ABSTRACT

Title of Thesis:	MULTI-VEHICLE ROUTE PLANNING FOR CENTRALIZED AND DECENTRALIZED SYSTEMS
	Ruchir Jayesh Patel, Master of Science, 2019
Thesis Directed By:	Professor Jeffrey W. Herrmann Professor Shapour Azarm Department of Mechanical Engineering

Multi-vehicle route planning is the problem of determining routes for a set of vehicles to visit a set of locations of interest. In this thesis, we describe a study of a classical multi-vehicle route planning problem which compared existing solutions methods on min-sum (minimizing total distance traveled) and min-max (minimizing maximum distance traveled) cost objectives. We then extended the work in this study by adapting approaches tested to generate robust solutions to a failure-robust multi-vehicle route planning problem in which a potential vehicle failure may require modifying the solution, which could increase costs. Additionally, we considered a decentralized extension to the multi-vehicle route planning problem, also known as the decentralized task allocation problem. The results of a computational study show that our novel genetic algorithm generated better solutions than existing approaches on larger instances with high communication quality.

MULTI-VEHICLE ROUTE PLANNING FOR CENTRALIZED AND DECENTRALIZED SYSTEMS

by

Ruchir Jayesh Patel

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2019

Advisory Committee:
Professor Jeffrey W. Herrmann, Chair and Advisor
Professor Shapour Azarm, Co-Advisor
Assistant Professor Michael Otte

Acknowledgements

The work presented in Chapter 3 was supported in part by the Naval Air Warfare Center Aircraft Division - Pax River, MD, through a cooperative agreement, N004211720018. This support is gratefully acknowledged.

The work presented in Chapter 4 was supported in part by a contract from Toyon Research Corporation as part of a U.S. Navy STTR project and a contract from the Air Force Research Lab (AFRL), contract number FA87501820114. This support is gratefully acknowledged.

The work presented in Chapter 5 was supported in part by a contract from the Air Force Research Lab (AFRL), contract number FA87501820114. This support is gratefully acknowledged.

Table of Contents

Abbreviations

ACBBA	Asynchronous Consensus-based Bundle Algorithm
CBAA	Consensus-based Auction Algorithm
CBBA	Consensus-based Bundle Algorithm
DH	Decentralized Hungarian
GA	Genetic Algorithm
HIPC	Hybrid Information and Plan Consensus
ILP	Integer Linear Programming
MST	Minimum Spanning Tree
mTSP	multiple Traveling Salesman Problem
NN	Nearest Neighbors
PIA	Performance Impact Algorithm
SA	Simulated Annealing
VRP	Vehicle Routing Problem
-mm	Min-max variant
-multi	Multi-objective variant

Symbols

a_i	Time spent at location i
c_{ij}	Cost of edge from node i to node
c_j	Cost of node j meaning the cost of a vehicle's route up to node j
c_{max}	Maximum cost of all nodes
$\boldsymbol{D}$	Set of depot locations
m	Number of vehicles
M	Two times the maximum edge cost plus one
n	Number of locations of interest
p	Maximum number of nodes a vehicle can visit
$\boldsymbol{q}$	Set of unvisited locations after failure occurs
s	Failure location (scenario)
T	Temperature used in simulated annealing
t_F	Time at which failure occurs
t_i	Time vehicle leaves location i
u_i	Position of node i in vehicle's route
v	Speed of vehicles
x_{ij}	Binary variable indicating if edge from node i to node j is used in a solution
x_{ijk}	Binary variable indicating if vehicle k uses the edge from node i to node j in a solution
y_{ijk}	Binary variable indicating if vehicle k uses the edge from node i to node j in a replanning solution

Chapter 1: Introduction

1.1 *Motivation*

Multi-vehicle systems have gained popularity in recent years in both commercial and military applications. Such systems are advantageous because they allow for coordination among vehicles which increases efficiency in accomplishing mission tasks or goals. Coordination (or planning) in these systems typically involves determining routes for vehicles which are defined as sequences of locations of interest for vehicles to visit. The costs of these routes often define the mission performance in multi-vehicle systems. Therefore, planning optimal routes for vehicles is of primary interest to mission planners. There exist many approaches for generating optimal or near-optimal routes in different types of problem scenarios. A well-studied scenario, often at the root of many multi-vehicle route planning problems, is that of the classical Vehicle Routing Problem (VRP) or multiple Traveling Salesman Problem (mTSP). In this scenario, the goal for the central planner is to determine optimal routes for vehicles to visit a set of known locations starting and ending at a singular point of origin called the depot. Although many approaches exist for solving this classical problem, understanding their advantages and disadvantages with different mission objectives is important for applying them in more complex planning scenarios.

One such scenario is multi-vehicle route planning considering uncertainty. This problem is of interest because it is relevant for practical applications such as search and rescue or drone delivery where many kinds of uncertainties exist. In these

applications, failure uncertainty is present meaning it is possible that vehicles fail while completing mission tasks potentially due to environmental or adversarial influence. When a vehicle fails, the routes initially planned for the remaining vehicles may not satisfy mission requirements. In such cases, new routes need to be generated through replanning which can significantly increase the mission cost if the initial routes interact poorly with the replanning step. This motivates the desire to mitigate this effect by considering the possibility of failure in the initial route planning. If a distribution of failure likelihood is known, then the uncertainty can be accounted for in the planning objective using an expected value over the distribution. However, it is likely that the uncertainty distribution is not known in practice. In such a case, the uncertainty can be accounted for by optimizing the worst-case cost of the initial route plan over all possible failure scenarios. We refer to the worst-case cost as the robustness of a solution. Thus, one solution is more robust than another if the first solution's worst-case cost is better than the second solution's worst-case cost.

Another planning scenario of practical interest is decentralized multi-vehicle route planning. This problem is commonly referred to as decentralized task allocation in the literature. The previous scenarios discussed consider centralized systems where a centralized planner can determine routes for all vehicles using global information about the system. In decentralized systems, each vehicle determines its own route based on local information. This removes the need for a centralized planner, thereby increasing the adaptability and reliability of the system. This scenario is especially of interest in communication-limited environments where communication can be lossy meaning messages sent between vehicles can be dropped.

1.2 *Research Questions*

Many multi-vehicle route planning approaches exist in the literature, each tailored for different problem scenarios. We explored what approaches exist and how they perform for centralized deterministic and stochastic scenarios, as well as a decentralized scenario. We considered min-sum and min-max objectives which are minimizing the total distance traveled and maximum distance traveled by the vehicles, respectively. Performance measures included computational time and solution quality. The particular questions we seek to answer in this thesis are:

- What are the best performing methods for planning vehicles' routes for a classical mTSP with min-sum and min-max objectives? How does the performance of these methods change as the problem size increases?
- How can we plan robust routes for vehicles when there is a chance that vehicles can fail in the classical routing problem?
- What are the best approaches for real-time decentralized route planning in communication-limited environments? How reliable are these methods at different levels of communication quality?

1.3 *Organization*

Chapter 2 discusses related previous work to multi-vehicle route planning covering classical routing, stochastic and robust routing problems, and decentralized task allocation. Chapter 3 discusses the classical route planning problem and presents results comparing popular solution methods. Chapter 4 discusses the failure-robust route planning problem and analyzes the robustness of solutions found by a genetic algorithm (GA). Chapter 5 discusses the decentralized task allocation problem and

presents results comparing standard approaches and a novel GA approach at varying levels of communication. Chapter 6 discusses a summary of conclusions drawn in the previous chapters, contributions made, and presents some ideas for future directions.

Chapter 2: Related Work

2.1 *Introduction*

This chapter describes previous work related to the topics considered in this thesis. This includes centralized classical and robust route planning problems as well as decentralized route planning which is also known as decentralized task allocation.

The rest of this chapter is organized as follows: Section 2.2 discusses work related to classical multi-vehicle route planning and Section 2.3 describes previous work on stochastic and robust route planning. Section 2.4 discusses work related to decentralized task allocation. Finally, Section 2.5 describes the research gaps present in the literature.

2.2 *Classical Vehicle Route Planning*

Classical route planning has been studied in the past, commonly seen as the VRP or mTSP where salesmen are vehicles. There already exist survey papers on the popular min-sum variant of the mTSP. For instance, Bektas [1] discussed various formulations and solution procedures used to solve the mTSP exactly and approximately. One approach transforms the mTSP into a Traveling Salesman Problem (TSP) and solves the resultant TSP exactly or heuristically. Jonker and Volgenant [2] described this transformation in detail. Bektas [1] and Singh [3] both described an integer linear programming formulation that can be solved using branch-and-bound. Although branch-and-bound and branch-and-cut can find exact solutions to the mTSP, they can be computationally expensive as the size of the problem increases. Indeed, the mTSP is considered to be NP-hard [4]. Therefore, search

algorithms (heuristics or meta-heuristics) are often used to generate high-quality solutions quickly. Bektas [1] described some common algorithms including simulated annealing, greedy algorithms, neural networks, and genetic algorithms. Singh [3] described how meta-heuristic algorithms, especially genetic algorithms, are becoming increasingly vital for solving larger instances of the mTSP. Meta-heuristics have also seen popularity in solving VRPs [5, 6] which can be considered a variant of the mTSP with additional constraints. Potvin [7] reviewed state-of-the-art evolutionary algorithms in detail and their many applications for solving the VRP.

Although the min-max mTSP is not as widely studied as the min-sum version, several researchers have considered the min-max mTSP and related problems such as the VRP. França et al. [8] claimed to be the first to solve the min-max mTSP exactly. They proposed two exact methods that utilize approaches for solving Distance-constrained VRPs (DVRPs). The DVRP is similar to the min-sum mTSP with the addition of a constraint on the maximum distance any salesman can travel. Along with these exact methods, they proposed a Tabu search algorithm to solve the problem inexactly. As with the min-sum mTSP, heuristics are popular for quickly generating high-quality solutions to the min-max mTSP. For example, Somhom et al. [9] proposed a competition-based neural network approach, and Ren [10] used a hybrid genetic algorithm to solve a Min-Max Vehicle Routing Problem (MMVRP). Applegate et al. [11] also considered the MMVRP and described a branch-and-cut search to solve the min-max problem. The model they proposed duplicates the problem for each vehicle. Bertazzi et al. [12] studied the min-sum VRP in comparison to the MMVRP. They derived relations between the two problems and a tight bound

on the optimal solution cost for min-sum and min-max formulations. Matsuura and Numata [13] proposed two new heuristic methods based on Tabu search and a chaotic neural network for solving the min-max mTSP. They compared common heuristic algorithms for the min-max mTSP formulation based on solution quality after a certain amount of iterations (solutions).

2.3 *Route Planning Considering Uncertainty*

Multi-vehicle route planning with uncertainty can be considered an extension of classical route planning. As such, others have studied VRPs and mTSPs with consideration for different kinds of uncertainties. Ritzinger et al. [14] presented a survey of stochastic VRPs and described the most commonly studied sources of uncertainty, including stochastic travel times, demands, customers, and combinations of these. The objective in stochastic VRPs and mTSPs is often to minimize the expected costs where a probability distribution is available for uncertain parameters. Sundar et al. [15] studied such a problem in the context of path planning for multiple heterogeneous unmanned vehicles with uncertain service times. They formulated the problem with an expected value objective and solved it using a branch-and-cut algorithm.

Dulai et al. [16] studied the "fault-tolerant" VRP, which is similar to the problem considered in Chapter 4 that considers failure uncertainty, and solved it using a heuristic method based on the Clarke and Wright savings algorithm [17]. They assumed that the failure scenarios are equally likely and optimized the expected value. In their problem, only one surviving vehicle is used to visit the locations on the

failed vehicle's route. However, in applications such as multi-vehicle search, using multiple vehicles to cover the unvisited locations can lead to lower costs.

Problems that utilize expected value require knowledge of probability distributions for uncertain parameters. Robust variants of these problems such as the robust VRPs studied in [18] and [19], however, do not require distributions to be known because the objective is to optimize the worst-case performance over all possible realizations of uncertain parameters.

Vehicle failure has been considered by researchers of autonomous multi-vehicle systems. Habib et al. [20] studied an Open Multiple Depot VRP (OMDVRP) in the context of cooperative Unmanned Aerial Vehicle (UAV) path planning in which each vehicle starts at its own depot and visits a set of locations without returning to the initial depot. They used a branch-and-bound algorithm to find solutions to a Mixed-Integer Linear Programming (MILP) formulation of the problem in which a UAV fails. Giger et al. [21] focused on mission replanning for Unmanned Underwater Vehicles (UUVs) after vehicle failure. They modeled the problem as a Multiple Depot mTSP (MDmTSP) and used a GA for replanning.

2.4 *Decentralized Task Allocation*

The previously discussed problem scenarios can be considered centralized meaning there exists a central planner that determines the routes for all vehicles using global information of the system. Decentralized task allocation is another extension of multi-vehicle route planning that removes the need for a centralized planner by making vehicles determine their own routes using local information. Khamis et al. [22] and Singhal and Dahiya [23] presented surveys of popular approaches for

solving decentralized task allocation problems which generally can be classified into market-based or optimization-based approaches. In market-based approaches, there exist auctioneers that decide how tasks are allocated based on bids from other vehicles. In decentralized auction approaches, every vehicle acts as an auctioneer and uses bids from all other vehicles to "auction" tasks. Khamis et al. [22] noted these approaches have advantages in scalability and adaptability. Optimization-based approaches generally include approaches that utilize distributed constraint optimization, game theory, metaheuristics, or other well-known optimization techniques. Khamis et al. [22] noted that some of these approaches aim to produce optimal task allocations. This can reduce their scalability since the task allocation problem is a variant of the mTSP which is NP-hard.

Market- or auction-based approaches have recently seen popularity in the literature with the introduction of consensus-based auction approaches. Most notably, the Consensus-Based Auction Algorithm (CBAA) and Consensus-Based Bundle Algorithm (CBBA) [24] have been widely cited due to their scalability and optimality bounds. These approaches attempt to maximize reward over the entire system, similar to minimizing total cost. They accomplish this by iterating between auction and consensus phases where in the auction phase, the assignment is greedily created using local information. The local information is then updated in the consensus phase via communication with other vehicles. Many practical extensions have been proposed to the CBBA such as the Asynchronous CBBA (ACBBA) [25] which improves performance when communication is asynchronous between vehicles. It modifies the consensus rules presented in [24] to accomplish this. Others have tried to directly

improve the performance of the CBBA. Zhao et al. [26] proposed the Performance Impact Algorithm (PIA) which modifies the CBBA to utilize "significance" instead of bids when evaluating task values. The results of their study showed the PIA outperformed the CBBA on solution quality in a number of problem instances. How [27] presented the Hybrid Information and Plan Consensus (HIPC) approach which modifies the CBBA to utilize a centralized task allocation approach inside of the decentralized framework. This enables vehicles to determine their own assignment based on what they predict the assignments of other vehicles to be. Nanjananth and Gini [28] proposed a parallel repeated auctions approach where every vehicle is an auctioneer and bidder in parallel allowing for more adaptability.

Optimization-based approaches sometimes share similarities to these market-based approaches. The Decentralized Hungarian (DH) algorithm [29] is similar to the CBAA but replaces the auction step with solving a task assignment problem via the Hungarian method on a cost matrix. The Hungarian method is an algorithm that generates the optimal task assignment from a cost matrix. The study showed that the DH approach outperformed CBAA in terms of solution quality and computational time. Choi and Kim [30] proposed a two-stage GA-based approach where vehicles first determine their own routes using a GA and then communicate with other vehicles to exchange tasks if it reduces costs. They compared the GA-based approach to a centralized MILP approach and showed the improved scalability of the GA-based approach.

Often, it is practical to consider communication limitations when working with decentralized systems. Rantanen et al. [31] studied the performance degradation

of the ACBBA in lossy communication environments. Lossy communication means messages sent between vehicles can be dropped. Using a Wi-Fi communication model, they found that the number of redundant task assignments increases significantly as communication becomes more lossy. Otte et al. [32] presented a comparison of centralized auction algorithms in lossy environments on solution quality versus message acceptance probability. They found the solution quality degraded for all algorithms at very low acceptance probabilities.

2.5 *Research Gaps*

There exist several gaps in the literature discussed in the previous sections. The following are the gaps that we have identified:

(i) Although many methods for solving the min-sum and min-max mTSP or similar problems have been reported, there is a gap in the literature for a comparison study of some of the key methods for solving both min-sum and min-max problems in terms of the solution quality and computational time.

(ii) While stochastic and robust routing problems have been studied previously, there has been no consideration for robustness against vehicle failure. Replanning has been considered in an online context but there is a lack of approaches that incorporate replanning costs to plan robust initial routes.

(iii) Many decentralized task allocation approaches have been proposed in the past, however, there exists a gap in the performance analysis and comparison of these methods under different levels of communication quality. Another gap is present in the objectives of many of these approaches. They tend to optimize a min-sum type objective which can sometimes work poorly when minimizing mission time. A min-

max objective is typically better-suited for minimizing mission time which is often a primary objective in practice. The literature also lacks studies on evolutionary approaches, particularly genetic algorithms, for decentralized task allocation. These approaches can be easily decentralized and can greatly benefit from a parallelized scheme.

In this thesis, we address each of these gaps in Chapters 3, 4, and 5, respectively. Chapter 3 provides a comparison of standard methods used to solve the mTSP with min-sum and min-max objectives. Chapter 4 presents a method for generating route plans that are robust against the possibility of vehicle failure. Finally, Chapter 5 provides results comparing standard task allocation approaches and a novel GA approach at different levels of communication quality.

Chapter 3: Comparison of Methods for Solving a Classical Multi-Vehicle Route Planning Problem

3.1 *Introduction*

The multi-vehicle route planning problem consists of creating a route plan for a set of vehicles to visit locations of interest, representing tasks that need to be done. The objective typically is to optimize the plan with respect to one or more performance measures such as the total distance or time needed to visit all locations. There are various complexities that can be considered in this problem such as uncertainty in task locations, hazard avoidance, mission re-planning, and real-time considerations.

This chapter describes the results of a study that tested a variety of algorithms for solving a classical form of the problem with no uncertainty where the vehicles must visit a given set of locations of interest. It is assumed that all vehicles begin from the same start point (the depot), visit all of the locations, and return to the same start point. Two separate objectives were considered: minimizing the total distance traveled by all vehicles (min-sum) and minimizing the maximum distance traveled by any vehicle (min-max). This problem is equivalent to the well-known multiple Traveling Salesman Problem (mTSP) where the locations of interest are cities and the vehicles are the salesmen. For solving mTSP problems with the min-sum and min-max objectives, three search methods including a branch-and-bound technique, genetic algorithm (GA), and simulated annealing (SA) approach were considered. Three construction heuristics were also considered including a nearest neighbors

algorithm, minimum spanning tree (MST) approximation, and Christofides algorithm [33] were considered. These algorithms were tested and compared on multiple mTSP instances of different problem sizes to understand how algorithm performance varies by problem size.

The rest of this chapter is organized as follows: Section 3.2 describes the problem formulation as well as the formulation for the representative mTSP. Section 3.3 describes exact and heuristic solution methods for the mTSP. The experimental setup and results are presented and discussed for each of the methods in Sections 3.4 and 3.5. Finally, Section 3.6 summarizes and concludes this chapter.

3.2 *Problem Formulation*

3.2.1 **Problem Definition**

The multi-vehicle route planning problem considered in this chapter can be defined as follows: Given a set of *n* fixed locations of interest including the depot, visit all locations using *m* vehicles. The objective is to optimize either the min-sum objective or the min-max objective. No uncertainty exists in this problem; therefore, perfect information is known about the objects' locations by all vehicles. All of the vehicles start and end their routes at the same single location (the depot).

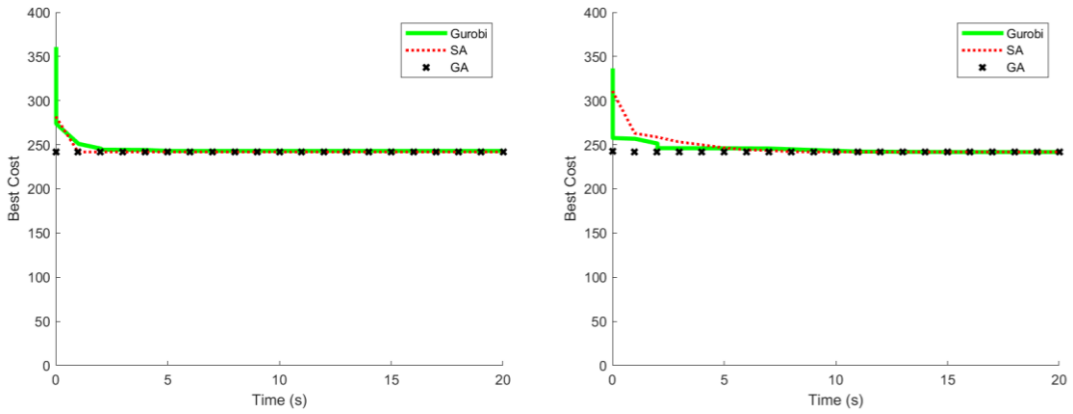
The mTSP representation of this problem shares a similar definition: Given a set of *n* cities including the depot, find routes for all *m* salesmen such that every city, except the depot, is visited by exactly one salesman, considering either the min-sum objective or min-max objective. All *m* salesmen start and end at the same single depot. The problem can be represented using a graph with *n* nodes, and *n* - 1 edges

from each node to every other node. Each edge has a cost c_{ij} which is the distance from node *i* to node *j*. The depot is city (or node) 1. The routes are paths in the graph.

The problem formulation makes the following assumptions: Except the depot, every location is visited by exactly one vehicle. Edges to and from the depot can be reused. This assumption is relevant when vehicles only visit one location and come immediately back to the depot. Each edge cost is symmetric, meaning c_{ij} is equal to c_{ji}. The cost of a path is independent of the vehicle.

For the min-sum objective, not all vehicles are required to be used in a solution. However, for the min-max objective, all vehicles are required to be used in a solution. Preliminary testing provided insights that led to these assumptions which improve the solution quality or convergence time of the search methods considered in this study. In preliminary tests, both objectives were considered with both assumptions. The three search methods were tested in these cases on a limited set of randomly generated problem instances and for ten trials on each case. The solution history was recorded for each method which refers to the cost of the best solution found by each method recorded over time. Time limits were set for each of the instances tested. For the min-sum objective, the results showed using less vehicles was better in improving solution quality. This is because using one vehicle to optimally visit two locations of interest will always require the same, if not lower, cost than using two vehicles since the routes are connected at the same depot. For the min-max objective, using more vehicles either improved or reached the same solution quality when compared to using less. This is shown in Figure 3.1 which shows the methods' solution histories for the min-max objective for both of the two

assumptions. When comparing the two histories, the best solutions found by all methods eventually end up with the same min-max quality by the time limit. Using the assumption that all vehicles need to depart improved the convergence time of the branch-and-bound approach (Gurobi). The convergence was slower for the SA approach in this particular instance; however, the difference is relatively small. Based on these and other preliminary results, we decided on the previously stated assumptions for these search methods. Note, the construction heuristics work with the assumption that not all vehicles need to depart with either objective.



(a) (b)

Figure 3.1. Solution histories generated from preliminary testing on an instance with 26 locations and 8 vehicles. The three search methods were tested with the min-max objective on the assumptions that all vehicles do not need to depart from the depot (a) and all vehicles need to depart from the depot (b).

3.2.2 **Min-Sum Formulation**

The binary variable $x_{ij} = 1$ if the edge from node i to node j is included in the solution, otherwise $x_{ij} = 0$. Let u_i denote the position of node i in a salesman's tour, and let p denote the maximum number of nodes that can be visited by any salesman. Following [1] and [34], the min-sum mTSP can be formulated as the following mixed-integer programming problem:

$$\min_{x} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.1}$$

$$\text{s. t.:} \sum_{j=2}^{n} x_{1j} \leq m \tag{3.2}$$

$$\sum_{i=2}^{n} x_{i1} - \sum_{j=2}^{n} x_{1j} = 0 \tag{3.3}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 2, \ldots, n \tag{3.4}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 2, \ldots, n \tag{3.5}$$

$$x_{ij} + x_{ji} \leq 1 \qquad i, j = 2, \ldots, n \tag{3.6}$$

$$u_i - u_j + p x_{ij} \leq p - 1 \qquad i, j = 2, \ldots, n \tag{3.7}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, n \tag{3.8}$$

$$u_i \in \{1, 2, \ldots, p - 1\} \qquad i = 2, \ldots, n \tag{3.9}$$

The min-sum objective (or cost) function, as given by Equation (3.1), represents the total distance traveled by all salesmen. Equations (3.2) - (3.7) represent the constraints. Equations (3.2) and (3.3) ensure that at most *m* salesmen exit and enter the depot node. Equations (3.4) and (3.5) ensure that every other node (which represents one city) is visited exactly once since it constrains the tours such that only one edge enters and exits each intermediate node. Equation (3.6) ensures that no symmetry occurs between edges. This means if an edge from node *i* to node *j* is used, then the edge from node *j* to node *i* cannot be used. Equation (3.7), which is known as the Miller-Tucker-Zemlin (MTZ) constraint [35], eliminates the possibility of sub-tours (cycles not connected to the depot). Equations (3.8) and (3.9) specify the

domain of variables x_{ij} and u_i. The optimizer will find values for the variables x_{ij} and u_i that minimize the total distance traveled by all salesmen.

3.2.3 **Min-Max Formulation**

The min-max mTSP can be formulated in a similar manner with a modification to the objective function and the addition of certain decision variables and constraints. Let c_j denote the cost of node j which is the cost of a salesman's tour up to node j. In other words, if node i is the node before node j in the tour, then the cost of node j, c_j, is equal to the cost of node i, c_i, plus the cost of the edge from node i to node j, c_{ij}. Let c_{max} equal the maximum cost of any node. Let M equal twice the maximum edge cost plus one. The parameter M is used in a Big M constraint to provide a sufficient lower bound on the difference between node costs in cases when x_{ij} is 0. The min-max mTSP can then be formulated as follows:

$$\min_{x} \; c_{max} \tag{3.10}$$

$$\text{s. t.} \quad \sum_{j=2}^{n} x_{1j} = m \tag{3.11}$$

$$\sum_{i=2}^{n} x_{i1} = m \tag{3.12}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 2, \ldots, n \tag{3.13}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 2, \ldots, n \tag{3.14}$$

$$x_{ij} + x_{ji} \leq 1 \qquad i, j = 2, \ldots, n \tag{3.15}$$

$$u_i - u_j + p x_{ij} \leq p - 1 \qquad i = 2, \ldots, n \quad j = 2, \ldots, n \tag{3.16}$$

$$c_i - c_j + Mx_{ij} \leq M - c_{ij} \quad i = 1, \ldots, n \quad j = 2, \ldots, n \tag{3.17}$$

$$c_i + c_{i1}x_{i1} \leq c_{max} \quad i = 1, \ldots, n \tag{3.18}$$

$$x_{ij} \in \{0, 1\} \quad i, j = 1, \ldots, n \tag{3.19}$$

$$u_i \in \{1, 2, \ldots, p - 1\} \quad i = 2, \ldots, n \tag{3.20}$$

$$c_i \geq 0 \quad i = 1, \ldots, n \tag{3.21}$$

The min-max objective function, as given by Eq. (3.10), represents the maximum distance travelled by any salesman. Equations (3.11) – (3.16) are equivalent to Equations (3.2) – (3.7) from the min-sum formulation but now it is enforced that all *m* salesmen are used in the solution. Equation (3.17) ensures that when an edge from node *i* to node *j* is used by a salesman, the cost of node *j* is constrained to be greater than or equal to the cost of node *i* plus the cost of the edge from node *i* to node *j*, c_{ij}. Equation (3.18) constrains c_{max} to be greater than or equal to the cost of any node plus the cost of the edge connecting the node back the depot (if used by a salesman). Equations (3.19) and (3.20) are equivalent to Equations (3.8) and (3.9) from the min-sum formulation. Equation (3.21) ensures all c_i are nonnegative. The optimizer will find values for the variables x_{ij}, u_i, c_i, and c_{max}.

3.3 *Solution Methods*

Both exact and heuristic solution methods were considered. An Integer Linear Programming (ILP) solver was used for finding exact solutions to the mTSP formulations presented in the previous section. The ILP solver utilizes a branch-and-bound algorithm to find an optimal solution. For the ILP solvers the maximum number of nodes a salesmen could visit, *p*, was set equal to the number of cities, *n*, in order to keep consistency in constraints among solution methods. Five heuristic

approaches to generate solutions to the mTSP were also considered. The heuristic approaches considered include a GA, SA approach, nearest neighbors algorithm, MST approximation, and Christofides algorithm. The GA and SA approaches are search algorithms that start with one or more random solutions to the mTSP, generate new solutions as they run, and save the best solution found. For these two approaches, constraints (3.2) and (3.3) are enforced for the min-sum objective and (3.11) and (3.12) for min-max. The nearest neighbors, MST, and Christofides approaches can be considered construction or one-shot heuristics that attempt to construct a single high quality solution. Constraints (3.2) and (3.3) are enforced when using the construction heuristics with either objective.

3.3.1 **Branch-and-bound**

The ILP solver used was Gurobi's Parallel Mixed Integer Programming solver [36]. The solver was accessed in MATLAB® and allows for the branch-and-bound search to be performed in parallel to utilize multiple cores on the computer. The solver was allowed to run until a specified time limit was reach.

3.3.2 **Genetic Algorithm**

The first heuristic algorithm was an open-source GA written in MATLAB® [37]. This GA initializes a population of 80 random solutions to the mTSP. The solutions in the population are evaluated, and ten of the best solutions are mutated to form a new population of children. There are 7 different mutations that are applied to each of these solutions: flip, swap, slide, breakpoint modification, flip and breakpoint modification, swap and breakpoint modification, and slide and breakpoint

modification. The route that is mutated by the GA is represented as a sequence of values, each representing a city. Breakpoints split the route into pieces, and each piece of the route is assigned to a salesman. The breakpoint modification mutation simply changes the breakpoints of the route thereby changing the cities assigned to each salesman. In a typical heuristic algorithm, this or a similar process is repeated for a user-specified amount of iterations. However, for the purposes of the comparison experiments in this chapter, the number of iterations for all algorithms was set to infinite and all were allowed to run until a set time limit was reached.

3.3.3 **Simulated Annealing**

An open-source SA algorithm written in MATLAB® for solving a VRP [38] was also considered. From an instance of the mTSP, a VRP instance was created by setting the vehicle capacity to infinite and giving all locations of interest the same arbitrary demand. In a typical capacitated VRP, each location has a demand that uses up vehicle capacity, and vehicles may not exceed their capacity constraint. By setting vehicle capacity to infinite, all vehicles can handle any amount of demand from locations thereby reducing this VRP into an mTSP. The SA algorithm starts with a random solution and sets the temperature T to an initial temperature of $T = 100$. A neighbor solution is created by performing one of three operations (swap, reversion, inversion) on the route randomly. The neighbor solution is compared to the current solution, and, if the neighbor is a better solution, then the algorithm accepts that as the new current solution; otherwise, it randomly accepts the neighbor as the new current solution. In each iteration, 80 neighbor solutions are generated and compared. After each iteration, the temperature is lowered by two percent: $T = 0.98 \times T$. As the

temperature decreases, the probability of accepting a lower quality neighbor solution as the new current solution decreases. As with the GA, the number of iterations was set to infinite and the algorithm returns the best solution found after a set time limit is reached.

3.3.4 **Nearest Neighbors**

The nearest neighbors algorithm used in this chapter is similar to that presented in [39] with slight modifications for the min-sum and min-max objectives. The approach works by iteratively appending nodes (locations) to the ends of vehicles' routes. In each iteration, the lowest cost unrouted node is appended to a vehicle's route. This procedure terminates when all nodes have been routed. For the min-sum objective, the cost of a node for a particular vehicle is evaluated as the distance between the node and the current end of the vehicle's route (not including the depot). For the min-max objective, the cost of a node for a particular vehicle is evaluated as the cost of the vehicle's route after appending the node to the end of the route which includes returning to the depot.

3.3.5 **Minimum Spanning Tree Approximation**

The MST approximation method considered is adapted from [40]. This procedure works with a graphical representation of the problem as presented in the previous section. However, for this approach, the depot node is cloned for each vehicle in the graph. Edges in the graph still have a cost of distance between nodes with the exception of edges between copies of the depot. These edges have "0" cost which ensures they appear in the MST of the graph. Edge costs were left the same for

both min-sum and min-max objectives. Once the graph is generated, the MST is computed using Prim's algorithm [41]. In this MST, there exists m - 1 edges connecting the depot copies together. These edges are removed to form a forest of trees, one for each vehicle. The approach then uses a preorder traversal on each tree and appends the original depot to that traversal to generate a route for each vehicle. This method gives a route at worst two times the cost of the optimal solution for the min-sum objective.

3.3.6 **Christofides Algorithm**

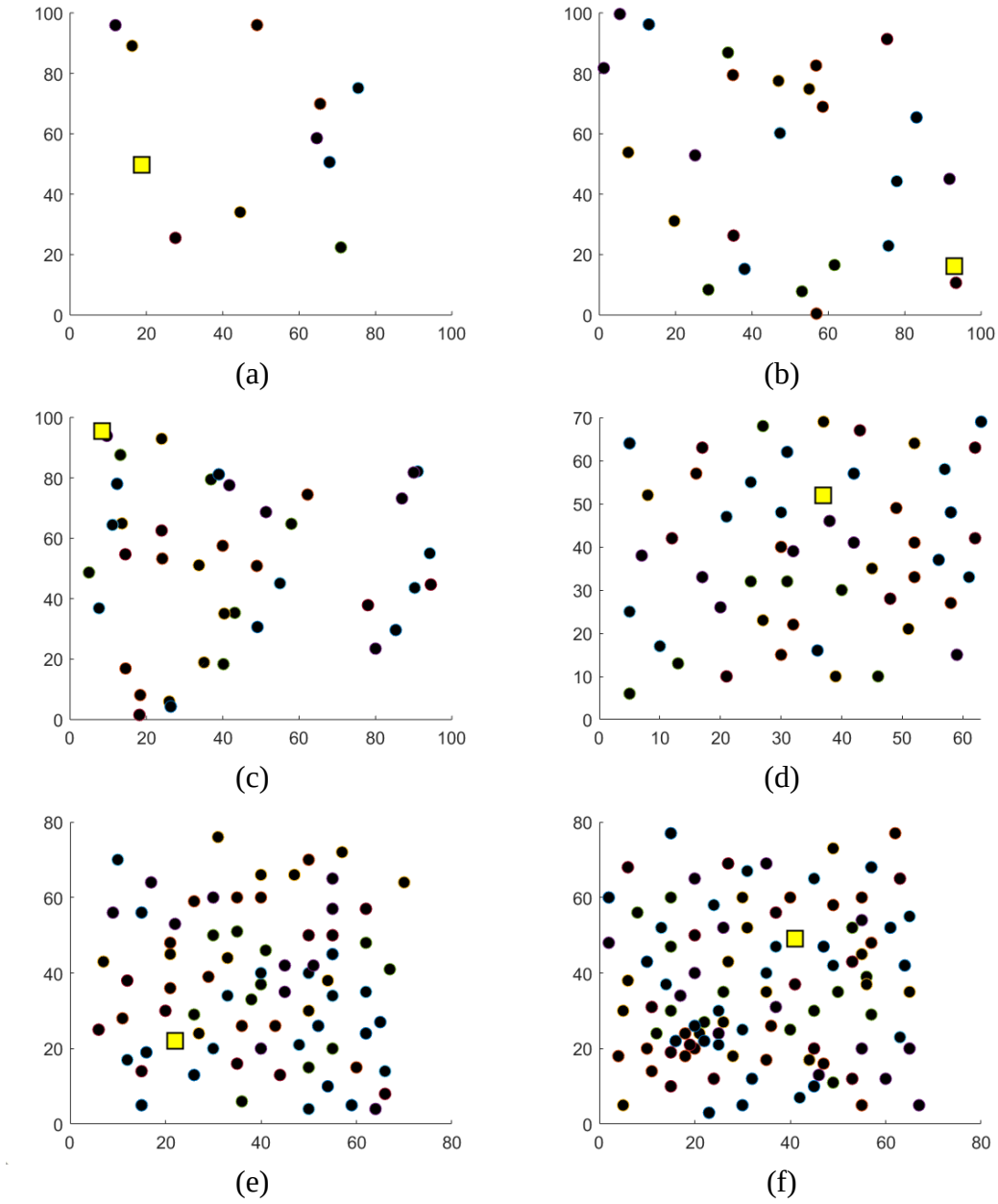
Christofides algorithm [33] is another construction heuristic similar to the MST approximation, but instead gives a route at worst 1.5 times the cost of the optimal min-sum solution. The forest of spanning trees is still generated as done in the MST approximation; however, the routes are no longer generated using a preorder traversal on each tree. A minimum-weight perfect matching and Eulerian circuit are used to generate the routes from each tree in the forest. The implementation of this was adapted from the TORSCHE Scheduling Toolbox for MATLAB [42].

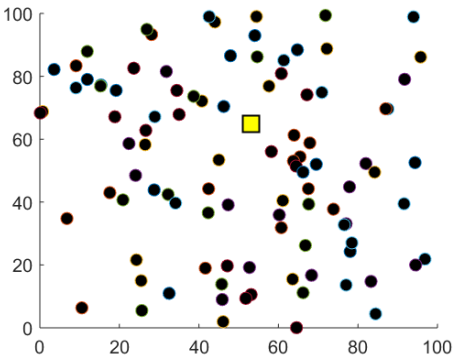
3.4 *Experimental Setup*

For testing the algorithms, four instances (or problems) of different sizes were randomly generated. The location of the depot and each city in each instance was selected from a uniform distribution over the two-dimensional region [0, 100] x [0, 100]. Additionally, three TSPLib [43] instances were tested: eil51, eil76, eil101. All algorithms were tested on the same seven problem sizes specified as follows: n = 11,

m = 5; n = 26, m = 8; n = 41, m = 12; n = 51, m = 7; n = 76, m = 12; n = 101, m = 17; n = 101, m = 20.

Figure 3.2 presents the problem instances used for testing. In each plot, the square indicates the depot, and the circles are the locations of interest. All algorithms were tested on an Intel® Xeon® CPU E3-1245 v5 @ 3.50 GHz 3.50 GHz with 16.00 GB of RAM.





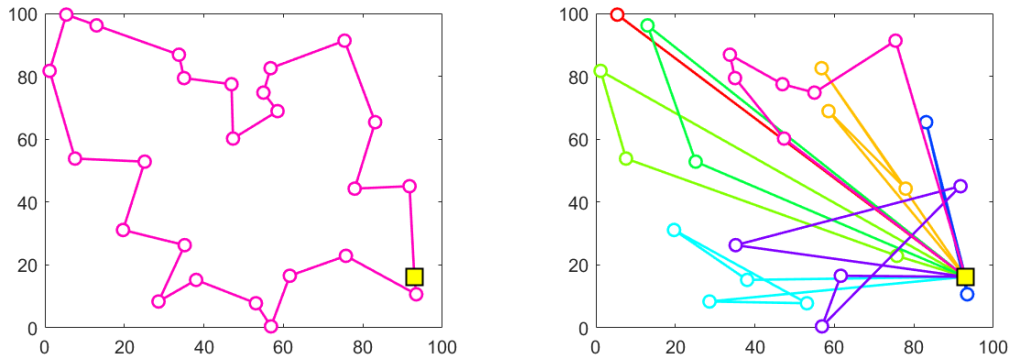
(g)

Figure 3.2. Plots of problem instances: (a) $n = 11$, $m = 5$; (b) $n = 26$, $m = 8$; (c) $n = 41$, $m = 12$; (d) $n = 51$, $m = 7$; (e) $n = 76$, $m = 12$; (f) $n = 101$, $m = 17$; (g) $n = 101$, $m = 20$. The yellow square is the depot and black circles are locations of interest.

3.5 *Results*

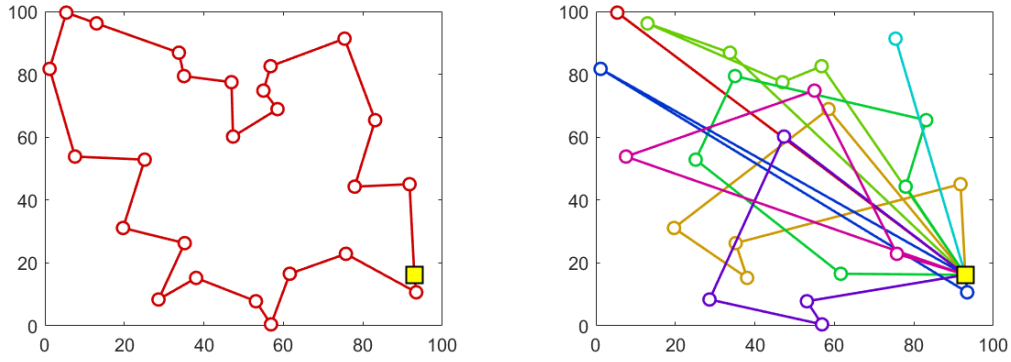
3.5.1 **Example Routes**

This section presents sample routes generated from each of the five methods on the medium instance with 26 locations of interest (including the depot) and 8 vehicles. The routes found by the GA, SA algorithm, MST approximation, Christofides algorithm, nearest neighbors approach, and the Gurobi approach are shown in Figures 3.3 to 3.8, respectively. Each method has a solution for the min-sum objective and a solution for the min-max objective other than the MST approximation and Christofides algorithm which are indifferent to the objective.



(a) (b)

Figure 3.3. (a) Min-sum and (b) min-max routes generated using the GA in the 26 x 8 instance.



(a) (b)

Figure 3.4. (a) Min-sum and (b) min-max routes generated using the SA approach in the 26 x 8 instance.

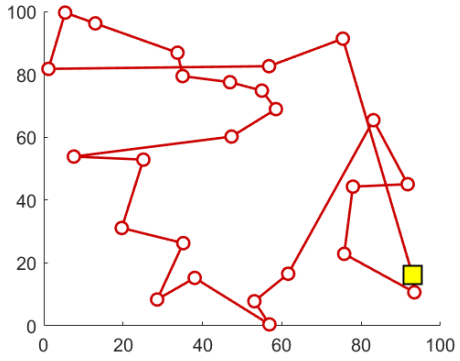


Figure 3.5. Routes generated using the MST approximation in the 26 x 8 instance. This solution is evaluated for both min-sum and min-max objectives.

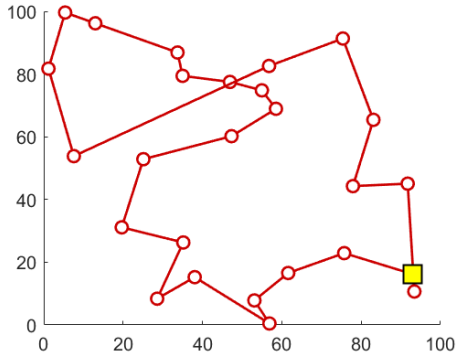
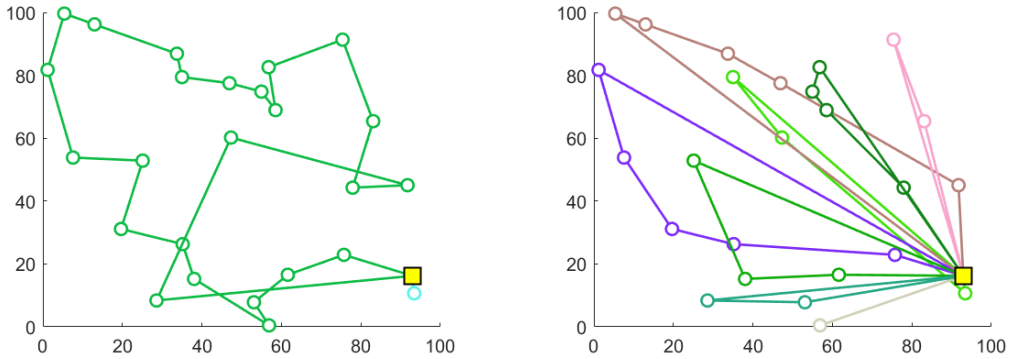
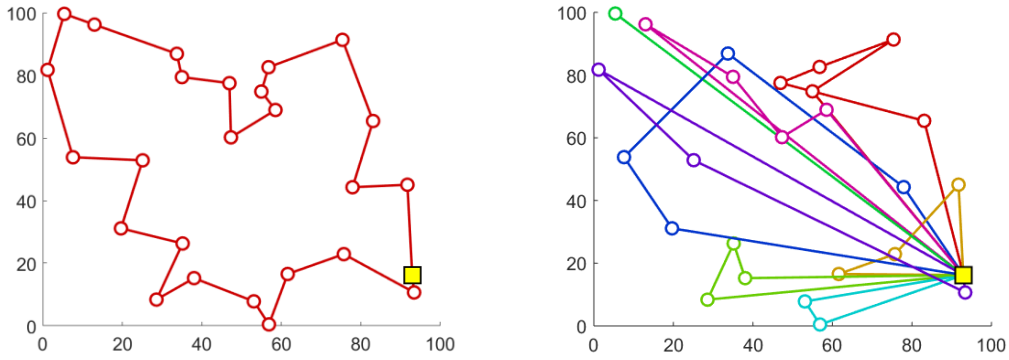


Figure 3.6. Routes generated using Christofides algorithm in the 26 x 8 instance. This solution is evaluated for both min-sum and min-max objectives.



(a) (b)

Figure 3.7. (a) Min-sum and (b) min-max routes generated using the nearest neighbors approach in the 26 x 8 instance.



(a) (b)

Figure 3.8. (a) Min-sum and (b) min-max routes generated using the Gurobi method in the 26 x 8 instance.

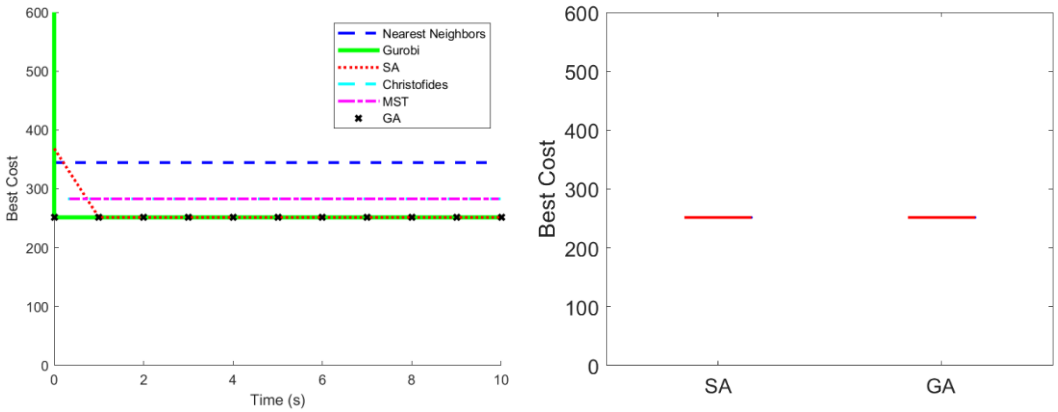
Comparing the two objectives, the min-sum routes tended to use only one vehicle to visit all locations which can be seen in all Figures 3.3 to 3.8. This behavior is typical for a single depot min-sum problem because all possible vehicle routes are connected at the depot. This means that one vehicle can be used to accomplish any multi-vehicle routes with at most the same total distance. Thus, using one vehicle is always as good if not better than using multiple for optimizing the min-sum objective in the particular route planning problem considered.

Across the methods considered, the min-sum routes are nearly identical in most cases. Although the min-max routes appear to be significantly different, they share similarities in the ways they balance loads and in the quality of the individual routes. For the most part, the individual vehicle routes avoid intraroute crossover meaning they can be considered nearly optimized for a TSP.

3.5.2 **Solution History Results**

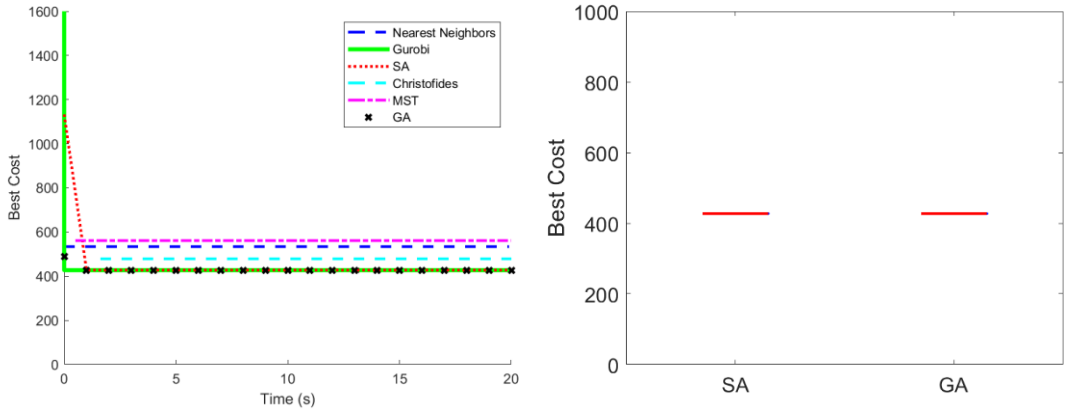
The computational time and solution quality were measured for each method in each instance. Computational time was recorded in terms of elapsed wall-clock time and was recorded from the time of function call to the time of algorithm termination. Solution quality was defined as the total distance traveled by all vehicles or the maximum distance traveled by any vehicle depending on the objective considered. The time limits for the problem instances were set as follows (n x m): 11 x 5: 10 seconds; 26 x 8: 20 seconds; 41 x 12: 35 seconds; 51 x 7: 40 seconds; 76 x 12: 45 seconds; 101 x 17: 50 seconds; 101 x 20: 50 seconds. These limits were set based on preliminary testing which showed how much time each method generally needed to either reach a high-quality solution or converge.

The solution quality of the best solution found by each method over time was recorded and overlaid onto single plots for each instance and each objective. Each heuristic algorithm was run 10 times for each problem instance. The average value at each time step of 1 second was used in the solution history plots which plot the cost of the current best solution versus time. Figures 3.9 to 3.22 illustrate these results for all instances and both objectives. The "Best Cost" refers to the cost of the best solution found at the particular time step. Box plots are included to illustrate the distribution of the average best solution found by each heuristic algorithm at the final time step for each instance.



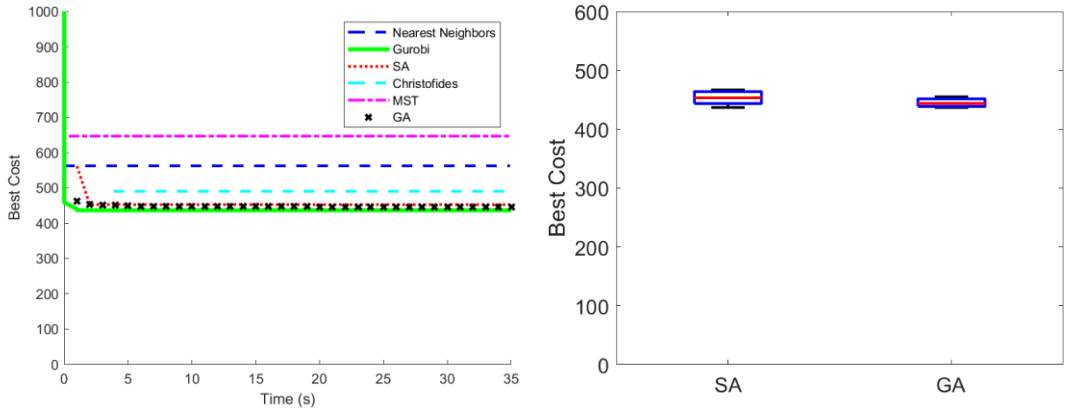
(a) (b)

Figure 3.9. (a) Solution history and (b) box plot for the 11 x 5 min-sum instance.



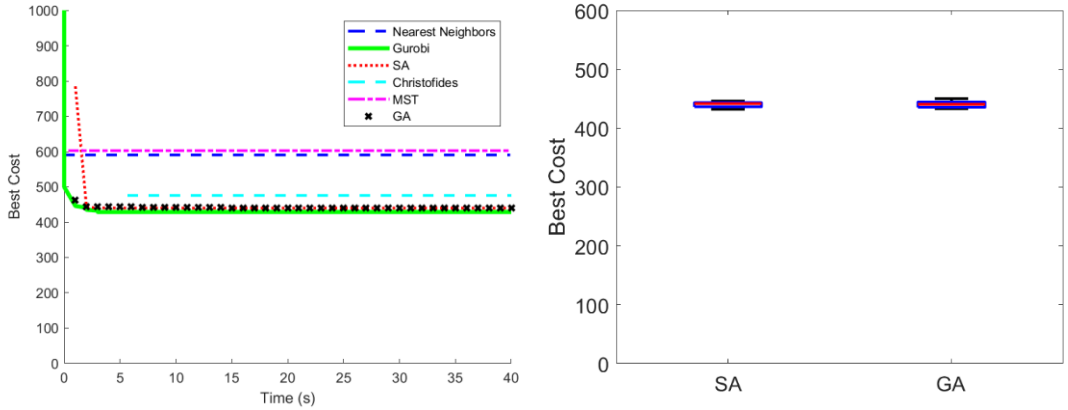
(a) (b)

Figure 3.10. (a) Solution history and (b) box plot for the 26 x 8 min-sum instance.



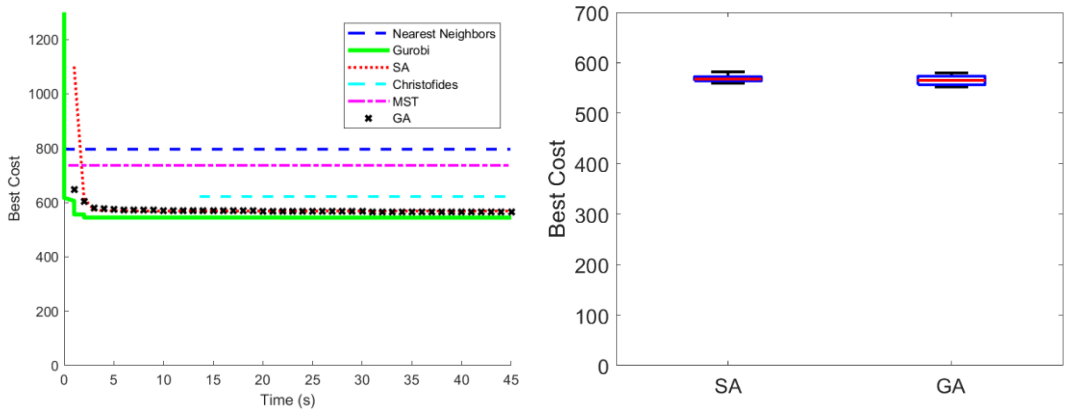
(a) (b)

Figure 3.11. (a) Solution history and (b) box plot for the 41 x 12 min-sum instance.



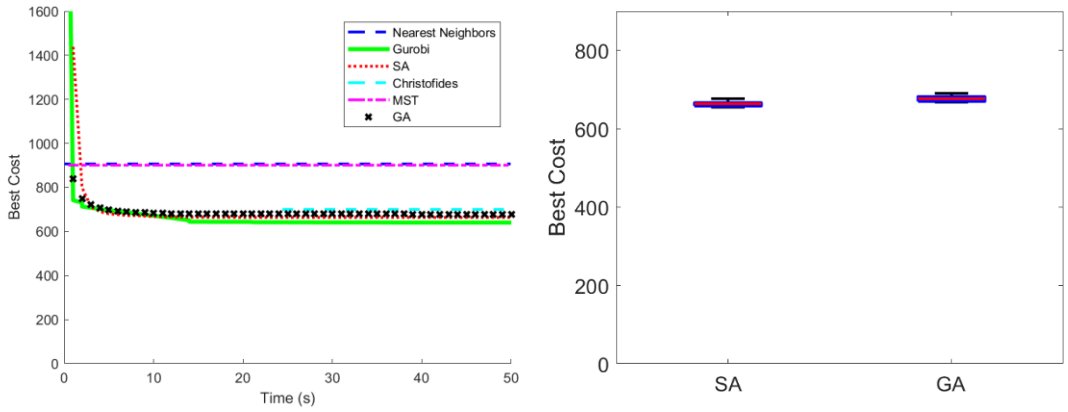
(a) (b)

Figure 3.12. (a) Solution history and (b) box plot for the 51 x 7 min-sum instance.



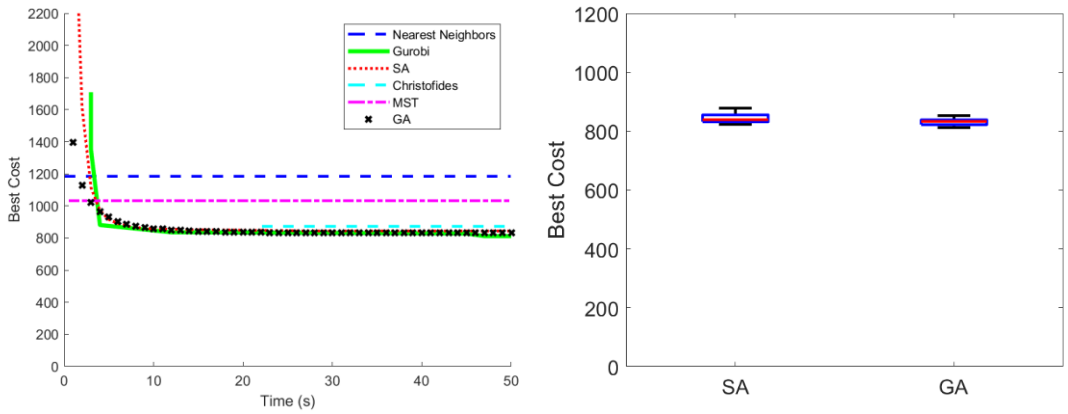
(a) (b)

Figure 3.13. (a) Solution history and (b) box plot for the 76 x 12 min-sum instance.



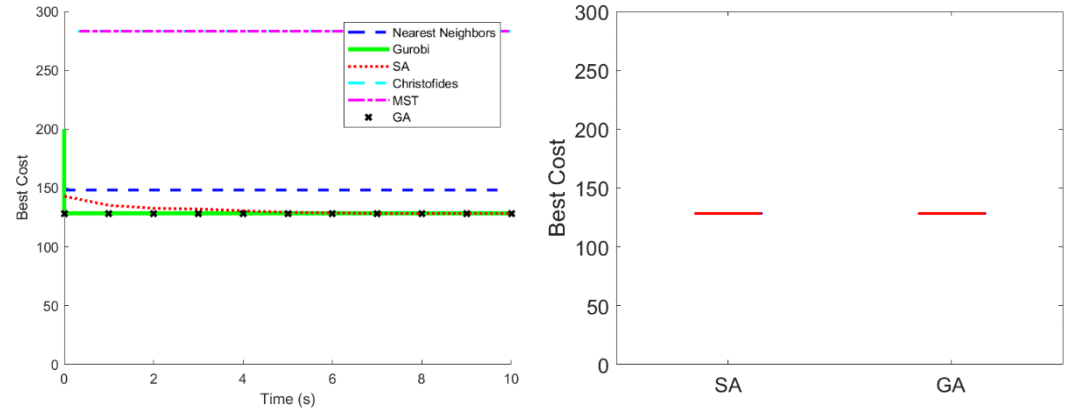
(a) (b)

Figure 3.14. (a) Solution history and (b) box plot for the 101 x 17 min-sum instance.



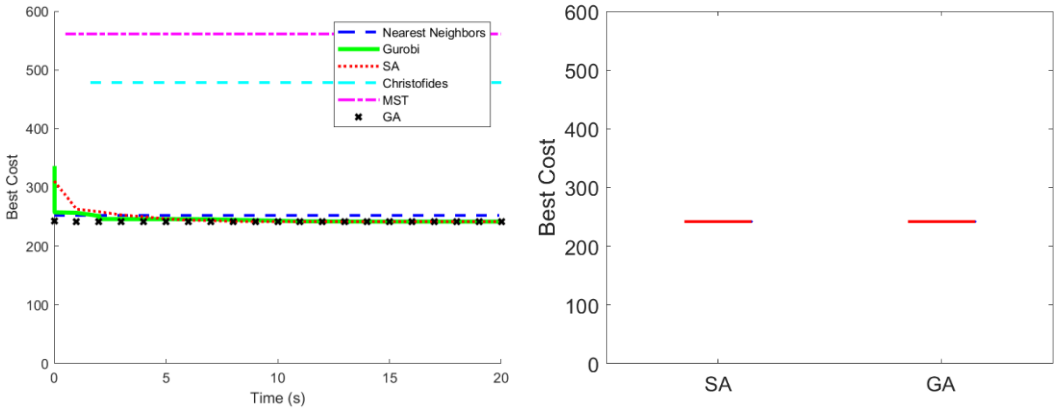
(a) (b)

Figure 3.15. (a) Solution history and (b) box plot for the 101 x 20 min-sum instance.



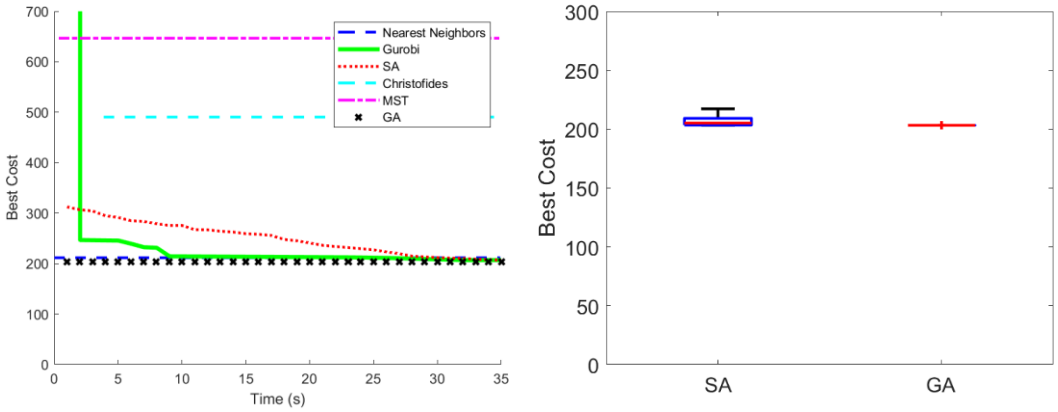
(a) (b)

Figure 3.16. (a) Solution history and (b) box plot for the 11 x 5 min-max instance.



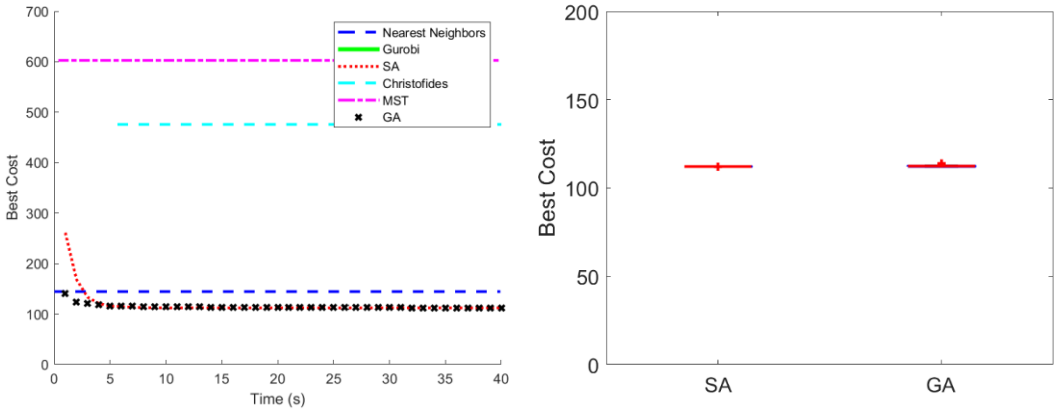
(a) (b)

Figure 3.17. (a) Solution history and (b) box plot for the 26 x 8 min-max instance.



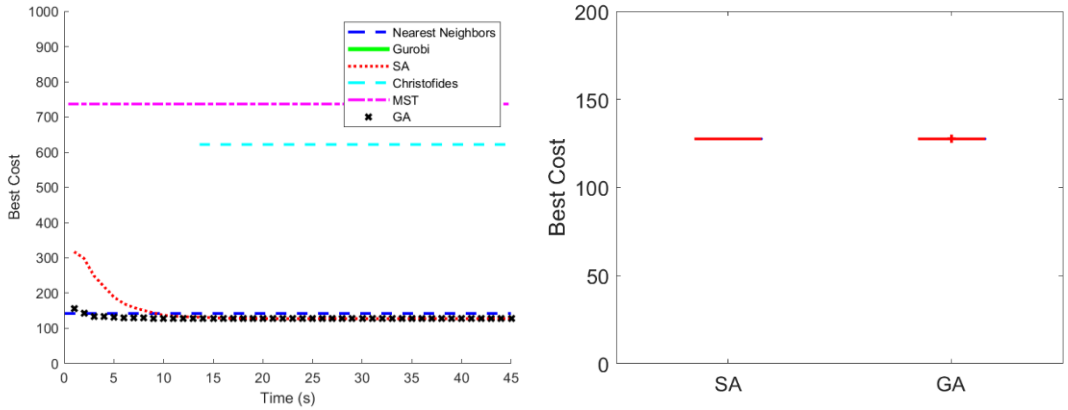
(a) (b)

Figure 3.18. (a) Solution history and (b) box plot for the 41 x 12 min-max instance.



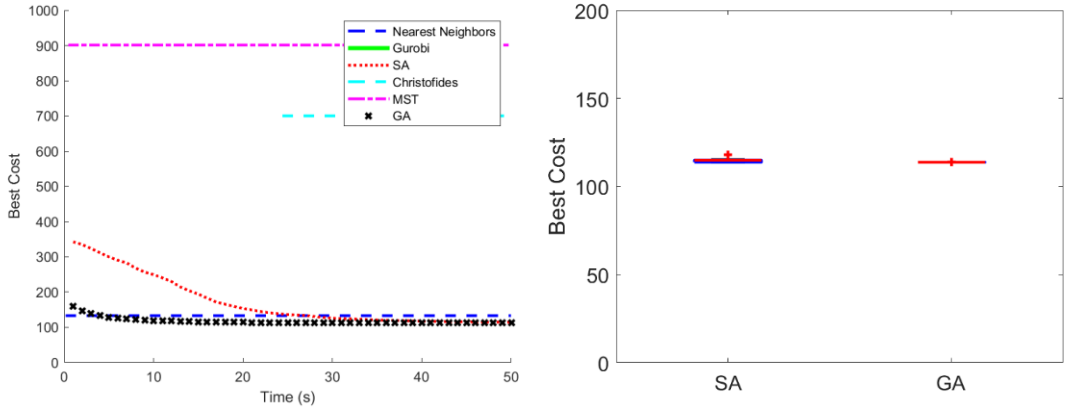
(a) (b)

Figure 3.19. (a) Solution history and (b) box plot for the 51 x 7 min-max instance. Gurobi solution history does not appear because it is not found within the time limit.



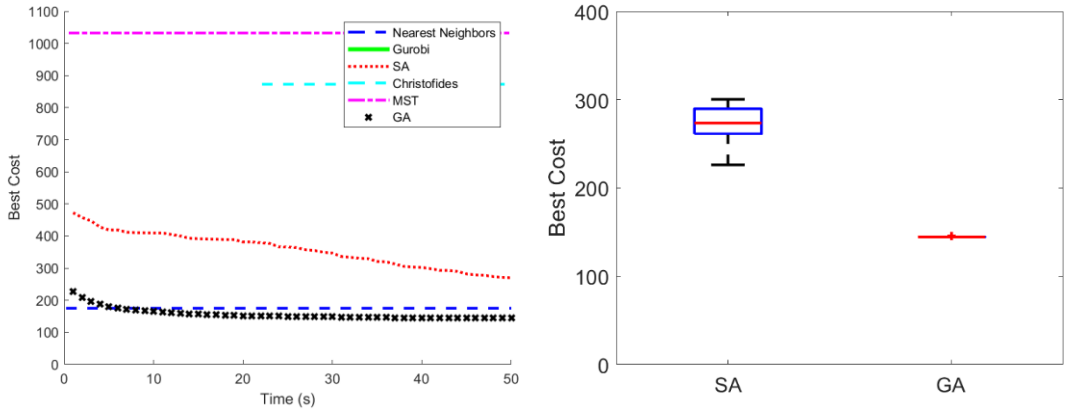
(a) (b)

Figure 3.20. (a) Solution history and (b) box plot for the 76 x 12 min-max instance. Gurobi solution history does not appear because it is not found within the time limit.



(a) (b)

Figure 3.21. (a) Solution history and (b) box plot for the 101 x 17 min-max instance. Gurobi solution history does not appear because it is not found within the time limit.



(a) (b)

Figure 3.22. (a) Solution history and (b) box plot for the 101 x 20 min-max instance. Gurobi solution history does not appear because it is not found within the time limit.

For the min-sum objective, the results indicate that the Gurobi approach generates the lowest cost solutions. Other than the largest instance, the min-sum algorithms were dominated by the Gurobi approach in all instances. In the largest instance (Figure 3.15(a)), the Gurobi approach took longer to find an initial solution and converge showing worse scalability when compared to the heuristic methods. In that instance, the GA found high-quality solutions faster than the Gurobi approach did. Of the heuristic algorithms, the GA and SA showed similar performance and were able to find high quality solutions quickly in all min-sum cases. However, in all min-sum instances, the GA found better solutions slightly faster than the SA. The constructions heuristics including the nearest neighbors algorithm, MST approximation, and Christofides algorithm always found lower-quality solutions across all min-sum instances. Of the three, the Christofides algorithm generated solutions with the highest quality. However, the Christofides algorithm took significantly longer to generate solutions as the problem size increased. This can partially be attributed to the implementation of the approach which was not optimized for use in MATLAB. These construction heuristics performed poorly because they only construct one solution to the problem. This sacrifices quality for speed which is apparent in these results with the exception of Christofides algorithm. The min-sum SA and GA showed similar amounts of variation over the 10 trials. This variation increased with problem size as seen in the box plots.

For the min-max objective, the results show the GA produced the lowest cost solutions. Unlike with the min-sum objective, the heuristic algorithms generally outperformed the Gurobi approach in the larger instances with the min-max objective.

Although the Gurobi approach was able to find high quality solution quickly in the smallest two instances, its performance quickly degraded and was unable to find any solutions in the largest four instances. This shows that the min-max ILP is significantly more difficult to solve than the min-sum. Along with added complexity from extra variables and constraints, there also exists an issue with many solutions having identical costs which slows the branch-and-bound procedure. This is because the objective only depends on the longest route, which means the other routes can change without changing the objective. Although this negatively impacts the Gurobi approach, the search-based heuristic approaches benefit from this since they are able to explore many solutions quickly. The GA was very quickly able to reach the highest quality solution of all on all instances. However, the SA was sometimes slow to converge as seen in Figures 3.18(a) and 3.22(a) unlike the behavior seen with the min-sum objective. The MST approximation method had the worst performance in every instance with Christofides algorithm close behind. The results show these two construction heuristics are much better suited for a min-sum objective. The min-max nearest neighbors algorithm was comparable to the GA and in the largest instances, it was able to find high quality solutions faster than the GA. The GA and SA showed significantly less variation in final solution quality with the min-max objective as seen in the box plots. While the GA showed almost no variation over the instances, the SA only had significant variation in the 41 x 12 and 101 x 20 instances as seen in Figures 3.18(b) and 3.22(b).

3.6 *Summary*

This chapter presented existing methods for solving a classical vehicle route planning problem were presented and compared. Methods tested included a GA, SA approach, Gurobi ILP approach, nearest neighbors algorithm, MST approximation, and Christofides algorithm. The results suggest that an exact ILP solver, particularly Gurobi which supports parallelism, performs well on this problem when the objective is minimizing the total distance traveled by all vehicles. With solvers such as Gurobi, optimal solutions can be found rapidly on smaller instances. In both smaller and larger instances, the Gurobi solver is able to find better quality solutions in shorter periods of time when compared to heuristic approaches. However, as the problem size increases beyond what was considered in this chapter, it is likely the heuristic approaches will eventually outperform the Gurobi approach on the min-sum objective. When the objective is to minimize the maximum distance traveled by any salesman, the heuristic algorithms, particularly the GA, are preferable as they are able to find high-quality solutions quickly, even on large problem sizes, for the problem instances considered. It was also seen that the nearest neighbors algorithm is able to construct high-quality min-max solutions rapidly, even for large problem instances.

The next chapter presents a study of a robust route planning problem that considered the possibility of vehicle failure. This problem is an extension to the classical route planning problem presented in this chapter. The GA considered in this chapter is adapted in the next chapter to generate robust routes.

Chapter 4: Robust Multi-Vehicle Route Planning Considering Vehicle Failure

(The material in this chapter originally appeared in [44].)

4.1 *Introduction*

Multi-vehicle systems are commonly seen in both commercial and military applications including drone delivery, search and rescue, fighting wildfires, surveillance, and hazard clearance. Planning routes for such systems is critical to accomplishing such tasks in the best way possible. A deterministic route planning problem was considered in the previous chapter. However, in all of these applications there may exist uncertainty in the environment, hazards, and vehicles (e.g., sensor or reliability uncertainty). Planning under uncertainty has been considered in the past assuming knowledge of probability distributions for uncertain parameters [14-16]. However, such distributions may not be known in practice. In this case, planning methods need to produce plans that are robust against these sources of uncertainty with guarantees on worst-case performance. Here, robustness is defined as the performance under worst-case realizations of uncertainty, which is especially relevant when there is no probability distribution for the uncertainty.

In this chapter, we discuss an approach for solving a multi-vehicle failure-robust route planning problem where the goal is to plan the most robust route for vehicles to visit a given set of locations of interest. Here, robustness is considered with respect to uncertainty in vehicle failure: a vehicle could fail at any location that it visits. This planning problem is motivated by hazardous applications and scenarios

in which an adversary or the environment may cause a vehicle to fail. Vehicles such as Unmanned Aerial Vehicles (UAVs) are more at risk to a catastrophic, mission-ending accidents due to the potential of a crash if something onboard fails, winds become dangerous, or some other interference. The objective is to minimize the worst-case cost, where the cost includes the cost of the initial plan up to the failure and the cost of the recovery plan generated to complete the mission after the failure, i.e., the locations that the failed vehicle did not visit must be visited by the surviving vehicles. Figure 4.1 shows an example of this where the unvisited locations along the failed vehicle's route would not be covered by the initial plan. Two objectives are considered and optimized: min-sum, where the total distance traveled by all vehicles is minimized, and min-max, the maximum distance traveled by any vehicle is minimized. The min-sum objective is useful when optimizing fuel consumption, while the min-max objective corresponds to minimizing total mission time.

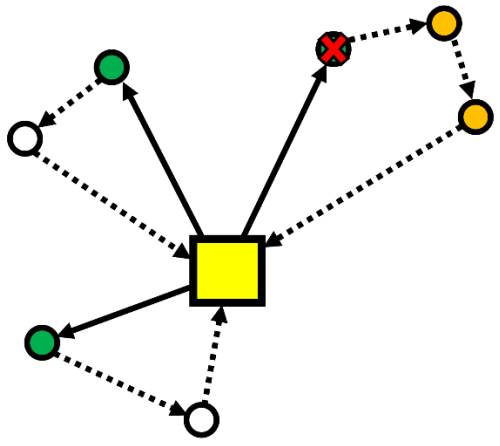


Figure 4.1. Example set of routes for 3 vehicles with a failure scenario. The square is the depot and circles are locations of interest. Traveled segments are denoted with solid lines and visited locations are filled green. Vehicle failure is shown by the red "x" over one location. The orange locations are the ones that the failed vehicle did not visit.

The rest of this chapter is organized as follows: Section 4.2 defines the multi-vehicle failure-robust routing planning problem and describes the problem formulations for the min-sum and min-max objectives. Section 4.3 describes the proposed Genetic Algorithm (GA) solution approach. Section 4.4 describes the experimental setup. Results are presented and discussed in Section 4.5. Finally, Section 4.6 summarizes and concludes this chapter.

4.2 *Problem Formulation*

The multi-vehicle failure-robust route planning problem can be defined as follows: a team of *m* vehicles is initially stored at a depot but is tasked with visiting *n*-1 specified locations in a region of interest (*n* equals the total number of locations, including the depot). The goal is to plan robust optimal routes for these vehicles such that every location is visited once and surviving vehicles end their routes at the depot. The objective function is the worst-case cost, where the cost includes the cost of the initial plan up to the failure and the cost of the recovery plan generated to complete the mission after the failure occurs. Maximizing robustness requires minimizing the worst-case cost. Two cost functions were considered: the total distance traveled by all vehicles (the min-sum objective) and the maximum distance traveled by any vehicle (the min-max objective). The recovery plan is created at the time of failure; this replanning step involves determining routes for the remaining vehicles so that they visit the unvisited locations and return to the depot.

The problem formulation is based on the following assumptions: A vehicle can fail only at locations visited upon arrival. The location at which a vehicle fails is considered visited and does not need to be visited by any other vehicle. At most one

vehicle can fail. Once one vehicle fails, the remaining vehicles will not fail. At each location, each vehicle spends a fixed amount of time to perform the required task (such as search) and a variable amount of idle time. Vehicle dynamics (such as acceleration and deceleration) are ignored. Vehicles travel at a constant speed (v) from location to location.

This problem can be represented as a two-stage robust optimization problem where the first stage determines the initial plan and the second stage is for replanning or recourse after failure uncertainty is realized. (In practice, the failure-robust problem could be solved again to mitigate the impact of another failure, if it could be solved quickly enough.)

The first stage can be thought of as an mTSP where the vehicles are the salesmen and the depot and the locations to visit are cities. The problem can be modeled using a graph in which the nodes represent the locations of interest and the depot and the edges represent the paths between these nodes. The second stage is an MDmTSP where the surviving vehicles visit the locations that have yet to be visited after a vehicle failure.

4.2.1 **Classical Min-Sum**

The following min-sum formulation was presented in the previous chapter and is restated for clarity and convenience. For the min-sum mTSP formulation, let the binary variable $x_{ij} = 1$, if the edge from node i to node j is included in the solution, otherwise $x_{ij} = 0$. Let u_i denote the position of node i in a salesman's tour, and let p denote the maximum number of nodes that can be visited by any salesman. Let c_{ij} be the cost (distance) of the edge from node i to node j. The first stage solution for the

min-sum formulation is the initial plan $\boldsymbol{x_1}$ which is comprised of x_{ij} and u_i for all i and j. It is assumed that not all salesmen need to be used in a min-sum solution. Based on [1] and [34], the classical integer programming formulation for the min-sum mTSP is the following:

$$\min_{\boldsymbol{x_1}} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{4.1}$$

$$\text{s. t.:} \sum_{j=2}^{n} x_{1j} \leq m \tag{4.2}$$

$$\sum_{i=2}^{n} x_{i1} - \sum_{j=2}^{n} x_{1j} = 0 \tag{4.3}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 2, \ldots, n \tag{4.4}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 2, \ldots, n \tag{4.5}$$

$$x_{ij} + x_{ji} \leq 1 \qquad i, j = 2, \ldots, n \tag{4.6}$$

$$u_i - u_j + p x_{ij} \leq p - 1 \qquad i, j = 2, \ldots, n \tag{4.7}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, n \tag{4.8}$$

$$u_i \in \{1, 2, \ldots, p-1\} \qquad i = 2, \ldots, n \tag{4.9}$$

The min-sum objective (or cost) function, as given by (4.1), represents the total distance traveled by all salesmen. Equations (4.2) - (4.7) represent the constraints. Equations (4.2) and (4.3) ensure that all m salesmen exit and enter the depot. Equations (4.4) and (4.5) ensure that every other node is visited exactly once since it constrains the tours such that only one edge enters and exits each intermediate node. Equation (4.6) ensures that no symmetry occurs between edges. This means if an edge from node i to node j is used, then the edge from node j to node i cannot be used. Equation (4.7), which is known as the Miller-Tucker-Zemlin (MTZ) constraint

[35], eliminates the possibility of sub-tours (cycles not connected to the depot). Equations (4.8) and (4.9) specify the domain of x_{ij} and u_i.

4.2.2 **Classical Min-Max**

The following min-max formulation differs from the one presented in Chapter 3 as it is better suited to explain the min-max robust formulation which is described in the next section. For the min-max formulation, let the binary variable $x_{ijk} = 1$, if the edge from node i to node j is included in the solution of salesman k, otherwise $x_{ijk} = 0$. Let u_i denote the position of node i in a salesman's tour, and let p denote the maximum number of nodes that can be visited by any salesman. Edge costs are denoted by c_{ij} which is the cost (distance) of the edge from node i to node j. For the min-max formulation the initial plan $\boldsymbol{x_1}$ is comprised of x_{ijk} and u_i for all i, j, and k. The classical three-index integer programming formulation for the min-max mTSP [45] is the following:

$$\min_{\boldsymbol{x_1}} \max_{k} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ijk} \right\} \tag{4.10}$$

$$\text{s. t.:} \sum_{j=2}^{n} x_{1jk} = 1 \qquad k = 1, \ldots, m \tag{4.11}$$

$$\sum_{i=2}^{n} x_{i1k} = 1 \qquad k = 1, \ldots, m \tag{4.12}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \qquad j = 2, \ldots, n \tag{4.13}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \qquad i = 2, \ldots, n \tag{4.14}$$

$$\sum_{i=1}^{n} x_{ijk} - \sum_{i=1}^{n} x_{jik} = 0 \quad j = 2, \ldots, n \quad k = 1, \ldots, m \tag{4.15}$$

$$\sum_{k=1}^{m} x_{ijk} + \sum_{k=1}^{m} x_{jik} \leq 1 \qquad i, j = 2, \ldots, n \tag{4.16}$$

$$u_i - u_j + p \sum_{k=1}^{m} x_{ijk} \leq p - 1 \qquad i, j = 2, \ldots, n \tag{4.17}$$

$$x_{ijk} \in \{0, 1\} \qquad i, j = 1, \ldots, n \quad k = 1, \ldots, m \tag{4.18}$$

$$u_i \in \{1, 2, \ldots, p - 1\} \quad i = 2, \ldots, n \tag{4.19}$$

Equation (4.10) represents the min-max objective function of maximum distance traveled by any salesmen. Equations (4.11) - (4.14) represent the same constraints as (4.2) - (4.5) but now all salesmen need to be used in a solution. Equation (4.15) ensures if salesman *k* enters node *j* then salesman *k* must exit node *j*. Equations (4.16) - (4.19) represent the same constraints as (4.6) - (4.9).

4.2.3 **Robust Min-Sum and Min-Max**

The objective of the classical min-sum or min-max mTSP can be modified to add failure-robustness to the formulation. Given a set of initial routes for the *m* vehicles, let *s* be a failure scenario that denotes the failure of a specific vehicle at one of the locations that it will visit. Let *S* be the set of all possible scenarios, including the no-failure scenario. At the time of one vehicle's failure, each surviving vehicle is either at a node or along an edge. Since these positions are needed to solve the second stage replanning problem, a timing model is needed to provide the vehicle positions with respect to the failed vehicle's final position. The timing model utilizes the initial plan ($\boldsymbol{x_1}$) and the failure scenario *s* to determine the current positions of the remaining $m - 1$ vehicles and the set of nodes left to be visited. The vehicles positions and leftover nodes are used to formulate the second stage replanning problem, which is an MDmTSP in which each vehicle's position is a depot and the leftover nodes are the

cities. The classical integer programming formulation for the MDmTSP is very similar to that of the mTSP [46].

Using the definitions of each stage, for robust min-sum optimization the objective (4.1) becomes:

$$\min_{\boldsymbol{x_1}} \max_{s \in S} \{ f_1(\boldsymbol{x_1}, s) + f_2(\boldsymbol{x_1}, s) \} \tag{4.20}$$

where $f_1(\boldsymbol{x_1}, s)$ is the objective value of the first stage mTSP solution up to the point of the vehicle failure in scenario s, the uncertain scenario which specifies the vehicle that fails and its location when it fails. This function computes the total sum of the costs of edges traveled by the vehicles up to the point of failure. Let the variable t_i denote the time at which a vehicle leaves location i. The variable a_i is the amount of time a vehicle spends at location i which is comprised of a fixed search time spent at each location plus a variable amount of idle time. The variable t_F denotes the time at which the failure occurs at location h. $f_1(\boldsymbol{x_1}, s)$ can be defined as follows:

$$f_1(\boldsymbol{x_1}, s) = \sum_{i,j} c_{ij} x_{ij} + \sum_{l,c} v(t_F - t_l) x_{lc} \tag{4.21}$$

$$i, j : \; t_j - a_j < t_F \; ; \qquad l, c : t_F \geq t_l \text{ and } t_F \leq t_c - a_c$$

$$t_F = t_h - a_h \tag{4.22}$$

Equation (4.21) represents the total distance traveled by all vehicles up to the point of failure which can be split into two summations. The first summation is for edges completely traversed at the time of failure and the second summation is for edges that are partially traversed which occurs when vehicles are still along edges at the time of failure. Equation (4.22) sets the time passed as the time upon exiting failure location h minus the time spent at location h.

$f_2(\boldsymbol{x_1}, s)$ is the objective value of the second stage MDmTSP solution that computes the cost of the approximate MDmTSP solution after the vehicle failure in scenario s. This function computes the total sum of the costs of edges traveled by all vehicles after the point of failure. Let the variable y_{ijk} denote the approximate solution to the MDmTSP replanning problem which equals 1 if the edge from node i to node j is included in the solution of vehicle k, otherwise $y_{ijk} = 0$. *APPROX* is a function that takes in the initial plan $\boldsymbol{x_1}$ and the scenario s, and outputs the approximate solution to the constructed MDmTSP instance. $f_2(\boldsymbol{x_1}, s)$ can be defined as follows:

$$f_2(\boldsymbol{x_1}, s) = \sum_i \sum_j \sum_k c_{ij} y_{ijk} \tag{4.23}$$

$$\{y_{ijk}\} = APPROX(\boldsymbol{x_1}, s) \tag{4.24}$$

Equation (4.23) represents the total distance traveled by all remaining vehicles after the point of failure which is a summation over all edges used in the replanning solution. Equation (4.24) sets y_{ijk} as the replanning solution outputted from the approximation method.

For the robust min-max formulation, the objective in Equation (4.10) becomes:

$$\min_{\boldsymbol{x_1}} \max_{s \in S} \max_{d \in D} \{g_1(\boldsymbol{x_1}, s, d) + g_2(\boldsymbol{x_1}, s, d)\} \tag{4.25}$$

For vehicle d in the set of vehicles D, $g_1(\boldsymbol{x_1}, s, d)$ is the total distance that vehicle d travels from time 0 until the vehicle failure in scenario s at time t_F.

$$g_1(\boldsymbol{x_1}, s, d) = \sum_{i,j} c_{ij} x_{ijd} + \sum_{l,c} v(t_F - t_l) x_{lcd} \tag{4.26}$$

$$i, j : t_j - a_j < t_F ; \qquad l, c : t_F \geq t_l \text{ and } t_F \leq t_c - a_c$$

$$t_F = t_h - a_h \tag{4.27}$$

Equation (4.26) is the same as (4.21) except the summation is only over vehicle *d*'s edges. Equation (4.27) is the same as (4.22).

For vehicle *d*, $g_2(\boldsymbol{x_1}, s, d)$ is the total distance that vehicle *d* travels after the vehicle failure in scenario *s*.

$$g_2(\boldsymbol{x_1}, s, d) = \sum_i \sum_j c_{ij} y_{ijd} \tag{4.28}$$

$$\{y_{ijk}\} = APPROX(\boldsymbol{x_1}, s) \tag{4.29}$$

Equation (4.28) represents the distance traveled by vehicle *d* after the point of failure which is a summation over all edges used by vehicle *d* in the replanning solution. Equation (4.29) sets y_{ijk} as the replanning solution outputted from the approximation method.

4.3 *Solution Method*

To solve the multi-vehicle failure-robust route planning problem, we modified the GA [37] presented in Chapter 3 and used it to generate robust solutions to the min-sum and min-max variants of the mTSP. The GA initializes a population of randomly generated solutions to the mTSP, and each solution's cost is evaluated based on Equation (4.20) or (4.25) depending on the objective function considered. The initial population also includes the non-robust min-sum solution found by the Gurobi Optimizer [36] cut off after twenty seconds if need be. Each solution is represented as a sequence of locations, a sequence of breakpoints that split the sequence of locations into distinct routes, and a sequence of idle times, one for each location. After evaluating the solutions in the population, the GA mutates some of the

best solutions to form a new population of children. The GA uses eight mutation operators: flip, swap, slide, breakpoint modification, flip and breakpoint modification, swap and breakpoint modification, slide and breakpoint modification, and idle time modification. Flip reverses a subsequence of the sequence of locations. Swap takes two locations in the sequence and switches them. Slide shifts a subsequence of the sequence of locations. Breakpoint modification replaces the current set of breakpoints with a randomly generated new one. Idle time modification replaces the current set of idle times with a randomly generated new one.

The GA repeats the process of generating a new population for a specified number of iterations (or generations) and returns the lowest cost solution found. The cost evaluation procedure generates every possible failure scenario for a solution and evaluates the cost of the solution for every scenario. The pseudocode for this procedure is given by the function COST_EVAL in Algorithm 4.1 below. The method for generating the set of scenarios from a solution is defined as follows: for each route in the solution, if the route contains more than one location to visit, then every location in that route other than the last is included in the set of scenarios. The last location does not need to be considered because if the vehicle were to fail there then the initial plan is still feasible and optimal. For this method, the number of scenarios will always equal $n - m$ including the scenario where no vehicle fails, if the problem is constrained such that every vehicle visits at least one location. In the unconstrained case where not all vehicles need to be used in a solution, the number of scenarios is bounded below by $n - m$ and above by $n - 1$ including the scenario where no vehicle fails.

For each scenario generated, the COST_EVAL procedure creates an instance of the MDmTSP with a set of m-1 depots ($\boldsymbol{D}$) and set of locations ($\boldsymbol{q}$) that need to be visited (Algorithm 4.1, line 5). Each depot represents the current position of each surviving vehicle, and the cities are the locations that have yet to be visited at the time of failure. This procedure also computes $f_1(\boldsymbol{x_1}, s)$ or $g_1(\boldsymbol{x_1}, s, d)$ for each vehicle depending on the objective considered.

After creating the MDmTSP instance, the MST_APPROX procedure uses a Minimum Spanning Tree (MST) approximation method [40] to rapidly generate a new solution in which the remaining locations are visited and the vehicles return to the initial depot. This method was tested in the previous chapter and was shown to produce solutions quickly. The pseudocode for this procedure is given by MST_APPROX in Algorithm 4.2. This procedure creates a complete graph with zero-cost edges between every pair of depots. The MST that is generated includes m - 2 edges connecting the depots. The MST_APPROX procedure removes these edges to form a forest of trees, each rooted at a surviving vehicle. The procedure then uses a preorder traversal on each tree and appends the original depot to that sequence to generate a route for each vehicle. For the min-sum objective, the COST_EVAL procedure adds the cost of the new route to the initial cost and computes the total cost in this scenario. For the min-max objective, the procedure determines the cost for each vehicle, and Line 7 (Algorithm 4.1) uses the greatest value of $g_1(\boldsymbol{x_1}, s, d) + g_2(\boldsymbol{x_1}, s, d)$ as the cost of the solution for this scenario. It repeats this for every scenario to determine the worst-case cost.

Algorithm 4.1: Pseudocode for evaluating the cost of the initial plan.

```
1:  function COST_EVAL(x_1, ini_dpt)
2:        scenarios ← scenarios_gen(x_1)
3:        worst_cost ← 0
4:        for each scenario s in scenarios
5:              {D, q, ini_cst} ← mdmtsp_gen(x_1, s)
6:              routes ← MST_APPROX(D, q, ini_dpt)
7:          worst_cost ← max(worst_cost, ini_cst + cost(routes))
8:        end for
9:        return worst_cost
10: end function
```

Algorithm 4.2: Pseudocode for generating a replanning solution using the MST approximation.

```
1:  function MST_APPROX(D, q, ini_dpt)
2:        nodes ← D ∪ q
3:        edges ← {}
4:        for each node i in nodes
5:          for each node j ≠ i in nodes
6:                if is_depot(i) and is_depot(j)
7:                      cost ← 0
8:                else
9:                  cost ← dist(i, j)
10:       end if
11:       edges ← edges ∪ {i, j, cost}
12:     end for
13: end for
14: G ← graph(nodes, edges)
15: mst ← PRIMS(G)
16: forest ← remove_depot_edges(mst)
17: routes ← {}
18: for each tree t in forest
19:   route ← preorder_traversal(t)
20:   route ← route.append(ini_dpt)
21:   routes ← routes ∪ route
22: end for
23: return routes
24: end function
```

4.4 *Experimental Setup*

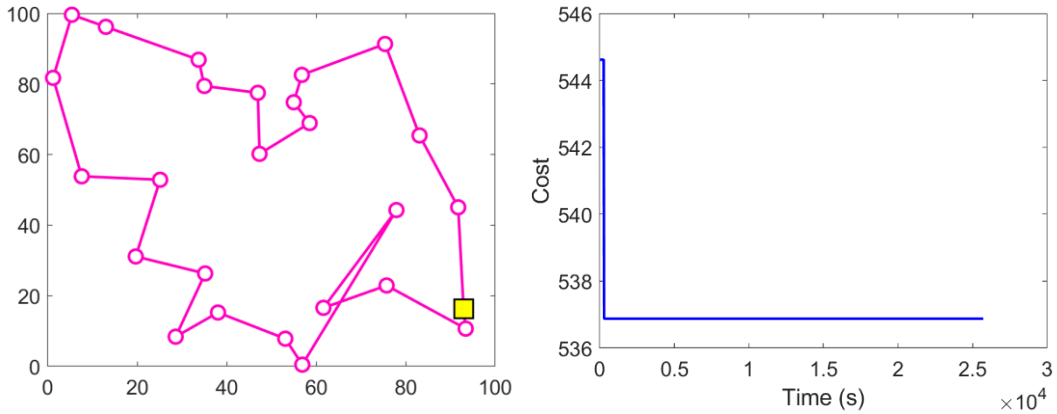
The approach described has been tested on the same instances used in the previous chapter which includes 11 x 5, 26 x 8, 41 x 12, and 101 x 20 (*n* x *m*)

randomly generated problem instances and three TSPLib [43] instances: eil51, eil76, eil101 with the number of vehicles being: m = 7, 12, and 17, respectively. The four randomly generated instances were created by selecting locations from a uniform distribution over a two-dimensional region [0, 100] x [0, 100]. The min-max objective was tested more extensively than min-sum because preliminary testing with the min-sum objective showed little to no improvement in robustness relative to the non-robust optimal solution in all instances. Evidence of this can be seen in Figure 4.2 which depicts the solution found in the 26 x 8 min-sum instance. The progression of best worst-case cost over time (Figure 4.2(b)) shows the insignificant improvement in worst-case cost of the "robust" solution over the initial non-robust optimal solution. Therefore, mainly min-max computational results are reported. The GA was run for 1000 iterations with a population of 90 individuals. The fixed task time was set to 10 seconds, and velocity was set to 2 units per second. The variable idle times were limited to the range from 0 to 10 seconds. The GA was run for five trials for each min-max case reported and three trials for the min-sum cases.

To evaluate the quality of the solutions that the GA found, we also constructed solutions using one-shot heuristics including a Time-Oriented Nearest Neighbors (TONN) construction heuristic based on [39] and the MST approximation described previously. We also generated solutions by solving the classical non-robust MILP using Gurobi (with a time limit of 1000 seconds).

We also ran the GA with a reduced scenario set to test the tradeoff between solution quality and computational effort. This reduced scenario set was comprised of the no-failure scenario and scenarios where the failed location is the first in a

vehicle's route. Preliminary testing indicated that those scenarios comprised the majority of worst-case scenarios for the min-max cases. The cardinality of the reduced scenario set was in the range from 2 to $m + 1$. Experiments were run in MATLAB® R2018a on an Intel® Xeon® CPU E3-1245 v5 @ 3.50 GHz with 16.00 GB of RAM. For each trial of the GA, the solution history was saved to analyze the convergence properties of the method for every case. The best solution history refers to the progression of the best worst-case cost over time.



(a) (b)

Figure 4.2. Sample (a) solution and (b) progression of best worst-case cost over time found by the GA (full) in the 26 x 8 min-sum instance.

4.5 *Results*

4.5.1 **Example Min-Max Routes**

Figure 4.3 shows an example of how replanning responds to an arbitrary failure for a less robust solution and a more robust solution for the 26 x 8 instance with the min-max objective function. The vehicle routes are shown with distinct colors. Figure 4.3(a) and 4.3(b) depict the failure scenario overlaid onto the two solutions; in each one, the vehicle failure is marked by the red "x". Figure 4.3(c) shows the recovery plan for the less robust solution for the less robust solution in

Figure 4.3(a); Figure 4.3(d) shows the recovery plan for the more robust solution in Figure 4.3(b). The locations of the surviving vehicles at the time of failure are indicated by black squares. In the recovery plan of the more robust solution (Figure 4.3(d)), the vehicles work together with balanced loads to cover the remaining locations that need to be visited after the failure. However, in the recovery plan of the less robust solution (Figure 4.3(c)), the purple vehicle has a higher load than the other vehicles which increases min-max cost.

Figure 4.4 shows another example of replanning but for the two solutions' worst-case scenarios. Figure 4.4(c) shows the recovery plan for the less robust solution for the less robust solution in Figure 4.4(a); Figure 4.4(d) shows the recovery plan for the more robust solution in Figure 4.4(b). In the worst-case scenario, a similar contrast is seen between the two recovery plans. While the loads between vehicles remain balanced in the more robust solution's recovery plan, the poor min-max behavior is compounded even further in the recovery plan of the less robust solution.

While the less robust and more robust solution have similar min-max costs without failure (241.8 and 250.7 units, respectively), the worst-case cost of the less robust solution is significantly higher than that of the more robust solution (563.5 and 260.9 units, respectively). This means robustness was achieved with little sacrifice to the nominal cost if no failure occurs. These results also show that the MST method of replanning can be inconsistent in generating quality min-max recovery routes as seen with the differences in Figures 4.3(c) and 4.4(c).

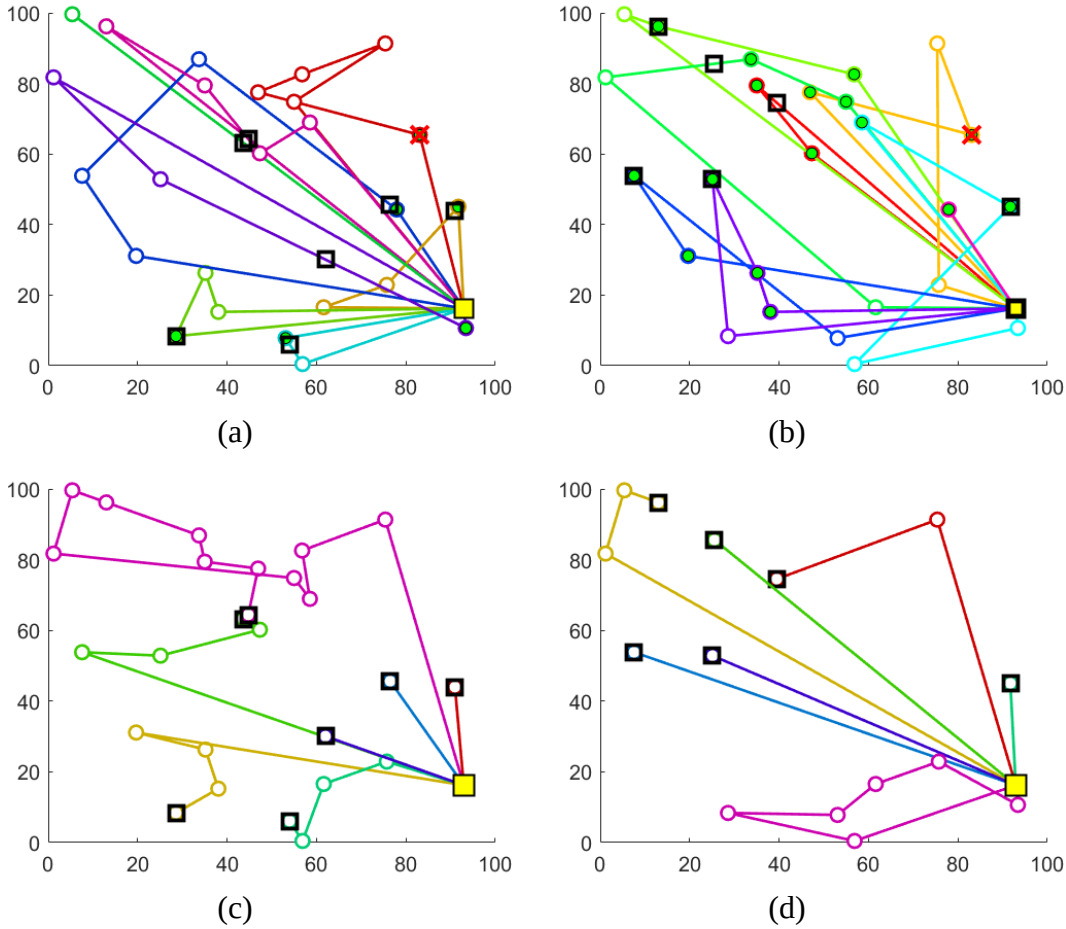


Figure 4.3. Comparison of original and recovery routes for an arbitrary failure scenario using the (a) Gurobi and (b) GA (full) solutions for the 26 x 8 min-max instance. The recovery plans for the Gurobi and GA solutions are shown in (c) and (d), respectively. The yellow square is the depot and circles are the locations of interest. Each color indicates the route of a different vehicle. Black squares indicate locations of surviving vehicles at the time of failure. For (a) and (b), green-filled circles are visited locations at the time of failure. The red "x" indicates the failure scenario.

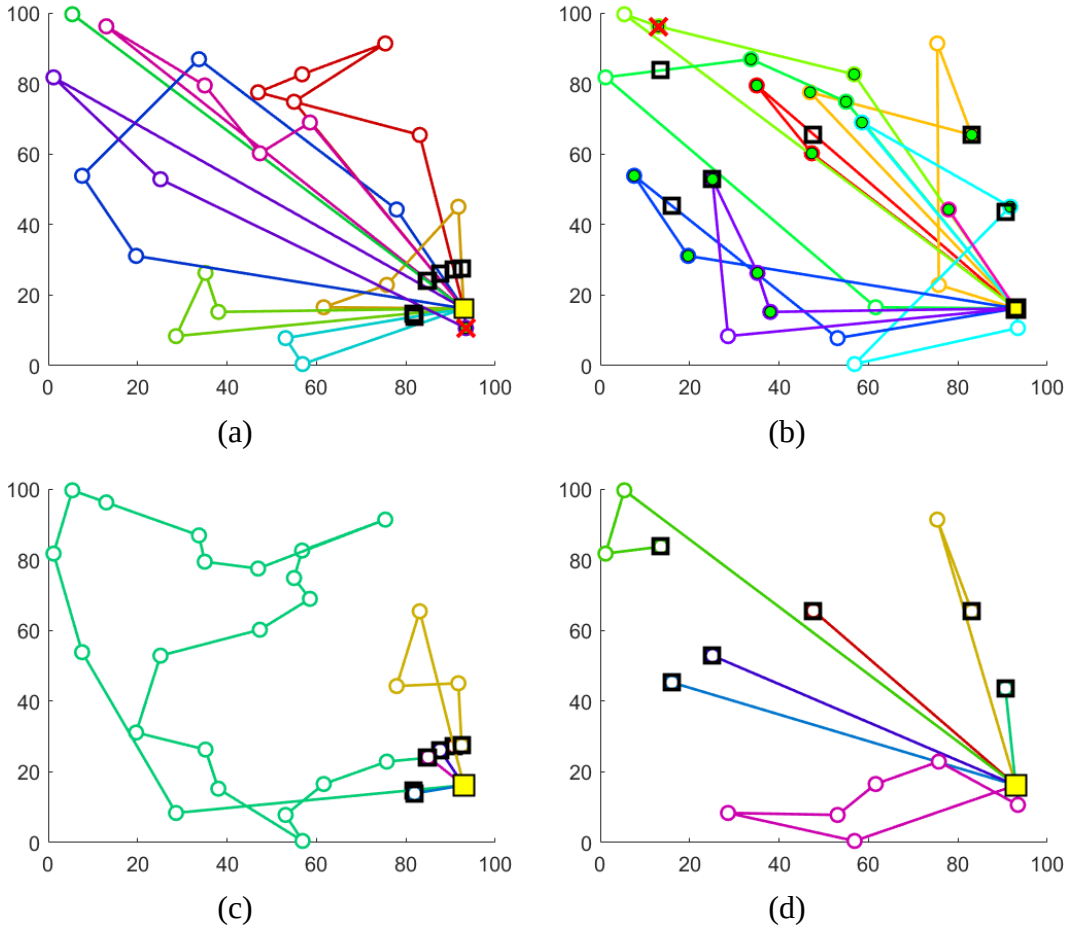


Figure 4.4. Comparison of original and recovery routes for worst-case failure scenarios using the (a) Gurobi and (b) GA (full) solutions for the 26 x 8 min-max instance. The recovery plans for the Gurobi and GA solutions are shown in (c) and (d), respectively. The yellow square is the depot and circles are the locations of interest. Each color indicates the route of a different vehicle. Black squares indicate locations of surviving vehicles at the time of failure. For (a) and (b), green-filled circles are visited locations at the time of failure. The red "X" indicates the failure scenario.

4.5.2 **Solution History Results**

Figure 4.5 shows an overlay of sample solution history plots, each describing results from five trials for the three TSPLib instances with the robust min-max objective GA. The figure shows that for smaller instances, high-quality solutions were found more quickly. Also, little to no variation was seen across trials as

indicated by the intervals which extend from the minimum to the maximum cost at that point in time over the five trials.

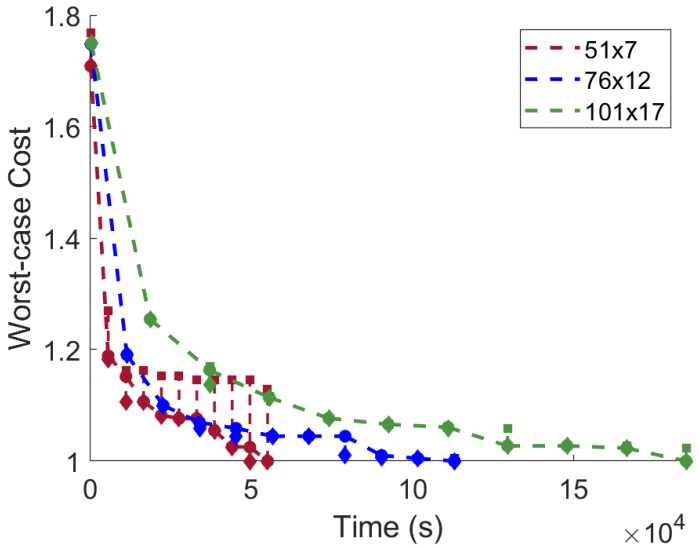


Figure 4.5. Min-max GA (full) solution histories for TSPLib instances. Worst-case cost is scaled by best cost found in each instance. Each marker shape (diamond, circle, and square) corresponds to minimum, median, and maximum costs over the five trials. The dashed line is the median with vertical intervals created using minimum and maximum of the trials at each point.

Figure 4.6 shows an overlay of sample solution history plots, each describing results from three trials for the three TSPLib instances with the robust min-sum objective GA. Unlike with the min-max objective, little improvement was seen in the best solution's robustness over time. There is also little variation over the trials with the min-sum objective.

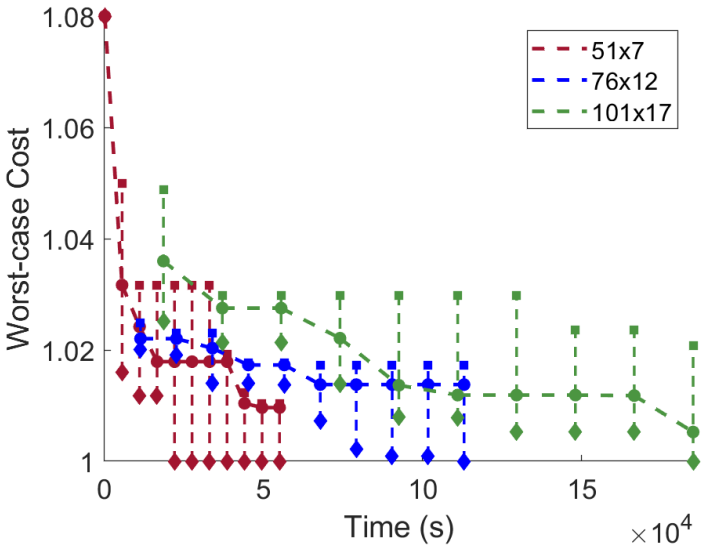


Figure 4.6. Min-sum GA (full) solution histories for TSPLib instances. Worst-case cost is scaled by best cost found in each instance. Each marker shape (diamond, circle, and square) corresponds to minimum, median, and maximum costs over the three trials. The dashed line is the median with vertical intervals created using minimum and maximum of the trials at each point.

4.5.3 **Min-Max Robustness**

Table 4.1 summarizes the min-max worst-case costs of the final solutions found by five different methods: (1) the GA with the full scenario set, (2) the GA with the reduced scenario set, (3) the TONN heuristic, (4) the MST approximation method, and (5) optimistic planning. The optimistic planning procedure uses the Gurobi optimization solver to find a solution to the classical problem (Equations 4.10-4.19) without considering the possibility of failure. Reported GA costs were averaged over five trials. Averaged over the seven instances, relative to the optimistic solutions, the solutions found by the GA with the full scenario set reduced the worst-case cost by 43%.

Table 4.1. Comparison of min-max worst-case costs for all methods. GA results are averaged over five trials. The boldface value is the best worst-case cost (in units) for each instance.

Instance	Optimistic	GA (Full)	GA (Red.)	TONN	MST
11 x 5	193.3	**138.7**	176.2	337.9	330.3
26 x 8	563.5	**260.9**	314.2	569.7	562.8
41 x 12	623.0	**257.7**	352.3	593.0	646.8
51 x 7	289.1	**231.1**	367.0	410.1	602.8
76 x 12	624.2	**249.7**	321.8	618.5	748.1
101 x 17	401.4	**238.3**	344.5	614.3	901.2
101 x 20	501.7	**291.1**	343.6	528.9	1036.4

The GA with the full scenario set was able to converge to solutions with lower worst-case cost on all instances when compared with the solutions found by other methods. For the min-max objective, among all instances tested, significant improvement was seen in the worst-case cost between the solutions generated by optimistic planning and the solutions generated by the GA, which considered the worst-case cost. It is likely that the poor worst-case costs of the optimistic solutions stem from the MST method of replanning which is poorly suited for the min-max objective. This can be seen in Figure 4.3(c) where the purple vehicle is assigned more locations than other remaining vehicles. To accurately determine the effects of the replanning method, similar studies utilizing different replanning methods are needed. However, it is also important to consider that replanning methods need to be sufficiently fast to run in real-time, so such inefficiencies may be unavoidable.

More variation was seen in the medium instances (51 x 7, 41 x 12) as seen in Figure 4.5. This may be attributed to the layout of the locations and depot in these instances. Additional trials are needed to fully understand the variation of the GA in different instances.

The high computational cost of the approach can be seen in the solution histories. As the problem size grows, the convergence rate tends to slow and the time to finish 1000 iterations increases significantly. The number of scenarios considered increases with the problem size. Each additional scenario requires constructing a replanning solution using the approximate MST method, which also increases in complexity with the problem size. This motivates using a reduced scenario set as described previously. As seen in Table 4.1, using the reduced scenario set yielded solutions with a worse robustness than those generated using the full scenario set, but these solutions were better than the solutions generated by the TONN, MST, and optimistic planning procedures (other than the 51 x 7 instance). Using the reduced scenario set required less computational effort than using the full scenario set but more computational effort than the construction heuristics (TONN and MST). For example, in one trial of the 101 x 17 min-max instance, the computation time of the GA using the full set of scenarios was on the order of 10^5 seconds while, with the reduced scenario set, it was 10^4 seconds. The time order of magnitude for the construction heuristics was 10^0 seconds. Thus, these results show that there is a tradeoff between the robustness of a solution and the computational effort needed to find it.

4.6 *Summary*

This chapter considered a new multi-vehicle route planning problem with a goal of planning robust routes for vehicles that can fail. The cost function in the problem is the cost of the initial plan up to the failure plus the cost of the recovery plan generated to complete the mission after the failure. In this way, the chapter

contributes to the problem of generating initial route plans that are robust against vehicle failure. Robustness provides guarantees on worst-case performance without the need of knowing probability distributions for uncertain parameters. This chapter formulated the route planning problem and presented a GA that can find solutions. We tested the approach on problem instances of different sizes and used the results to evaluate the method's strengths and weaknesses. The results showed the method can find high quality solutions, although it requires more computational effort than running construction heuristics or solving an integer programming problem to find a solution without considering uncertainty. This motivates the desire for more efficient methods with less sacrifice in solution quality. Using a reduced scenario set required less computational effort but yielded solutions that were not as robust.

The next chapter presents a study of a different extension to the multi-vehicle route planning problem in which the system of vehicles is decentralized. In such a case, the vehicles must determine their own routes individually with communication and collaboration occurring in real time. The chapter studies and compares the performance of different solution methods at different levels of communication quality.

Chapter 5: Decentralized Multi-Vehicle Route Planning in Communication-Lossy Environments

5.1 *Introduction*

In previous chapters, multi-vehicle route planning was considered for centralized systems. In such systems, a centralized planner with global knowledge decides the routes for all vehicles in the system. Although centralized planners generally achieve better coordination among vehicles, they present a significant risk when operating in hazardous and communication-limited environments. If the communication link between the centralized planner and the vehicles were to fail, then the entire system could fail to complete its mission. This motivates the use of a decentralized system in which each vehicle plans its own route, thereby removing the dependence of the vehicles on a single centralized planner. Vehicles in such systems can collaborate via communication with one another to deconflict or improve their routes. However, communication can be lossy between vehicles, so that messages sent by vehicles may not be received by intended vehicles.

This chapter presents a study of a decentralized multi-vehicle route planning problem which is also known as the problem of decentralized task allocation in the literature. In this problem, each vehicle decides its own routes while communicating and collaborating with other vehicles in the system. In contrast to the problems considered in previous chapters, this scenario requires online planning approaches meaning a route decided by a vehicle can change during the mission, possibly due to new information received from communication with other vehicles. In the considered

scenario, vehicles start at different locations (depots) and do not return to their depot once the mission is complete. The goal is for vehicles to visit a set of locations of interest known by all vehicles in advance. At the time that the vehicles make a decision, they consider a fixed set of locations and decide which vehicles should visit which locations and the routes that they should take. A vehicle may need to change its route if it learns that another vehicle has visited a location that it was planning to visit. The mission terminates when all vehicles are aware that all locations of interest have been visited. Two objectives are considered: minimizing the total distance traveled by all vehicles (min-sum) and minimizing the time needed to visit all locations of interest (min-time). In Chapters 3 and 4, a min-max objective was considered, and in those scenarios, min-max was equivalent to min-time because routes were fixed before execution. However, in this chapter, min-time is explicitly considered because the routes can change in real time which means there may exist periods of time where a vehicle does not move. Several existing approaches were considered including the Decentralized Hungarian approach [29], Consensus-based Auction Algorithm [24], Asynchronous Consensus-based Bundle Algorithm [25], Performance Impact Algorithm [26], and the Hybrid Information and Plan Consensus approach [27]. A new approach is also proposed that utilizes the genetic algorithm [37] described in previous chapters. These approaches are tested and compared on five problem instances of varying size and at three levels of communication quality. The communication quality refers to how lossy the communication between vehicles is, i.e., how often messages are dropped when the vehicles communicate.

The rest of this chapter is organized as follows: Section 5.2 describes the problem formulation for the decentralized multi-vehicle route planning problem. Section 5.3 discusses the solution methods considered. Section 5.4 details the experimental setup including the problem instances and communication model. Section 5.5 presents and discusses the results. Section 5.6 summarizes and concludes this chapter.

5.2 *Problem Formulation*

The decentralized multi-vehicle route planning problem considered is defined as follows: Given a set of n fixed locations of interest, visit all locations using m vehicles. The objective is to optimize either the min-sum objective or the min-time objective. The following are assumed: The system is decentralized. Vehicles start from different positions. Every vehicle initially knows of all locations of interest but not the locations of other vehicles in the system. Each vehicle can communicate with all other vehicles in the system. Vehicles all travel with the same constant speed when visiting locations.

The problem can be formulated as a decentralized task allocation problem [25] which has a binary integer programming formulation similar to the mTSP. In this case, tasks are defined as visiting locations of interest. Let the binary variable $x_{ij} = 1$ if vehicle i is assigned task j, otherwise $x_{ij} = 0$. The route vehicle i takes can be defined as a sequence of x_{ij} called $\boldsymbol{x_i}$. Let $\boldsymbol{x}$ be the set of routes for the system meaning the set of $\boldsymbol{x_i}$ for all i. Let $c(\boldsymbol{x})$ be the cost function to be minimized which is either the min-sum or min-time objective. Thus, the following is the binary integer program for the problem:

$$\min_{x} c(\boldsymbol{x}) \tag{5.1}$$

$$\text{s. t.:} \sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{5.2}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, m \quad j = 1, \ldots, n \tag{5.3}$$

Equation (5.1) is the objective to be minimized. Equation (5.2) exists to ensure all tasks are completed meaning all locations are visited. Equation (5.3) defines x_{ij} as a binary variable.

For min-sum, the objective can be defined as follows where $c_d(\boldsymbol{x_i})$ is the distance traveled by vehicle *i* to follow the route defined by $\boldsymbol{x_i}$:

$$c(\boldsymbol{x}) = \sum_{i=1}^{m} c_d(\boldsymbol{x_i}) \tag{5.4}$$

For min-time, the objective is similar to min-max and can be defined in terms of $c_t(\boldsymbol{x_i})$ which is the time taken by vehicle *i* to follow the route defined by $\boldsymbol{x_i}$. $c_d(\boldsymbol{x_i})$ is not used because the distance traveled by a vehicle is not equivalent to the time taken since vehicles may stop moving for periods of time. The min-time objective is defined as follows:

$$c(\boldsymbol{x}) = \max_{i} c_t(\boldsymbol{x_i}) \tag{5.5}$$

5.3 *Solution Methods*

Both optimization-based and auction-based methods were considered. The optimization-based approaches included the Decentralized Hungarian (DH) approach and a novel decentralized Genetic Algorithm (GA) approach. These approaches utilize well-known optimization techniques when assigning tasks to vehicles. The

auction-based approaches include the well-known Consensus-based Auction Algorithm (CBAA) and Asynchronous Consensus-based Bundle Algorithm (ACBBA). More recent auction-based approaches were also considered including the Performance Impact Algorithm (PIA) and Hybrid Information and Plan Consensus (HIPC) approach. We also considered a greedy nearest neighbors (NN) approach that served as a simple baseline to compare against.

Since the system is decentralized, each method is run separately on each vehicle. All methods are adapted to work with asynchronous communication meaning vehicles can decide their routes independently of communication with other vehicles. All methods considered continuously run until the mission is complete. Generally, vehicles do not generate new routes until the current assigned route is complete. Exceptions to this include the GA approach and when a task in the current route was completed by another vehicle. Figure 5.1 illustrates the basic structure shared by all methods implemented with the exception of the GA approach. The execution thread is run continuously onboard each vehicle. The methods iterate between assignment and consensus phases and return a new route to execute if no route is currently being executed.

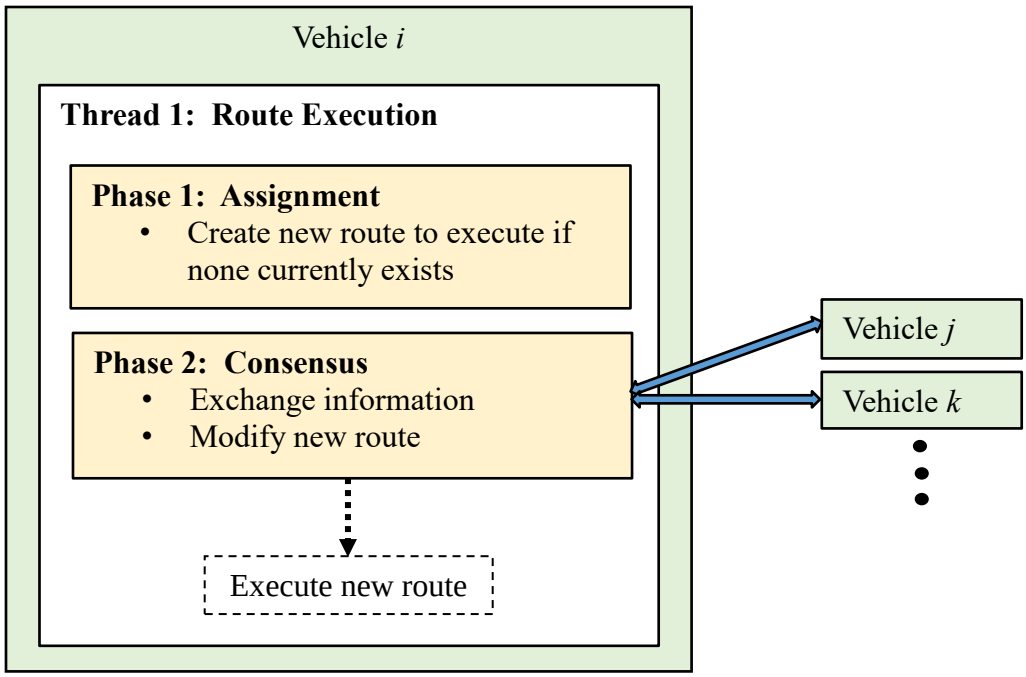


Figure 5.1. Basic structure of methods implemented (excluding GA). A new assignment is determined in the first phase if no route is currently being executed by the vehicle. Information is exchanged and updated in the second phase. The new route may be modified based on the updated information. Finally, if a new route is determined by the task allocation method, it is executed.

The following subsections describe how each approach determines the task assignment for a vehicle. For most of the methods, two cost functions are considered. One aims to optimize the min-sum objective while the other is a min-max function that attempts to optimize the min-time objective by minimizing the longest route of all vehicles.

5.3.1 **Greedy Nearest Neighbors (NN)**

The greedy NN method is a simple approach where each vehicle assigns itself the lowest cost task that has not been completed. Cost for this method is defined as the distance between the location to be visited and the vehicle's current location. This method differs from the nearest neighbors algorithm presented in Chapter 3 because it

only assigns one task. There is no consensus on the assignment, therefore, vehicles act greedily without regard for other vehicles. Vehicles only communicate which tasks they know to be completed.

5.3.2 **Decentralized Hungarian (DH)**

The DH approach [29] is a task allocation method that decides the task assignment based on a cost matrix. Elements of this matrix define the cost of each task to each vehicle. This method is constrained to assign a single task at a time to the vehicle. There are two phases to the approach. In the first phase, if the vehicle is not currently assigned a task, the vehicle runs the well-known Hungarian algorithm [47] on the cost matrix to obtain the optimal one-to-one task assignment for the system. In the second phase, the vehicle attempts to exchange cost matrices with all other vehicles. The vehicle then updates its own cost matrix based on information from received cost matrices. Two cost functions are implemented. The first is the distance between the vehicle and the location to visit and the second is the same distance plus a bias based on how far the vehicle has traveled. The latter is considered the min-max (mm) variant for the approach.

5.3.3 **Consensus-based Auction Algorithm (CBAA)**

The CBAA [24] is similar to the DH approach, however, instead of utilizing a cost matrix it uses a list of bids. Bids in this case are defined as the same costs implemented for the DH approach, however, only the best bid on each task is stored. Each vehicle greedily assigns itself the lowest cost task that it has outbid other vehicles for. Instead of exchanging cost matrices, the vehicles now exchange the list

of bids with each other and perform consensus on these bids. For consensus, each vehicle updates its bids list with the lowest bids received from all vehicles. The original algorithm is written to iterate between the auction and consensus phases until a convergence criterion is met. Then, the vehicle would be assigned a task. However, to avoid lengthy computational time for this and other methods a limit on the number of iterations is set. As with the DH approach, the same two cost functions are implemented yielding the min-sum and the min-max (mm) variants.

5.3.4 **Asynchronous Consensus-based Bundle Algorithm (ACBBA)**

The ACBBA [25] is a generalization of the CBAA that assigns a bundle of tasks instead of a single one. The ACBBA is an extension to the original CBBA [24] which is used for asynchronous systems. Along with a list of bids, it maintains other information such as the winning vehicles and timestamps that are used for consensus. Since multiple tasks are being assigned at a time, bids for tasks depend on tasks already assigned. Again, two cost functions are considered. In this case, the first is the distance between the previous task in the bundle and the current task. The second is the distance expected to be traveled by the vehicle up to the current task from the start of the mission. The latter is considered the min-max (mm) variant.

5.3.5 **Performance Impact Algorithm (PIA)**

The PIA [26] is another consensus-based approach that modifies the CBBA to utilize a different kind of bid evaluation called “significance”. It also makes improvements in how conflict resolution is achieved in the consensus phase. Originally this method used the consensus rules defined for the CBBA. This was

modified to instead use the ACBBA consensus rules. For this method, only the min-sum cost function was implemented.

5.3.6 **Hybrid Information and Plan Consensus (HIPC)**

The HIPC approach [27] is an extension to the CBBA that utilizes a global task assignment procedure on each vehicle instead of a local greedy approach. This means each vehicle solves the task allocation problem for the entire system to determine its own routes. In this way, vehicles attempt to predict what tasks other vehicles will complete and determine their own routes based on this. The assignment procedure implemented was the min-max nearest neighbors algorithm presented in Chapter 3. As with PIA, this approach was modified to use the ACBBA consensus rules. For this method, only the min-max cost function was implemented.

5.3.7 **Decentralized Genetic Algorithm (GA)**

The decentralized GA is a new approach in which each vehicle continuously runs a GA in parallel with route execution. The GA is based on an existing implementation [37] and was used in previous chapters to find routes for the entire system. The implementation is modified to remove tasks from solutions in the population whenever those tasks are completed. Also, the initial population includes the solution of the min-max nearest neighbors algorithm presented in Chapter 3. The cost evaluation is also modified for different variants of the objective. The first variant of cost evaluation uses the min-sum objective. The second variant evaluates the min-max objective. This is called the min-max (mm) variant. The third variant uses the min-max objective with a small min-sum bias. This is called the multi-

objective (multi) variant. This bias is the total distance scaled by 0.01 and helps reduce issues seen with the pure min-max objective which will be presented in the Results section. This cost function is given by Equation 5.6. The method is also modified to work with multiple depots since the vehicles start from different positions.

$$c(\boldsymbol{x}) = \max_i c_d(\boldsymbol{x_i}) + 0.01 * \sum_{i=1}^{m} c_d(\boldsymbol{x_i}) \tag{5.6}$$

The route execution thread queries the GA for its current best solution at a fixed rate. However, if the number of vehicles is greater than the number of locations left to be visited, it queries the min-max nearest neighbors algorithm for the solution. This is done because the referenced GA cannot handle this case. The solution is a set of routes for the entire system. It uses the queried solution to determine the current route for the vehicle to execute regardless of what the route currently being executed is. In the same thread, the vehicle attempts to exchange its current best solution with all other vehicles. These solutions are incorporated by the vehicle's GA into its population. In this way, the method indirectly attempts to reach consensus on the routes. This method has a key advantage in being able to converge to a globally optimal solution. However, since the method is stochastic, routes may change undesirably across different missions and even during a mission. The pseudocode for the route execution thread is provided in Algorithm 5.1. Note, when sending the current best route, the vehicle also sends information on tasks that have been completed and the vehicle's current location. Figure 5.2 illustrates the basic structure of this approach. As described, two threads are continuously run in parallel. In one

thread, the GA is run and in the other the route execution is run. This route execution thread is the same as the ROUTE_EXECUTION function detailed in Algorithm 5.1.

Algorithm 5.1: Pseudocode for route execution run in a separate thread on vehicle *i*.

```
1:  function ROUTE_EXECUTION()
2:      while mission not complete:
3:          if num_vehicles > num_locations
4:              current_solution ← solution from nearest neighbors
5:          else
6:              current_solution ← current best solution from GA
7:          vehicle_route ← current_solution{i}
8:          execute(vehicle_route)
9:          send current_solution to all other vehicles
10:         receive solution from whoever responds
11:         store received solutions for incorporation into population by GA
12: end function
```

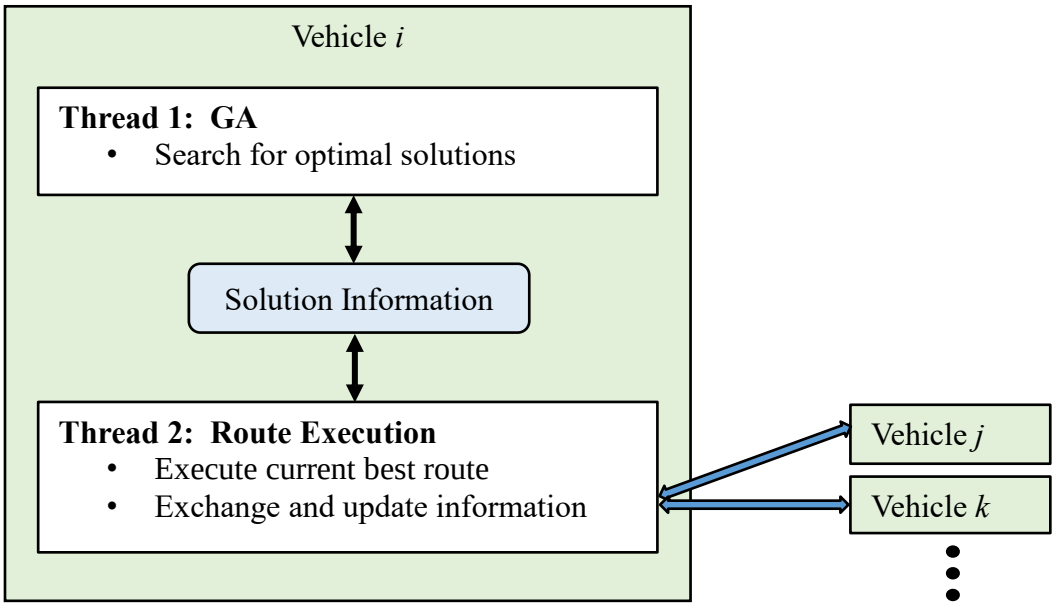


Figure 5.2. Basic structure of proposed decentralized GA approach. The two threads are continuously run in parallel. The GA continuously searches for better solutions in one thread. The current best route is queried and executed in the other thread. Additionally, information is exchanged and updated in the second thread. The two threads share information including the current best solution and the solutions received from other vehicles.

5.4 *Experimental Setup*

Each of the methods described were tested on problem instances of different size and at three levels of communication quality. Each case was run for 10 trials. The methods were implemented in Python and simulated using Robot Operating System (ROS) [48] Kinetic. Vehicles were simulated in separate threads, similar to how a typical decentralized system would operate. The speed of each vehicle is set to 5 units per second. Based on preliminary testing, the ACBBA and PIA variants were restricted to a bundle size of at most five and were run for five iterations at a time. Simulations were run on an AMD Ryzen 7 2700 Eight-Core 3.20 GHz Processor with 16.0 GB of RAM. Simulations were terminated when all vehicles were aware that all locations have been visited.

5.4.1 **Problem Instances**

Five problem instances were tested. The first instance shown in Figure 5.3(a) is the same 11 x 5 instance tested in Chapters 3 and 4 but now the vehicle start positions are randomized. In this chapter, the instance (n x m) is now considered 10 x 5 as there does not exist a singular depot anymore. The other four instances shown in Figure 5.3(b) to Figure 5.3(d) were uniformly randomly generated over a [0, 100] x [0, 100] region and are of sizes 20 x 4, 30 x 3, 35 x 5, and 40 x 6, respectively. Note, the five instances are of varying location-to-vehicle ratio.

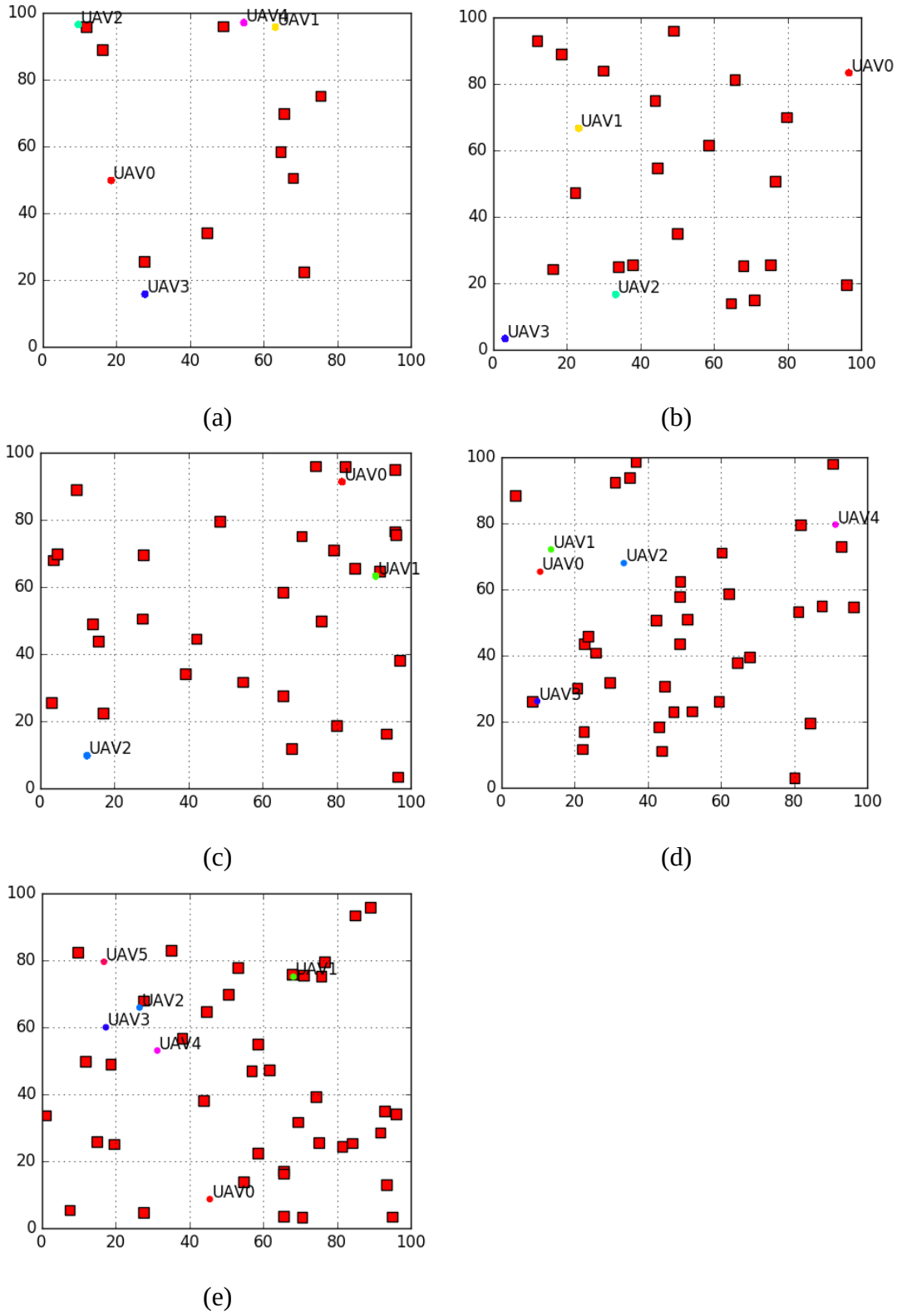


Figure 5.3. Problem instances of size: (a) 10 x 5, (b) 20 x 4, (c) 30 x 3, (d) 35 x 5, and (e) 40 x 6. UAVs refer to the vehicles in this problem. Vehicles are colored circles while locations to visit are red squares.

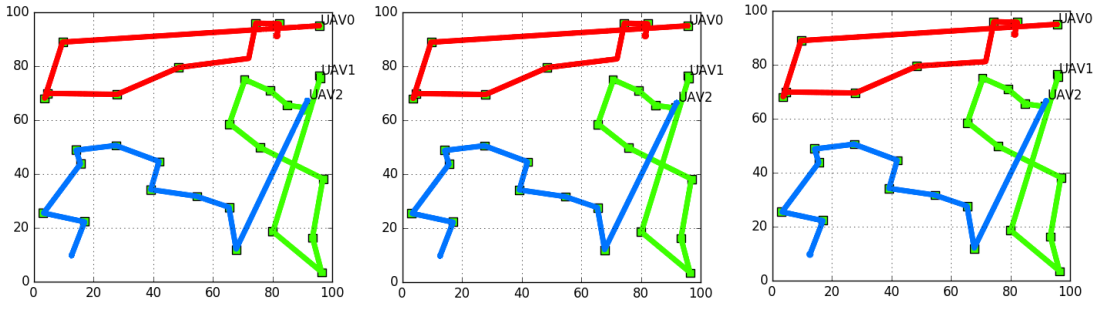
5.4.2 **Bernoulli Communication Model**

The communication among vehicles is modeled using a Bernoulli communication model [32]. In this simplistic model, each message sent between vehicles has a probability *b* of being successfully received and 1 – *b* of being dropped. This parameter *b* is fixed throughout each simulation. Three values of *b* were tested and include 1, 0.5, and 0.01. This gives a range of conditions from high to very low communication quality.

5.5 *Results*

5.5.1 **Example Routes**

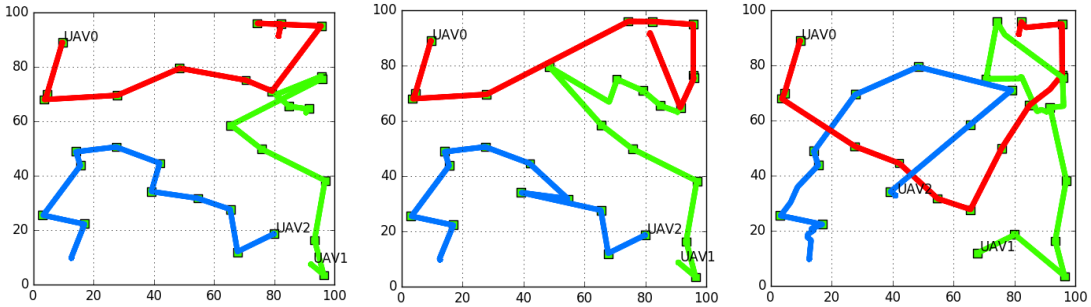
Figures 5.4, 5.5, and 5.6 present routes traveled by vehicles in the 30 x 3 instance using CBAA-mm, GA-mm, and GA-multi, respectively. Each vehicle's route is indicated by a distinct color (red, green, or blue). In each case, routes at different levels of communication quality are shown. For each figure, subfigure (a) shows the routes produced by running at *b* = 1. Subfigure (b) corresponds to the *b* = 0.5 case and subfigure (c) corresponds to *b* = 0.01.



(a) (b) (c)

Figure 5.4. Routes produced by running vehicles with CBAA-mm on the 30 x 3 instance at (a) *b* = 1, (b) *b* = 0.5, and (c) *b* = 0.01.

In Figure 5.4, we see that CBAA-mm was mostly unaffected by degradation in communication quality for this particular instance. This can partially be attributed to the simplicity of the approach. Since it only assigns one task at a time, the routes are not as reliant on communication with other vehicles and there is less chance for conflict. While the method is reliable at different levels of communication, the quality of the routes is poor for both min-sum and min-time objectives when compared to multiple-assignment methods like the GA. In particular, the red and green routes can be improved significantly.



(a) (b) (c)

Figure 5.5. Routes produced by running vehicles with GA-mm on the 30 x 3 instance at (a) $b = 1$, (b) $b = 0.5$, and (c) $b = 0.01$.

In contrast to CBAA-mm, the GA-mm approach was significantly affected by communication quality in this instance as seen in Figure 5.5. At high communication quality, the routes traveled are low cost and contain little to no overlap. These routes are significantly better than those produced by the CBAA-mm at $b = 1$ and $b = 0.5$. At $b = 0.5$, the routes are noticeably worse than the routes produced at full communication which is apparent with the blue route in Figure 5.5(b). However, at low communication quality, the routes produced by the GA-mm approach are significantly worse. The significant change in routes between the levels can also be attributed to the stochasticity of the method.

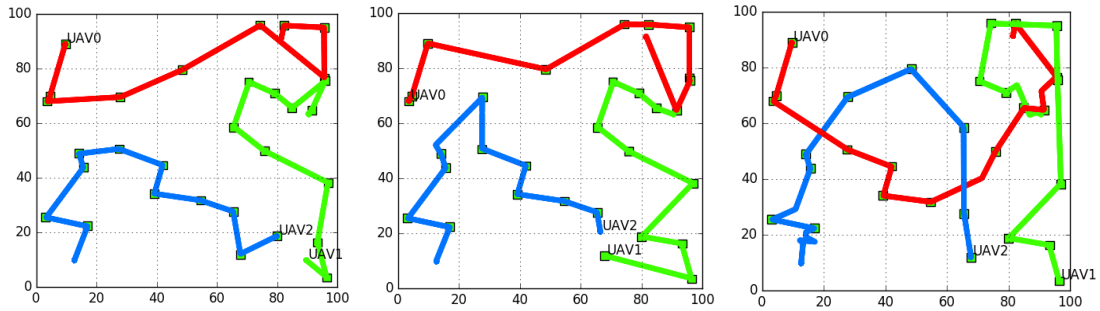


Figure 5.6. Routes produced by running vehicles with GA-multi on the 30 x 3 instance at (a) $b = 1$, (b) $b = 0.5$, and (c) $b = 0.01$.

The GA-multi approach was also affected by communication quality as seen in Figure 5.6 but was able to reduce some of the issues present with GA-mm. Comparing Figures 5.6 and 5.5, each vehicle's route is optimized when using the GA-multi approach. This is indicated by the lack of intraroute overlap across communication qualities. This is the result of adding the small min-sum bias to the cost evaluation. With a pure min-max objective, only the longest route is considered by the cost function. This means the other routes can change however they want with potentially no effect on the cost of a solution. This leads to the undesirable behaviors seen with the green and blue routes in Figure 5.5(a) and 5.5(b), respectively. The bias term solves this issue which is apparent in Figure 5.6.

5.5.2 **Example Convergence**

Figure 5.7 shows example convergence plots for vehicles running the GA-multi approach on the 30 x 3 instance. The plots illustrate the cost of the current best solution found by the GA-multi approach over time for each vehicle. The costs are averaged over 10 trials for each vehicle and 95% confidence intervals are shown at each point.

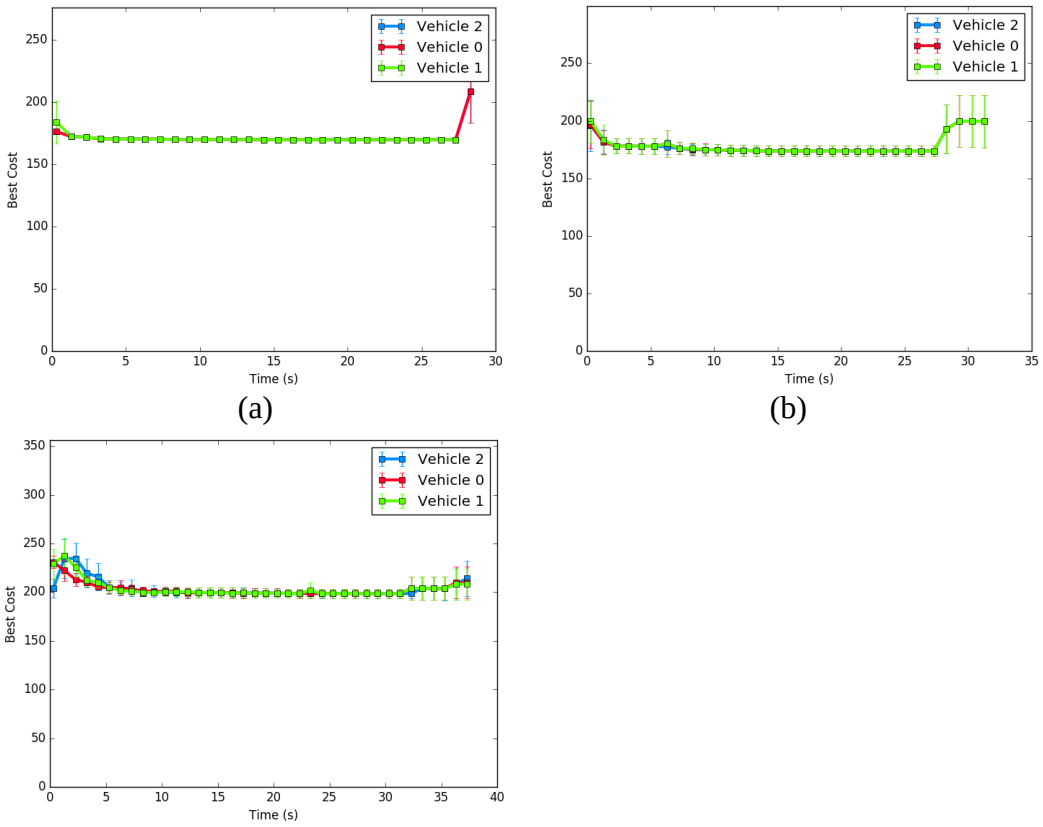


Figure 5.7. Convergence plots produced by running vehicles with GA-multi on the 30 x 3 instance at (a) b = 1, (b) b = 0.5, and (c) b = 0.01. Each point represents the cost of the current best solution for the vehicle averaged over 10 trials.

In the full communication case shown in Figure 5.7(a), all three vehicles rapidly converged to the same solution and this solution did not change for most of the execution time. Towards the end of the trials, the vehicles' costs increased. This is because the GA used only generates and considers solutions where each vehicle is assigned at least one location to visit. Such solutions may not be optimal depending on the objective, especially towards the end of the mission when there are fewer tasks to assign. This causes the solution's cost to increase since it will force vehicles to complete tasks when it is not optimal for it to do so. Although the cost of the solution can increase significantly as shown in Figure 5.7, the min-time and min-sum objective values are not significantly impacted since the approach switches to using

the nearest neighbors solution when the number of tasks is less than the number of vehicles. As the communication quality worsened, the vehicles took longer to converge as shown in Figures 5.7(b) and 5.7(c).

5.5.3 **Total Time**

Both total time and total distance performance metrics were considered. These metrics were evaluated for all methods and variants to analyze which variants perform best on each metric. This reflects how well each method optimizes the min-sum and min-time objectives. Time was recorded from the start of the trial until the point all locations of interest were visited. Distance was recorded until the end of the simulation when all vehicles were aware all locations have been visited. Tables 5.1 to 5.3 report the mean time to visit all locations of interest for each method on each instance at b = 1, 0.5, and 0.01, respectively. Standard deviations (SDs) are also reported. The best result is bolded and shaded in green for each case.

Table 5.1. Mean times to visit all locations and standard deviations (SDs) for each method on each instance at b = 1. Times are reported in seconds.

***b* = 1**	10 x 5		20 x 4		30 x 3		35 x 5		40 x 6	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
NN	19.6	0.1	45.4	0.1	41.6	0.02	32.4	4.1	26.2	0.1
CBAA	14.7	2.3	28.8	0.1	41.7	1.1	31.1	2.6	25.9	0.4
CBAA-mm	15.2	2.2	28.8	0.03	41.3	0.02	27.3	2.1	26.1	0.1
DH	**11.6**	0.02	29.6	0.05	41.6	1.1	25.2	1.1	23.8	1.0
DH-mm	**11.6**	0.02	29.6	0.05	42.7	1.7	25.9	1.3	23.7	1.5
ACBBA	14.9	1.1	23.5	1.6	34.4	0.04	27.5	1.4	26.0	4.3
ACBBA-mm	14.0	2.1	25.5	0.03	34.4	0.03	24.6	0.03	24.9	0.2
PIA	19.2	2.0	34.4	5.6	49.0	0.6	31.4	3.6	35.7	8.0
HIPC	12.0	1.2	28.0	1.2	34.4	0.04	24.0	0.02	24.7	0.05
GA	11.9	0.1	25.8	1.6	33.9	0.03	**21.8**	0.2	23.4	2.1
GA-mm	12.5	1.4	23.6	2.0	**33.7**	0.4	23.1	1.3	23.5	2.4
GA-multi	14.2	1.8	**22.8**	1.5	**33.7**	0.6	22.2	1.3	**22.2**	0.7

At b = 1, the proposed GA approach is dominant on the time objective on four of five instances. In the 10 x 5 instance, the DH and DH-mm approaches which are single assignment methods outperformed the others. However, in the instances with higher location-to-vehicle ratios, the multiple-assignment methods including the ACBBA variants, HIPC, and GA variants significantly outperformed the single assignment methods. In the 20 x 4 instance, the GA-multi approach had the best mean time of all methods with the GA-mm variant close behind. In the 30 x 3 instance, the GA variants outperformed all other methods on mean time which is also the case in the 35 x 5 instance. Finally, in the 40 x 6 instance, the GA-multi approach again outperformed the other methods by a relatively significant margin. These results show the GA variants, particularly the GA-multi approach, typically outperform the other methods on mean time at high communication quality.

Table 5.2. Mean times to visit all locations and standard deviations (SDs) for each method on each instance at b = 0.5. Times are reported in seconds.

***b* = 0.5**	10 x 5		20 x 4		30 x 3		35 x 5		40 x 6	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
NN	19.6	0.1	45.5	0.1	41.6	0.02	29.3	5.1	26.3	0.1
CBAA	14.7	2.4	29.0	0.7	41.6	1.1	30.4	3.4	26.0	0.1
CBAA-mm	14.3	2.4	28.8	0.05	41.7	1.1	27.0	2.3	26.0	0.1
DH	12.1	1.6	29.2	1.9	41.3	0.1	25.4	0.8	**23.6**	1.1
DH-mm	12.7	2.2	29.1	1.9	42.0	1.4	25.4	0.8	23.9	1.3
ACBBA	15.7	2.9	28.0	5.0	34.4	0.03	26.9	3.8	26.4	3.1
ACBBA-mm	16.6	2.3	25.7	0.1	34.4	0.04	24.6	0.03	25.1	0.8
PIA	17.6	2.8	30.4	4.5	49.8	2.1	32.7	4.3	33.0	2.7
HIPC	13.1	2.4	28.0	1.2	34.4	0.03	**24.0**	0.02	24.6	0.04
GA	**11.8**	0.8	24.6	3.1	35.3	2.4	25.3	2.9	25.8	2.8
GA-mm	12.6	1.4	**22.4**	2.0	**34.0**	2.3	24.8	2.2	24.5	1.9
GA-multi	14.3	1.7	23.2	1.4	34.4	1.2	24.5	2.2	23.8	2.3

For the b = 0.5 case, the proposed GA and its variants deteriorated in performance and were no longer dominant across many of the instances. In the 10 x 5 instance, the GA approach was the best performer with the DH approach in close

second. The GA-mm approach had the best mean time in the 20 x 4 instance as well as the 30 x 3 instance while its other variants were close behind. Although the GA variants performed well in the 35 x 5 and 40 x 6 instances, the HIPC and DH approaches had the best mean times in the respective instances. These results indicate the GA variants may not perform as well on the time objective as communication quality degrades.

Table 5.3. Mean times to visit all locations and standard deviations (SDs) for each method on each instance at $b = 0.01$. Times are reported in seconds.

***b* = 0.01**	10 x 5		20 x 4		30 x 3		35 x 5		40 x 6	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
NN	20.3	0.4	45.9	0.4	58.0	12.0	46.1	6.4	26.3	0.2
CBAA	17.0	0.2	31.3	0.03	41.6	0.2	33.8	1.8	26.1	0.1
CBAA-mm	17.0	0.1	29.2	0.2	41.8	0.6	32.4	3.0	26.6	1.1
DH	17.3	0.5	34.8	2.5	42.7	1.1	34.4	4.3	**25.9**	0.2
DH-mm	17.2	0.2	34.5	1.8	42.9	1.1	34.3	4.7	**25.9**	0.1
ACBBA	14.6	1.6	25.2	1.7	**34.4**	0.04	27.1	1.7	26.3	4.6
ACBBA-mm	12.7	0.6	26.3	1.5	34.5	0.05	**24.7**	0.5	26.1	0.1
PIA	18.7	3.7	33.8	4.2	48.1	2.6	36.5	4.7	37.9	5.9
HIPC	15.6	1.5	**23.2**	0.5	45.7	2.1	32.6	1.2	27.5	2.9
GA	13.6	1.2	25.9	2.0	41.8	2.7	29.3	2.4	32.3	1.6
GA-mm	**12.5**	2.0	23.3	1.9	41.5	1.9	28.6	1.9	33.5	3.2
GA-multi	13.5	2.2	24.1	1.1	39.8	1.8	27.4	1.8	31.8	1.6

In the $b = 0.01$ case, the results differ significantly. As seen in the previous section, the performance of the GA variants significantly deteriorated at low communication quality. However, the CBAA and ACBBA variants remained relatively unaffected by the low communication quality. The GA variants no longer dominated in the 20 x 4 instance but instead outperformed other methods in the 10 x 5 instance. The insensitivity of the ACBBA to communication quality is shown with its performance in the 30 x 3 and 35 x 5 instances where the ACBBA and ACBBA-mm had the best mean times. The time taken for these two methods remained generally the same across different levels of communication quality in these instances. In the 30

x 3 and 40 x 6 instances, the mean times of the GA variants significantly increased when compared to their values at higher communication quality. At this level of communication quality, even the greedy nearest neighbors approach outperformed the GA variants in the 40 x 6 instance.

5.5.4 **Total Distance**

Tables 5.4 to 5.6 report the mean total distance traveled for each method on each instance at b = 1, 0.5, and 0.01, respectively. Standard deviations (SDs) are also reported. The best result is bolded and shaded in green for each case.

Table 5.4. Mean total distance traveled and standard deviations (SDs) for each method on each instance at b = 1.

***b* = 1**	10 x 5		20 x 4		30 x 3		35 x 5		40 x 6	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
NN	485.6	2.4	903.3	0.7	618.5	0.2	800.8	102.3	773.5	2.7
CBAA	298.3	28.3	518.7	9.5	606.7	20.3	688.4	52.1	713.6	16.9
CBAA-mm	306.1	27.3	521.0	8.0	599.7	0.3	619.3	21.2	720.2	11.3
DH	259.6	6.4	520.3	13.5	603.8	21.4	576.6	22.3	655.1	10.0
DH-mm	259.9	8.7	511.2	8.5	628.3	31.9	583.1	35.7	644.0	23.5
ACBBA	356.3	31.5	**381.3**	48.8	440.3	1.8	638.9	53.1	675.2	40.1
ACBBA-mm	319.2	62.9	429.1	0.6	**439.7**	0.0	519.9	0.02	606.9	1.5
PIA	203.6	18.6	485.4	51.6	559.2	23.8	501.3	16.1	591.0	20.3
HIPC	**198.9**	1.2	402.7	8.3	445.0	0.0	**443.0**	5.7	589.0	0.0
GA	221.5	6.1	436.9	31.6	456.5	0.5	459.7	2.8	**588.5**	43.7
GA-mm	269.0	30.7	431.3	26.7	470.4	10.2	541.4	33.2	642.4	33.9
GA-multi	247.1	13.5	397.2	13.6	456.6	5.1	499.5	28.9	597.2	15.2

For b = 1, the results for total distance differ from the time results. The HIPC approach achieved the best total distance on the 10 x 5 and 35 x 5 instances. The ACBBA had the best performance in the 20 x 4 instance with the GA-multi approach relatively close behind. In the 30 x 3 instance, the ACBBA and ACBBA-mm outperformed the other methods. The GA approach had the mean best distance in the 40 x 6 instance and generally had good performance on the other instances.

Table 5.5. Mean total distance traveled and standard deviations (SDs) for each method on each instance at $b = 0.5$.

***b* = 0.5**	10 x 5		20 x 4		30 x 3		35 x 5		40 x 6	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
NN	486.4	3.3	904.6	0.8	618.6	0.4	724.2	127.1	772.0	2.9
CBAA	300.1	35.8	520.5	13.0	606.7	20.0	660.0	42.8	721.1	9.6
CBAA-mm	303.2	35.4	518.0	0.7	606.5	20.2	616.8	24.8	716.6	13.4
DH	300.4	40.9	579.0	38.5	613.5	2.2	596.6	23.7	657.0	22.3
DH-mm	314.4	55.2	576.7	37.2	623.4	21.2	603.0	20.3	661.2	31.1
ACBBA	328.3	58.1	476.4	75.8	**439.7**	0.0	618.6	50.8	710.4	74.2
ACBBA-mm	353.6	49.0	430.4	0.7	**439.7**	0.0	529.4	34.5	629.0	43.0
PIA	**194.0**	29.2	453.9	45.6	553.4	33.1	498.6	15.2	602.3	21.1
HIPC	216.4	17.8	**407.1**	16.9	445.0	0	**443.0**	5.7	**589.9**	0.0
GA	214.5	15.3	412.3	44.3	479.4	30.3	526.1	47.7	658.8	71.6
GA-mm	268.5	33.3	417.9	35.2	473.4	35.7	565.8	36.9	660.4	37.6
GA-multi	245.5	19.8	416.0	21.1	473.4	21.8	547.7	47.0	646.8	49.4

For $b = 0.5$, the PIA outperformed the other methods in the 10 x 5 instance. Previously, we saw the PIA often had the worst mean times but this was caused by some vehicles not being used. This behavior can reduce total distance traveled which is seen here. The HIPC approach had the best performance on three of five instances including the 20 x 4, 35 x 5, and 40 x 6 instances. Again, the ACBBA and ACBBA-mm outperformed the other methods in the 30 x 3 instance little to no variation. Matching with the time results, the GA variants increased in mean distance as the communication quality decreased.

Table 5.6. Mean total distance traveled and standard deviations (SDs) for each method on each instance at $b = 0.01$.

***b* = 0.01**	10 x 5		20 x 4		30 x 3		35 x 5		40 x 6	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
NN	504.8	9.2	912.5	7.4	865.6	179.2	1142.5	159.6	774.1	8.5
CBAA	391.1	22.9	560.8	1.5	603.5	2.9	757.9	71.9	733.9	12.4
CBAA-mm	366.3	28.1	544.5	12.8	606.3	8.6	729.7	68.9	707.9	57.9
DH	434.1	10.8	694.1	50.9	638.1	16.8	866.9	96.4	775.4	4.0
DH-mm	430.3	4.9	689.7	36.8	640.7	16.7	856.3	116.0	774.7	3.4
ACBBA	341.4	34.1	**423.6**	49.9	**439.7**	0.0	630.6	63.3	710.8	69.8
ACBBA-mm	286.2	20.6	458.6	35.8	**439.7**	0.0	**576.3**	8.5	745.2	44.4
PIA	383.6	66.3	597.0	97.5	593.4	65.2	713.4	102.7	894.8	133.1

HIPC	283.3	23.1	431.6	10.0	621.1	41.7	645.7	29.7	**636.6**	33.9
GA	**258.5**	24.3	465.3	41.6	589.1	36.3	634.6	56.3	848.6	23.0
GA-mm	263.3	34.5	428.8	25.0	602.7	23.3	667.3	33.2	928.7	86.4
GA-multi	278.2	33.3	431.5	15.3	571.4	21.3	637.6	38.8	866.6	46.1

For $b = 0.01$, the GA variants achieved the best mean distance on the 10 x 5 instance, similar to the time results. However, in larger instances there was significant deterioration of the GA at low communication quality. Again, the ACBBA variants remained relatively unaffected by degradation in communication. This insensitivity allowed them to achieve the best mean distances on a majority of the instances at low communication quality. Although the HIPC approach was affected by the low communication quality, the impact was not enough for it to lose in total distance to the other methods in the 40 x 6 instance. Overall, the HIPC approach and the ACBBA variants performed well on the distance metric across different communication qualities.

5.6 *Summary*

This chapter considered a decentralized multi-vehicle route planning problem where vehicles plan their own routes using local information and collaboration. Min-sum and min-time objectives were considered. Several existing solution methods were considered including the DH, CBAA, ACBBA, PIA, and HIPC approaches. A novel GA approach was proposed along with variants of most methods to evaluate the min-max objective which is closely correlated with min-time. The methods were tested on five problem instances of varying size. Additionally, the communication quality was varied at three levels from high to very low to test its effect on method performance.

The results showed that single assignment methods like the DH approach perform well on problems with low location-to-vehicle ratio; however, multiple assignment methods typically dominate at higher ratios. We also saw that min-max variants of the approaches can sometimes outperform min-sum variants on time taken. The proposed GA and its variants had the best times on most instances when the communication quality was high. However, as the communication quality degraded, the GA's performance deteriorated significantly relative to other methods. The ACBBA variants were more resilient to degrading communication quality and were able to maintain consistent performance across all levels of quality. These variants along with the HIPC approach were able to achieve the best performance on the total distance metric across most instances and communication qualities.

The next chapter summarizes the work presented in this thesis and details the key contributions. Future work is also discussed for each of the three studies discussed in this thesis.

Chapter 6: Conclusions

6.1 *Summary*

This thesis explored the problem of multi-vehicle route planning in detail and compared a number of approaches for solving different types of route planning problems. Chapter 3 considered a classical multi-vehicle route planning problem. The problem is represented as an mTSP, which is a fundamental problem typically used to solve more complex multi-vehicle route planning problems. We considered two objectives including the min-sum and min-max objectives. Existing exact and heuristic solution methods were tested and compared. These included a branch-and-bound based ILP solver, GA, SA approach, nearest neighbors algorithm, MST approximation, and Christofides algorithm. We compared the methods on solution quality and computational time. The results showed that the ILP solver was able to reach the lowest cost solutions the fastest when working with the min-sum objective. However, with the min-max objective, the heuristics methods dominated the ILP solver. In particular, the GA showed the best performance with the min-max objective.

We then adapted the GA used in Chapter 3 to find solutions to the failure-robust multi-vehicle route planning problem presented in Chapter 4. In this problem, vehicles can fail at locations visited and if a vehicle fails, replanning is needed to generate new routes that satisfy mission requirements. The goal was to generate robust solutions to the problem which are solutions that have the best possible worst-case cost over all possible failure scenarios. We modified the GA to evaluate the

worst-case cost of solutions and then tested and compared the modified approach to optimistic planning approaches that did not consider uncertainty. The results showed that the GA was able to generate high quality min-max robust solutions when compared to the optimistic planning solutions, although at high computational effort.

Finally, Chapter 5 considered a decentralized extension to the multi-vehicle route planning problem. This is also known as the decentralized task allocation problem. In this problem, routes still need to be determined for vehicles to take to visit locations of interest. However, each vehicle now determines its own route in real time using local information along with some collaboration. Several existing decentralized task allocation methods were considered along with a novel GA approach. We tested and compared the methods on instances of varying location-to-vehicle ratios and at varying levels of communication quality. We considered min-sum (total distance) and min-time (total time) performance metrics. Results showed the consensus-based auction methods were consistent across different instances and at different communication qualities. They often outperformed the other methods on the total distance metric but not the time metric. The proposed GA approach performed the best on the time metric at higher communication quality. However, as the communication quality worsened, the GA's performance significantly degraded compared to other methods.

6.2 *Contributions*

1. A comparison of methods for solving a classical multi-vehicle route planning problem that contributes new insights on how the methods perform on solution quality and computational time for min-sum and min-max objectives.

2. A formulation and comparison of solution methods for a new failure-robust multi-vehicle route planning problem including a GA approach that generates robust solutions.
3. A comparison of methods for solving a decentralized multi-vehicle route planning problem that contributes new insights on the performance of the methods on min-sum and min-time objectives at different levels of communication quality.
4. A novel GA approach for decentralized task allocation that outperforms standard methods on larger instances at high communication quality.

6.3 *Future Work*

There are a number of future directions for the problems explored in this thesis. For the classical multi-vehicle route planning problem, other approaches should be tested with min-sum and min-max objectives. These include the Clarke and Wright savings algorithm [17], which is popular in the VRP literature, and possibly an improved implementation of the Christofides algorithm which we used in the study. Multi-objective optimization is another avenue for future work on this topic. Instead of considering the min-sum and min-max objectives separately, combining the two as a bi-objective function could show interesting results. We saw this with the GA in the decentralized task allocation problem where we added a min-sum bias that sometimes improved the performance of the method.

For the failure-robust multi-vehicle route planning problem, future work should consider alternatives to the proposed reduced scenario set such as iteratively building up the scenario set. Also, other replanning approaches should be tested such

as the nearest-neighbors approach which can generate better min-max solutions. Other heuristics that can better exploit the structure of the robust planning problem should also be explored using the insights gained from the study. In order to understand the optimality gap of solutions found by the GA and other heuristics, an MILP should be formulated to exactly solve the robust problem. Future studies should also consider the possibility of multiple vehicle failures. Considering multiple failures significantly increases the complexity of this already complex problem, therefore, simplifying assumptions or heuristics are needed. Solutions found by considering a single vehicle failure may perform well even if multiple failures did occur. One possible approach is to solve the one-failure problem every time a vehicle fails so that the recovery plan is robust against the next failure. Such solutions should be evaluated in multi-failure scenarios to analyze how robust they are to multiple failures.

The approaches for decentralized task allocation should be tested on different problem scenarios and with different communication models. Instead of considering a scenario with known stationary locations of interest, scenarios with unknown and moving locations may produce different results, especially for the multiple assignment methods. Also, alternatives to the Bernoulli communication model should be tested such as the Gilbert-Elliott model [49] which can be considered more practical. The proposed GA approach can be improved by tuning the population size and possibly adding genetic operations such as crossover. Additionally, the GA should be modified to remove the constraint enforcing vehicles to be assigned at least one task. This variant should be tested to explore if this improves performance.

References

[1] T. Bektas, “The multiple traveling salesman problem: an overview of formulations and solution procedures,” *Omega,* vol. 34, no. 3, pp. 209–219, 2006.

[2] R. Jonker and T. Volgenant, “An improved transformation of the symmetric multiple traveling salesman problem,” *Operations Research,* vol. 36, no. 1, pp. 163-167, 1988.

[3] A. Singh, “A review on algorithms used to solve multiple traveling salesman problem,” *International Research Journal of Engineering and Technology (IRJET),* vol. 3, no. 4, pp. 598-603, 2016.

[4] M. R. Garey, and D. S. Johnson, “Computers and intractability: a guide to the theory of NP-completeness”, *New York: Freeman*, 1979.

[5] R. Scanlon, Q. Wang, and J. Wang, J., "Ant Colony Optimisation Model for Vehicle Routing Problem With Simultaneous Pickup and Delivery," in *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 4, pp. V004T05A035-V004T05A035, 2016.

[6] B. Scholz-Reiter, H. Rekersbrink, B. Wenning, and T. Makuschewitz, "A survey of autonomous control algorithms by means of adapted vehicle routing problems," in

ASME 2008 9th Biennial Conference on Engineering Systems Design and Analysis, vol. 2, pp. 411-416, 2008.

[7] J. Potvin, "State-of-the Art Review-Evolutionary algorithms for vehicle routing," *INFORMS Journal on Computing*, vol. 21, no. 4, pp. 518-548, 2009.

[8] P. M. França, M. Gendreau, G. Laporte, and F. M. Müller, "The m-traveling salesman problem with minmax objective," *Transportation Science*, vol. 29, no. 3, pp. 267-275, 1995.

[9] S. Somhom, A. Modares, and T. Enkawa, "Competition-based neural network for the multiple travelling salesmen problem with minmax objective," *Computers & Operations Research*, vol. 26, no. 4, pp. 395-407, 1999.

[10] C. Ren, "Solving min-max vehicle routing problem," *Journal of Software*, vol. 6, no. 9, pp. 1851-1856, 2011.

[11] D. Applegate, W. Cook, S. Dash, and A. Rohe, "Solution of a min-max vehicle routing problem," *INFORMS Journal on Computing*, vol. 14, no. 2, pp. 132-143, 2002.

[12] L. Bertazzi, B. Golden, and X. Wang, “Min-max vs. min-sum vehicle routing: A worst-case analysis,” *European Journal of Operational Research,* vol. 240, no. 2, pp. 372-381, 2015.

[13] T. Matsuura and K. Numata, “Solving min-max multiple traveling salesman problems by chaotic neural network,” in *2014 International Symposium on Nonlinear Theory and its Application,* pp. 237-240, 2014.

[14] U. Ritzinger, J. Puchinger, and R. F. Hartl, “A survey on dynamic and stochastic vehicle routing problems,” *International Journal of Production Research,* vol. 54, no. 1, pp. 215–231, 2016.

[15] K. Sundar, S. Venkatachalam, and S. G. Manyam, “Path planning for multiple heterogeneous Unmanned Vehicles with uncertain service times,” in *2017 International Conference on Unmanned Aircraft Systems (ICUAS),* pp. 480–487, 2017.

[16] T. Dulai, Á. Werner-Stark, and K. M. Hangos, “Algorithm for directing cooperative vehicles of a vehicle routing problem for improving fault-tolerance,” *Optimization and Engineering,* vol. 19, p. 239, June 2018.

[17] G. Clarke and J. W. Wright, “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points,” *Operations Research,* vol. 12, no. 4, pp. 519–643, 1964.

[18] I. Sungur, F. Ordonez, and M. Dessouky, “A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty,” *IIE Transactions*, vol. 40, no. 5, pp. 509–523, 2008.

[19] J. Han, C. Lee, and S. Park, “A Robust Scenario Approach for the Vehicle Routing Problem with Uncertain Travel Times,” *Transportation Science*, vol. 48, no. 3, p. 373-390, 2014.

[20] D. Habib, H. Jamal, and S. A. Khan, “Employing multiple unmanned aerial vehicles for co-operative path planning,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, p. 235, 2013.

[21] G. Giger, M. Kandemir, and J. Dzielski, “Reliable mission execution using unreliable UUVs,” *AUVSIs Unmanned Systems North America*, 2008.

[22] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot task allocation: A review of the state-of-the-art,” in *Cooperative Robots and Sensor Networks 2015*, pp. 31-51, 2015.

[23] V. Singhal and D. Dahiya, “Distributed task allocation in dynamic multi-agent system,” in *International Conference on Computing, Communication & Automation*, pp. 643-648, 2015.

[24] L. Brunet, H. L. Choi, and J. How, "Consensus-based auction approaches for decentralized task assignment," in *AIAA guidance, navigation and control conference and exhibit*, p. 6839, 2008.

[25] L. Johnson, S. Ponda, H. L. Choi, and J. How, "Asynchronous decentralized task allocation for dynamic environments," in *Infotech@ Aerospace 2011*, p. 1441, 2011.

[26] W. Zhao, Q. Meng, and P.W. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Transactions on Cybernetics*, vol. 46, no. 4, pp. 902-915, 2015.

[27] L. B. Johnson, H. L. Choi, and J. P. How, "Hybrid information and plan consensus in distributed task allocation," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, p. 4888, 2013.

[28] M. Nanjanath and M. Gini, "Repeated auctions for robust task execution by a robot team," *Robotics and Autonomous Systems*, vol. 58, no. 7, pp. 900-909, 2010.

[29] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 23-28, 2017.

[30] H. J. Choi, Y. D. Kim, and H. J. Kim, “Genetic algorithm based decentralized task assignment for multiple unmanned aerial vehicles in dynamic environments,” *International Journal of Aeronautical and Space Sciences*, vol. 12, no. 2, pp. 163-174, 2011.

[31] M. Rantanen, J. Modares, N. Mastronarde, F. Ghanei, and K. Dantu, “Performance of the asynchronous consensus based bundle algorithm in lossy network environments,” in *2018 IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pp. 311-315, 2018.

[32] M. Otte, M. Kuhlman, and D. Sofge, “Multi-robot task allocation with auctions in harsh communication environments,” in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 32-39, 2017.

[33] N. Christofides, “Worst-case analysis of a new heuristic for the traveling salesman problem,” Graduate School of Industrial Administration, Carnegie Mellon University, Technical Report 388, 1976.

[34] D. Huizing, “Solving the mTSP for fresh food delivery,” B.S. Thesis, *Delft University of Technology*, Netherlands, 2015.

[35] C. E. Miller, A. W. Tucker, and R. A. Zemlin, “Integer programming formulation of traveling salesman problems,” *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326-329, 1960.

[36] “Gurobi Optimizer Reference Manual”, *Gurobi Optimization*, 2018, [Online]. Available: https://www.gurobi.com/wp-content/plugins/hd_documentations/content/pdf/reference_manual.pdf. [Accessed: July 11, 2019].

[37] J. Kirk, “Fixed start/end point multiple traveling salesmen problem - genetic algorithm,” *MATLAB® Central File Exchange,* 2014, https://www.mathworks.com/matlabcentral/fileexchange/21299, [Accessed: July 11, 2019].

[38] S. Mostapha Kalami Heris, “Vehicle routing problem (VRP) using simulated annealing (SA),” *MATLAB® Central File Exchange*, 2015, https://www.mathworks.com/matlabcentral/fileexchange/53113. [Accessed: July 11, 2019].

[39] M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[40] S. Rathinam, R. Sengupta, and S. Darbha, “A resource allocation algorithm for multivehicle systems with nonholonomic constraints,” *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 98–104, 2007.

[41] R. C. Prim, “Shortest connection networks and some generalizations,” *Bell System Technical Journal*, vol. 36, no. 6, pp. 1389-1401, 1957.

[42] P. Šůcha, M. Kutil, M. Sojka, and Z. Hanzálek. “TORSCHE Scheduling Toolbox for Matlab,” in *IEEE International Symposium on Computer-Aided Control Systems Design. Munich*, Germany, 2006.

[43] “MP-TESTDATA - The TSPLIB Symmetric Traveling Salesman Problem Instances.” [Online]. Available: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html. [Accessed: July 11, 2019].

[44] R. Patel, E. Rudnick-Cohen, S. Azarm, and J. W. Herrmann, “Robust multi-UAV route planning considering UAV failure,” in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[45] R. Necula, M. Breaban, and M. Raschip, “Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems,” in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 873–880, 2015.

[46] I. Kara and T. Bektas, “Integer linear programming formulations of multiple salesman problems and its variations,” *European Journal of Operational Research*, vol. 174, no. 3, pp. 1449–1458, 2006.

[47] H. W. Kuhn, “The Hungarian Method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, pp. 83-97, 1955.

[48] M. Quigley, K. Conley, B. Gerkey, J. Faust, J., T. Foote, J. Leibs, ... and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.

[49] G. Haßlinger, and O. Hohlfeld, “The Gilbert-Elliott model for packet loss in real time services on the Internet,” in *14th GI/ITG Conference-Measurement, Modelling and Evalutation of Computer and Communication Systems*, pp. 1-15, 2008.