the Collab-MTSP

We identified the following three axes of core importance to the structure of a Collab-MTSP problem.

1. The number of skills the robots have as a proportion of the total number of skills.
2. The number of skills the tasks requires as a proportion of the total number of skills.
3. How well covered the tasks are by the swarm.

Regarding the first point, we defined five classes of robot skill distribution: super-overpowered (SOPR), overpowered (OPR), mixed (MR), weak (WR), and super-weak (SWR). A problem has an overpowered swarm when every robot has all of the available skills. This removes the need for coalition formation, so it is not an interesting case. Overpowered swarms consist of robots that all have $> 50\%$ of the available skills. Weak swarms consist of robots that all have $< 50\%$ of the available skills, and super-weak swarms consist of robots that each only have one skill. Mixed swarms can have robots with any number of skills.

There is an analogous classification for how skills are distributed among tasks, and they are super-large (SLT), large (LT), mixed (MT), small (ST), and super-small tasks (SST).

For coverage, we define two classes - large and small coverage (LC and SC, respectively).In problems with large coverage, there are at least as many robots as there are tasks. Problems with small coverage have significantly fewer robots than tasks.

With these classifications types, we can categorize a problem as a tuple from $\{SOPR, OPR, MR, WR, SWR\} \times \{SLT, LT, MT, ST, SST\} \times \{LC, SC\}$ similar to the taxonomy from Korsah *et al.* [32]. For example, a problem with a weak swarm, large tasks, and small coverage is WS-LT-LC.

4.4.1 Explored Problem Space

Since the problem space is vast, we limited our studies to a subset of the possible classifications as we have defined them.There are 50 unique combinations of these classes; however, not all of them are interesting to investigate. Any example with SOPR is not interesting, since it removes all need for collaboration under our formulation. For our experiments, 10 classes of problems were studied at 3 different scales. These classes are

- SWR-LT-LC
- SWR-LT-SC
- SWR-ST-LC
- SWR-ST-SC
- MR-MT-LC
- MR-MT-SC
- WR-LT-LC
- WR-LT-SC
- WR-ST-LC
- WR-ST-SC

at scales $\sigma = \{16, 32, 64\}$. These include only skill distributions of MT and weaker because when robots are overpowered there is no need for collaboration, thus not capturing the essence of the Collab-MTSP. Super-Small tasks are left out via the same rationale, and Super-Large tasks are left out to maintain variety in the skills required by each task.

For the cases with large coverage, the number of robots (R), number of tasks (T), and number of skills (S) is set to

$$R = \sigma$$
$$T = \sigma$$
$$S = 4$$

And for cases with small coverage, we have

$$R = 8$$
$$T = \sigma$$
$$S = 4$$

4.5 Experiments Against an Optimal Baseline

We collected a small dataset of optimal solutions using a linear programming (MILP) technique. Due to the limits of the MILP technique, collecting many optimal solutions to problems larger than 4 robots, 8 tasks, and 2 skills was impractical due to runtime constraints. We ran the standard DR-TACO algorithm on these datasets to evaluate how close it can get to optimal baselines on easy (small) problems. All of these benchmarks are MR-MT-SC problems. The results from some of these trials are visualized in figure 4.1.

DR-TACO can consistently achieve within 10-20% of the optimal bounds across all of the benchmark problems. The promising results from these experiments motivated further testing of this algorithm on more problems at larger scales.

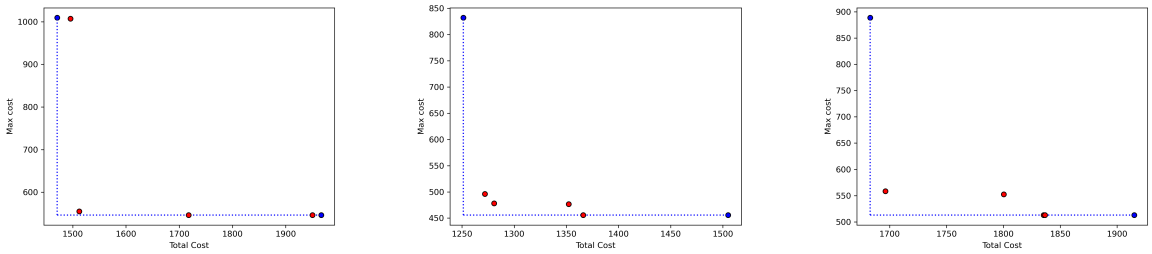

Figure 4.1: Charts showing the results of the solutions generated by DR-TACO (red points) compared to the optimal bounds produced by a MILP solver (blue lines). The blue points represent the solutions generated by the MILP solver. One solution optimizes for the total cost and the other for the max cost.

4.6 Experiments on Problems of Increasing Scale

Since the main goal of using an ACO approach is to improve scaling properties over an exact approach like MILP, we also evaluated our algorithm's performance on problems of increasing scale. For this, we generated 30 problem instances selected from the 10 problem classes highlighted previously at three scales $\sigma = \{16, 32, 64\}$ each. The results on all problems are in table 4.3. Boxplots visualizing some of the results from table 4.3 are in figure 4.2.

We can see from these results that the standard DR-TACO algorithm is able to effectively explore the solutions spaces and improve solution quality in some instances but not in others, as seen in table 4.3. In 25/30 cases, the DR-TACO algorithm achieved significantly lower IGD+ than the greedy baseline. In the other five cases, the algorithm did not improve on the greedy solution at all, so both got IGD+ values equal to zero. Of the five cases with no improvement, two are of scale $\sigma = 32$ and three are of scale $\sigma = 64$. All of the problems were large coverage problems, meaning that $R = T = \sigma$. This indicates that the standard DR-TACO algorithm struggles to explore the solution space when scaling up the number of tasks and robots simultaneously.

4.7 Evaluating Pheromone Approximation and New Heuristics

Three modifications of the base DR-TACO algorithm were experimented with to try and improve solution quality: using a pheromone approximator to make pheromone a function of robot arrival time, using the completion heuristic $\eta_{rs}^{\text{completion}}$, and using the combined heuristic $\eta_{rs}^{\text{combined}}$. Evaluated these modifications on the same benchmarks as we did for the experiments in the previous section. The results of these experiments are in figure 4.4. The algorithms using the completion heuristic and the pheromone approximator often perform worse than the standard DR-TACO algorithm with higher mean IGD+ values. The algorithm using the combined heuristic often achieves the lowest IGD+ score, but it is only significantly lower than that of DR-TACO on five of the problems. The modifications were also unable to achieve significantly better performance on the problems that DR-TACO had no improvement on, although the completion algorithm was able to generate a few better solutions to 2/5 difficult problems.

4.8 Discussion

In this section we discussed our extension of the TACO algorithm, TACO with Deadlock Reversal (DR-TACO), to the Collab-MTSP. This is, to our knowledge, the first ACO-based solution attempted to solve the heterogeneous multi-robot coalition formation and task allocation problem as we present it. DR-TACO can generate near-optimal multi-objective solutions to small-scale instances of this problem. In scaling up the problem size, DR-TACO struggles to scale and gets suck in local optima. We presented modifications to the algorithm to mitigate this effect, new heuristics and a novel method for approximating pheromone, to mixed results. One potential reason for the difficulty of solving the Collab-MTSP with this approach is that we

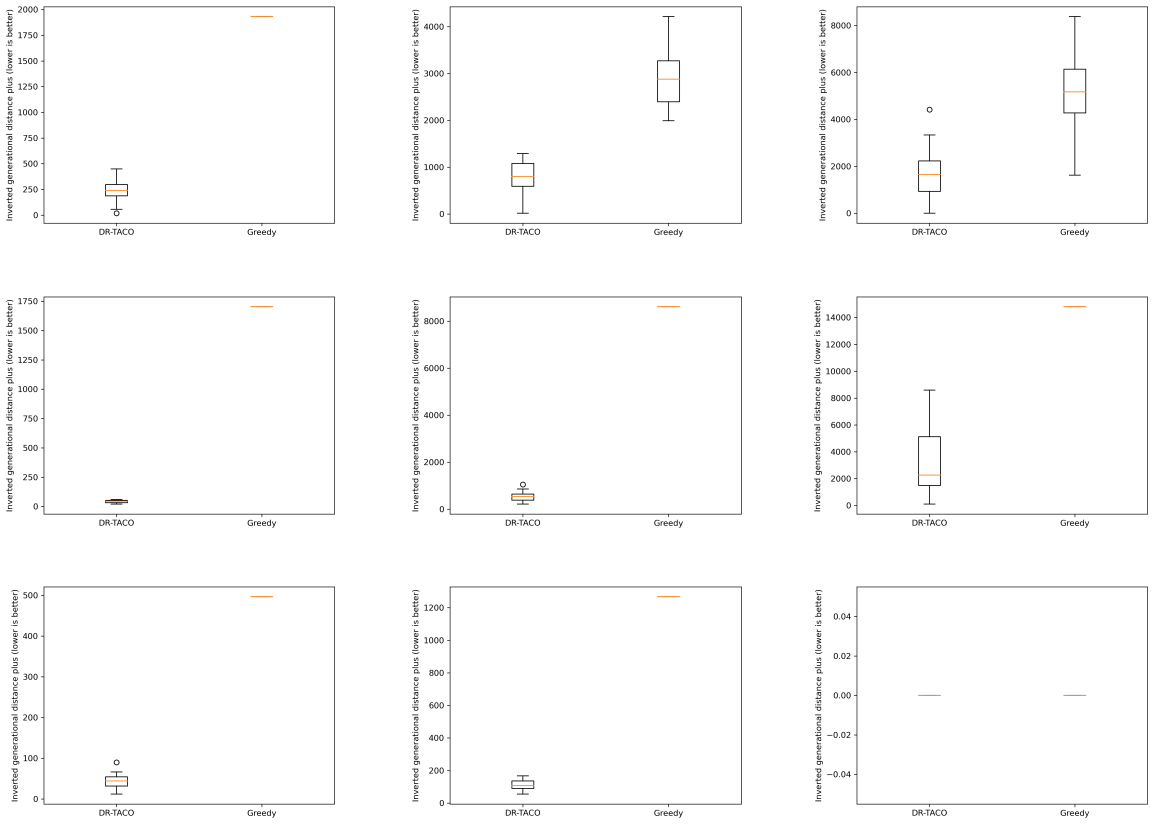

Figure 4.2: Example IGD+ boxplots for 3 problems at 3 scales. For IGD+, lower values are better. Both algorithms have a value of zero indicates that they both had the same output, *i.e.* DR-TACO was unable to improve on the initial seed solution from the greedy algorithm.

Problem	DR-TACO Mean	DR-TACO STD	Greedy Mean	Greedy STD
0_16_16_4_0_WR_LT_LC	**477.97**	246.63	960.14	0
1_32_32_4_0_WR_LT_LC	0.0	0	0	0
2_64_64_4_0_WR_LT_LC	0.0	0	0	0
4_8_16_4_0_WR_LT_SC	**710.71**	402.22	3008.31	169.31
5_8_32_4_0_WR_LT_SC	**780.16**	480.87	2366.74	0
6_8_64_4_0_WR_LT_SC	**1064.15**	888.54	3487.05	704.01
8_16_16_4_0_SWR_LT_LC	**1052.7**	504.48	4928.94	0
9_32_32_4_0_SWR_LT_LC	0.0	0	0	0
10_64_64_4_0_SWR_LT_LC	0.0	0	0	0
12_8_16_4_0_SWR_LT_SC	**233.71**	103.8	1932	0
13_8_32_4_0_SWR_LT_SC	**792.59**	339.05	2885.88	587.37
14_8_64_4_0_SWR_LT_SC	**1746.28**	982.79	5220.91	1512.53
16_16_16_4_0_WR_ST_LC	**58.26**	25.21	1198.02	0
17_32_32_4_0_WR_ST_LC	**169.56**	48.31	2226.31	0
18_64_64_4_0_WR_ST_LC	**583.36**	220.87	817.72	0
20_8_16_4_0_WR_ST_SC	**39.01**	8.97	1056.83	0
21_8_32_4_0_WR_ST_SC	**218.2**	114.66	1631.92	0
22_8_64_4_0_WR_ST_SC	**2319.5**	1253.46	8377.45	1913.96
24_16_16_4_0_SWR_ST_LC	**43.04**	16.38	496.32	0
25_32_32_4_0_SWR_ST_LC	**111.39**	28.97	1267.38	0
26_64_64_4_0_SWR_ST_LC	0.0	0	0	0
28_8_16_4_0_SWR_ST_SC	**61.91**	37.23	802.87	0
29_8_32_4_0_SWR_ST_SC	**175.14**	74.96	3104.87	1471.17
30_8_64_4_0_SWR_ST_SC	**2562.67**	907.41	7098.35	1973.44
32_16_16_4_0_UR_UT_LC	**26.84**	7.65	2280.48	0
33_32_32_4_0_UR_UT_LC	**125.85**	44	4616.29	0
34_64_64_4_0_UR_UT_LC	**391.41**	217.62	1912.84	0
36_8_16_4_0_UR_UT_SC	**42.32**	11.29	1703.15	0
37_8_32_4_0_UR_UT_SC	**538.5**	203.95	8618.86	0
38_8_64_4_0_UR_UT_SC	**3230.91**	2455.98	14807	0

Figure 4.3: Inverted generational distance plus (IGD+) statistics for DR-TACO and the greedy baseline. Bolded values indicate a statistically significant difference according to the Wilcoxon Ranked Sign test.

Problem	DR-TACO Mean	Greedy Mean	Comp.Mean	Phero(t) Mean	C*D Mean
0_16_16_4_0_WR_LT_LC	477.97	960.14	785.38	907.15	**441.79**
1_32_32_4_0_WR_LT_LC	0.0	0	0	0.0	0.0
2_64_64_4_0_WR_LT_LC	0.0	0	0	0.0	0.0
4_8_16_4_0_WR_LT_SC	712.72	3014.9	1569.84	1835.45	**618.43**
5_8_32_4_0_WR_LT_SC	789.24	2205.3	1007.51	2205.3	**556.04**
6_8_64_4_0_WR_LT_SC	1870.62	4368	2531.29	3220.47	**1552.71**
8_16_16_4_0_SWR_LT_LC	1052.7	4928.94	2290.19	1521.01	**908.71**
9_32_32_4_0_SWR_LT_LC	91.11	91.11	91.11	91.11	**88.08**
10_64_64_4_0_SWR_LT_LC	0.0	0	0	0.0	0.0
12_8_16_4_0_SWR_LT_SC	301.51	2092.61	487.99	763.91	**281.61**
13_8_32_4_0_SWR_LT_SC	786.82	2857.06	1132.86	1914.45	**672.92**
14_8_64_4_0_SWR_LT_SC	**1746.28**	5220.91	2148.4	2779.3	1771.62
16_16_16_4_0_WR_ST_LC	**71.03**	1207.99	104.12	159.75	80.73
17_32_32_4_0_WR_ST_LC	203.67	2498.48	318.81	475.94	**188.13**
18_64_64_4_0_WR_ST_LC	**581.64**	776.05	776.05	719.98	586.59
20_8_16_4_0_WR_ST_SC	39.85	1086.45	62.72	97.49	**38.69**
21_8_32_4_0_WR_ST_SC	**277.17**	1717.31	715.27	627.28	282.71
22_8_64_4_0_WR_ST_SC	2198.7	8239.64	3980.23	4593.54	**1575.9**
24_16_16_4_0_SWR_ST_LC	70.88	588.16	83.3	103.12	**65.51**
25_32_32_4_0_SWR_ST_LC	128.72	1219.76	358.67	143.65	**101.47***
26_64_64_4_0_SWR_ST_LC	105.11	105.11	105.11	**98.56**	103.99
28_8_16_4_0_SWR_ST_SC	66.55	802.53	70.04	134.63	**46.73***
29_8_32_4_0_SWR_ST_SC	440.59	3594.51	843.09	620.13	**335.9***
30_8_64_4_0_SWR_ST_SC	3123.01	7667.34	3635.79	4120.69	**2906.77**
32_16_16_4_0_UR_UT_LC	**29.66**	2364.47	37.76	109.36	30.3
33_32_32_4_0_UR_UT_LC	172.53	4077.48	332.26	408.56	**169.08**
34_64_64_4_0_UR_UT_LC	795.99	2562.25	2456.64	991.17	**723.88**
36_8_16_4_0_UR_UT_SC	**65.74**	1675.5	65.75	151.81	66.1
37_8_32_4_0_UR_UT_SC	554.55	8507.97	1321.89	1380.12	**405.96***
38_8_64_4_0_UR_UT_SC	4310.54	16534.4	10739	14601.44	**2741.93***

Figure 4.4: Inverted generational distance plus (IGD+) statistics for the variants of DR-TACO and the greedy baseline. Bolded values indicate the lowest mean, while starred values are significantly lower according to the Wilcoxon Ranked Sign test.

are attempting to solve both the coalition formation and scheduling problems (both difficult combinatorial optimization problems by themselves) at once. This motives an attempt to decouple these two problems and solve them in a pseudo-hierarchical manner. The next section of this paper discusses that research.

Chapter 5

Swarm Ant System for the Collab-MTSP

Despite the success of the TACO algorithm on the MTSP, extending it directly to the Collab-MTSP proved difficult due to the significantly higher degree of complexity of the Collab-MTSP compared to the standard MTSP. Much of this complexity to due to the fact that the Collab-MTSP can actually be viewed as two difficult combinatorial optimization problems in one, coalition formation and task scheduling. The TACO algorithm creates robot coalitions and schedules simultaneously, essentially trying to solve both problems at once. This approach, while effective for small problem sizes, does not scale well to larger problems. This approach also allows for deadlocks to arise that limit the ability of the TACO algorithm to effectively explore the solution space of the Collab-MTSP. These two qualities are undesirable, so we propose a new ACO-based algorithm that decouples coalition formation and scheduling to sidestep the deadlock problem. This approach eliminates the possibility of deadlock by making ants represent the *entire swarm* instead of just single robots. This allows ants to explore the solution space without any dependencies on other ants. We call this new method *Swarm Ant System (SAS)*. As the ants visit tasks, they assign members of the swarm to that task based on an *assignment function*. The selection of the assignment function is a critical design decision, and we compare the performance of SAS on three different assignment functions. We also evaluated a modified SAS that uses another pheromone matrix and heuristic function to perform assignments. Due to the "dual" nature of this solution using separate pheromone matrices to optimize for each of route planning and coalition formation, we call this method *Dual Swarm Ant System (DSAS)*. The details of these algorithms and the experimental evaluation is described in this chapter.

5.1 Swarm Ant System

Much of SAS is similar to what has been described in earlier sections. A greedy solution is used to seed the Pareto set and establish the initial value of the pheromones and many Pareto-optimal solutions are returned when it terminates. The most novel addition to this method is the solution construction phase that is described next.

5.1.1 Solution construction

The solutions construction step of SAS is relatively simple compared to that of DR-TACO, by design. At each SAS iteration, we run N_a ants to generate N_a candidate solutions. Note the subtle difference between this and DR-TACO: in DR-TACO, many *ants* co-constructed a single candidate solution. In SAS, a single ant forms a single solution.

To form a solution, an ant incrementally steps through the unassigned tasks, selecting them based on a single pheromone similar to that of Ant System. At each task, the ant assigns robots to the task using the assignment function. Many assignment functions are evaluated in this work. The paths of the individual robots are determined by the path of the ant. This is clearly shown in figure 5.1. Robots that are not

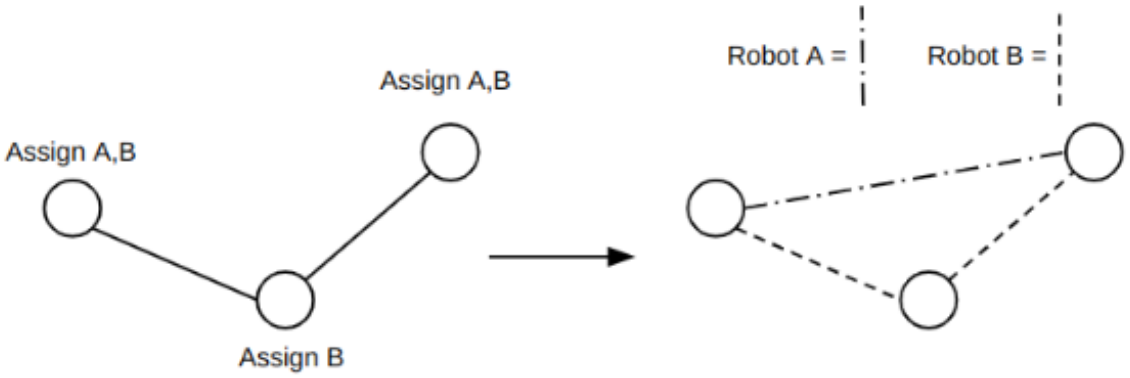

Figure 5.1: A diagram showing how robot paths get deconstructed from an ant's path. On the left, the ant visits 3 tasks and assigns robots to cover those tasks. On the right are the corresponding robot paths.

Algorithm 7: Solution Construction

Result: N_g solutions to the Collab-MTSP
$S \leftarrow empty$;
for $i = 1$ *to* N_g **do**
  $F \leftarrow x \forall\ x\ \in \mathbf{T} \mid x < k$;
  $P_i \leftarrow [0], \forall\ i \in \mathbf{N} \mid i < k$;
  while F *is not empty* **do**
    Choose task *next* according to (**??**);
    Assign coalition *contributing* based on assignment function;
    Append *next* to $P_i\ \forall\ i \in$ ids(*contributing*);
    $F \leftarrow F - \{next\}$;
  end
  Append 0 to the end of all paths P_i
end
return S

assigned to a given task are not affected by the ant's movement to that task, and all robots to a task are assigned at once. This means that there is no "waiting" in SAS as there is in DR-TACO. This allows the ant to more freely explore the solution graph to the problem and thus generate solutions more efficiently than in DR-TACO. The pheromone for the swarm is updated using the same methodology as from TACO and DR-TACO.

5.1.2 SAS Assignment Function

The general structure of the proposed assignment functions for SAS is shown in Algorithm 8. The viable robots (robots that have the skills to contribute to the task) are sorted based on an evaluation metric. They are assigned in a first-come-first-serve manner based on their order and if they can contribute to the task. We ran experiments with 3 different metrics to sort by which are the distance from the robot to the current task, the path cost so far, and the sum of both. The assignment function prioritizes allocating robots with the smallest metrics.

5.1.3 Dual Swarm Ant System Assignment Function

The only difference between DSAS and SAS is the use of a pheromone-based assignment function. The size of this pheromone matrix is $R \times T$ since there is a pheromone for each robot as each task. The pheromone

Algorithm 8: SAS Assignment

Data: n is the current task
Data: sorted is a list of robot_ids sorted by the selected metric
Result: Coalition assigned to task n
$C \leftarrow empty$;
$S \leftarrow \text{skillset}(T_{\text{next}})$;
for robot_id *in* sorted **do**
  if $\text{skillset}(R_{\text{robot\_id}}) \subset S$ **then**
    $C = C + \text{robot\_id}$;
    $S = S - \text{skillset}(R_{\text{robot\_id}})$;
  end
  break if S is empty;
end
return C

$\phi_i^{(r)}$ for robot r at task i is used with a heuristic $h_i^{(r)}$ to select the next robot to assign to the task. Robot r is assigned to task i with probability $\frac{\phi_i^{(r)} h_i^{(r)}}{\sum_{q \in \mathbf{R}} \phi_i^{(q)} h_i^{(q)}}$ where $\mathbf{R}$ is the set of remaining robots that have the skills needed to contribute to task i. The assignment pheromones are updated in the same way as the path pheromones. The heuristics used for DSAS mirror the assignment function from SAS, being a distance, path-so-far, and combined heuristic.

5.2 Results

To evaluate the performance of SAS and DSAS, we evaluated their performance on the same benchmarks as used for TACO in Chapter 4 and used TACO as a baseline. Boxplots for 3 problems are shown in figure 5.2. Eight algorithm variants are compared: DR-TACO (DT), Greedy (GDT), SAS with distance heuristic (SAS-C), SAS with path-so-far heuristic (SAS-P), SAS with a combined heuristic (SAS-T), and DSAS with the same heuristic (DSAS-C, DSAS-P, and DSAS-T, respectively). The overall best-performing algorithm was Swarm Ant System with the closeness assignment function (SAS-C) with the lowest mean IGD+ on 24/36 problems. In general, DSAS performed worse than its SAS counterpart with the same heuristic. DR-TACO tended to perform worse than SAS-C, but was usually comparable or superior to the rest of the algorithms. All approaches outperformed the greedy baseline on most problems.

5.3 Discussion

Swarm Ant System represents a fundamentally different ACO formulation of the Collab-MTSP, representing the whole swarm as an ant rather than individual robots. This facilitates exploring the solution space more easily than with DR-TACO because deadlocks cannot occur in solution construction. As a tradeoff for this ease of exploration, a new design decision in designing an *assignment function* is introduced. We evaluated two classes of assignment function - one based deterministically on features of the path states (SAS) and another that uses a second pheromone matrix to assign tasks (DSAS). The results showed that the SAS approaches consistently outperformed the DSAS approaches, but the overall quality of the solutions was highly dependent on the choice of assignment function. Some of the difficulty in using a pheromone-based assignment function is that the optimal assignment of robots for a given task has a complicated dependency on the paths that have been created so far and the remaining tasks to be completed which is likely difficult to capture in a matrix of values.

SAS-C performed at least as well as DR-TACO on most of the benchmark problems and outperformed it on many, especially on the problems at the largest scales. Though on smaller-scale problems, TACO could still perform the best. As a result, we now have two algorithms that can produce good solutions to the Collab-MTSP and seem to perform best on different types of problems. Further analysis is needed to

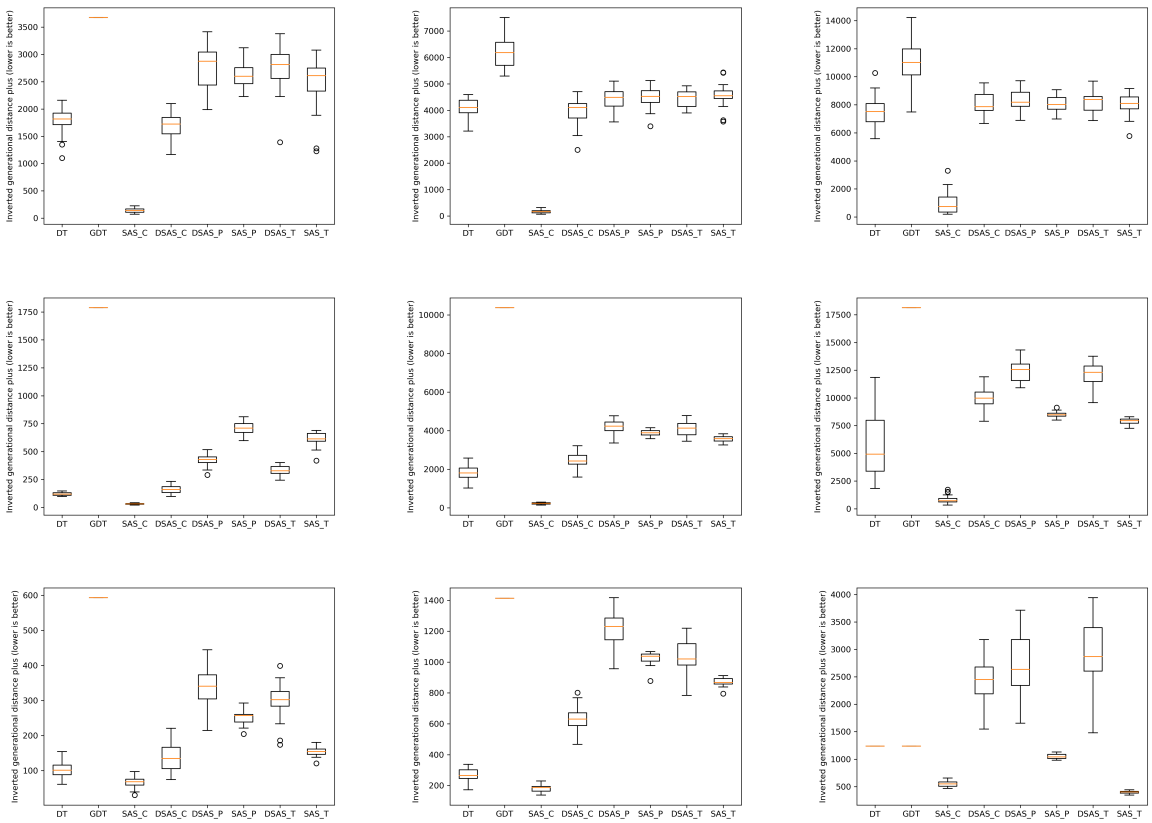

Figure 5.2: Example IGD+ boxplots for 3 problems at 3 scales. For IGD+, lower values are better.

evaluate which problem classes these algorithms perform best. Future research will deeply explore wall-clock time costs and solution quality on problems with different spatial qualities and of increasing scale.

Chapter 6

Conclusions

This work presents an exploration of Ant Colony Optimization (ACO) methods on two formulations of the Multi-Robot Task Allocation (MRTA) problem: the standard Multiple Travelling Salesmen Problem (MTSP) and an extension of the MTSP with synchronous collaboration constraints we called the Collab-MTSP. The Collab-MTSP introduces an embedded coalition formation problem entangled with the already difficult to solve MRTA problem. We formulated both problems as multi-objective optimization problems (MOOPs) where we care about optimizing both the summed cost of all robot paths as well as the maximum cost of any robot in the swarm. As a result, we explore ACO techniques that return sets of many Pareto-optimal solutions that assume no *a priori* knowledge of the downstream preferences between both objectives.

With regards to the MTSP, we introduced a novel ACO-based method called Territorial Ant Colony Optimization (TACO) that outperforms competing state-of-the-art metaheuristic techniques for the MTSP on commonly used TSPLIB benchmark datasets. The main novelties of TACO are the introduction of *territorial pheromones* and *willingness*. Taking advantage of the fact that each ant represents a single robot, each ant is assigned its own pheromone to guide its search. Additionally, the willingness of an ant to move to a new task is affected by the distribution of other ant pheromones at that node, effectively allowing ants to claim territory over tasks.

We present two new methods for solving the Collab-MTSP, an extension of TACO called TACO with Deadlock Reversal (DR-TACO) and another method called Swarm Ant System (SAS). When extending TACO to the Collab-MTSP, a new challenge arises which is handling *deadlocks*. Since robots need to form coalitions to complete tasks, it becomes possible for circular dependencies to occur while iteratively constructing a solution. We propose a *deadlock reversal* step that allows ants to undo actions they have previously taken to remove deadlocks and continue constructing a solution. SAS reformulates the problem completely by casting the entire swarm as an ant, rather than an individual robot. This sidesteps the deadlocks problem by constructing robot paths based on the ant path and by introducing an *assignment function* that assigns a coalition of robots to a task at once. Our results show that both DR-TACO and SAS are effective solution techniques for the Collab-MTSP, achieving near-optimal performance on small benchmarks and significantly outperforming a greedy baseline on larger problems. SAS generally outperforms DR-TACO in terms of solution quality, especially as the problem scales to more tasks and robots. We attribute this to the higher degree of simplicity in SAS as well as the ease with which it can explore the solution space of the Collab-MTSP with its ant-as-swarm formulation.

The most significant contribution of this work is the comparison of algorithms that represent two fundamentally different ways of formulating ACO for MTRA, these being DR-TACO for the ant-as-robot formulation and SAS for the ant-as-swarm formulation. Recent literature has gravitated toward the former [15, 24], likely due to the straightforward intuition that underpins its design. However, this work shows that this method of the formulation does not scale well as more constraints are added to the problem since effectively handling these changes is not straightforward as they become increasingly complicated to implement and hinder the exploration ability of ACO. On the other hand, SAS presents a fundamentally different way of using ACO for these problems that is more scalable with problem size and extendable with additions to the problem formulation.

Accordingly, we recommend further analysis of the strengths and weaknesses of this technique. In partic-

ular, how these techniques fair with different classes of MRTA problems. A detailed analysis of the empirical wall-clock time cost of these methods is also needed and not included in this work. In further research, we are interested in exploring these critical research questions as well as boosting SAS with machine learning techniques and further extending it to more constrained versions of the Collab-MTSP.

Bibliography

[1] A. Aswale and C. Pinciroli, "Heterogeneous Coalition Formation and Scheduling with Multi-Skilled Robots." https://arxiv.org/abs/2306.11936v1, June 2023.

[2] M. Dorigo and T. Stützle, "Ant Colony Optimization: Overview and Recent Advances," in *Handbook of Metaheuristics* (M. Gendreau and J.-Y. Potvin, eds.), vol. 272, pp. 311–351, Cham: Springer International Publishing, 2019.

[3] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy," *Computer Science Review*, vol. 40, p. 100369, May 2021.

[4] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative Heterogeneous Multi-Robot Systems: A Survey," *ACM Computing Surveys*, vol. 52, pp. 29:1–29:31, Apr. 2019.

[5] B. P. Gerkey and M. J. Matarić, "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.

[6] G. Oh, Y. Kim, J. Ahn, and H.-L. Choi, "PSO-based Optimal Task Allocation for Cooperative Timing Missions," *IFAC-PapersOnLine*, vol. 49, pp. 314–319, Jan. 2016.

[7] C. Mouradian, J. Sahoo, R. Glitho, M. Morrow, and P. Polakos, "A Coalition Formation Algorithm for Multi-Robot Task Allocation in Large-Scale Natural Disasters," Apr. 2017.

[8] M. U. Arif, "Robot coalition formation against time-extended multi-robot tasks," *International Journal of Intelligent Unmanned Systems*, vol. ahead-of-print, Jan. 2021.

[9] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," p. 26, 1996.

[10] M. Dorigo, *Ant Colony Optimization*. Cambridge, Mass: MIT Press, 2004.

[11] M. Lippi and A. Marino, "A Mixed-Integer Linear Programming Formulation for Human Multi-Robot Task Allocation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pp. 1017–1023, Aug. 2021.

[12] Y. Emam, S. Mayya, G. Notomista, A. Bohannon, and M. Egerstedt, "Adaptive Task Allocation for Heterogeneous Multi-Robot Teams with Evolving and Unknown Robot Capabilities," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7719–7725, May 2020.

[13] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An Optimal Task Allocation Strategy for Heterogeneous Multi-Robot Systems," in *2019 18th European Control Conference (ECC)*, pp. 2071–2076, June 2019.

[14] A. Koubâa, O. Cheikhrouhou, H. Bennaceur, M.-F. Sriti, Y. Javed, and A. Ammar, "Move and Improve: A Market-Based Mechanism for the Multiple Depot Multiple Travelling Salesmen Problem," *Journal of Intelligent & Robotic Systems*, vol. 85, pp. 307–330, Feb. 2017.

[15] S. Wang, Y. Liu, Y. Qiu, Q. Zhang, F. Huo, Y. Huangfu, C. Yang, and J. Zhou, "Cooperative Task Allocation for Multi-Robot Systems Based on Multi-Objective Ant Colony System," *IEEE Access*, vol. 10, pp. 56375–56387, 2022.

[16] X. Chen, P. Zhang, G. Du, and F. Li, "Ant Colony Optimization Based Memetic Algorithm to Solve Bi-Objective Multiple Traveling Salesmen Problem for Multi-Robot Systems," *IEEE Access*, vol. 6, pp. 21745–21757, 2018.

[17] C. Wei, Z. Ji, and B. Cai, "Particle Swarm Optimization for Cooperative Multi-Robot Task Allocation: A Multi-Objective Approach," *IEEE Robotics and Automation Letters*, vol. 5, pp. 2530–2537, Apr. 2020.

[18] Y. Shuai, S. Yunfeng, and Z. Kai, "An effective method for solving multiple travelling salesman problem based on NSGA-II," *Systems Science & Control Engineering*, vol. 7, pp. 108–116, Nov. 2019.

[19] Z. Wang and J. Zhang, "A task allocation algorithm for a swarm of unmanned aerial vehicles based on bionic wolf pack method," *Knowledge-Based Systems*, vol. 250, p. 109072, Aug. 2022.

[20] M. Lopez-Ibanez and T. Stutzle, "The automatic design of multiobjective ant colony optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 861–875, 2012.

[21] I. Alaya, C. Solnon, and K. Ghédira, "Ant colony optimization for multi-objective optimization problems," in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, vol. 1, pp. 450–457, 2007.

[22] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A Multiobjective Evolutionary Algorithm Using Decomposition and AntColony," *IEEE Transactions on Cybernetics*, vol. 43, pp. 1845–1859, Dec. 2013.

[23] W. Deng, J. Xu, and H. Zhao, "An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.

[24] C. Bao, Q. Yang, X.-D. Gao, Z.-Y. Lu, and J. Zhang, "Ant colony optimization with shortest distance biased dispatch for visiting constrained multiple traveling salesmen problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, (New York, NY, USA), pp. 77–80, Association for Computing Machinery, July 2022.

[25] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored Traveling Salesman Problem," *IEEE Transactions on Cybernetics*, vol. 45, pp. 2390–2401, Nov. 2015.

[26] S. Han, M. Xu, X. Dong, Q. Lin, and F. Shen, "Hybrid ITÖ algorithm for multi-scale colored traveling salesman problem," *Journal of Computer Applications*, vol. 42, p. 695, Mar. 2022.

[27] H. Zhang, Q. Zhang, L. Ma, Z. Zhang, and Y. Liu, "A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows," *Information Sciences*, vol. 490, pp. 166–190, July 2019.

[28] U. M. Yildirim and B. Çatay, "A time-based pheromone approach for the ant system," *Optimization Letters*, vol. 6, pp. 1081–1099, Aug. 2012.

[29] C. J. Shannon, L. B. Johnson, K. F. Jackson, and J. P. How, "Adaptive mission planning for coupled human-robot teams," in *2016 American Control Conference (ACC)*, pp. 6164–6169, July 2016.

[30] H. Zhao and C. Zhang, "An ant colony optimization algorithm with evolutionary experience-guided pheromone updating strategies for multi-objective optimization," *Expert Systems with Applications*, vol. 201, p. 117151, Sept. 2022.

[31] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *INFORMS Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[32] 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.