Multiple Autonomous Vehicle Mission Planning and Management

by

Wei Zhao

B.S.E.E, Tsinghua University (1990)
M.S.M.E. & M.S.E.E, University of Cincinnati (1992, 1993)

Submitted to the System Design and Management Program in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering and Management

at the
Massachusetts Institute of Technology

June 1999

© 1999 Wei Zhao. All rights reserved.

The author hereby grants MIT the permission to reproduce and distribute publicly paper or electronic copies of this thesis in whole or in part.

Signature of Author .................................................................

System Design and Management Program
June, 1999

Certified by .................................................................

Professor Thomas L. Magnanti
Dean, MIT School of Engineering
Thesis Supervisor

Certified by .................................................................

Dr. Stephan Kolitz
Principal Member of the Technical Staff
Draper Laboratory
Technical Supervisor

Accepted by .................................................................

John R. Williams
Associate Professor of Civil and Environmental Engineering
Co-Director, MIT System Design and Management Program
Multiple Autonomous Vehicle Mission Planning and Management

by

Wei Zhao

Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering and Management

Abstract

This thesis investigates multiple autonomous vehicle mission planning and management. It begins by introducing the basic concepts and objectives of the multi-vehicle mission-planning problem. Then it formulates the problem mathematically and analyzes parameters in the objective function. The solution approach uses a hierarchical mission-planning scheme to take advantage of a scalable architecture.

We develop a heuristic-based algorithm to solve the multiple-vehicle mission-planning problem. The algorithm has two phases: goal-point partitioning and routing. Goal-point partitioning uses a sweep procedure to group goal-points. Routing uses an implementation of simulated annealing combined with well-known TSP heuristics. Through the computational experiments conducted on both traveling salesman problem test cases, the TSPLIB library, and randomly generated test data, the routing algorithm performs quite well. It has been able to find TSP tours within one percent of optimality, and typically within one-half of one percent. The integration of the two-phase approach provides a solution to the multiple autonomous vehicle mission planning problem.

Thesis Supervisor: Professor Thomas L. Magnanti
Dean, MIT School of Engineering

Technical Supervisor: Dr. Stephan Kolitz
Charles Stark Draper Laboratory
ACKNOWLEDGMENTS

I would like to thank Dr. Stephan Kolitz, who provided technical supervision as well as overall guidance and encouragement, and especially for making the Draper fellowship available. In addition, I would like to thank the other members of Mission Planning Team of the Decision Systems Group at Draper, for their supporting and enthusiasm toward the project.

I would also like to express my sincere thanks to Professor and Dean Thomas L. Magnanti. It is the innovative System Design and Management Program, co-founded and previously co-directed by Professor Magnanti, that gave me the opportunity to pursue further education in both engineering and management at MIT. Furthermore, Professor Magnanti’s diligent efforts on reviewing and editing the thesis, not only provided me tremendous help toward finishing the program, but also established a role model for me to aim high and to pursue the best quality. It’s my great pleasure and honor obtaining advice from both senior advisors.

Finally, I would like to express my appreciation to my wife, Qing Xiao, for her understanding and support.

Wei Zhao
June 1999
ACKNOWLEDGMENTS (continued)

This thesis was supported by the Charles Stark Draper Laboratory Inc., under Contract IR&D-927.

Publication of this thesis does not constitute approval by The Charles Stark Draper Laboratory, Inc., of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Permission is hereby granted by the Author to the Massachusetts Institute of Technology to reproduce any or all of this thesis.

Wei Zhao
June 1999
ASSIGNMENT

Draper Laboratory Report Number T – 1349.

In consideration for the research opportunity and permission to prepare my thesis by and at The Charles Stark Draper Laboratory, Inc., I hereby assign my copyright of the thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts.

Wei Zhao
June 1999
TABLE OF CONTENTS

Chapter 1 Introduction
  1.1 Background
  1.2 Mission Planning Scenario
  1.3 Hierarchical Mission Planning Architecture
  1.4 Overview of the Thesis

Chapter 2 Problem Definition
  2.1 Fundamental Concepts
  2.2 Vehicle Routing Problem (VRP)
  2.3 Traveling Salesman Problem (TSP)
  2.4 Multi-Vehicle Mission-Planning Problem
    2.4.1 Problem Formulation
    2.4.2 Observations on the model

Chapter 3 Algorithm Study
  3.1 Introduction
  3.2 Traditional Methods for Solving the VRP
    3.2.1 Optimization-Based Approaches
    3.2.2 Heuristic Algorithms
  3.3 Algorithm Development Strategy
  3.4 Goal-Point Partitioning Algorithm
  3.5 Goal-Point Routing Algorithm
    3.5.1 Simulated Annealing (SA)
    3.5.2 Local Search Structure
    3.5.3 Fast Local Search
    3.5.4 Tour Routing
  3.6 Total Multi-Vehicle Mission-Planning Algorithm
  3.7 Future Extensions
Chapter 4 Evaluation Results

4.1 H Parameter Evaluation

4.2 Travel Distance Constrained Mission-Planning

4.3 Test Condition and Environment

4.4 Evaluation Tests

4.4.1 Solution Quality Test of SA_N

4.4.2 Relative Performance of SA_N

4.4.3 Run-time Performance of SA_N

4.4.4 Run-time Comparison of SA_N and other Heuristics

4.4.5 Cooling Schedule Evaluation

4.4.6 Unconstrained Multi-Vehicle Mission-Planning Evaluation

4.4.7 Travel Distance Constrained Multi-Vehicle Mission-Planning Evaluation

4.4.8 Multi-Vehicle Mission-Planning Statistical Run-time Evaluation

4.4.9 Mission-Planning Tour Length versus Number of Vehicle Evaluation

Chapter 5 Summary

Reference

Appendix A Evaluation-Related Test Case Exhibit
Chapter 1 Introduction

1.1 Background

The application of autonomous vehicles is becoming increasingly important in a variety of problem settings. For tasks such as explosive material clearing, deep-sea operations or missions in hostile environments, an autonomous vehicle is not only preferable, but also sometimes the only feasible choice. For example, the recent well-publicized NASA Mars Pathfinder program successfully employed an autonomous vehicle.

The objective of autonomous vehicle mission planning and management is to use a computer to generate a set of vehicle motion control commands that will guide the vehicle to its mission activity locations, carry out its mission tasks and return safely. For situations with multiple mission activity locations, the mission planner is responsible for generating a route so that the vehicle will achieve the maximum total mission value from performing mission tasks. Researchers have conducted a substantial amount of work for situations requiring single autonomous vehicles.

When a mission task contains more activity locations than that is feasible for a single vehicle, mission planners might use multiple autonomous vehicles. Mission planning then includes an additional responsibility: assigning each vehicle a set of activity locations to maximize the total mission value achieved by performing the mission tasks.

This thesis presents an approach for solving the multi-vehicle mission-planning problem.

1.2 Mission Planning Scenario

Figure 1-1 depicts an example of the mission-planning scenarios that we study. In this case, autonomous vehicles are remotely controlled helicopters. "Home" is located at the lower left corner in the figure. The mission uses multiple autonomous vehicles. Each vehicle starts from the home position, travels to a set of assigned mission activity locations, and finally returns to the home position.
We refer to a mission activity location as a goal-point, which is a location with a value representing the value of the mission activity performed there, e.g., surveillance. Each autonomous vehicle travels from one goal-point to another, starting from the home position. Between each pair of goal-points on its path, an autonomous vehicle travels in a straight-line, if there are no obstacles in the path. If there is an obstacle ("no-fly zone"), the mission planner must modify the vehicle travel path by inserting way-points to guide the vehicle around the obstacle area as indicated by the dash line. The irregular geometric shaded area is a no-fly zone, while the triangle in Figure 1-1 is a way-point.

Since not all mission activities have the same importance, each goal-point has its own mission value, as indicated by the dots of various sizes in Figure 1-1. The larger the dot, the greater the value of the goal-point. Since the total distance a vehicle can travel is constrained by fuel and other considerations, it is possible that the vehicle will not be able to visit all the goal-points.

![Figure 1-1: A Multiple Vehicle Mission Planning Scenario](image.png)

1.3 Hierarchical Mission Planning Architecture

The core of mission planning is an Autonomous System. The functions of an autonomous system can be decomposed into the following planning and decision-making sub-functions: Monitoring, Diagnosis, Plan Generation and Plan Execution. As illustrated in Figure 1-2, whether implemented via hardware or software, these sub-functions contribute a fundamental autonomous planning unit. The research work described in this
thesis is to develop, implement, and evaluate a new class of algorithms to generate mission plans. The implementation of the entire autonomous system unit is not within the scope of this document.

Figure 1-2: A Typical Autonomous System Unit

One way to approach a complex problem is to decompose the problem into many simpler sub-problems that can be solved independently, as shown in Figure 1-3. Each level of the architecture has the same structure.

Figure 1-3: Hierarchical Architecture
By properly coordinating the decomposition to generate solutions for each sub-problem, in the end, we can combine a set of solutions for all the sub-problems into a near-optimal complete solution for the original, more complex problem [17].

Typically, we apply a hierarchical architecture based upon the scope of mission planning. As indicated in Figure 1-4, each level of planning addresses a different spatial resolution of mission planning and management. For instance, the top level represents strategic planning, addressing the broadest scope (e.g., the entire mission) but with a low fidelity (e.g., to determine the order for visiting the goal-points). On the other hand, the lowest level considers the smallest mission scope with the highest fidelity, e.g., trajectory planning, which is to determine the exact path to navigate between adjacent two goal-points. This approach decomposes the original mission-planning problem into smaller problems that can be solved in a timely fashion.

![Hierarchical Decomposition for Single Vehicle Mission Planning](image)

**Figure 1-4: Hierarchical Decomposition for Single Vehicle Mission Planning**

Because of its scalability, the hierarchical architecture is also effective for addressing the multiple-vehicle mission-planning problem. By inserting a new planning level as indicated in Figure 1-5 that addresses overall management of multiple vehicles, the complex multi-vehicle mission-planning problem can make use of existing solutions for single vehicle mission-planning problems. This thesis presents and analyzes an algorithm used in the plan generation function of the top level in Figure 1-5.
The input information for the new planning layer is a set of mission activity locations (or regions) and their relative priorities. The output of the new planning level is an ordered sequence of goal-points for each of the N vehicles. Each sequence is feasible and is determined by minimizing the total distance traveled by the vehicle.

From the implementation perspective, the hierarchical architecture facilitates reusable code. Furthermore, the functional and hierarchical decomposition of the architecture naturally enables multi-threading, enabling the developer to take full advantage of the computing power of multi-processing computers. As indicated in Figure 1-5, within the scope of overall mission planning, this thesis focuses on solutions to goal-point partitioning and tour routing.
1.4 Overview of the Thesis

Chapter 1 introduces background on multi-vehicle mission planning, the basics of mission planning, the objectives of multi-vehicle mission planning, and defines the scope of this thesis.

Chapter 2 first describes some fundamental concepts related to mission planning such as graph theory, the basic Traveling Salesman Problem (TSP), and the basic Vehicle Routing Problem (VRP). Later, it presents and discusses a mathematical formulation of the problem.

Chapter 3 reviews traditional approaches and presents newly developed heuristic algorithms for multi-vehicle mission-planning problem.

Chapter 4 illustrates preliminary data evaluation and analysis on the newly developed heuristic algorithms.

Chapter 5 concludes the thesis and proposes suggestions for future research in multi-vehicle mission planning and management.
Chapter 2 Problem Definition

This chapter first introduces a few fundamental concepts, including elements of graph theory, the vehicle routing problem (VRP), and the traveling salesman problem (TSP). It also presents the problem addressed in this thesis, including a detailed problem description and a mathematical formulation.

2.1 Fundamental Concepts

A graph is defined as a collection of nodes and edges (i.e., pairs of nodes). Two nodes are adjacent if the graph contains an edge between them. If \([i, j]\) is an edge of a graph, we also say that node \(i\) (or node \(j\)) and this edge are adjacent. If each edge is required to have unique orientation, the graph is called directed. Often each edge of a graph has an associated edge length.

A path is a sequence of adjacent nodes and edges. A cycle is a path that starts and ends at the same node. The distance between any two nodes along a path is the sum of the length of the edges the path contains. The shortest path between two nodes is the shortest length path from among all possible paths between these two nodes.

A route, or a tour, is a path that starts at one node, visits all the nodes and returns to the starting node. A subgraph is a graph consisting of nodes and edges that are a subset of another graph. A tree is a connected subgraph containing no cycles. A spanning tree is a tree containing all the nodes of the original undirected graph. A graph is Eulerian if it is possible to traverse the edge set on a path traversing every edge exactly once.

2.2 Vehicle Routing Problem (VRP)

The classic vehicle routing problem seeks the efficient use of a fleet of vehicles to visit customers' locations to pick up and/or deliver products. The problem is to specify the customers to be visited by each vehicle, and in what order, to minimize the total vehicle travel distance subject to a variety of constraints such as vehicle total travel distance [2].
The information required includes the number of vehicles, the customer locations coordinates, and a travel distance between every pair of customer locations. Sometimes the travel distance is the Euclidean distance between the customer coordinates, and the travel cost is proportional to the travel distance. The VRP problem usually imposes constraints on the total travel distance or total travel time of any vehicle [11].

The VRP is a classic problem in operations research. The extensive literature on this topic provides rich resources and references for the study of multi-vehicle mission planning [15], [26].

2.3 Traveling Salesman Problem (TSP)

In the traveling salesman problem (TSP), a salesman must visit a set of cities in an order that minimizes the total travel distance [4]. The TSP can be considered as a special case of the VRP, and it often is a sub-problem in vehicle routing applications [16].

The Prize-Collecting TSP has a "prize" associated with each city and the salesman not only has to decide the order to visit the cities, but also which cities to visit. Due to the reward, the salesman must make a tradeoff between collecting the most prizes and minimizing the travel cost.

The TSP is NP-hard, which implies that no optimization method is likely to solve the TSP in polynomial time. Researchers have devoted much work into solving the TSP. Several articles detail extensive background information on the TSP ([13], [23] and [24]).

The VRP and the TSP are closely related. If a classic VRP contains only one vehicle and no constraint other than the condition to visit each point, then it becomes a traveling salesman problem. Techniques used for solving the VRP can be applied to the TSP. On the other hand, a VRP with m vehicles and no constraints is a version of the TSP problem with m salesman (m-TSP)[3]. For m salesman at the depot, the objective is to construct m routes that visit every node exactly once and that minimize the total distance traveled by all m salesmen. The problem can be formulated as a TSP by duplicating the depot m times. The TSP connects the duplicated depot points to the customers in the same way as original VRP does. After solving the TSP, we recover the solution to the original VRP by aggregating the multiple copies of the depot in the TSP solution. The concept to
break down a VRP into multiple TSPs underlies the algorithm developed for the multi-
vehicle mission-planning problem in this thesis.

2.4 Multiple Vehicle Mission Planning Problem

In the multi-vehicle mission-planning problem, each goal-point represents a
mission activity location, e.g., a location where the vehicle is to perform some task. In
order to represent the relative importance of each mission activity, we assign a numerical
mission value to each goal-point. Mission value is defined as the total mission value,
summed over all goal-points visited. The longer distance a vehicle in a mission travels, the
more fuel it consumes and the greater risk it bears. In this research, we define the mission
travel cost as the total distance traveled by all the vehicles.

The overall multi-vehicle mission-planning problem has two basic objectives: maximizing
total mission value and minimizing travel cost. The two objectives have different priorities, with maximizing mission value the higher priority.

In reality, many factors influence the execution of mission planning including the
quantity of resources available for each vehicle, environmental issues, and the topography of
the mission area. For example, the vehicle travel time can be different depending on the
travel direction between two locations. Moreover, at different mission-planning stages, the
objective can be further defined as follows.

At the pre-mission stage, mission planning is responsible for high-level goal-point
planning, mission task partitioning and routing the grouped goal-points. This planning
stage will assign each vehicle a set of goal-points, attempting to maximize the overall
mission values while minimizing the travel cost.

During the mission, mission planning is responsible for dynamical real-time mission
re-planning, plan refining, and plan coordination to achieve the maximum mission value. There are many uncertainties during a mission’s execution. For example, if a mission
planner receives new information during mission execution, the mission-planning system
will dynamically re-plan the mission vehicle assignments to maintain the maximum
possible mission value. The dynamically altered mission plan will, for example, consider
such factors as changing wind flow patterns for navigation.
At the post-mission stage, mission planning is responsible for learning from the decision-making process. Since the decision-making model is based on a heuristic solution and incomplete data, the ability to learn, for example by updating values of the initial parameters of heuristic tour routing algorithm, will help improve the mission planning system.

To address the multiple-vehicle mission-planning problem analytically and quantitatively, we first describe the problem mathematically as follows. Assume that we are given a set of goal-points, indexed as $i = 2, \ldots, n$ with index $i = 1$ referring to the home location. We have $m-1$ vehicles, indexed as $k = 1, \ldots, m$, with $k = m$ representing a dummy vehicle. Based on certain rules, we will partition these goal-points into $m-1$ groups and assign each group to one vehicle. Possibly, the vehicles cannot or will not travel to all the goal-points. Group $m$ represents a collection of all un-assigned goal-points. By employing a dummy vehicle, we will be able to formulate the problem so that it resembles a classic vehicle routing problem.

Each goal-point $i$ has an assigned mission value $W_i$. We let $CT_{i,j}$ denote the distance from goal points $i$ to $j$, which is the travel cost for a vehicle. Let $Q_k$ denote travel distance capacity for each vehicle $k$. $Y_{ik}$ is a binary decision variable indicating whether vehicle $k$ visits goal-point $i$. $X_{ijk}$ is a binary decision variable indicating whether vehicle $k$ visits goal-point $j$ after visiting goal-point $i$. Figure 2-1 illustrates the multiple vehicle mission planning problem scenario. The next section specifies a mixed integer linear programming formulation of the problem.
2.4.1 Problem Formulation

Let

\[ X_{ijk} = \begin{cases} 
1 & \text{if vehicle } k \text{ visits goal } j \text{ immediately after goal } i, \\
0 & \text{otherwise.} 
\end{cases} \]

\[ Y_{ik} = \begin{cases} 
1 & \text{if vehicle } k \text{ visits goal-point } i, \\
0 & \text{otherwise.} 
\end{cases} \]

The objective function of the problem is

\[
\text{Maximize } H_1 \sum_{i=1}^{n} \left( W_i \sum_{k=1}^{m-1} X_{ik} \right) - H_2 \sum_{i=1}^{n} \sum_{j=1}^{n} \left( C_{ij} \sum_{k=1}^{m-1} X_{ijk} \right) 
\]  \hspace{1cm} (2.1)

In this expression, 
\[ \sum_{i=1}^{n} (W_i \sum_{a=1}^{m} X_{ia}) \] is the total mission value summed over the m-1 vehicles and 
\[ \sum_{i=1}^{n} \sum_{a=1}^{m} (CT_{ia} \sum_{a=1}^{m} X_{ia}) \] is the total travel cost summed over the m-1 vehicles. \( H_1 \) and \( H_2 \) are relative weights for the total mission value and the total travel cost in the mission objective. The model has the following constraints.

Each goal-point is visited only once, by one vehicle, either real or dummy. All m vehicles, including the dummy visit the home location.

\[ \sum_{k} Y_{ik} = \begin{cases} 
1 & \text{for } i = 2, \ldots, n \\
\ m & \text{for } i = 1. 
\end{cases} \]  \hspace{1cm} (2.2)

- For each vehicle except for the dummy, the total distance traveled, which is also defined as the travel cost, must not exceed vehicle travel capacity. Therefore,

\[ \sum_{ij} CT_{ij} \ast X_{ijk} \leq Q_k \hspace{1cm} \text{For } k = 1, \ldots, m - 1. \] \hspace{1cm} (2.3)

- If vehicle \( k \) arrives or departs from node \( i \), then it must visit goal-point \( i \). If a vehicle arrives at any goal-point, it must also leave that goal-point.

\[ \sum_{j} X_{ijk} = Y_{ik} \]

\[ \sum_{j} X_{jik} = Y_{ik} \]

\[ \text{for } i = 1, \ldots, n \]

\[ \text{for } k = 1, \ldots, m. \] \hspace{1cm} (2.4)
No route, including the one for the dummy vehicle, contains a sub-tour. Let $|S|$ denote the cardinality of the set $S$. Using a now standard model originally due to Dantzig, Fulkerson & Johnson ([1] and [4]), we can use the following constraints to eliminate sub-tours:

$$\sum_{i,j \in S} X_{ij} \leq |S| - 1$$

for all $S \subseteq \{2, \ldots, n\}$

for $k = 1, \ldots, m$. \hspace{1cm} (2.5)

$X_{ijk}$ and $Y_{ik}$ are both integer decision variables.

$$Y_{ik} = [0, 1]; \quad i = 1, \ldots, n; \quad k = 1, \ldots, m$$ \hspace{1cm} (2.6)

$$X_{ijk} = [0, 1]; \quad i, j = 1, \ldots, n; \quad k = 1, \ldots, m.$$ \hspace{1cm} (2.7)

$H_1$ and $H_2$ are relative weights for the total mission value and the total travel cost of the mission. The objective function integrates both maximizing the mission value and minimizing the travel cost.

### 2.4.2 Observations on the model

If the vehicles travel everywhere at a common speed, the travel time is proportional to the travel distance. Therefore, the travel cost, $CT_{i,j}$, can be regarded as the distance or travel time between goal-points $i$ and $j$. This is not true if the problem contains no-fly zones as shown in Figure 1-1. However, mission planners do the routing around a no-fly zone at a lower planning level in the hierarchy. Also, winds could affect the travel time but not the distance. We could handle issues at this level by altering the value of the travel cost $CT_{i,j}$ so that it is edge specific and need not to be proportional to the travel distance.

If we need to visit all the goal-points and have no travel constraints, the multi-vehicle mission-planning problem becomes a multi-vehicle TSP or m-TSP problem. For situations with a single vehicle, the original problem is a prize-collecting traveling
salesman problem. So, we might view the mission-planning problem as a prize-collecting m-TSP problem with an added travel distance constraint. The key distinction between the multi-vehicle mission-planning problem and a classic VRP is that mission planning needs to maximize the mission value and choose the goal-points to visit. As in the VRP problem, not only do we need to determine the order in which each vehicle visits a series of the goal-points, but we must also decide the fundamental question, which goal-points are assigned to each vehicle.
Chapter 3 Algorithm Study

3.1 Introduction

This chapter describes various approaches for solving the multi-vehicle mission-planning problem. First, we focus on traditional methods. In particular, we first explore traditional solution methods for the VRP problem, since this problem is so similar to the multi-vehicle mission-planning problem. We examine both optimization-based approaches, which seek an optimal solution, and heuristic approaches, which primarily search for a near optimal solution.

Later in this chapter, we present a heuristic-based algorithm for addressing the multi-vehicle mission-planning problem. The proposed algorithm solves the problem in a two-stage approach. First, it partitions the mission goal-points into a cluster for each vehicle. Secondly, it seeks an optimal route (tour) for each partitioned goal-point group.

During the last several decades, researchers have published hundreds of papers on vehicle routing and scheduling problems. Several researchers have written excellent reviews of work-to-date, including Christofides et al. [8], Magnanti [18], Bodin et al. [2], Golden and Assad [12], and Laporte [15]. Much of the work has focused on the classic Vehicle Routing Problem. Optimization approaches are limited in the size of the problems that they can handle. Consequently, the literature has focused more on heuristic methods.

3.2 Traditional Methods for Solving the VRP

3.2.1 Optimization-Based Approaches

The simplest conceptual approach for solving an NP-hard optimization problem such as VRP is to enumerate all the feasible solutions, evaluate their objective function values, and chose the best solution. It’s theoretically possible to formulate any NP-hard problem as an integer program, often using integer variables (usually of value 0 or 1), and solve them using variants of a method generally known as branch and bound [30]. Branch and bound is an efficient enumeration scheme that avoids the complete enumeration of solutions by building a search tree of the solutions. The search process will prune the tree,
attempts to reduce the number of solutions that needs to be considered and often finds an optimal solution in a reasonable amount of time.

Laporte & Nobert [15] have explored the relationship between the VRP and the multiple TSP problem (m-TSP). They have solved randomly generated capacitated VRPs on super computers using days of computing time. Christofides and et al. [3] embedded a lower bound in a branch and bound scheme and have solved VRPs with up to 25 nodes. Eilon, Watson-Gandy and Christofides [3] implemented dynamic programming for VRPs to develop lower bounds on optimal solutions for 10 to 25 node problems. Balinski and Quandt [8] were among the first to propose a set partitioning formulation for the VRP, in conjunction with a branch-and-bound algorithm. Desrochers, Desrosiers and Soloman [5] have used this approach for VRPs with time windows. Other formulations of the VRP include set covering formulations, vehicle flow based formulations, and commodity flow based formulations.

Optimization-based approaches have proven to be useful for only small-scale problems. Ideally, multi-vehicle mission planning is a real-time decision-making process and requires a quick response time. The planning time frame is expected to be on the order of seconds or single digit minutes. Currently, the optimization-based approaches are not appropriate for the multi-vehicle problem.

3.2.2 Heuristic Algorithms

Although the efficiency of optimization algorithms continues to improve as more powerful algorithms become known and more computing power resources become available, it still takes a long time to optimally solve large VRP problems. To meet the time concerns, researchers have developed a variety of heuristics approaches.

Due to the close relationship between the VRP and TSP problems, researchers have derived many heuristic algorithms for the VRP from the heuristic procedures for the TSP. In addition to algorithms originating from the TSP, researchers have developed other heuristics specifically for VRPs.

The Clarke and Wright algorithm intends to solve capacitated VRPs in which the number of vehicles is not fixed. The method starts with $n$ disjoint sub-tours connecting the
depot to each of the n points. Each iteration merges two routes, choosing the merger that generates the largest saving in distance. This algorithm is known to perform well in practice [15]. Researchers have developed derivatives of the basic method. Golden, Magnanti and Nguyen have used computer science techniques to substantially reduce the running time of the Clarke and Wright algorithm [10].

The Sweep algorithm is commonly attributed to Gillette and Miller [11], but can be traced back to the work of Wren and Holiday [11]. This method is used for travel length constrained or other capacitated VRPs with one or more depots. The sweep procedure starts by representing all locations by polar coordinates. A depot-arched ray sweeps the territory to partition the customer locations into clusters. The method then applies a TSP algorithm to each cluster to optimize the individual vehicle routes.

In addition to these TSP derived VRP algorithms, researchers have proposed other classes of heuristic approaches. The Tabu search algorithm is a good example that has been successfully applied to a number of classical VRP test problems [28]. This method constructs a sequence of possible vehicle routes that might not be feasible. Their degree of feasibility is measured by a penalty in the objective function. The last step of this algorithm uses a post-optimization method to improve the routes. Since Tabu search requires intensive and complicated management and manipulation of computer memory for its implementation, it is not the preferred methodology for solving the multi-vehicle problem.

Two basic heuristic approaches for the VRP problem are “route building” and “cluster first - route second” methods [29]. The sweep heuristic for the Euclidean VRP proposed by Gillett & Miller [29], a sequential approach proposed by Mole & Jameson, and a generalized assignment problem method proposed by Fisher & Jaikumer [9] are all good examples [9] of the cluster first – route second approach.

The clustering step might not be easy for more complex problem. As a result, Bodin and Sexton proposed a swap heuristic, which seeks local improvement by transferring single demand points between tours for a dial-a-ride problem [2]. For the same reason, Cullen and Jarvis proposed a Cluster-Chain-Route heuristic for a precedence-constrained problem [9]. Their algorithm requires users’ interaction to generate feasible
route-segments that are then chained together to form vehicle routes. Multi-vehicle mission planning seeks a high level of autonomy in the execution of mission planning.

In summary, an optimization-based approach appears not to be suitable for multi-vehicle mission planning and a heuristic alternative seems to be the only feasible choice. The *cluster-first, route-second* VRP approach, particularly the sweeping algorithm, seems to provide an alternative approach developing solutions for the multi-vehicle mission-planning problem.

### 3.3 Algorithm Development Strategy

Figure 3-1 illustrates the strategy used to solve the problem, which is to decompose the problem into two independent sub-problems, goal-point partitioning and goal-point routing.

![Diagram of Decomposition Strategy]

We first developed a new heuristic algorithm for goal-point partitioning. The goal-point partitioning considers vehicle travel capacity constraints such as fuel, workload, and battery capacities. It uses mission value as partitioning criteria.

Tradeoffs must be made between mission value and vehicle travel distance, when the tour length of goal-point group exceeds the maximum travel distance of a single vehicle. If the vehicle travel distance constraint is not satisfied for a group of goal-points, higher priority is given to these goal-points with larger mission values. The algorithm checks vehicle travel distance constraint by implementing a simplified shortest path routing algorithm, which is described in the goal-point routing algorithm section of the chapter.
In addition to the top-level goal-point-partitioning, we have further developed a goal-point routing algorithm for each low-level mission-planning unit as indicated in Figure 3-1. The goal-point routing algorithm is based on simulated annealing heuristic and a combination of TSP local modification heuristics, such as 2-switch and edge insertion. By integrating the simulated annealing heuristic, we have developed a complete goal-point routing algorithm to maximize the total mission value for mission planning problem.

3.4 Algorithm for Goal-Point Partitioning

Goal-point partitioning is a key function of the new hierarchical planner. Based on the requirements of the multi-vehicle mission-planning problem, we have designed the new heuristic for the mission partition to be simple and straightforward. It groups goal-points in terms of both their geometric locations and mission values.

The heuristic is based on the sweep concept. The method starts by generating two sorted goal-point lists. The first list contains all the goal-points sorted in increasing polar coordinate angle and the second list contains all the goal-points sorted in decreasing mission value.

The method divides the overall mission goal-points into a number of groups equal to the pre-assigned number of participating vehicles. Each group will have approximately the same total mission value.

We perform the procedure by sweeping in a forward direction or in a backward direction, sorting the goal-points either in an increasing or a decreasing order of the polar coordinate angles.

Beginning with the smallest polar angle, we add the goal-points to the current group according to the polar angle list, as shown in the Figure 3-2. The current group accumulates the indicated goal-point's mission values. Assume there are m participating vehicles. When the total mission value for the group reaches the total goal-point mission value divided by m, the algorithm completes the current group and starts a new group. This process continues until completing all the goal-points. The goal-point partitioning will create m (total vehicle number) sets of goal-point groups.
The next step is checking the vehicle travel distance constraint. At this stage, the mission planner generates a new sorted list for each partitioned group. Within each goal-point group, the mission planner sorts all the goal-points based upon their distance to the home location.

For each group of goal-points, the mission planner uses a fast routing algorithm, which we describe in the next section of this chapter, to estimate the length of a tour. One output of the fast routing algorithm is an upper bound for the length of the ultimately selected tour. If the tour is infeasible, then there are two possible outcomes and the mission planner will handle them as follows.

1) The tour has no goal-point with smaller mission value than the current goal-point the farthest distance from home. In this case, the mission planner will remove the goal-point the farthest from home and continue to check the tour length constraint.

2) The tour has at least one goal-point with smaller mission value than the goal-point farthest from home. In this case, the mission planner will label all such goal-points as a test group, remove the goal-point with the least mission value from the group and check the tour length constraint. The removed goal-point becomes a temporarily unrouted goal-point. All the temporarily removed goal-points will rejoin the tour if the mission planner does not find a feasible solution after removing all the goal-point in the test group.
If a partitioned group contains unrouted goal-points, the routed goal-points do not necessarily represent the optimal solution. To improve the total mission value of each group, we implement a goal-point swapping procedure between the unrouted goal-points and the routed goal-points within the same group. Suppose the unrouted goal-points in the group have a total mission value of $V$. We swap all the unrouted goal-points with one routed goal-point that has the highest mission value smaller than $V$.

The mission planner checks for tour length feasibility after the swap. If the resulting solution is feasible, the mission planner retains the solution. Otherwise, it reverses the swap. The process continues until no goal-point with smaller mission value is available to swap. Figure 3-4 and 3-5 depict the swapping concept.
Figures 3-6 and 3-7 describe details of the goal-point partitioning algorithm.

**Goal-Point Partitioning Algorithm**

Step 1: Sort all the goal-points to form two lists: list-1 in increasing order of polar coordinate angles and list-2 in decreasing order of mission values.

Step 2: Beginning with the goal-point that has the least polar angle,

(2.1) Start a new group by adding goal-points according to their order in list-1, while accumulating the associated mission value. Mark the corresponding goal-point in list-2 with the group number.

(2.2) Once the accumulated mission value reaches the total mission value of all the goal-points divided by the given number of vehicles, start a new group.

(2.3) Repeat steps (2.1) and (2.2) until the process has grouped all the goal-points.

Step 3: Sort all the goal-points in a group in increasing order of their distance from the home location, forming list-3. Use the simplified SA_N algorithm, which we will describe in Figure 3-11 of this chapter, to route a tour and estimate the tour length.

(3.1) Check the tour length constraint. If it is feasible, go to step (4.1).

(3.2) Let U be the mission value of the goal-point the farthest from home. If the tour has no goal-point with mission value smaller than MV, remove the goal-point farthest from home. The removed goal-point becomes unrouted. Go to step (3.1).

(3.3) Group all the goal-points with mission value smaller than U as a test group. Set all these goal-points as active.

(3.4) Select the active goal-point with the smallest mission value from the test group.

(3.5) If there is a tie, select the goal-point farthest from home.
(3.6) Remove the selected goal-point. It becomes inactive. Check for feasibility. If the tour is feasible, mark all inactive goal-points as unrouted and go to step (4.2). Otherwise, continue to next step.

(3.7) If there is goal-point left in the test group, go to step (3.4).

(3.8) Otherwise, restore all the goal-points in test group to the tour. Remove the goal-point farthest from home. The removed goal-point becomes unrouted. Go to step (3.1).

(3.10) Repeat step (3.1) to step (3.10) until all the goal-points are grouped.

Step 4: Mark all routed and unrouted goal-points.

(4.1) If the tour has no removed goal-point, terminate the process and continue to the next group. Go to step (3.1).

(4.2) Let V be the sum of the mission value of the unrouted goal-points. If the tour has no other goal-point with mission value smaller than V, terminate the process and continue to the next group.

(4.3) Otherwise, select the goal-point with the greatest mission value V' smaller than V.

(4.4) If the tour has more than one such goal-point, select the goal-point farthest from home.

(4.5) Swap all unrouted goal-points with the selected goal-point. Check for feasibility.

(4.6) If the resulting solution is feasible, keep the new routed set terminate the process and continue to the next group. Otherwise, reverse the swap.

(4.7) If the tour has any goal-point of mission value V', select the goal-point farthest from home. Go to step (4.9).

(4.8) Otherwise, if the tour has any goal-point of mission value smaller than V', select the goal-point with the greatest mission value.

(4.9) Repeat step (4.4) through (4.8) until completing all the partitioned goal-point groups.

End of Goal-Point Partitioning Algorithm

Figure 3-6: Goal-Point Partitioning Algorithm
Figure 3-7: Travel Distance Constrained Tour Routing
Figure 3-8 illustrates details of the three goal-point lists during goal-point partitioning. Figure 3-9 shows a sample output of the partition process. In this case, there are three groups of goal-points for three vehicles. After the partition, the grouped goal-points are ready for tour routing.

![Diagram of goal-point lists and partitioning process](image)

**Figure 3-8: Three Goal-point lists of Partition Process**

![Sample of partitioned goal-point groups](image)

**Figure 3-9: Sample of Partitioned Goal-point Groups**
3.5 Goal-Point Routing Algorithm

After partitioning the goal-points, we pass the grouped goal-points, together with the unrouted goal-points, to the lower mission planning level for tour routing. The multiple-vehicle mission-planning routing algorithm uses a simulated annealing heuristic and a combination of various local search methods.

3.5.1 Simulated Annealing (SA)

The simulated annealing technique originates from statistical mechanics and was inspired by the physical process of annealing used for cooling solids so that they form perfect crystals. SA is a randomized scheme that reduces the risk of getting trapped in local minima by allowing moves to inferior solutions. Given a neighborhood of a combinatorial minimization problem, the algorithm randomly selects a potential new point \( s' \) from this set. The method accepts solution \( s' \) as a replacement of solution \( s \) if

\[ s' \text{ is better than } s \ (s' < s) \text{ or } \]
\[ s' \text{ is worse than } s \ (s' > s), \text{ but } e^{\frac{s}{T}} > x, \]

\( T \) is a control parameter called the temperature and \( x \in [0,1] \) is a uniform random number. The temperature parameter \( T \) is initially set to a high value, allowing many non-improving moves to be accepted and it is gradually reduced to a value whereby nearly all non-improving moves are rejected. In this way, the algorithm avoids becoming trapped in a local minimum until the final stages of the search procedure when the temperature is very low and the algorithm has already settled into a region with a good solution.

There have been many studies on the convergence properies of SA. Research using the theory of Markov chains has proved that if the temperature is lowered slowly enough, SA will eventually converge to a global minimum [14]. Unfortunately, the same research shows that this choice of temperature change will, in general, require an enormous number of iterations.

Although simulated annealing provide fairly robust heuristic performance, the algorithm requires that several parameters be fairly well tuned in order to deliver efficient performance. We discuss this issue in the following sections.
Cooling Schedule

The simulated annealing process will lower the temperature according to a scheme known as the annealing schedule. A typical annealing cooling schedule includes: (i) an initial starting value of the temperature parameter T, (ii) a cooling rate r, for updating the temperature, and a rule for updating the temperature, (iii) the number of iterations to be performed at each temperature, and (iv) a termination criterion of the algorithm. All these parameters have influences on both speed and accuracy of the simulated annealing process.

Initial Temperature

The algorithm should start with a high enough initial temperature so that it will accept almost all initial moves. In practice, this requires some knowledge of the magnitude of neighboring solutions. Trial runs help to gain this knowledge. Without such knowledge, one might choose what appears to be a large value and run the algorithm for a short time and observe the acceptance rate. If the rate is high enough, the temperature value might be used to start the process. On average, an acceptance rate of between 40% and 60% seems to deliver good results [14]. We have used a collection of TSPLIB cases to obtain some sense of a good starting temperature. Our evaluation results indicate that an initial temperature of 100 is reasonable for vehicle mission-planning purposes.

Cooling Rate and Cooling Rule

We have used an annealing schedule that retains the temperature as a constant for a given number of iterations and then reduces it. We used a geometric reduction function for the rule reducing the temperature from t to r*t, for some constant r < 1. Best performances are reported in the literature [22] for values of r in the range 0.8 < r < 0.99. For our path routing purpose, we set the reduction factor at its norm, 0.9.

Final Temperature

In theory, we should continue the simulated annealing procedure until the final temperature becomes zero, but in practice it is sufficient to terminate when the chance of accepting an uphill move has become negligible. This choice is problem-dependent, and as
in the case of selecting an initial temperature, involves some monitoring of the ratio of acceptance. Through experimental trials on TSP cases, on average, a final temperature of 1 or less demonstrates efficient performance.

➢ Number of Iterations

The total number of iterations depends on the initial temperature, the final temperature, the cooling schedule, and the number of iterations at each temperature. The number of iterations at each temperature should ensure enough statistical opportunity for the neighborhood search to explore local optima. We choose the number of iterations per temperature to be a product of a large constant time the number of goal-points. Through preliminary evaluation, we determine the constant to be 500.

➢ Initial Tour

Traditionally, the SA heuristic constructs its initial tours from scratch randomly. This procedure does not exploit any additional knowledge concerning the problem. With this type of unguided initiation, the method could take significant extra effort before it is on the right track to the global minimum. We apply Prim’s heuristic to construct an initial tour. Our heuristic will first find an Eulerian tour and then create a Hamiltonian tour from it.

Let $G = (V, E)$ ($V$ = a set of $n$ nodes and $E$ = a set of $m$ edges) be a connected graph with edge weights $C_{u,v}$ for all edges $u$ and $v$. A Minimum Spanning Tree (MST) of $G$ is a non-cyclic connected subset $T \subseteq E$ that contains all the nodes and has the minimum possible value of $C(T) = \sum_{(u,v) \in T} C_{u,v}$. A Hamiltonian Tour is a cycle of length $n$ in a graph on $n$ nodes. An Eulerian Tour is a closed path that traverses every edge of a graph exactly once. A spanning tree contains $n-1$ edges.

Let $v_{i0}, v_{i1}, ..., v_{ik}$ be the sequence in which the nodes are visited when traversing an Eulerian tour starting at node $v_{i0}$ and returning to node $v_{ik} = v_{i0}$. We next discuss two methods for constructing a tour for initializing the SA procedure. Figure 3-10 illustrates the procedure to obtain a Hamiltonian tour. For details, see reference [23].
Obtaining Hamiltonian Tour Procedure

Step 1: Set $Q = \{v_{i0}\}, T = 0$, $v = v_{i0}$, and $l = 1$.

Step 2: As $|Q| < n$, perform the following steps.

(2.1) If $v_{il} \notin Q$, then set $Q = Q \cup \{v_{il}\}$, $T = T \cup \{vv_{il}\}$, and $v = v_{il}$

(2.2) Set $l = l + 1$.

Step 3: Set $T = T \cup \{vv_{il}\}$.

Step 4: T is a Hamiltonian tour.

End of Obtaining Hamiltonian Tour Procedure

Figure 3-10: Obtaining a Hamiltonian Tour from an Eulerian Tour

We considered two procedures to create the initial tour.

(I) MST-Based Heuristic - We first compute the MST (minimum spanning tree) using Prim’s algorithm [23]. We then construct a two-node loop by doubling the shortest edge on the MST. The next step is to select the node in the MST that is adjacent to the nodes on the loop and has the shortest edge. We construct a new loop by inserting the new node between two adjacent nodes on the loop. Figure 3-11 illustrates the procedure to obtain an initial tour through MST-based Heuristic.

MST-based Tour Procedure

Step 1: Compute the MST (minimum spanning tree).

Step 2: Construct a two-node loop by doubling the shortest edge on the MST.

Step 3: Select the node that is adjacent to the nodes on the loop and has the shortest edge.

Construct a new loop by inserting the node, placing the selected node between two adjacent nodes on the loop.

Step 4: Obtain a tour by repeating step 3 until completing all the nodes.

End of MST-based Tour Procedure

Figure 3-11: MST-Based Tour Procedure

(II) Christofides Heuristic - Christofides [8] suggested a method to convert a spanning tree into an Eulerian tour – it is sufficient to add a perfect matching on the odd-degree nodes of the tree. A Perfect Matching of a node set $W$, $|W| = 2k$, is a set of $k$ edges so that each node of $W$ is incident to exactly one of these edges. After adding all the edges of this perfect matching, all node degrees are even and hence the graph is Eulerian.
Since a minimum spanning tree is no longer than a shortest Hamiltonian tour and the matching computed in Christofides heuristic has weight at most half of the length of an optimal tour, the Christofides heuristic produces a tour at 1.5 times but takes longer time ([6] and [7]). By using the MST-based heuristic to construct an initial tour, the SA method is likely to provide a good starting point for the local search procedure.

3.5.2 Local Search Structure

The neighborhood structure plays an important role in determining the efficiency of the SA method. After reviewing the local search literature, we constructed a structure based on some basic local search algorithms while maintaining both efficiency and effectiveness.

- **2-Switch** - is a neighborhood search procedure that obtains a solution from the current solution by replacing two edges, reversing one of the paths and reconnecting the tour as shown in Figure 3-12. P0 and P1 are two starting points on the tour. Nodes a and b are adjacent and so are nodes c and d. Note that we imposed a direction on the tour to be able to decide which pair of new edges to add to form a new tour.

![Figure 3-12: 2-Switch Local Search](image)

Figure 3-13 gives a description of the 2-switch procedure.
2-Switch Procedure

Step 1: Randomly select two nodes P0 and P1 on the tour.

Step 2: As shown in Figure 3-12, set
\[
\begin{align*}
a &= \text{node P0-1}; \\
b &= \text{node P0}; \\
c &= \text{node P1}; \\
d &= \text{node P1+1}; \\
e &= \text{node P2}; \\
f &= \text{node P1+1};
\end{align*}
\]

Step 3: Disconnect node b from node a and node d from node c.

Connect node a to node c and node b to node d to obtain a tour.

End of 2-Switch Procedure

Figure 3-13: 2-Switch Procedure

- **Edge Insertion** - Edge insertion removes an edge from the tour, reinserting it at another place in the tour between nodes that are connected by another edge in the tour. Note that at every insertion point there are two possibilities for connecting the eliminated edge obtained by switching its endnotes. Figure 3-14 and 3-15 describe the edge insertion pictorially and procedurally.

![Figure 3-14: Edge Insertion](image)

Edge Insertion Procedure

Step 1: Randomly select three nodes P0, P1 and P2 on the tour.

Step 2: As shown in Figure 3-14, set
\[
\begin{align*}
a &= \text{node P0-1}; \\
b &= \text{node P0}; \\
c &= \text{node P1}; \\
d &= \text{node P1+1}; \\
e &= \text{node P2}; \\
f &= \text{node P1+1};
\end{align*}
\]

Step 3: Disconnect node b from node a, node d from node c and node f from node e.

Connect node d to node a, node b to node e and node f to node c to obtain a new tour.

End of Edge Insertion Procedure

Figure 3-15: Edge Insertion Procedure
3.5.3 Fast Local Search

Local search procedures seek tour improvement by exchanging individual edges in the current tour and for a combination of two, three, or more other edges from the tour. A faster implementation [27] partitions the neighborhood into sub-neighborhood so that, for a problem with N goal-points, there are N sub-neighborhoods. Initially, all sub-neighborhoods are active. If the sub-neighborhood contains no improvement, then the sub-neighborhood becomes inactive. If no sub-neighbored contains an improvement, then the procedure terminates.

We incorporate the Fast Search idea into the solution for the multiple-vehicle mission-planning problem by randomly choosing nodes on a tour for edge insertion and 2-switch modification algorithm.

3.5.4 Tour Routing

Based on the SA procedure and a combination of neighborhood search improvements, we have developed a heuristic for mission-planning routing. This algorithm inherits globally optimal heuristics from a classic SA and integrates various local search procedure such as 2-switch and edge insertion, as well as a minimum spanning tree based initial tour described in Figure 3-11. Since the tour routing algorithm is a combination of both SA and various local neighborhood improvement procedures, we refer to it as SA_N. Figure 3-16 describes the procedure.

SA_N Algorithm

Step 1: Compute an initial tour T using the MST-based heuristic. Choose an initial
temperature $t = 100$ and a temperature reduction factor $r = 0.9$.

Step 2: Perform the following steps for total N goal-points.

(2.1) At each temperature, iterate the following steps 500 * N times:

(2.1.1) With probability of 50%, perform either a 2-switch procedure, as shown in Figure 3-12, or an Edge Insertion procedure, as shown in Figure 3-14, on the current tour. Let $T'$ be the tour created. Let

$C(T)$ be the total length of the tour. Compute $\Delta T = C(T') - C(T)$.

(2.1.2) Compute a uniform random variable $X$, $X \in [0, 1]$.

(2.1.3) If $\Delta T < 0$ or $X < \exp (-\Delta T / t)$, accept the new tour and set $T = T'$.
(2.1.4) If the total improvement exceeds 60*N, continue to step (2.2); otherwise go to step (2.1.1) until completing all 500*N modifications.

(2.2) Update the temperature to become \( t = r \cdot t \).

Step 3: Repeat Step 2 until reaching the final temperature, then output the tour.

End of SA_N Algorithm

Figure 3-16: SA_N Algorithm

Figure 3-17 illustrates some details of the algorithm. Figure 3-18 indicates the probability of randomly selecting each local modification. We assume the tour has N goal-points. The letters a, b, c, d, e, f are representative goal-points. \( D(a, b) \) is the tour length from goal-point a to goal-point b. \( \Delta T \) is the difference in total tour length.
Figure 3-17: SA_N Local Search Block Diagram
The step (2.1.1) includes two categories of major local modifications, 2-switch and edge insertion. The particulars of the software implementation are not within the scope of the thesis.

The most appealing advantages of SA methods are that they are simple to implement ([19] and [25]) and able to be implemented to run quickly. By applying various cooling parameters, we adjust the SA's process time. As mentioned in Session 3.4, we implement a simplified SA_N method for goal-point partitioning. With a shorter cooling schedule and fewer iterations, the simplified SA_N requires much less time to estimate a tour's length. Figure 3-19 illustrates the simplified SA_N algorithm.

**Simplified SA_N Algorithm**

Step 1: Compute an initial tour T using the MST-based heuristic. Choose an initial temperature $t = 100$ and a temperature reduction factor $r = 0.6$.

Step 2: Perform the following steps for N goal-points.

(2.1) At each temperature, iteration the following steps 100*N times:

(2.1.1) With probability of 50%, perform either a 2-switch procedure, as shown in Figure 3-12, or an Edge Insertion procedure, as shown in Figure 3-14, on the current tour. Let $T'$ be the tour created. Let $C(T)$ be the total length of the tour. Compute $\Delta T = C(T') - C(T)$.

(2.1.2) Compute a uniform random variable $X$, $X \in [0, 1]$.

(2.1.3) If $\Delta T < 0$ or $X < \exp(-\Delta T / t)$, accept the new tour and set $T = T'$.

(2.1.4) If total improvement exceeds 10*N, continue to step (2.2); otherwise go to step (2.1.1) until completing all 100*N modifications

(2.2) Update the temperature to become $t = r*t$.

Step 3: Repeat Step 2 until reaching the final temperature, then output the tour.

End of Simplified SA_N Algorithm

Figure 3-19: Simplified SA_N Algorithm
The SA_N algorithm does not account for the vehicle travel length constraint and it does not account for the mission value of the goal-points. We have developed a complete goal-point routing algorithm accounts for both of these problem ingredients. Since the simplified SA_N generates a coarse estimation of tour length, it is necessary to modify the goal-point list in the tour routing process. The modification process uses the same procedure in step 3 and step 4 described in Figures 3-6 and 3-7 of the goal-point partitioning procedure. Figure 3-20 describes a complete goal-point routing algorithm.

Complete Goal-Point Routing Algorithm

Step 1: Sort all the goal-points in a group in increasing order of their distance from the home location and form list-3.

Step 2: Apply SA_N on goal-points with a "routed" tag within a group.

Step 3: Sort all the goal-points in a group in increasing order of their distance from the home location, forming list-3. Use the simplified SA_N algorithm, which we will describe in Figure 3-11 of this chapter, to route a tour and estimate the tour length.

(3.1) Check the tour length constraint. If it is feasible, go to step (4.1).

(3.2) Let \( U \) be the mission value of the goal-point the farthest from home. If the tour has no goal-point with mission value smaller than \( MV \), remove the goal-point farthest from home. The removed goal-point becomes unrouted. Go to step (3.1).

(3.3) Group all the goal-points with mission value smaller than \( U \) as a test group. Set all these goal-points as active.

(3.4) Select the active goal-point with the smallest mission value from the test group.

(3.5) If there is a tie, select the goal-point farthest from home.

(3.6) Remove the selected goal-point. It becomes inactive. Check for feasibility. If the tour is feasible, mark all inactive goal-points as unrouted and go to step (4.2). Otherwise, continue to next step.

(3.7) If there is goal-point left in the test group, go to step (3.4).

(3.8) Otherwise, restore all the goal-points in test group to the tour. Remove the goal-point farthest from home. The removed goal-point becomes unrouted. Go to step (3.1).

(3.10) Repeat step (3.1) to step (3.10) until all the goal-points are grouped.

Step 4: Mark all routed and unrouted goal-points.

(4.1) If the tour has no removed goal-point, terminate the process and continue to the next group. Go to step (3.1).
(4.2) Let V be the sum of the mission value of the unrouted goal-points. If the tour has no other goal-point with mission value smaller than V, terminate the process and continue to the next group.

(4.3) Otherwise, select the goal-point with the greatest mission value V' smaller than V.

(4.4) If the tour has more than one such goal-point, select the goal-point farthest from home.

(4.5) Swap all unrouted goal-points with the selected goal-point. Check for feasibility.

(4.6) If the resulting solution is feasible, keep the new routed set terminate the process and continue to the next group. Otherwise, reverse the swap.

(4.7) If the tour has any goal-point of mission value V', select the goal-point farthest from home. Go to step (4.9).

(4.8) Otherwise, if the tour has any goal-point of mission value smaller than V', select the goal-point with the greatest mission value.

(4.9) Repeat step (4.4) through (4.8) until completing all the partitioned goal-point groups.

Step 5: 

(5.1) If the tour has no unrouted goal-points, terminate the process. Otherwise, continue.

(5.2) Beginning with the unrouted goal-point closest to home, add the unrouted goal-point to the tour. Check for feasibility. If the resulting solution is feasible, keep the tour and continue to the next unrouted goal-point the farthest from home. The process continues until no unrouted goal-point exists or the resulting solution is infeasible.

(5.3) If the resulting tour is infeasible, remove the added goal-point and terminate the process.

End of Complete Goal-Point Routing Algorithm

Figure 3-20: Complete Goal-Point Routing Algorithm
3.6 Total Multi-Vehicle Mission-Planning Algorithm

We present the total solution for multi-vehicle mission-planning problem in Figure 3-21, which incorporates both the goal-point partitioning algorithm as described in Figure 3-6 and the complete goal-point routing algorithm as described in Figure 3-20.

**Total Multi-Vehicle Mission Planning Algorithm**

Step 1: Initialize mission-planning environment including the vehicle travel distance capacity, goal-points, goal-point values, and total number of vehicles assigned.

Step 2: Apply the goal-point partitioning algorithm as in Figure 3-21 to obtain partitioned goal-point groups as in Figure 3-9.

Step 3: Apply the complete goal-point routing algorithm as in Figure 3-20 to each group of goal-points.

**End of Total Multi-Vehicle Mission Planning Algorithm**

**Figure 3-21: Total Multi-Vehicle Mission-Planning Algorithm**

The total multi-vehicle mission-planning solution reflects our algorithm design strategy as described in Figure 3-1 and fits well into the multi-vehicle mission-planning hierarchical architecture, as described in Figure 1-5. Figure 3-22 depicts the total solution in a diagram.

![Diagram](image.png)

**Figure 3-22: Total Multi-Vehicle Mission-Planning Algorithm Diagram**
Figure 3-23 plots a flow chart of the total multi-vehicle mission-planning algorithm.
3.7 Future Extensions

To further improve the total multi-vehicle mission-planning solution, we could exchange goal-points between partitioned groups after completing routing all the groups. This operation requires a longer run-time, which might not be feasible in practice. Figure 3-24 describes a conceptual algorithm for this procedure.

Figure 3-24 is a heuristic-based improvement algorithm. The exchanging of goal-points could be implemented randomly among the routed and unrouted goal-points across various groups.

![Diagram of Neighborhood Exchanging Improvement (3 vehicles)](image)

Figure 3-24: Neighborhood Exchanging Improvement (3 vehicles)
Chapter 4: Evaluation Results

This chapter presents results from computational studies to evaluate the heuristic SA_N for the multi-vehicle mission-planning problem.

We first discuss the parameters \( H_1 \) and \( H_2 \) in the objective function. Subsequently, we describe the test data sets and test results. We used two different test data sets: (i) examples chosen from a traveling salesman problem library, and (ii) some randomly generated problems. We used two measures of performance for the algorithm, CPU computation time and deviation from optimality. We compared the algorithm with existing algorithms. The results indicate that the new heuristic performs well.

4.1 H Parameter Evaluation

For the original objective function, \( H_1 \) and \( H_2 \) are the relative weights for the mission value and travel cost. In the real problem, the first priority is to maximize the total mission value; minimizing the travel cost is secondary. Therefore, when the vehicle travel distance constraint is binding, the goal is to visit a set of goal-points with the total highest mission value. If travel distance is not a binding constraint so that the mission can visit all the goal-points, then the objective is to find the tour with the lowest total travel cost.

The analysis of the \( H_1 \) and \( H_2 \) parameters focuses on finding the appropriate value so that the solution will meet these objectives. By scaling the objective function to simplify our discussion, we set \( H_1 \) to 1 without loss of generality.

To illustrate the relationship between \( H_2 \) and the goals described above, we introduce and discuss a simple example. We implemented the model in a Microsoft Excel spreadsheet on a Window platform, using the Excel solver to obtain the solution. The example uses one vehicle.
Example 4-1

The example contains seven goal-points whose X-Y locations and values are listed in Figure 4-1. Since the example contains only one vehicle, the variable $X_{ijk}$ becomes $X_{ij}$. Similarly, $Y_{ik}$ becomes $Y_i$. We choose the travel cost $C_{ij}$ between goal-point $i$ and $j$ to be the Euclidean distance $D_{ij}$ between goal-point $i$ and $j$. The objective function for the integer programming, as in Equation 4-2, is the summation of each individual goal values times the decision variable $Y_i$, and the product of $H_2$ and the travel cost for the goal-points on the tour.

$$\text{Maximize } \sum_{i=1}^{n} (W_i Y_i) - H_2 \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} X_{ij}$$

(4.1)

The most challenging task in creating and solving the integer programming is to establish sub-tour-eliminating equations. For seven goal-points, the model contains a total of 126 equations. We used the Excel solver to obtain the solution, setting the solver tolerance to 1% and the convergence limit to 1e-6. See the Microsoft Excel manual for the definitions of these parameters.

<table>
<thead>
<tr>
<th>Goal-Points</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Position</td>
<td>0</td>
<td>40</td>
<td>40</td>
<td>0</td>
<td>80</td>
<td>86</td>
<td>40</td>
</tr>
<tr>
<td>Vertical Position</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>40</td>
<td>0</td>
<td>46</td>
<td>80</td>
</tr>
<tr>
<td>Reward</td>
<td>1000</td>
<td>900</td>
<td>600</td>
<td>800</td>
<td>400</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 4-1: Goal-Points X-Y Location and Mission Value

Figure 4-2 shows a plot of all seven goal-points. The size of a circle centered at each goal-point represents its mission value.
Situation 1: $H_2 = 0$.

The original objective function becomes:

$$
\text{Maximize } \sum_{i=1}^{n} (W_i Y_i).
$$

Figure 4-4 graphically specifies the integer programming solution provided by the Excel solver. All the $Y_i$ variables are of value 1, indicating that the tour visits all the goal-points. The objective function maximizes only the mission value, and does not consider the travel cost. Note that the total distance traveled by the vehicle is not the minimum possible. The variables $X_12 = X_{25} = X_{56} = X_{67} = X_{74} = X_{43} = X_3 = 1$ and all the other $X$ variables are at value 0.

<table>
<thead>
<tr>
<th>Mission Value</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_2$</td>
<td>0</td>
</tr>
<tr>
<td>Distance</td>
<td>330</td>
</tr>
<tr>
<td>Objective Value</td>
<td>4000</td>
</tr>
</tbody>
</table>

Figure: 4-3: Value of the Objective Function at Optimality when $H_2 = 0$

Since the weight of the travel cost is zero, or $H_2 = 0$, as shown in Figure 4-3, the objective function is the total mission value. The mission planning objective, therefore, is maximizing the total mission value regardless of the travel cost, or distance.
Situation 2: \( H_2 = 1 \).

In this case, the objective function has two components, mission value and travel distance. The objective function not only maximizes the mission value, but also minimizes the travel cost. Figure 4-6 depicts the integer solution provided by the Excel solver. Note that all 7 goal-points are selected. The total distance traveled is not only less than that in the previous case, but is actually the minimum among all tours that visit all the goal-points.

<table>
<thead>
<tr>
<th>Mission Value</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_2 )</td>
<td>1</td>
</tr>
<tr>
<td>Distance</td>
<td>297</td>
</tr>
<tr>
<td>Objective Value</td>
<td>3703</td>
</tr>
</tbody>
</table>

Figure: 4-5: Objective Function When \( H_2 = 1 \)

Situation 3: \( H_2 = 6 \).
In this case, the relative weight given to the distance component of the objective function is so high that the optimal solution does not visit all the goal-points even though it is feasible for the vehicle to do so. Figures 4-7, 4-8 and 4-9 specify the situation.

<table>
<thead>
<tr>
<th>Xij</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Sum</th>
<th>Yi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4-7: Decision Variables Under** $H_2 = 6$

<table>
<thead>
<tr>
<th>Mission Value</th>
<th>3900</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_2$</td>
<td>6</td>
</tr>
<tr>
<td>Distance</td>
<td>240</td>
</tr>
<tr>
<td>Objective Value</td>
<td>2460</td>
</tr>
</tbody>
</table>

**Figure: 4-8: Objective Function When** $H_2 = 6$

Since maximizing the total mission value is always the first priority, placing too much weight in the objective function on the travel cost would lead to a solution that doesn’t meet the real problem’s goals.

**Figure 4-9: Shortest Path When** $H_2 = 6$

As shown by Example 4-1, when $H_2 / H_1$ is positive, but substantially small enough, the optimal solution will maximize the mission value, and among all such
solutions minimizes the travel cost. When \( H_2 / H_1 \) is large enough, the optimal solution will not visit all the goal points, since the gained mission value of visiting a goal-point would not justify the travel cost to the location. Choosing the correct parameter values is essential to defining a right objective function.

Figure 4-10 depicts the optimal objective value as a function of the parameter \( H_2 \) in Example 4-1, where \( H_1 = 1 \). Each straight line represents the value of the objective function for a certain set of goal-points. For example, the line with lowest \( H_2 \) intercept is the objective function for all 7 goal-points. The line with second lowest \( H_2 \) intercept is the objective function for 6 of the goal-points. The combination of these lines constructs all possible options of goal-points. The upper envelop of all the linear functions for various \( H_2 \) values forms a frontier line for the \( H_2 \) parameter. Thus the \( H_2 \) frontier is the ultimate relationship between \( H_2 \) and the objective function. For the vehicle mission-planning problem, \( H_2 \) should be a small value between zero and the first non-zero turning point as indicated on the plot.

![Figure 4-10: H and Objective Function](image-url)
4.2 Travel Distance Constrained Mission Planning

In this section, we present a case study of the vehicle routing problem a single vehicle that cannot possibly visit all the goal points because of a travel distance constraint. With a limitation of the travel distance, the vehicle is unable to visit all the assigned goal-points. In this case, the mission planner must decide which goal-points to visit so that the mission objective maximizes the overall accumulated mission value.

The vehicle routing problem with a limiting travel length constraint is a variant of the classic vehicle routing problem. Instead of trying to route all the goal-points with the shortest tour length, the mission planner must make a more important decision concerning which goal-points to visit. The challenge is that there are many possible goal-points combinations. With all the related parameters defined in Section 4.1, the mission-planning objective function for a single vehicle becomes,

\[
\text{Maximize } \sum_{i=1}^{n} (W_i Y_i) - H_2 \sum_{i=1}^{n} \sum_{j=1}^{n} (D_{ij} X_{ij}).
\]

(4.3)

In particular, the vehicle must also satisfy its total travel distance constraint,

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} X_{ij} \leq Q.
\]

(4.4)

As in Section 4-1, we set \(H_1 = 1\). We also set \(H_2 = 1\). As we saw in Example 4-1, doing so ensures that maximizing the mission value is the first priority and minimizing travel distance has the second priority in the mission-planning objective function. The optimal solution to a travel length constrained vehicle routing problem is the feasible solution with the maximum mission value.

Depending on various values of the travel length \(Q\), the mission-planning problem has various optimal solutions. In order to examine the possible situations for a travel length constrained mission vehicle, we present a computational example.
Example 4-2

We use the same data set as in Example 4-1. The mission goal-points are again listed in Figure 4-11. One vehicle is available for the mission.

<table>
<thead>
<tr>
<th>Goal-Points</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Position</td>
<td>0</td>
<td>40</td>
<td>40</td>
<td>0</td>
<td>80</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>Vertical Position</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>40</td>
<td>0</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>Reward</td>
<td>1000</td>
<td>900</td>
<td>600</td>
<td>800</td>
<td>400</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 4-11: Goal-points Locations and Their Mission Value

We use the same integer programming setup as in Example 4-1. In addition, we constrain total travel distance, \( \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} X_{ij} \leq Q \). Using the integer program, we compute the optimal solution to the mission-planning problem with various values of \( Q \).

Situation 1: \( Q = 300 \).

In this case, the vehicle length constraint is greater than the shortest tour of all 7 goal-points. Therefore, the vehicle is able to visit all 7 mission goal-points. The result is the same as in Figure 4-6.

Situation 2: \( Q = 240 \).

In this case, the vehicle is not able to visit all 7 goal-points due to its limited travel length. With total travel length of 240, the mission planning system has several feasible solutions. Figure 4-12 lists three such situations. Figure 4-13 lists the objective values for these three situations, with the situation (a) being the optimal solution.
Situation 3: $Q = 160$.

In this case, the vehicle is able to visit fewer goal-points due to its more limited travel distance capacity. With total travel length of 160, the mission-planning problem still has several feasible solutions. Figure 4-14 lists three such situations and Figure 4-15 shows their objective values. Situation (a) in Figure 4-14 is the optimal solution.
As we have seen, the optimal solution to the mission-planning problem could vary as a function of various values of travel length limit $Q$. In practice, we face another challenge due to the possible large number of mission goal-points and resulting large number of combinations of feasible solutions for problem with constrained vehicle travel length. We will present further computational experiments in a later section of this chapter as we explore possible general solution strategies.
4.3 Test Condition and Environment

We used two evaluation criteria to evaluate the SA_N algorithm: solution quality and run-time. We selected some of the test data sets from a public library TSPLIB of TSP problems. Researchers have already optimally solved most of the problems included in TSPLIB and these problems are widely used in the TSP literature. TSPLIB is electronically distributed and is available at the web site http://softlib.rice.edu/softlib/tsplib.

The test conditions are summarized as follows.
(1) We performed the tests on a 150MHz Pentium PC with 144-MB of memory and equipped with the Windows 95 operating system. No other application programs were running when we conducted the tests.
(2) We implemented all test algorithms in ANSI C using the WATCOM Integrated Development Environment (IDE) and 32-bit environment as the target platform.
(3) All CPU times are given in seconds.
(4) The test data sets for were taken from the TSPLIB or were randomly generated.
(5) The TSPLIB data file contained three columns, corresponding to goal-point names, and the x and y coordinates of the goal-points.
(6) We computed distances $D_{i,j}$ using the Euclidean distances between the goal points.
(7) We computed an average over several runs of the same problem to remove some possible random noise in the test runs.
(8) In most cases, we chose the value of each goal-point to be the same.

4.4 Evaluation Tests

We measured the solution quality as the percentage deviation from optimality, also defined as the excess:

$$Excess = \frac{solution\_cost \ - \ optimal\_cost}{optimal\_cost}.$$  \hspace{1cm} (4.5)
4.4.1 Solution Quality Test of SA_N

This test evaluated the performance of SA_N using a group of small and medium sized TSPLIB cases. In the name used to refer to any TSPLIB case, the numerical data specifies the total goal-points. For example, “att48” has 48 goal-points. The number of goal-points in the test group varies from 48 to 318.

In Figure 4-16, the first column is a list of TSPLIB test cases that we used, sorted alphabetically. The second column specifies the optimal tour length for each case. The third column is the tour length generated by SA_N, and the last column is the relative deviation from optimality, or the excess. The relative performance of the SA_N algorithm is quite good on these TSPLIB test cases.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Best</th>
<th>SA_N</th>
<th>Excess</th>
</tr>
</thead>
<tbody>
<tr>
<td>att48</td>
<td>10628</td>
<td>10640</td>
<td>0.00</td>
</tr>
<tr>
<td>berlin52</td>
<td>7526</td>
<td>7526</td>
<td>0.00</td>
</tr>
<tr>
<td>ch130</td>
<td>6100</td>
<td>6100</td>
<td>0.00</td>
</tr>
<tr>
<td>d198</td>
<td>15780</td>
<td>16923</td>
<td>0.07</td>
</tr>
<tr>
<td>eil51</td>
<td>426</td>
<td>426</td>
<td>0.00</td>
</tr>
<tr>
<td>eil76</td>
<td>538</td>
<td>538</td>
<td>0.00</td>
</tr>
<tr>
<td>kroa100</td>
<td>21282</td>
<td>21642</td>
<td>0.02</td>
</tr>
<tr>
<td>krob100</td>
<td>22141</td>
<td>22327</td>
<td>0.01</td>
</tr>
<tr>
<td>kroc100</td>
<td>20749</td>
<td>21118</td>
<td>0.02</td>
</tr>
<tr>
<td>krod100</td>
<td>21294</td>
<td>21294</td>
<td>0.00</td>
</tr>
<tr>
<td>lin105</td>
<td>14379</td>
<td>14379</td>
<td>0.00</td>
</tr>
<tr>
<td>lin318</td>
<td>42039</td>
<td>42039</td>
<td>0.00</td>
</tr>
<tr>
<td>pr76</td>
<td>108159</td>
<td>109631</td>
<td>0.01</td>
</tr>
<tr>
<td>rd100</td>
<td>7910</td>
<td>7914</td>
<td>0.00</td>
</tr>
<tr>
<td>st70</td>
<td>675</td>
<td>675</td>
<td>0.00</td>
</tr>
<tr>
<td>u159</td>
<td>42080</td>
<td>42800</td>
<td>0.02</td>
</tr>
<tr>
<td>ulyss22</td>
<td>7013</td>
<td>7013</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 4-16: Excess Test Results of SA_N

Figure 4-17 provides a graphical representation of the excess column of Figure 4-16. The y-coordinate represents the excess and the x-coordinate represents TSPLIB test cases. The average excess for these cases is 0.02 overall.
4.4.2 Relative Performance of SA_N

Figure 4-18 provides a comparison of the SA_N algorithm on the TSPLIB test cases with other heuristic algorithms: the Nearest Neighborhood algorithm with a random starting point, the Fast Saving algorithm, a Greedy algorithm, a combination of 2-opt and Nearest Neighborhood, and a combination of 3-opt and Nearest Neighborhood. We have obtained the reported computational performance of the other algorithms from the literature [23]. For detailed descriptions of the methods, please see reference [23].

<table>
<thead>
<tr>
<th>Problem</th>
<th>Near N.</th>
<th>F. Saving</th>
<th>Greedy</th>
<th>2opt/NN</th>
<th>3opt/NN</th>
<th>SA_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>att48</td>
<td>14.22</td>
<td>5.64</td>
<td>6.01</td>
<td>3.42</td>
<td>1.97</td>
<td>0.11</td>
</tr>
<tr>
<td>berlin52</td>
<td>12.74</td>
<td>3.54</td>
<td>4.53</td>
<td>2.74</td>
<td>1.24</td>
<td>0.09</td>
</tr>
<tr>
<td>ch130</td>
<td>11.94</td>
<td>5.21</td>
<td>5.79</td>
<td>2.45</td>
<td>2.21</td>
<td>0.07</td>
</tr>
<tr>
<td>d198</td>
<td>15.51</td>
<td>6.96</td>
<td>6.69</td>
<td>3.18</td>
<td>5.27</td>
<td>7.13</td>
</tr>
<tr>
<td>lin318</td>
<td>22.52</td>
<td>8.34</td>
<td>14.13</td>
<td>5.94</td>
<td>2.80</td>
<td>0.07</td>
</tr>
<tr>
<td>fl417</td>
<td>17.84</td>
<td>13.74</td>
<td>9.48</td>
<td>7.25</td>
<td>3.68</td>
<td>4.33</td>
</tr>
<tr>
<td>pcb442</td>
<td>22.91</td>
<td>10.20</td>
<td>14.62</td>
<td>7.82</td>
<td>2.66</td>
<td>3.21</td>
</tr>
<tr>
<td>u574</td>
<td>22.11</td>
<td>12.36</td>
<td>10.20</td>
<td>7.02</td>
<td>3.98</td>
<td>2.83</td>
</tr>
<tr>
<td>p654</td>
<td>18.75</td>
<td>10.66</td>
<td>18.82</td>
<td>12.37</td>
<td>3.80</td>
<td>3.01</td>
</tr>
<tr>
<td>rat783</td>
<td>24.76</td>
<td>9.88</td>
<td>10.04</td>
<td>8.39</td>
<td>3.47</td>
<td>2.27</td>
</tr>
<tr>
<td>pr1002</td>
<td>25.18</td>
<td>10.24</td>
<td>12.96</td>
<td>8.48</td>
<td>3.61</td>
<td>3.42</td>
</tr>
</tbody>
</table>

Figure 4-18: Excess (x100) % Results of SA_N Compared with other Heuristics

As indicated by Figure 4-18, the SA_N performance compares quite favorably with the other heuristics. Figure 4-19 depicts the results in Figure 4-18 as a chart.
4.4.3 Run-time Performance of SA_N

We have evaluated the run-time of the SA_N algorithm using test data randomly sampled from test case “fnl4461” of TSPLIB (see Appendix A). For each test set with goal-points ranging from 10 to 50, we obtained ten random samples from the test case “fnl4461”. Figure 4-20 depicts the average computational time versus the number of goal-points. We conducted the test to see how the run-time of the SA_N algorithm varied as a function of the number of goal-points.
Figure 4-21 shows the run-time of the cases in Figure 4-18 as a function of the number of goal-points. We found that within a range of 200 goal-points, the run-time increases linearly with goal-points.

![Figure 4-21: Run-time comparison of cases in Figure 4-18](image)

4.4.4 Run-time Comparison of SA_N and other Heuristics

We have also conducted evaluation tests to compare four different simulated annealing algorithms: (i) SA_N; (ii) SA with 2-switch local search and the minimum-spanning-tree-based initial tour; (iii) SA with edge insertion and the minimum-spanning-tree-based initial tour, and (iv) SA with combined local search with a random starting tour.

Figure 4-22 reports the computational time for the four heuristics on a representative list of TSP instances. Figure 4-23 reports on the excesses for these same test cases. As these results indicate, the algorithm SA_N performs quite well on these test problems.

As indicated by these experiments, the selection of a good initial tour (the MST heuristic instead of a random initial tour) has a significant impact on the algorithm’s performance, measuring by its run-time and the excess of the final solution. The implementation with either a 2-switch or edge insertion neighborhood performed similarly and the implementation using both, as in SA_N, performed much better.
Figure 4-22: Computing Time of various Heuristic Algorithms

Figure 4-23: Excess Test of various Heuristic Algorithms
4.4.5 Cooling Schedule Evaluation

We have experimented with various starting and ending temperatures for the cooling schedules to determine an optimal setting for general mission-planning applications. Figure 4-24 and Figure 4-25 illustrate two sample temperature schedules generated by SA_N for the first 100 points from test case "fnl4461". We have identified that a starting temperature of 100 and a final temperature of 1 are generally good settings for mission goal-point routing.

![Figure 4-24: Cooling Schedule starts at 500](image)

![Figure 4-25: Cooling Schedule starts at 100](image)
4.4.6 Unconstrained Total Multi-Vehicle Mission-Planning Evaluation

We performed an evaluation of the total multi-vehicle mission-planning algorithm described in Figure 3-21, based on five data sets drawn randomly from the TSP problem “fnl4461”. For these examples, the number of goal-points ranged from 10 to 50. Example 4-3 presents the computational experiments and evaluation results.

Example 4-3

To evaluate the total multi-vehicle mission-planning algorithm for the multi-vehicle mission-planning problem, we first generated five groups of test data. These data are randomly sampled from the TSPLIB test case “fnl4461” and are plotted in Figure 4-26.

![Figure 4-26: Test Data for Mission-Planning](image)

We used three vehicles in the evaluation. Therefore, the mission planner will partition these goal-points into three groups and route these grouped goal-points with the SA_N algorithm. In this experiment, we did not impose any vehicle travel distance constraint, since we were interested in assessing the run-time of overall mission planning. We assigned the same mission value to all goal-points.

We begin the goal-point-partitioning process with the goal-point that has the least polar angle. Each partitioned group contains about one-third the total mission value. After grouping the goal-points, the mission partitioning process does not check for the vehicle
travel distance constraint, since we are only interested in an unconstrained travel distance in this example.

Figure 4-27 depicts the process flow chart.

![Unconstrained Mission-Planning Flow-Chart (3 vehicles)](image)

After partitioning the goal-points, we apply the SA_N routing algorithm on three grouped goal-points. We denote these three groups as T-1, T-2 and T-3. We conduct the routing process sequentially. The overall mission-planning process terminates after we have completed all the group routing.

Based on the run-time output data generated by the program, we reprint four bars in Figure 4-28, representing the partitioning run-time and the three routing run-times. The five data sets represent the five goal-point test groups, with total number of goal-points ranging from 10 to 50.

Figure 4-28 indicates that the run-times for goal-point partitioning is essentially the same for all five-test groups, while the routing run-time increases with the number of goal points.
Figure 4-28: Mission-Planning Run-time Evaluation (3 vehicles)

Figure 4-29 presents the total accumulated run-time for each test group versus the number of goal-points.

Figure 4-29: Total Mission-Planning Run-time Evaluation (3 vehicles)
4.4.7 Length Constrained Mission-Planning Evaluation

Since mission planning includes both constrained and unconstrained mission-planning problems [13], in this section we perform an evaluation of constrained multi-vehicle mission-planning evaluation. Example 4-4 describes the problem setting including the test environment, test data, test procedures, and test results.

Example 4-4

In order to conduct a length constrained multi-vehicle mission-planning test, we first generated a set of test data with 50 mission goal-points based on a more realistic mission-planning scenario than we have been discussed before. We assigned each goal-point with a mission value ranging from 40 to 120, in increments of 20. We set the home position to be (6731, 4437). Figure 4-30 lists the 50 goal-points, their locations, and their mission values, as well as the home location. Each goal-point was indexed from 1 to 50. We index the home location as location 51.

<table>
<thead>
<tr>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Value</th>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8235</td>
<td>6700</td>
<td>40</td>
<td>31</td>
<td>6182</td>
<td>6908</td>
<td>80</td>
</tr>
<tr>
<td>2</td>
<td>8356</td>
<td>6325</td>
<td>40</td>
<td>32</td>
<td>6034</td>
<td>6687</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>8424</td>
<td>6588</td>
<td>40</td>
<td>33</td>
<td>6324</td>
<td>6678</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>8245</td>
<td>6448</td>
<td>40</td>
<td>34</td>
<td>6345</td>
<td>6667</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>8358</td>
<td>6917</td>
<td>40</td>
<td>35</td>
<td>6054</td>
<td>6933</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>7660</td>
<td>6200</td>
<td>60</td>
<td>36</td>
<td>6196</td>
<td>6522</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>7938</td>
<td>6153</td>
<td>60</td>
<td>37</td>
<td>6047</td>
<td>6522</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>7751</td>
<td>6029</td>
<td>60</td>
<td>38</td>
<td>5586</td>
<td>6010</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>7580</td>
<td>6129</td>
<td>60</td>
<td>39</td>
<td>5918</td>
<td>6021</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>7357</td>
<td>6278</td>
<td>60</td>
<td>40</td>
<td>5946</td>
<td>6189</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>7897</td>
<td>6000</td>
<td>60</td>
<td>41</td>
<td>5445</td>
<td>6079</td>
<td>40</td>
</tr>
<tr>
<td>12</td>
<td>7514</td>
<td>6563</td>
<td>80</td>
<td>42</td>
<td>5568</td>
<td>6210</td>
<td>40</td>
</tr>
<tr>
<td>13</td>
<td>7932</td>
<td>6769</td>
<td>80</td>
<td>43</td>
<td>5842</td>
<td>6276</td>
<td>40</td>
</tr>
<tr>
<td>14</td>
<td>7736</td>
<td>6810</td>
<td>80</td>
<td>44</td>
<td>5747</td>
<td>5900</td>
<td>40</td>
</tr>
<tr>
<td>15</td>
<td>7497</td>
<td>6790</td>
<td>80</td>
<td>45</td>
<td>5076</td>
<td>6700</td>
<td>120</td>
</tr>
<tr>
<td>16</td>
<td>7961</td>
<td>6950</td>
<td>80</td>
<td>46</td>
<td>5340</td>
<td>6930</td>
<td>120</td>
</tr>
<tr>
<td>17</td>
<td>7676</td>
<td>6950</td>
<td>80</td>
<td>47</td>
<td>5500</td>
<td>6670</td>
<td>120</td>
</tr>
<tr>
<td>18</td>
<td>7622</td>
<td>6654</td>
<td>80</td>
<td>48</td>
<td>5038</td>
<td>6892</td>
<td>120</td>
</tr>
<tr>
<td>19</td>
<td>6800</td>
<td>6933</td>
<td>60</td>
<td>49</td>
<td>5568</td>
<td>6859</td>
<td>120</td>
</tr>
<tr>
<td>20</td>
<td>7165</td>
<td>6695</td>
<td>60</td>
<td>50</td>
<td>5310</td>
<td>6588</td>
<td>120</td>
</tr>
<tr>
<td>21</td>
<td>6824</td>
<td>6777</td>
<td>60</td>
<td>51</td>
<td>6731</td>
<td>4437</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>6896</td>
<td>6588</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>7142</td>
<td>6974</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>6669</td>
<td>6925</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>6665</td>
<td>6079</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>6651</td>
<td>6251</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>7005</td>
<td>8112</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>6845</td>
<td>6300</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>6901</td>
<td>5900</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4-30: Test Data Table for Constrained Mission-Planning Evaluation
Figure 4-31 depicts the test data set. Different sizes of dots represent various associated mission values. Goal-points within a region have the same mission values. We label all the goal-points with their indexing numbers as in Figure 4-30. Each region represents an area of interest identified by a higher level in the mission planning system.

![Figure 4-31: Test Data Set for Travel Length Constrained Mission-Planning Evaluation](image)

We assign three identical vehicles to the mission-planning task. Each vehicle has the same travel distance capacity. After setting up the test environment, we first ran the program with an unconstrained travel distance capacity to obtain a sense of the requirements needed to visit all the goal-points. Consequently, we reduced the vehicle travel distance capacity in stages, running the total multi-vehicle mission-planning algorithm, as described in Figure 3-21, each time. We implemented the entire mission-planning process as described in Figure 3-21.

Next, we present some detailed results of the evaluation.
Figure 4-32 illustrates the results from the goal-point partitioning; Figure 4-33 depicts the corresponding graph. Note that we have duplicated the home location three times so as to assign each partitioned group with one home location.

<table>
<thead>
<tr>
<th>Group 1</th>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Value</th>
<th>Polar</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6358</td>
<td>6325</td>
<td>40</td>
<td>0.860</td>
<td>2492</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>8424</td>
<td>6588</td>
<td>40</td>
<td>0.904</td>
<td>2737</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>8245</td>
<td>6448</td>
<td>40</td>
<td>0.925</td>
<td>2517</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>8358</td>
<td>6917</td>
<td>40</td>
<td>0.990</td>
<td>2965</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>7751</td>
<td>6059</td>
<td>60</td>
<td>1.001</td>
<td>1891</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>7977</td>
<td>6276</td>
<td>60</td>
<td>1.062</td>
<td>2106</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>7932</td>
<td>6749</td>
<td>60</td>
<td>1.095</td>
<td>2623</td>
<td>34</td>
</tr>
<tr>
<td>9</td>
<td>7890</td>
<td>6120</td>
<td>60</td>
<td>1.104</td>
<td>1885</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>7961</td>
<td>6950</td>
<td>80</td>
<td>1.116</td>
<td>2798</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>7600</td>
<td>6200</td>
<td>60</td>
<td>1.134</td>
<td>2056</td>
<td>31</td>
</tr>
<tr>
<td>12</td>
<td>7736</td>
<td>6810</td>
<td>80</td>
<td>1.170</td>
<td>2577</td>
<td>36</td>
</tr>
<tr>
<td>13</td>
<td>7622</td>
<td>6654</td>
<td>60</td>
<td>1.169</td>
<td>2389</td>
<td>35</td>
</tr>
<tr>
<td>14</td>
<td>7676</td>
<td>6850</td>
<td>80</td>
<td>1.211</td>
<td>2685</td>
<td>32</td>
</tr>
<tr>
<td>15</td>
<td>7497</td>
<td>6790</td>
<td>80</td>
<td>1.256</td>
<td>2475</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>7185</td>
<td>6695</td>
<td>60</td>
<td>1.372</td>
<td>2303</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>7005</td>
<td>6112</td>
<td>100</td>
<td>1.409</td>
<td>1697</td>
<td>51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group 2</th>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Value</th>
<th>Polar</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.410</td>
<td>2566</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.453</td>
<td>2516</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.455</td>
<td>2513</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.531</td>
<td>2342</td>
</tr>
<tr>
<td>6</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.543</td>
<td>2493</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.585</td>
<td>2489</td>
</tr>
<tr>
<td>8</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.611</td>
<td>1643</td>
</tr>
<tr>
<td>9</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.670</td>
<td>1823</td>
</tr>
<tr>
<td>10</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.728</td>
<td>2460</td>
</tr>
<tr>
<td>11</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.743</td>
<td>1618</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.750</td>
<td>2278</td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.789</td>
<td>2531</td>
</tr>
<tr>
<td>14</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.822</td>
<td>2153</td>
</tr>
<tr>
<td>15</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.836</td>
<td>2556</td>
</tr>
<tr>
<td>16</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.871</td>
<td>2355</td>
</tr>
<tr>
<td>17</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.871</td>
<td>2355</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group 3</th>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Value</th>
<th>Polar</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.860</td>
<td>2492</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.453</td>
<td>2516</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.455</td>
<td>2513</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.531</td>
<td>2342</td>
</tr>
<tr>
<td>6</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.543</td>
<td>2493</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.585</td>
<td>2489</td>
</tr>
<tr>
<td>8</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.611</td>
<td>1643</td>
</tr>
<tr>
<td>9</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.670</td>
<td>1823</td>
</tr>
<tr>
<td>10</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.728</td>
<td>2460</td>
</tr>
<tr>
<td>11</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.743</td>
<td>1618</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.750</td>
<td>2278</td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.789</td>
<td>2531</td>
</tr>
<tr>
<td>14</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.822</td>
<td>2153</td>
</tr>
<tr>
<td>15</td>
<td>37</td>
<td>6247</td>
<td>6132</td>
<td>60</td>
<td>1.836</td>
<td>2556</td>
</tr>
</tbody>
</table>

Figure 4-32: Goal-Point Partition Table (3 Groups)

Figure 4-33: Goal-Point Partition Graph (3 Groups)
Figures 4-34 and 4-35 show the resulting tours in table and graph form respectively.

### Figure 4-34: Unconstrained Goal-Point Routing Table (3 Groups)

<table>
<thead>
<tr>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Reward</th>
<th>Polar</th>
<th>Distance</th>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Reward</th>
<th>Polar</th>
<th>Distance</th>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Reward</th>
<th>Polar</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>7897</td>
<td>6000</td>
<td>60</td>
<td>0.930</td>
<td>1950</td>
<td>23</td>
<td>7142</td>
<td>6974</td>
<td>60</td>
<td>1.410</td>
<td>2570</td>
<td>37</td>
<td>6247</td>
<td>6322</td>
<td>80</td>
<td>1.888</td>
<td>2194</td>
</tr>
<tr>
<td>8</td>
<td>7751</td>
<td>6029</td>
<td>60</td>
<td>1.001</td>
<td>1891</td>
<td>22</td>
<td>6986</td>
<td>6588</td>
<td>65</td>
<td>1.453</td>
<td>2166</td>
<td>40</td>
<td>5946</td>
<td>8169</td>
<td>40</td>
<td>1.936</td>
<td>1902</td>
</tr>
<tr>
<td>7</td>
<td>7898</td>
<td>6153</td>
<td>60</td>
<td>0.958</td>
<td>2042</td>
<td>29</td>
<td>6901</td>
<td>5905</td>
<td>100</td>
<td>1.453</td>
<td>1473</td>
<td>49</td>
<td>5568</td>
<td>6599</td>
<td>120</td>
<td>2.018</td>
<td>2087</td>
</tr>
<tr>
<td>10</td>
<td>7752</td>
<td>6726</td>
<td>60</td>
<td>1.062</td>
<td>2126</td>
<td>28</td>
<td>6845</td>
<td>6350</td>
<td>100</td>
<td>1.510</td>
<td>1866</td>
<td>43</td>
<td>5842</td>
<td>6578</td>
<td>40</td>
<td>2.021</td>
<td>2043</td>
</tr>
<tr>
<td>6</td>
<td>7600</td>
<td>6390</td>
<td>60</td>
<td>1.134</td>
<td>2056</td>
<td>21</td>
<td>6824</td>
<td>6777</td>
<td>60</td>
<td>1.531</td>
<td>2342</td>
<td>39</td>
<td>5918</td>
<td>6351</td>
<td>40</td>
<td>2.045</td>
<td>1780</td>
</tr>
<tr>
<td>9</td>
<td>7580</td>
<td>6120</td>
<td>60</td>
<td>1.104</td>
<td>1885</td>
<td>19</td>
<td>6800</td>
<td>6833</td>
<td>60</td>
<td>1.543</td>
<td>2497</td>
<td>47</td>
<td>5500</td>
<td>6670</td>
<td>120</td>
<td>2.275</td>
<td>2550</td>
</tr>
<tr>
<td>2</td>
<td>8358</td>
<td>6325</td>
<td>40</td>
<td>0.860</td>
<td>2432</td>
<td>24</td>
<td>6669</td>
<td>6425</td>
<td>60</td>
<td>1.596</td>
<td>2849</td>
<td>40</td>
<td>5340</td>
<td>6930</td>
<td>120</td>
<td>2.080</td>
<td>2855</td>
</tr>
<tr>
<td>17</td>
<td>7006</td>
<td>6112</td>
<td>100</td>
<td>1.409</td>
<td>1697</td>
<td>25</td>
<td>6665</td>
<td>6279</td>
<td>100</td>
<td>1.611</td>
<td>1843</td>
<td>47</td>
<td>5588</td>
<td>6910</td>
<td>40</td>
<td>2.151</td>
<td>2120</td>
</tr>
<tr>
<td>20</td>
<td>7185</td>
<td>6685</td>
<td>60</td>
<td>1.372</td>
<td>2303</td>
<td>26</td>
<td>6551</td>
<td>6261</td>
<td>100</td>
<td>1.670</td>
<td>1823</td>
<td>50</td>
<td>5310</td>
<td>6468</td>
<td>120</td>
<td>2.155</td>
<td>2578</td>
</tr>
<tr>
<td>12</td>
<td>7514</td>
<td>6663</td>
<td>80</td>
<td>1.218</td>
<td>2296</td>
<td>34</td>
<td>6345</td>
<td>6867</td>
<td>80</td>
<td>1.728</td>
<td>2460</td>
<td>44</td>
<td>5747</td>
<td>6950</td>
<td>40</td>
<td>2.163</td>
<td>1753</td>
</tr>
<tr>
<td>18</td>
<td>7622</td>
<td>6654</td>
<td>80</td>
<td>1.169</td>
<td>2369</td>
<td>30</td>
<td>6447</td>
<td>6920</td>
<td>100</td>
<td>1.743</td>
<td>1858</td>
<td>48</td>
<td>5038</td>
<td>6922</td>
<td>120</td>
<td>2.175</td>
<td>2992</td>
</tr>
<tr>
<td>15</td>
<td>7497</td>
<td>6790</td>
<td>80</td>
<td>1.256</td>
<td>2475</td>
<td>33</td>
<td>6324</td>
<td>6678</td>
<td>80</td>
<td>1.750</td>
<td>2278</td>
<td>38</td>
<td>5586</td>
<td>6610</td>
<td>40</td>
<td>2.200</td>
<td>1946</td>
</tr>
<tr>
<td>17</td>
<td>7676</td>
<td>6950</td>
<td>80</td>
<td>1.211</td>
<td>2685</td>
<td>31</td>
<td>6182</td>
<td>6926</td>
<td>80</td>
<td>1.789</td>
<td>2531</td>
<td>45</td>
<td>5078</td>
<td>6720</td>
<td>120</td>
<td>2.202</td>
<td>2654</td>
</tr>
<tr>
<td>14</td>
<td>7736</td>
<td>6817</td>
<td>80</td>
<td>1.170</td>
<td>2577</td>
<td>36</td>
<td>6156</td>
<td>6552</td>
<td>80</td>
<td>1.822</td>
<td>2153</td>
<td>41</td>
<td>5445</td>
<td>6797</td>
<td>40</td>
<td>2.225</td>
<td>2968</td>
</tr>
<tr>
<td>13</td>
<td>7691</td>
<td>6765</td>
<td>80</td>
<td>1.095</td>
<td>2623</td>
<td>35</td>
<td>6054</td>
<td>6339</td>
<td>80</td>
<td>1.836</td>
<td>2586</td>
<td>51</td>
<td>6731</td>
<td>4437</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>7961</td>
<td>6950</td>
<td>80</td>
<td>1.116</td>
<td>2798</td>
<td>32</td>
<td>6034</td>
<td>6687</td>
<td>80</td>
<td>1.871</td>
<td>2355</td>
<td>1080</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8356</td>
<td>6917</td>
<td>40</td>
<td>0.985</td>
<td>2988</td>
<td>51</td>
<td>6731</td>
<td>4437</td>
<td>0</td>
<td></td>
<td></td>
<td>6910</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8424</td>
<td>6588</td>
<td>40</td>
<td>0.904</td>
<td>2737</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1280</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8245</td>
<td>6448</td>
<td>40</td>
<td>0.925</td>
<td>2217</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>6731</td>
<td>4437</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mission Value: 1280
Tour Length: 8255

### Figure 4-35: Unconstrained Goal-Point Routing Graph (3 Groups)

- **Capacity:** 8255
- **Tour Length:** 8255 (Group 1), 7104 (Group 2), 6910 (Group 3)
Next, we used the length of the shortest tour (6910 of vehicle 3) as the distance constraint for the other two vehicles. Figures 4-36 and 4-37 show the results.

Figure 4-36: Constrained Goal-Point Routing Table (3 Groups)

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>27</td>
<td>7055</td>
<td>6112</td>
</tr>
<tr>
<td>8</td>
<td>7360</td>
<td>6120</td>
</tr>
<tr>
<td>5</td>
<td>7231</td>
<td>6029</td>
</tr>
<tr>
<td>11</td>
<td>7897</td>
<td>6000</td>
</tr>
<tr>
<td>7</td>
<td>7833</td>
<td>6153</td>
</tr>
<tr>
<td>5</td>
<td>7600</td>
<td>6020</td>
</tr>
<tr>
<td>12</td>
<td>7757</td>
<td>6776</td>
</tr>
<tr>
<td>12</td>
<td>7614</td>
<td>6583</td>
</tr>
<tr>
<td>20</td>
<td>7185</td>
<td>6695</td>
</tr>
<tr>
<td>12</td>
<td>7822</td>
<td>6654</td>
</tr>
<tr>
<td>15</td>
<td>7497</td>
<td>6790</td>
</tr>
<tr>
<td>2</td>
<td>8358</td>
<td>6325</td>
</tr>
<tr>
<td>14</td>
<td>7236</td>
<td>6810</td>
</tr>
<tr>
<td>13</td>
<td>7902</td>
<td>6769</td>
</tr>
<tr>
<td>5</td>
<td>8358</td>
<td>6917</td>
</tr>
</tbody>
</table>

Mission Value: 1080
Tour Length: 6910

Figure 4-37: Constrained Goal-Point Routing Graph (3 Groups)
The next four sets of tables show the results from vehicle travel distance capacities of 90% (6219), 80% (5520), 60% (4140) and 55% (3800) of 6910 respectively.

Figure 4-38: Multi-Vehicle Mission Planning For Travel Capacity of 6129 (90% of 6910)

Figure 4-39: Multi-Vehicle Routing Graph For Travel Capacity of 6129 (90% of 6910)
Figure 4-40: Multi-Vehicle Mission Planning For Travel Capacity of 5520 (80% of 6910)

Figure 4-41: Multi-Vehicle Routing Graph For Travel Capacity of 5520 (80% of 6910)
Figure 4-42: Multi-Vehicle Mission Planning For Travel Capacity of 4146 (60% of 6910)

Figure 4-43: Multi-Vehicle Routing Graph For Travel Capacity of 4146 (60% of 6910)
Figures 4-44 and 4-45 illustrate resulting constrained goal-point routing with length 3800.

<table>
<thead>
<tr>
<th>Title: Limited Vehicle Travel Capacity Tour Routing (Capacity = 3800)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group 1</strong></td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>27</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>51</td>
</tr>
</tbody>
</table>

Mission Value: 100
Tour Length: 3526

Figure 4-44: Multi-Vehicle Mission Planning For Travel Capacity of 3800 (55% of 6910)

![Image of multi-vehicle mission planning graph with travel capacity of 3800](image)

Figure 4-45: Multi-Vehicle Routing Graph For Travel Capacity of 3800 (55% of 6910)

Capacity: 3800
Tour Length: 3526 (Group 1)
2946 (Group 2)
3526 (Group 3)

Page 76
Figures 4-46 and 4-47 illustrate the relationship between vehicle travel capacity and total goal-points captured by all three vehicles. The travel length is defined as the maximum possible travel distance for any single mission vehicle.

<table>
<thead>
<tr>
<th>Per Vehicle Capacity</th>
<th>Total Mission Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8255</td>
<td>3640</td>
</tr>
<tr>
<td>6910</td>
<td>3220</td>
</tr>
<tr>
<td>6219</td>
<td>2200</td>
</tr>
<tr>
<td>5520</td>
<td>1620</td>
</tr>
<tr>
<td>4146</td>
<td>620</td>
</tr>
<tr>
<td>3800</td>
<td>240</td>
</tr>
</tbody>
</table>

Figure 4-46: Total Mission Value versus Per Vehicle Travel Capacity versus (3 Groups)

Figure 4-47: Total Mission Value versus Vehicle Travel Capacity Chart (3 Groups)
4.4.8 Mission-Planning Statistical Run-time Evaluation

In this section, we perform a statistical evaluation of the run-time of the total multi-vehicle mission-planning algorithm. We use randomly generated goal-points in the evaluation.

We generated five groups of random data with the number of goal-points ranging from 10 to 50. Each group contains 10 sets of test data. The test data are evenly distributed over a 100 by 100 square area, with the lower left corner as the home location.

We conducted test runs on each set of test data by first partitioning the goal-points into three groups and then routing the three goal-points into tours. We used a travel distance constraint in this test run. Figure 4-48 shows the mean, standard deviations, high and low values of the run-time for each group.

<table>
<thead>
<tr>
<th>Goal-points</th>
<th>Total Run-time for Mission-Planning (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>10  20  30  40  50</td>
</tr>
<tr>
<td>Test 2</td>
<td>5   15  27  39  55</td>
</tr>
<tr>
<td>Test 3</td>
<td>7   20  35  45  56</td>
</tr>
<tr>
<td>Test 4</td>
<td>9   20  34  45  56</td>
</tr>
<tr>
<td>Test 5</td>
<td>8   19  34  48  62</td>
</tr>
<tr>
<td>Test 6</td>
<td>8   21  33  50  61</td>
</tr>
<tr>
<td>Test 7</td>
<td>10  19  33  50  58</td>
</tr>
<tr>
<td>Test 8</td>
<td>8   19  34  46  55</td>
</tr>
<tr>
<td>Test 9</td>
<td>5   19  34  45  57</td>
</tr>
<tr>
<td>Test 10</td>
<td>6   19  38  43  64</td>
</tr>
<tr>
<td>Mean</td>
<td>7.30 19.20 33.60 45.50 58.30</td>
</tr>
<tr>
<td>Std</td>
<td>1.55 1.60 2.56 3.14 2.77</td>
</tr>
<tr>
<td>High</td>
<td>10  21  38  50  64</td>
</tr>
<tr>
<td>Low</td>
<td>5   15  27  39  55</td>
</tr>
</tbody>
</table>

Figure 4-48: Mission-planning Run-time Statistics (3 vehicles)

Figure 4-49 provides a graphical representation of the distribution the run-times. Note that the standard deviation increases with the total number of goal-points and the mean increases faster than does the standard deviation.
Figure 4-49: Mission-planning Statistical Run-time (3 vehicles)
4.4.9 Total Tour Length versus Number of Vehicles Evaluation

We conducted another evaluation to investigate the relationship between total tour length of all vehicles and total number of vehicles assigned. The relationship provides possible managerial insight regarding mission planning and operations.

To conduct the evaluation test, we used a sample test case “lin318” from TSPLIB as the test data. Figure 4-50 is a graph of the “lin318” data set locations. It contains 318 total goal-points. Figure 4-51 and Figure 4-52 illustrate the test results.

As indicated by Figure 4-51, the increase in the total routing tour length (summed over all vehicles) is significant when the mission assigns fewer than three vehicles. For more than three vehicles, the increase in distance is relatively minor, while total run-time increases considerably. This means that it is not worthwhile to use more than three vehicles in this case since doing so will not decrease total travel cost, or distance.
Figure 4–51: Total Tour Length versus Numbers of Vehicles

Figure 4–52: Total Routing Length versus Various Numbers of Vehicles
Chapter 5

Summary

We have investigated the multi-vehicle mission-planning problem. We have discussed various aspects of the problem including the problem specification and formulation, the problem analysis, and development of a solution procedure.

In this thesis, we contributed to the following areas.

1) The hierarchically structured approach to goal-point partitioning and goal-point routing results in a feasible and practical solution to the multi-vehicle mission-planning problem. By decomposing the problem hierarchically into a partitioning problem and multiple TSPs, we have constructed a scalable solution to the original complicated mission-planning problem.

2) We presented and developed a goal-point-partitioning algorithm for multi-vehicle mission-planning problem.

3) We developed a heuristic algorithm based on simulated annealing for solving the TSP problem. Computational experiments and evaluations based on both TSPLIB cases and randomly generated test data demonstrate that the SA_N algorithm has a very good performance.

4) By varying the parameters of the SA_N algorithm, we are able to implement the algorithm for various purposes, both routing final goal-points and quickly estimating tour lengths.
References


Appendix A

Evaluation-Related Test Case Exhibit

All test instances are plotted here for reference. These graphs are not drawn on the same scale.