Air Force Institute of Technology
AFIT Scholar

Theses and Dissertations | Student Graduate Works

3-2000

A Hybrid Jump Search and Tabu Search Metaheuristic for the Unmanned Aerial Vehicle (UAV) Routing Problem

Gary W. Kinney Jr.

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Operational Research Commons

Recommended Citation

Kinney, Gary W. Jr., "A Hybrid Jump Search and Tabu Search Metaheuristic for the Unmanned Aerial Vehicle (UAV) Routing Problem" (2000). *Theses and Dissertations*. 4816.
https://scholar.afit.edu/etd/4816

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.

A HYBRID JUMP SEARCH AND TABU SEARCH METAHEURISTIC FOR THE UNMANNED AERIAL VEHICLE (UAV) ROUTING PROBLEM

THESIS

Gary W. Kinney Jr., Captain, USAF

AFIT/GOA/ENS/00M-5

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

20000613 071

DTIC QUALITY INSPECTED 4

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2000	Master's Thesis

4. TITLE AND SUBTITLE

A HYBRID JUMP SEARCH AND TABU SEARCH METAHEURISTIC FOR THE UNMANNED AERIAL VEHICLE (UAV) ROUTING PROBLEM

5. FUNDING NUMBERS

6. AUTHOR(S)

Gary W. Kinney Jr., Captain, USAF

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOA/ENS/00M-05

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Mark O'Hair, Lt Col, USAF
UAV Battlelab mark.ohair@eglin.af.mil
1003 Nomad Way, Suite 107 Comm: (850) 882-5940 x208
Eglin AFB, FL 32542-6867 DSN: 872-5940 x208

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Advisor: Maj Raymond R. Hill, ENS, DSN: 785-3636, ext. 4327

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

ABSTRACT (*Maximum 200 Words*)

In this research, we provide a new meta-heuristic, a jump search / tabu search hybrid, for addressing the vehicle routing problem with real-life constraints. A tour construction heuristic creates candidate solutions or jump points for the problem. A tabu search algorithm uses these jump points as starting points for a guided local search. We provide statistical analysis on the performance of our algorithm and compare it to other published algorithms. Our algorithm provides solutions within 10% of the best known solutions to benchmark problems and does so in a fraction of the time required by competing algorithms. The timeliness of the solution is vitally import to the unmanned aerial vehicle (UAV) routing problem. UAVs provide the lion's share of reconnaissance support for the US military. This reconnaissance mission requires the UAVs to visit hundreds of target areas in a rapidly changing combat environment. Air vehicle operators (AVOs) must prepare a viable mission plan for the UAVs while contending with such real-life constraints as time windows, target priorities, multiple depots, heterogeneous vehicle fleet, and pop-up threats. Our algorithm provides the AVOs with the tools to perform their mission quickly and efficiently.

14. SUBJECT TERMS

Air Force Research, Operations Research, Optimization, Combinatorial Analysis, Algorithms,
Remotely Piloted Vehicles, Surveillance Drones, Multiple Depots, Time Windows (Jump Search, Tabu Search, Vehicle Routing Problem, Java, Heuristics, Traveling Salesman Problem).

15. NUMBER OF PAGES

69

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the US Government.

AFIT/GOA/ENS/00M-5

A HYBRID JUMP SEARCH AND TABU SEARCH METAHEURISTIC FOR THE UNMANNED AERIAL VEHICLE (UAV) ROUTING PROBLEM

THESIS

Presented to the Faculty of the Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operational Analysis

Gary W. Kinney Jr.,
Captain, USAF

March 2000

Approved for public release; distribution unlimited

AFIT/GOA/ENS/00M-5

A HYBRID JUMP SEARCH AND TABU SEARCH METAHEURISTIC FOR THE UNMANNED AERIAL VEHICLE (UAV) ROUTING PROBLEM

Gary W. Kinney Jr.,
Captain, USAF

Approved:

Raymond R. Hill — 1 Mar 00

Maj Raymond R. Hill (Chairman) — date

James T. Moore — 1 Mar 00

Dr James T. Moore (Member) — date

Acknowledgements

I would like to take the time to thank several people who helped me to complete this work. Thanks to Lt Col (ret) Glenn Bailey, my initial thesis advisor before going off to greener pastures, for choosing me for the mission. Thanks to Lt Col Mark O'Hair, my thesis sponsor, for providing customer focus, the necessary information, and the funds to go get it. Thanks to Dr. James Moore, my responsive reader, for undertaking the grueling task of reading and editing this work. Thanks to Capt Kevin O'Rourke for laying down the groundwork.

I would like to give special thanks to Maj Ray Hill, my thesis advisor, for providing direction while allowing me to choose my own way, and for his advice and assistance in all aspects of this undertaking. I would also like to give special thanks to Lt Robert Harder, the Java guru and my wingman, whose help was crucial to the completion of this effort and for coming along for the ride.

Finally, I would like to thank my friends and classmates for listening to me babble on about this project when they had projects of their own to contend with and for providing a much-appreciated distraction when I needed it. Most of all, I would like to thank my family for providing the support I needed over the last year and the 32 years before.

Gary W. Kinney Jr.

Table of Contents

Chapter 1. Background and Statement of the Problem 1

1.1 Background 1
1.2 Problem Statement 2
1.3 Scope and Contribution 3
1.4 Report Overview 4

Chapter 2. Literature Review 5

2.1 Vehicle Routing and Traveling Salesman Problems 5
2.2 Heuristic Approaches 9
2.2.1 Tour Construction and Tour Improvement Algorithms 9
2.2.2 Solomon's Insertion Heuristic 11
2.2.3 k-opt Improvement Algorithms 13
2.2.4 Tabu Search 13
2.2.5 Jump Search 15
2.3 Earlier UAV Routing Efforts 16
2.4 Analysis of Heuristics 17
2.5 Conclusion 18

Chapter 3. Methodology 20

3.1 Solution Representation 20
3.2 Tour Construction Heuristic 21
3.2.1 Parameter Settings 22
3.2.2 Tour Initialization 22
3.2.3 Customer Insertion 23
3.2.4 Completed Solutions 23
3.3 Local Search Heuristics 24
3.3.1 First Best Local Search 24
3.3.2 Global Best Local Search 27
3.3.3 Global Best Tabu Search 29
3.3.4 Reactive Tabu Search 31
3.4 The Jump Search Algorithm 32
3.5 Conclusion 33

Chapter 4. Results 34

4.1 Description of Solomon Instances 34
4.2 Tour Construction Parameters 35
4.3 Tabu List Length 37
4.4 Jump Points 38
4.5 Heuristic Analysis 41
4.6 Comparison to O'Rourke's Reactive Tabu Search Algorithm 43
4.7 Comparison to Best Known Solutions for Solomon's MVRPTW Instances 48
4.8 Multiple Depot Problems 50
4.9 Conclusions 52

Chapter 5. Recommendations for Further Research 53

5.1 Modeling UAV Realism 53
5.2 Tour Construction 53
5.3 Search Techniques 54
5.4 Extensions to VRP 54

Bibliography 55

List of Tables

Table 1 – Parameter Value Goodness-of-fit Test 36
Table 2 – Tabu Length Test 37
Table 3 – Average Best Jump Point Index 38
Table 4 – Jump Point Thresholds for JFB 39
Table 5 – Jump Point Thresholds for JGB 39
Table 6 – Jump Point Thresholds for JTS 40
Table 7 – Threshold / Iteration Combinations for JTS 40
Table 8 – Minimum Distances Found by All Algorithms 42
Table 9 – Average Solve Times for All Algorithms 43
Table 10 – Comparison on Solomon 25 Customer Problems 45
Table 11 – Comparison on Solomon 50 Customer Problems 46
Table 12 – Comparison on Solomon 100 Customer Problems 47
Table 13 – Comparison to Best Known Solutions for Solomon Instances 49
Table 14 – Comparison on Cordeau *et al* MD VRPs 50
Table 15 – Comparison on MD VRPs from Literature 51

Abstract

In this research, we provide a new meta-heuristic, a jump search / tabu search hybrid, for addressing the vehicle routing problem with real-life constraints. A tour construction heuristic creates candidate solutions or jump points for the problem. A tabu search algorithm uses these jump points as starting points for a guided local search. We provide statistical analysis on the performance of our algorithm and compare it to other published algorithms. Our algorithm provides solutions within 10% of the best known solutions to benchmark problems and does so in a fraction of the time required by competing algorithms. The timeliness of the solution is vitally import to the unmanned aerial vehicle (UAV) routing problem. UAVs provide the lion's share of reconnaissance support for the US military. This reconnaissance mission requires the UAVs to visit hundreds of target areas in a rapidly changing combat environment. Air vehicle operators (AVOs) must prepare a viable mission plan for the UAVs while contending with such real-life constraints as time windows, target priorities, multiple depots, heterogeneous vehicle fleet, and pop-up threats. Our algorithm provides the AVOs with the tools to perform their mission quickly and efficiently.

Keywords: Air Force Research, Operations Research, Combinatorial Analysis, Algorithms, Remotely Piloted Vehicles, Surveillance Drones, Multiple Depots, Time Windows, Jump Search, Tabu Search, Vehicle Routing Problem, Java, Heuristics, Traveling Salesman Problem.

A HYBRID JUMP SEARCH AND TABU SEARCH METAHEURISTIC FOR THE UNMANNED AERIAL VEHICLE (UAV) ROUTING PROBLEM

Chapter 1. Background and Statement of the Problem

1.1 Background

Unmanned aerial vehicles (UAVs) play an increasingly important role in military operations. In recognition of this fact, the Air Force established the UAV Battlelab at Eglin AFB, FL in 1997. The UAV Battlelab's mission is to "rapidly identify and demonstrate the military worth of innovative concepts that exploit the unique characteristics of UAVs to advance Air Force combat capability" (USAF Unmanned Aerial Vehicle Battlelab homepage, 1999).

The bulk of the UAV mission is reconnaissance. A reconnaissance mission involves the UAV flying over a number of target areas within established time windows and/or outside of restricted time windows, collecting images for a minimum (though potentially longer) amount of time, and returning to base. The air vehicle operators (AVOs) are responsible for creating a viable flight plan for each reconnaissance mission, for each UAV under their control.

1.2 Problem Statement

The AVOs must determine the routing for multiple UAVs to cover designated target areas while conforming to established time window restrictions and remaining within UAV endurance limits. The routings must account for the wind and weather conditions at various altitudes, no-fly zones, high threat areas, and target priorities if complete target coverage is impossible. Currently, AVOs determine these routing manually.

In a rapidly changing combat environment, new targets often arise. It is common for UAV missions to receive a new target tasking during a mission, a dynamic re-tasking. It is also common for there to be several re-taskings during the course of a UAV mission. AVOs currently re-route the UAV manually to accommodate the re-tasking and try to complete as much of the original plan as possible. AVOs need a way to quickly add new targets to the route while minimizing the coverage impact on any targets not already visited.

Mathematically, the problem is to minimize the 'cost' of coverage, (*e.g.* flight time, man-hours, *etc.*) if target coverage is feasible or, alternatively to maximize coverage with the available resources. We know the targets and their time window restrictions and priorities, the number of vehicles available, current weather data, and threat areas. We solve the problem by assigning tours to the available vehicles. Each tour consists of an ordered list of targets. We are constrained by the time windows for the target areas, the threat areas we must circumnavigate, and the endurance of the vehicles. Formally, this problem is a multiple travelling salesman problem (TSP) with side constraints.

1.3 Scope and Contribution

This research continues the efforts of O'Rourke (1999) in support of the UAV Battlelab. O'Rourke's algorithm provides the AVOs with near-optimal tours accounting for time windows, threat areas, multiple vehicles, and asymmetric route lengths due to wind. Our algorithm extends O'Rourke's effort in four areas. The first area adds a priority scheme to the target areas to accommodate resource-constrained environments. The second area adds the ability to handle heterogeneous vehicle types from multiple starting locations, or depots. The third area adds the ability to route vehicles so as to avoid restricted time windows or time walls. The final area provides a quicker solution using a jump search / tabu search (JTS) hybrid algorithm.

We do not perform any target preprocessing. Operationally, the AVO receives a target list and often the UAV can capture more than one target in a single snapshot. Mathematically, this represents a coverage problem; however, for our purposes we assume coverage is accounted for in the target area list provided to our algorithm.

We also do not account for vehicle turning radius or approach angles. Although this can be an important aspect of vehicle routing, such capabilities vary based on vehicle type. Since our algorithm handles multiple vehicle types, we assume AVOs handle the flight profile execution detail.

Finally, we do not account for changes in terrain. Although terrain may affect route feasibility, certain assumptions must be made in order to return an answer in a reasonable amount of time. We will consider terrain as flight profile execution and again leave that to the AVOs.

Our contribution lies in assisting the AVOs in two important ways. First, the jump search portion of our algorithm provides the AVOs with a very quick, feasible, high-quality solution. This is an important capability for dynamically routing an airborne UAV. Time permitting, the AVOs can further refine a quickly obtained solution by engaging the tabu search portion of the algorithm. Second, we provide enhancements to known heuristic approaches to address customer prioritization, time walls, multiple depots, and non-homogeneous vehicles.

In terms of operations research, this effort provides a new meta-heuristic approach, a JTS hybrid algorithm, for solving complex vehicle routing problems. This algorithm is based on the hypothesis that the speed of tabu search is improved with a quality starting solution and that multiple starting solutions are an effective diversification technique for the search. We prove both of these hypotheses through empirical testing.

1.4 Report Overview

Chapter 2 presents a brief review of the literature pertaining to this research, while Chapter 3 presents a proposed methodology for conducting the research. Chapter 4 presents our test of the algorithm and analysis of the test results. Chapter 5 provides avenues for further research.

Chapter 2. Literature Review

2.1 Vehicle Routing and Traveling Salesman Problems

The vehicle routing problem (VRP) and the traveling salesperson (agent) problem (TSP) are two classic problems of operations research. The literature contains many examples of different varieties of these problems, some of which we describe below. Lawler *et al* (1985) provides comprehensive coverage of the TSP and its variants.

The two problems are closely related. In the TSP, a 'salesman' must visit a list of cities and return home, visiting each city only once. The objective is to find the minimum tour length. A tour or route consists of an ordered list of cities visited. By its classical definition, a sub-tour is an ordered cycle of one or more cities that does not include all of the cities. The presence of sub-tours in the solution of a TSP makes the solution infeasible.

For our purposes, we may not have the resources to visit all the cities. Therefore, we define a sub-tour as an ordered cycle of two of more cities that does not include the starting city or depot. The VRP is an extension of the TSP, in which the vehicle either delivers or picks up items from the cities subject to volume and weight capacity constraints.

Carlton (1995) creates a hierarchical classification scheme for the General VRP (GVRP). His classification establishes tiers for the basic TSP, VRP, and pickup and delivery problems (PDP). In a VRP, the vehicles perform either delivery or pickup operations exclusively. A PDP extends the VRP to where vehicles can make one or more pickups from customers along the route for delivery to other customers along the route.

Each tier allows for any combination of special cases of each of the problems. The problem can have a single vehicle (SV), multiple homogeneous vehicles (MVH), or multiple non-homogeneous vehicles (MV~~H~~). The vehicles can depart from a single depot (SD), or multiple depots (MD), and the tour can be constrained by time windows (TW) and route length (RL) (Carlton 1995). Using Carlton's classifications, our problem is a MD MV~~H~~ TSP with TW and RL. The route length constraint represents the endurance of the vehicle. In addition, we must contend with added constraints accommodating customer priorities and restricted time windows.

We base our mathematical formulation of the problem on Carlton's formulation for the MD MV~~H~~ PDP with TW and RL and Ryan's formulation for the MVH SD TSP with TW and RL (Carlton 1995, Ryan *et al* 1999). We have $k=1...V$ vehicles located at $r=1...D$ depots which must service N customers indexed by i and j.

Each customer has a service time s_i, a time window defined by earliest arrival e_i and latest departure l_i, and a restricted time window defined by earliest restricted time er_i and latest restricted time lr_i. Vehicles arriving early may wait with waiting time W_i at customer i, equal to the earliest arrival time e_i minus the actual arrival time. Vehicles arriving during the restricted time must also wait with waiting time W_i at customer i, equal to the latest restricted time lr_i minus the actual arrival time. Vehicles arriving before the earliest restricted time er_i must be able to complete service before the earliest restricted time er_i or must also wait with waiting time W_i at customer i, equal to the latest restricted time lr_i minus the actual arrival time. If a customer is not visited, it has an associated penalty p_i based on its priority.

Each segment between customers i and j has an associated cost c_{ijkr}, time required to travel the segment t_{ijkr}, and segment penalty sp_{ijkr} for time spent in high threat areas and no fly zones; all of these values differ based on vehicle type. Each vehicle begins and ends its tour at its depot, customer index 0, and has an endurance (maximum route length) u_{kr}.

We assign a value of 1 to X_{ijkr} if vehicle k from depot r travels from customer i to customer j. We assign the starting service time for customer i to T_i. We assign a value of 1 to n_i if customer i is not visited. The objective is to minimize

$$Z = \sum_{r\in D}\sum_{k\in V}\sum_{i\in N}\sum_{j\in N}\left(c_{ijkr} + sp_{ijkr}\right)\cdot X_{ijkr} + \sum_{i\in N} W_i + \sum_{i\in N} p_i \cdot n_i. \tag{1}$$

Subject to tour constraints,

$$\sum_{r\in D}\sum_{k\in V}\sum_{\substack{i\in N\\ i\neq j}} X_{ijkr} = 1 \quad \forall j \in N \tag{2a}$$

[one vehicle enters each customer]

$$\sum_{i\in N} X_{ijkr} - \sum_{i\in N} X_{jikr} = 0 \quad \forall j \in N, k \in V, r \in D \tag{2b}$$

[same vehicle that enters each customer leaves it]

time window constraints,

$$X_{ijkr} = 1 \Rightarrow T_j = T_i + s_i + t_{ijkr} + W_j \quad \forall i, j \in N, \forall k \in V, \forall r \in D \tag{3a}$$

[time precedence]

$$e_i \le T_i \text{ and } T_i + s_i \le l_i \quad \forall i \in N \tag{3b}$$

[time windows]

$$lr_i \le T_i \text{ or } T_i + s_i \le er_i \quad \forall i \in N \tag{3c}$$

[restricted time windows]

route length constraints,

$$\sum_{i\in N}\sum_{j\in N}\left(t_{ijkr} + W_j + s_j\right)\cdot X_{ijkr} \le u_{kr} \quad \forall k \in V, r \in D \tag{4}$$

visitation constraints,

$$n_i = 1 - \sum_{r \in D} \sum_{k \in V} \sum_{\substack{j \in N \\ j \neq i}} X_{ijkr} \quad \forall i \in N \tag{5}$$

and binary constraints.

$$X_{ijkr} \in \{0,1\} \quad \forall i, j \in N, \forall k \in V, \forall r \in D \tag{6a}$$

$$n_i \in \{0,1\} \quad \forall i \in N \tag{6b}$$

Finally, we need sub-tour breaking constraints. Let N^{kr} represent the subset of available customers visited by vehicle k from depot r. For each vehicle, we add the following constraints

$$\sum_{i \in Q} \sum_{j \notin Q} X_{ijkr} \geq 1 \quad \forall \textit{ nonempty subset } Q \subseteq N^{kr}. \tag{7}$$

In Chapter 3, we show our solution structure implicitly enforces these constraints as well as the standard tour (2) and binary constraints (6). The time window constraints (3) ensure vehicles service the customers within required time windows. The route length constraints (4) ensure the tour is within vehicle endurance limits.

In addition, we accommodate customer priorities, asymmetric route lengths, threat areas and no-fly zones. We encapsulate these restrictions in our objective function (1), where the asymmetric route lengths are reflected in the segment cost c_{ijkr}. We penalize segments that enter threat areas and no-fly zones based on the time spent in these areas and add the penalty sp_{ijkr} to solutions containing the offending segment. Lastly, we add the priority penalty p_i for each unvisited customer.

2.2 Heuristic Approaches

In terms of computational complexity, the TSP belongs to the class NP-hard (Lawler *et al* 1985). A polynomial-time algorithm does not exist for members of this class, and it is unlikely one will ever be discovered (Parker and Rardin 1982a, 1982b). The number of possible solutions to the TSP grows at a factorial rate as the number of customers increases, which makes enumeration algorithms unappealing. Consequently, heuristic approaches dominate the solution techniques for the TSP and VRP (Brandao and Mercer 1997, Carlton 1995, Clarke and Wright 1964, Gendreau *et al* 1994, Gendreau *et al* 1998, O'Rourke 1999, Rochat and Semet 1994, Ryan *et al* 1999, Semet and Taillard 1993, Solomon 1987, Tsubakitani and Evans 1998). Heuristic approaches provide no guarantee of optimality, although most provide at least a feasible solution in a relatively short amount of time. Timeliness of a solution is very important for our implementation, as UAV operations are typically time-sensitive.

2.2.1 Tour Construction and Tour Improvement Algorithms

Laporte (1992a, 1992b) surveys current optimal and heuristic techniques for both the TSP and VRP and notes that heuristic techniques fall into two categories: tour construction algorithms and tour improvement algorithms. Tour construction algorithms start with all customers unassigned and attempt to build a near-optimal solution. Conversely, tour improvement algorithms start with the customers assigned and attempt to improve the solution by changing the order in which the vehicles visit the customers or changing which vehicles visit which customers.

Some common tour construction algorithms include nearest neighbor (Rosenkrantz *et al* 1977), the Clarke and Wright savings (Clarke and Wright 1964), sweep (Gillett and Miller 1974), and their insertion versions. At each iteration, nearest neighbor adds the nearest customer to the end of a current tour until all customers are visited. The vehicle then returns to the starting point after the last customer is added. While very quick, this approach's solutions are generally poor. The Clarke and Wright savings heuristic starts with all customers visited via independent tours. It then chooses the next customer to add to the current tour based on the net savings of visiting the customer pair on a single tour versus two separate tours. The sweep heuristic attempts to cluster the customers first by 'sweeping' in a circle from the depot. The insertion versions are more complex and allow for customers to be inserted anywhere in the tour. Insertion algorithms generally produce higher quality solutions (Laporte 1992a, 1992b).

Solomon (1987) modifies some of the common tour construction heuristics to handle the VRP with the addition of time windows (TW). He modifies the nearest neighbor, savings, and sweep heuristics, and provides an insertion-based heuristic with three different insertion criteria. He also provides the MVH VRP TW test cases. These test cases form the literature's standard for measuring MVH VRP TW algorithm performance. His most robust insertion algorithm achieves the best results for 27 of the 56 problems tested and a lower bound within 8.3% of best known solutions for the remaining 29 problems (Solomon 1987). (All further references to Solomon's insertion heuristic are based exclusively on the most robust version.)

2.2.2 Solomon's Insertion Heuristic

Solomon suggests two methods for initializing tours in his insertion heuristic. The first routes to the farthest customer, and the second routes to the customer with the earliest deadline. Once the tour is initialized, remaining customers are inserted until either the vehicle is at maximum capacity or no other customers can be added without violating time window feasibility or vehicle endurance. At this point, another tour is initialized and the process continues.

The algorithm inserts customer u between customers i and j based on two criteria: *C1(i,u,j)* and *C2(i,u,j)*. *C1(i,u,j)* determines the best insertion point for each unassigned customer u as

$$C1(i(u),u,j(u)) = \min\left(c1(i_p,u,j_{p+1})\right) \quad p=1,...,m \tag{8a}$$

and

$$c1(i_p,u,j_{p+1}) = \alpha 1\left(d_{iu} + d_{uj} - \mu \cdot d_{ij}\right) + \alpha 2\left(b_{ju} - b_j\right) \tag{8b}$$

where d_{ij} is the distance between customers i and j, b_{ju} is the beginning service time of customer j with customer u inserted before it, b_j is the beginning service time of customer j without customer u inserted and p is the position, from 1 to m, within the current tour. Parameters $\alpha 1$ and $\alpha 2$ must be positive and sum to one, while parameter μ must be positive.

Equation (8a) attempts to minimize the cost of inserting customer u into an emerging tour in terms of distance added ($d_{iu} + d_{uj} - \mu d_{ij}$) and delay in service to the following customer ($b_{ju} - b_j$). Parameter μ determines how much of the original distance between customers i and j is subtracted from the distance between customer i to customer

u to customer j, while parameters $\alpha 1$ and $\alpha 2$ balance the relative importance of distance and time window feasibility. All three parameters are user set.

Next, the algorithm chooses which customer to insert based on criteria *C2(i,u,j)* as

$$C2(i(u^*),u^*,j(u^*)) = \max\left(\lambda \cdot d_{ou} - C1(i(u),u,j(u))\right) \quad \forall \textit{ unrouted customer } u. \qquad (9)$$

Equation (9) inserts the unassigned customer u with the largest 'savings' compared to the distance between the customer and the depot 0. Parameter λ is the multiplier for the distance. The insertion continues until the algorithm has routed all customers (Solomon 1987).

Time window feasibility is maintained throughout the algorithm. Solomon's Lemma 1.1 states that if a customer is inserted into a tour that is time window feasible, it remains time window feasible if the insertion does not result in a delay to the following customer (1987). Therefore, when we insert a new customer, we need only check from the insertion point until we (1) find a customer whose service time is not delayed, (2) we find a time window violation, or (3) we reach the end of the tour.

Since a VRP is very similar to the TSP, we can easily modify many of the algorithms developed for one problem to find solutions for the other. For example, an algorithm developed for the VRP can be used for the TSP by relaxing the vehicle capacity constraints.

We use a tour construction algorithm based on Solomon's insertion heuristic in our JTS algorithm. Solomon develops his algorithm for the VRP with time windows. We use the same algorithm, but remove vehicle capacity constraints.

2.2.3 k-opt Improvement Algorithms

The *k*-opt algorithms are common tour improvement procedures. In the *k*-opt algorithm, *k* routes are dropped and replaced at each iteration until no further improving moves exist. For example, a move that swaps two adjacent customers, thus exchanging two routes, is a two-opt move (assuming the routes are symmetric).

2.2.4 Tabu Search

The more robust algorithms contain both tour construction and *k*-opt improvement moves. Of the heuristic approaches, Laporte (1992b) states that the tabu search heuristic may be one of the best for TSP and VRP. Indeed, tabu search's proven record in solving these types of problems (Brandao and Mercer 1997, Carlton 1995, Gendreau *et al* 1994, Gendreau *et al* 1998, O'Rourke 1999, Rochat and Semet 1994, Ryan *et al* 1999, Semet and Taillard 1993) motivates its use in this effort.

Tabu search (TS) is a meta-heuristic developed by Glover (1989, 1990a). TS provides a methodology to escape local optima by use of recency-based memory. The search moves from solution to solution while maintaining a list of recent moves. The moves in this list are tabu or off-limits. This stops the algorithm from cycling back to local optima after taking a non-improving move (Glover and Laguna 1997). The literature provides many variations and extensions of TS implementation (see Glover and Laguna 1997).

Many tabu search implementations use a heuristic to build a starting solution (Brandao and Mercer, 1997; Carlton, 1995; Gendreau *et al* 1994, Gendreau *et al* 1998; Rochat and Semet, 1994). This approach generally improves the quality of the solution

and the speed of the algorithm. Carlton (1995) and Rochat and Semet (1994) implement Solomon's insertion heuristic for this purpose, with both noting the degree to which the starting solution improved the overall solution depends upon the parameters used and the configuration of the customers.

Carlton (1995) develops algorithms to solve the VRPTW using a reactive tabu search (RTS) meta-heuristic. Reactive tabu search (RTS) (Battiti 1996) allows the length of the tabu list to change based on the quality of the search. When the search appears to be cycling through the same solutions, the algorithm increases the tabu length to force search diversification and break the cycle.

Two important facets of any good TS algorithm are intensification and diversification (Glover and Laguna 1997). Intensification is the process of conducting a more thorough search in the areas of the solution space where the algorithm has found good solutions. By contrast, diversification drives the search into new, previously unexplored, areas of the solution space.

One popular intensification technique is a candidate or elite list. The TS starts by diversifying and quickly scanning the solution space. The top candidate solutions found are saved in an elite list. After the diversification period, the search moves to each candidate in the elite list and intensifies the search in the neighborhoods of those elite solutions. Our approach is similar to the elite list, except we begin the TS with an elite list already in place. We obtain this elite list using tour construction heuristics.

2.2.5 Jump Search

Tsubakitani and Evans (1998) developed the jump search (JS) meta-heuristic as a way to generate good candidate solutions using a quick tour-construction heuristic and then use these candidate solutions as 'jump points' for a local search. The idea springs from the notion that there exists plateaus of good solutions within the solution space, and these jump points provide quick access to those plateaus. In their study, Tsubakitani and Evans use JS to guide two-opt and three-opt improvement algorithms to solve a 1-TSP without side constraints.

Their JS uses six different tour construction heuristics to generate a list of candidate jump points. The algorithm orders the candidate jump points based on objective function value. If two heuristic solutions differ by a single move, the algorithm keeps the solution with the best objective function value. The algorithm then launches a local search from the best available jump point. When JS finds a local optimum, the search moves to the next jump point on the candidate list. This process continues until all jump points are exhausted or the algorithm reaches a predetermined time limit or number of iterations.

JS produces equal or better solutions than TS for nearly all of the test cases Tsubakitani and Evans examine. However, they compare their JS algorithm to a very basic TS algorithm, one without intensification or diversification, and on a very basic problem, 1-TSP (Tsubakitani and Evans 1998). We hypothesize that in complex problems, these good plateaus may contain many local optima; consequently, it may be unwise to give up the search when we find the first local optimum.

TS provides a mechanism to escape the local optima and continue the search. We can search the plateau more thoroughly and therefore intensify the search in promising areas. When the rate of improvement in the solution quality begins to level off, the procedure moves to the next jump point and diversifies to an unexplored area of the solution space. Tsubakitani and Evans suggest that a JS / TS hybrid (JTS) could be a very effective search algorithm. We explore this suggestion and apply it to the UAV routing problem.

2.3 Earlier UAV Routing Efforts

Our UAV research effort is one of many performed in recent years. Sisson (1997) constructs a RTS algorithm based on Carlton's (1995) RTS approach to solve the UAV routing problem while accounting for wind effects and attrition due to enemy actions. Sisson's approach determines the minimum number of vehicles required to cover a specified target area given a risk assessment based on enemy threat and provides insight into the minimum tour and minimum risk involved in providing the necessary coverage. The routes are passed into a Monte Carlo simulation to assess vehicle losses and expected coverage of targets (Sisson 1997).

Ryan *et al* (1999) centers on finding the 'robust tour', using the minimum number of vehicles. He defines the 'robust tour' as the route least affected by changes in threat, target area service times, and weather conditions. Using an embedded optimization approach, his Monte Carlo simulation generates random weather conditions and probabilities of survival for each target. He then passes these values into a reactive tabu

search (RTS) algorithm to search for the best solution under each set of variations, with the robust solution defined as the solution appearing most often.

While Ryan's results are useful and appropriate for autonomous UAVs in which missions are preplanned, they do not address the concerns of the more dynamic missions of UAVs such as the US Air Force's RQ-1A Predator (O'Rourke 1999). USAF pilots in a ground control station control the Predator vehicle remotely. Consequently, they are able to change the Predator routes as the mission dictates. O'Rourke recognizes the need for a program capable of returning the best available tour based on current conditions and return this solution in a short enough time for it to be of use.

O'Rourke's efforts focus on a dynamic routing algorithm. Building upon the RTS algorithm from Ryan *et al* (1999) and Carlton (1995), he develops a Java-based application to solve this problem. O'Rourke adds functionality to the algorithm to account for, and take advantage of, multiple wind tiers. Additionally, his program incorporates time windows, no-fly zones, and threat areas as in previous efforts. He adds the ability to perform dynamic routing, so vehicles may start from their current location and return to the depot, and he improves the performance of the algorithm using a reactive penalty scheme.

2.4 Analysis of Heuristics

As Hooker (1995) observes, too much of heuristic research is reduced to competitive testing. Researchers are forced to show that their new algorithms are faster or produce higher quality solutions than existing algorithms to be published. What is typically missing is any kind of explanation as to why the algorithms perform as they do

and any statistical support that the superior performance of the algorithm extends beyond the test problems.

We develop and test five local search algorithms using empirical experimentation to gain insight into which approach is superior and why. Due to differences in programming techniques, testing competing algorithms head-to-head is problematic. We overcame this problem by modifying a common algorithm to isolate the differences due to the particular modifications made to that algorithm.

2.5 Conclusion

Although a tremendous amount of research exists on the TSP and its variants, few have endeavored to incorporate multiple non-homogeneous vehicles and multiple depots into the solution algorithms. These side constraints are very important to real-life problems in the military and civilian sector. While prioritizing customers for inclusion into tours is not tremendously difficult, it seems to have been neglected entirely, suggesting that customer prioritization is prominent mainly in military applications.

TS is a popular heuristic for solving TSPs and it provides good results. Many TS algorithms are primed with an initial solution provided by a tour construction heuristic that improves the performance of the algorithm. Carlton (1995) and O'Rourke (1999) demonstrate that a robust TS algorithm can overcome an arbitrary starting solution. We investigate TS performance when provided multiple, high-quality initial solutions.

When the tour construction heuristic generates a good starting solution for the TS, it finds high quality solutions rather quickly. However, the quality of the solutions generated depends on the configuration of the customers and the tour construction

heuristic used. Unfortunately, we do not know the customer configuration when the algorithm is developed. We can overcome this dilemma by generating multiple initial solutions using a tour construction heuristic with different initialization schemes and multiple parameter settings. In the next chapter, we discuss how we implement this approach.

Chapter 3. Methodology

3.1 Solution Representation

The first step in building our JTS hybrid algorithm is determining how we represent the solution in the algorithm. We use a solution representation developed by Harder (2000). Figure 1 shows the representation of the problem solution.

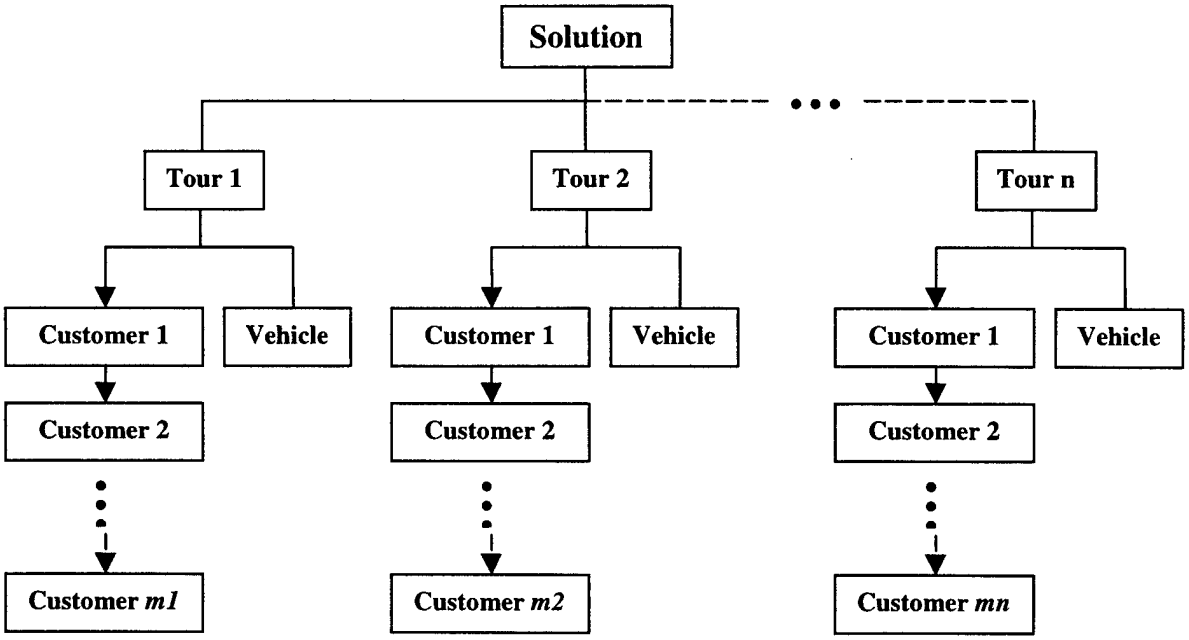

Figure 1 - Representation of Problem Solution

Each solution is comprised of n tours, where n is the number of available vehicles. Each tour has an associated vehicle and an ordered list of customers. We evaluate each tour individually for solution quality based on the associated vehicle's attributes and depot location. If we lack the vehicle resources to cover all targets, a dummy tour contains customers that are not visited, and the solution is penalized based on the priority

of those customers. A significant advantage of this representation is it implicitly captures the standard tour (2), binary (6), and sub-tour breaking (7) constraints.

3.2 Tour Construction Heuristic

We build our 'jump point' solutions using an insertion heuristic based on Solomon (1987). While Solomon's insertion heuristic arbitrarily chooses the next tour to construct, we build a tour for the vehicle closest to the initializing customer, which allows us to take advantage of multiple depots. We account for a heterogeneous vehicle fleet by evaluating each vehicle by its individual capabilities and only assigning the vehicle customers it is capable of servicing. Restricted time windows, or time walls, are handled in a similar way to time windows by maintaining time windows/wall feasibility throughout the algorithm. To incorporate customer priorities, we remove customers from the dummy tour based on their priority (*i.e.*, all priority 1 customers are assigned to real vehicles before priority 2 customers are considered). The logic flow for the tour construction algorithm is:

Step CON-1: Calculate cost matrix for each available vehicle.

Step CON-2: Build parameter matrix.

Step CON-3: Select the next set of parameters.

Step CON-4: Initialize the next tour based on the selected initialization method.

Step CON-5: Insert customers based on equations (8) and (9).

Step CON-6: When the tour is full return to CON-4.

Step CON-7: When all tours are full or all customers are assigned add the solution to the jump point array.

Step CON-8: If all parameters have been used, return the jump point array, STOP.

Otherwise, return to CON-3.

3.2.1 Parameter Settings

To develop a complete list of diverse starting solutions, we use a variety of parameter settings in our tour construction heuristic. We use each set of parameters in conjunction with both initialization criteria discussed in Section 3.2.2. We employ the following parameter ranges suggested by Rochat and Semet (1994)

$$\alpha 1 = 0.0; 0.1; ...; 1.0$$

$$\alpha 2 = 1 - \alpha 1$$

$$(\lambda, \mu) = \begin{cases} 1)\ \lambda = 1.25; 1.50; 1.75; 2.00 \ \ and\ \mu = \lambda - 1 \\ 2)\ \lambda = 0.0; 0.5; 1.0; 1.5 \ \ and\ \mu = 1 \end{cases}$$

These parameter ranges, combined with both initialization methods, yield 176 different combinations.

3.2.2 Tour Initialization

Since we attempt to generate a list of diverse solutions, we use two initialization techniques. We modify the first initialization method to include penalties for time spent in threat areas or no-fly zones, effectively assigning the costliest customers first. Since we have multiple depots, we find the customer with the largest minimum cost to any of the remaining vehicles. We assign this customer to the nearest remaining vehicle and build a tour for that vehicle.

For the second initialization method, we find the customer with the earliest deadline (latest arrival time). We assign the customer to the nearest remaining vehicle

and build a tour for that vehicle. Since we have a heterogeneous vehicle fleet, only vehicles capable of servicing the customer are considered. In both cases, priority of customers takes precedence over other selection criteria in initializing a tour.

3.2.3 Customer Insertion

Customers are inserted based on Solomon's original algorithm using equations (8) and (9). However, only customers the vehicle is capable of servicing are considered. The insertion considers customers in priority order and continues until all customers are assigned or all tours are full.

Time window and time wall feasibility is maintained throughout the process. Due to Solomon's lemma, we are able to check time window and time wall feasibility very efficiently. Vehicle's arriving before a customer's time window must wait until the time window begins. Vehicle's unable to complete service before a customer's time wall begins must wait until the end of the time wall to start service. Only feasible customer insertions are considered.

3.2.4 Completed Solutions

A tour is full when any further insertions will violate time window/time wall feasibility or vehicle endurance. The solution is complete when all available vehicles have full tours or all customers have been assigned. We insert completed solutions into the jump point array in descending order of solution quality.

A solution possessing the same number of vehicles, travel distance, waiting time, and penalty as an existing solution in the jump point array is considered equal to the solution and is discarded. In their original jump search algorithm, Tsubakitani and Evans

(1998) retained only the jump points from mutually exclusive neighborhoods. Even with the simplest of neighborhoods, this neighborhood determination becomes a complex and computationally expensive task. Through preliminary empirical testing on the Solomon MVH VRP TW test cases, we determined this approach was not cost-effective. The rate of duplicate best solutions found by the local search algorithms is less than 7% when all jump points are explored. Consequently, we simply retain all unique jump points.

3.3 Local Search Heuristics

We explore each jump point provided by our tour construction algorithm with three different local search heuristics: a first best local search, a global best local search, and a global best tabu search. As an experimental control, we test a reactive tabu search initialized two different ways. First, we generate all of the jump points and initialize the RTS with the best jump point (RTS-best). Second, we initialize the RTS after generating a single jump point (RTS-one).

3.3.1 First Best Local Search

The first best local search (FBLS) makes the first improving move found and continues doing so until no more improving moves can be found indicating we have reached a local optimum. For this algorithm, we consider only single customer insertion moves. A single customer can be removed from one tour and inserted into another tour or a different spot in the same tour. Only feasible insertions are considered. The logic flow for the FBLS algorithm is:

Step FBLS-1: Improve tours individually by rearranging customers within tours.

Step FBLS-2: Improve the solution further by rearranging customers between tours.

Step FBLS-3: If we found at least one improving move, try to assign customers from the dummy tour.

Step FBLS-4: If we found at least one improving move, return to FBLS-1.

Step FBLS-5: If we did not find an improving move, return the solution, STOP.

3.3.1.1 Rearranging Customers within Tours

Starting with the first customer in the first tour, we temporarily remove individual customers while noting the current cost to service that customer. The cost of servicing a customer is the travel distance, the wait time, and the penalty associated with the segments connecting the customer to the tour. These values for customer u currently between customers i and j can be calculated with the following equations

$$d_{iu} + d_{uj} - d_{ij} \quad \text{where } d_{ij} \text{ is the distance between } i \text{ and } j \text{ [travel distance]} \tag{10}$$

$$w_{iu} + w_{uj} - w_{ij} \quad \text{where } w_{ij} \text{ is the wait time when } j \text{ follows } i \text{ [wait time]} \tag{11}$$

$$p_{iu} + p_{uj} - p_{ij} \quad \text{where } p_{ij} \text{ is the penalty between } i \text{ and } j \text{ [penalty]} \tag{12}$$

Using equations (10), (11), and (12), we calculate the cost to service the customer at every other position in the tour starting from the first position. If the new cost of service is less than the current cost of service, we check time window/time wall and vehicle endurance feasibility. If the move is feasible, we insert the customer in the new position and check the next customer. If we cannot find any feasible improving moves for the customer, we return it to its original position and check the next customer.

When we cannot find any feasible improving moves for the current tour, we check the next tour. When we can no longer improve any of the tours, we attempt to rearrange customers between tours.

3.3.1.2 Rearranging Customers between Tours

Starting with the first customer in the first tour, we try to find a cheaper position for the customer in another tour. We evaluate the current cost of service for the customer using equations (10), (11), and (12).

We consider other tours in lexicographical order if the vehicle type is appropriate and there is sufficient capacity. If the vehicle can service the customer, we calculate the new cost of service for each position in the tour starting with the first position. If the new cost of service is less than the current cost of service, we check time window/time wall and vehicle endurance feasibility. If the move is feasible, we insert the customer in the new tour position and check the next customer. If we cannot find any feasible improving moves for the customer, we check the next customer.

When we cannot find any feasible improving moves for any of the customers in the tour, we check the next tour. When we can no longer improve any of the tours by rearranging customers between them, we attempt to empty the dummy tour.

3.3.1.3 Assigning Customers from the Dummy Tour

By reducing the costs of the tours, we hope that we have made room for customers not currently assigned. Since we are maximizing coverage first and minimizing cost second, we assign customers from the dummy regardless of the increase in costs.

Considering each dummy tour customer in priority order, we consider inserting the customer into the other tours with appropriate vehicle type and sufficient capacity. If a vehicle can service the customer, we try to find the best feasible position for it in the tour. If we assign any customers from the dummy tour, we return to FBLS-1 and attempt to improve the solution further.

3.3.2 Global Best Local Search

The global best local search (GBLS) is similar to the FBLS algorithm; however, we check all possible moves in the current neighborhood before choosing the best move. As before, we consider only feasible single customer insertion moves. We stop when we cannot find any feasible improving moves indicating we have reached a local optimum. The logic flow of the global best local search algorithm is:

Step GBLS-1: Check single customer insertions within tours.

Step GBLS-2: Check single customer insertions between tours.

Step GBLS-3: Make the best move found.

Step GBLS-4: If we found an improving move, try to assign customers from the dummy tour.

Step GBLS-5: If we found an improving move, return to GBLS-1.

Step GBLS-6: If we did not find an improving move, return the solution, STOP.

3.3.2.1 Check Insertions within Tours

Considering each customer in each tour, we note the current cost to service that customer using equations (10), (11), and (12). We then determine the cost of inserting the customer into every other position in the current tour.

The current cost of service minus the new cost of service represents the savings achieved by making the move. If the amount saved is positive and greater than the best savings found so far, we check time window/time wall and vehicle endurance feasibility. If the move is feasible, the move is retained as the best so far. If we cannot find any feasible improving moves for the customer, we check the next customer.

When we cannot find any feasible improving moves for any of the customers in the tour, we check the next tour. When we have checked all insertions within tours, we check insertions between tours.

3.3.2.2 Rearranging Customers between Tours

Starting with the first customer in the first tour, and considering all customers in all tours, we try to find a cheaper position for the customer in another tour whose vehicle is appropriate for the customer and has sufficient capacity. We evaluate the current cost of service for the customer using equations (10), (11), and (12).

If the vehicle is capable of servicing the customer, we calculate the new cost of service for each position in the tour starting with the first position. If the amount saved is positive and greater than the best savings found so far, we check time window/time wall and vehicle endurance feasibility. If the move is feasible, the move is retained as the best so far. If we cannot find any feasible improving moves for the customer, we check the next customer.

When we have checked all insertion moves within and between tours, we make the best move found. If we did not find a feasible improving move, we have reached a

local optimum and we return the solution. If we were able to make an improving move, we attempt to empty the dummy tour.

3.3.2.3 Assigning Customers from the Dummy Tour

Assigning customers from the dummy tour is accomplished exactly as in the FBLS algorithm explained in section 3.3.1.3.

3.3.3 Global Best Tabu Search

The global best tabu search (GBTS) is similar to the global best local search algorithm; however, we implement a tabu list to escape local optimum. The algorithm considers the same insertion moves as the first best and global best local search algorithms.

At every iteration, all possible insertion moves are evaluated and the best feasible non-tabu move is chosen. The best move in this case may actually be non-improving. The algorithm continues for a minimum of 100 iterations. If the algorithm has found a new best solution within the final 10 iterations, the search continues until 10 iterations are performed without improving the best solution. We choose these values based on empirical testing. The flow of the global best tabu search algorithm is:

Step GBTS-1: Check single customer insertions within tours.

Step GBTS-2: Check single customer insertions between tours.

Step GBTS-3: Check the tabu status of the best move found. If move is tabu and does not produce a solution better than the best found so far, check the next best move. If all feasible moves are tabu, the oldest half of the tabu list is discarded and the moves are checked again.

Step GBTS-4: After the move is made, it is added to the tabu list.

Step GBTS-5: Check to see if the new solution is the best found so far.

Step GBTS-6: If an improving move was made, try to assign customers from the dummy tour.

Step GBTS-7: If we have not reached 100 iterations or we have found a new best solution in the last 10 iterations, return to GBTS-1.

3.3.3.1 Check Insertions within Tours

Checking for insertion moves within tours is the same as in the global best local search algorithm described in section 3.3.2.1. However, we accept non-improving moves, *i.e.* negative savings, and since our best move may be tabu, we retain the top 150 moves.

3.3.3.2 Rearranging Customers between Tours

Again, this is done the same as in the global best local search algorithm described in section 3.3.2.2 with the exceptions noted in the previous paragraph.

3.3.3.3 Checking the Tabu Status

We use the customer number to denote the tabu status of a move. The length of the tabu list is set to 35% of the number of customers. How a move is denoted, the length of the tabu list, and the neighborhood size are closely tied. These factors must be balanced to avoid cycling and still conduct a thorough exploration of the search space near the jump point. We chose to set these factors based on empirical testing.

Aspiration criterion is a TS technique that allows a move to be made in spite of its tabu status. A very common aspiration criterion is to accept the move if that move leads to a new best solution. We use this criterion in our GBTS algorithm. If the top 150 moves found are all tabu and do not meet the aspiration criterion, we discard the oldest half of the tabu list.

3.3.3.4 Assigning Customers from the Dummy Tour

Assigning customers from the dummy tour is accomplished exactly as in the FBLS algorithm explained in section 3.3.1.3.

3.3.4 Reactive Tabu Search

The reactive tabu search (RTS) algorithm is identical to the GBTS algorithm with two exceptions. The first difference is the number of iterations. The RTS algorithm executes for a minimum of 2000 iterations and continues until 200 iterations are performed without finding a new best solution.

The second, and key, difference is that we adjust the length of the tabu list based on the productivity of the search. If we have performed 100 iterations without finding a new best solution, we increase the length of the tabu list by one. We continue increasing the length of the tabu list at each iteration until we find a new best solution or the length of the tabu list reaches 50% of the number of customers. At this point, we reset the length of the tabu list to 35% of the number of customers. As in the GBTS algorithm, if the top 150 moves found are all tabu and do not meet the aspiration criteria, we discard the oldest half of the tabu list. The flow of the reactive tabu search algorithm is:

Step RTS-1: If we have performed 100 iterations without finding a new best solution and the length of the tabu list is less than 50% of the number of customers, increase the length of the tabu list by 1. If the length of the tabu list is 50% of the number of customers, reset the length to 35% of the number of customers.

Step RTS-2: Check single customer insertions within tours.

Step RTS-3: Check single customer insertions between tours.

Step RTS-4: Check the tabu status of the best move found. If move is tabu and does not produce a solution better than the best found so far, check the next best move. If all feasible moves are tabu, the oldest half of the tabu list is discarded and the moves are checked again.

Step RTS-5: After the move is made, add it to the tabu list.

Step RTS-6: Check to see if the new solution is the best found so far. If so, reset the length of the tabu list to its original length.

Step RTS-7: If an improving move was made, try to assign customers from the dummy tour.

Step RTS-8: If we have not reached 2000 iterations or we have found a new best solution in the last 200 iterations, return to RTS-1.

3.4 The Jump Search Algorithm

Our JS algorithm uses the tour construction heuristic described in section 3.2 to generate up to 176 unique solutions. As each solution is generated, JS stores the solution in an array sorted in descending order by solution quality. JS then initializes a local

search using one of the local search algorithms described in section 3.3 for each jump point starting from the first.

For the FBLS algorithm, we perform local searches from each jump point until 25 jump points are searched without improving the best solution found. For the GBLS algorithm, we perform local searches from each jump point until 15 jump points are searched without improving the best solution found. For the GBTS algorithm, we perform searches from each jump point until 10 jump points are searched without improving the best solution found. These values are established through empirical testing described in section 4.4. The RTS algorithm explores only a single jump point, either the best jump point found (RTS-best) or a single generated jump point (RTS-one).

3.5 Conclusion

We design a representation of the solution that makes it easier to incorporate multiple heterogeneous vehicles and multiple depots. This solution also implicitly enforces the standard tour (2), binary (6), and sub-tour breaking (7) constraints.

Our tour construction algorithm builds multiple high-quality jump point solutions. We then use three different local search algorithms, first best local search (FBLS), global best local search (GBLS), and global best tabu search (GBTS) to explore these jump points.

Chapter 4. Results

4.1 Description of Solomon Instances

We use the Solomon MVH VRP TW problem instances to test the performance of JTS against published solutions and to establish our parameter values. The Solomon problems were randomly generated to model several factors common to VRP TW problems. The factors modeled include geographic positioning, number of customers a vehicle can service, and time window characteristics such as the tightness of the time windows and the percentage of customers with time windows.

The 56 Solomon problems are divided into six problem sets: R1, C1, RC1, R2, C2, and RC2. Each problem set has between 8 and 12 problems . Each problem contains data for 100 customers. The problems in sets R1 and R2 contain customers with random geographic coordinates. Sets C1 and C2 contain customers with clustered geographic coordinates. Sets RC1 and RC2 contain customers with both random and clustered coordinates.

Each problem set contains problems in which 25%, 50%, 75%, or 100% of the customers have time windows. In sets R1, C1, and RC1, the time windows are short and vehicle capacities small. In sets R2, C2, and RC2, the time windows are long and vehicles capacities large allowing each vehicle to service more customers (Solomon, 1987).

4.2 Tour Construction Parameters

Our first task is to determine if all of the parameter values suggested by Rochat and Semet (1994) are needed. If any parameter is unnecessary, we can avoid generating all 176 starting solutions with the tour construction heuristics. We solve each instance of the Solomon test problems with the jump search first best (JFB), jump search global best (JGB), and jump search tabu search (JTS) algorithms. We note the parameter values used to generate the best solution found for each problem by each algorithm.

If each parameter value has the same probability to generate the best solution, we would expect the values associated with the best solutions to appear in the same percentage as they appear in the parameter sets. Since we know the expected distribution of the parameter values, we can perform a goodness-of-fit test to determine if the actual distribution matches the expected distribution.

We do not wish to tune our algorithm specifically to the Solomon problem sets. Therefore, we require a high degree of confidence, 95% overall, before rejecting the hypothesis that the actual distribution of parameter values is equal to the expected distribution. Parameter $\alpha 2$ is derived from parameter $\alpha 1$ and has the same distribution. Since we have four parameter distributions and three algorithms, we must perform each hypothesis test at 99.58% to have a 95% percent confidence overall.

Table 1 contains the expected distribution for each parameter and the actual distribution that produced the best solutions for each algorithm. When more than one parameter set produced the same best solution, a fractional count is added to the value cell. In all cases, the test statistic, X^2, is less than the critical value so we fail to reject the

hypothesis that the data values fit the expected distribution. Therefore, we retain all parameter values for our algorithm.

Table 1 – Parameter Value Goodness-of-fit Test

parameter α1	JFB	JGB	JTS	expected value
0.0	2.37	2.08	0.72	5.09
0.1	5.58	4.15	3.80	5.09
0.2	3.80	4.56	4.01	5.09
0.3	2.16	2.71	3.66	5.09
0.4	2.85	4.45	3.66	5.09
0.5	7.16	2.00	8.45	5.09
0.6	1.92	3.29	6.03	5.09
0.7	9.01	9.67	5.93	5.09
0.8	6.44	7.55	5.38	5.09
0.9	5.54	5.54	5.33	5.09
1.0	9.12	9.98	9.04	5.09
X^2	13.93	15.75	10.74	
X^2(10,.0042)	25.68	25.68	25.68	
parameter μ	**JFB**	**JGB**	**JTS**	**expected value**
0.25	4.61	3.39	9.31	7
0.50	5.02	5.72	6.83	7
0.75	4.71	10.10	5.97	7
1.00	41.66	36.75	33.89	35
X^2	3.39	3.56	0.95	
X^2(3,.0042)	13.21	13.21	13.21	
parameter λ	**JFB**	**JGB**	**JTS**	**expected value**
0.00	1.33	1.67	2.53	7
0.50	6.48	4.71	5.74	7
1.00	9.22	12.69	9.72	7
1.25	4.61	3.39	9.31	7
1.50	18.90	14.39	13.45	14
1.75	4.71	10.10	5.97	7
2.00	10.75	8.98	9.28	7
X^2	10.63	13.24	5.81	
X^2(6,.0042)	18.98	18.98	18.98	
init method	**JFB**	**JGB**	**JTS**	**Expected**
0.0	38.14	32.18	34.33	28
1.0	17.86	23.82	21.67	28
X^2	7.34	1.25	2.86	
X^2(1,.0042)	8.20	8.20	8.20	

4.3 Tabu List Length

Having established our tour construction parameters, we next determine the length of our tabu list. Fixing the number of iterations to a base of 100 and increases of 10 as described in section 3.3.3, we vary the length of the tabu list from 5% to 50% of the number of customers in 5% increments.

Table 2 contains the sum of the average distances for each problem set produced by the JTS algorithm for problems with 25, 50, 75, and 100 customers. The 25, 50, and 75 customer problems are generated by using the first 25, 50, and 75 customers of the 100 customer problems. To simplify comparisons, we minimized distance only when solving each problem. The tabu list length appears in the column on the left.

Since we believe the UAV problems will typically contain at least 100 customers, we give more credibility to the larger problems. While the difference in total distance is relatively small across the different tabu list lengths, there is a definite sweet spot between 30% and 40%. Therefore, we set the initial tabu list length to 35% of the number of customers.

Table 2 – Tabu Length Test

Length	25 Customers	50 Customers	75 Customers	100 Customers
5%	2031.77	3662.49	5242.55	6256.46
10%	2025.41	3646.82	5213.47	6219.27
15%	2021.52	3640.00	5187.80	6202.68
20%	2023.99	3628.76	5170.97	6193.59
25%	2017.30	3606.87	5173.58	6186.90
30%	2014.27	3614.20	5182.11	6179.16
35%	2006.05	3608.99	5166.91	6179.06
40%	2003.72	3597.96	5184.97	6178.50
45%	2005.87	3597.00	5161.07	6187.97
50%	2010.41	3596.24	5182.01	6200.39

4.4 Jump Points

For the Solomon problem sets, the tour construction heuristic generates an average of 141.82 unique jump points. With 176 different parameter sets, the average percentage of duplicate jump points generated is 19%. Recall the generated jump points are ordered. When all jump points are explored, the average jump point index leading to the best solution is 42.84 for the JFB algorithm, 41.66 for the JGB algorithm, and 42.05 for the JTS algorithm. This implies the better jump points lead to the best solutions.

Table 3 – Average Best Jump Point Index

	JFB	JGB	JTS
average best jump point	42.84	41.66	42.05
standard deviation	43.46	48.68	42.06

Since timeliness of the solution is important to us, we do not want to explore all jump points. Rather than explore a fixed number of jump points, we establish the number of jump points to explore without finding a new best solution. If we explore this threshold number of jump points without improving the best solution so far, the overall search process is halted. By setting this threshold higher, we may improve the solution; however, we will definitely increase the runtime of the algorithm. This tradeoff between solution time and quality is a common tradeoff in heuristic implementation.

Table 4 contains the average distance found at different jump point thresholds for the JFB algorithm. The column on the far right shows how much we can improve the solution if we simply continued on from that threshold point to explore all points. When we reach a point within 1% of the best distance, we feel we have explored a sufficient

number of jump points. For the JFB algorithm, this point is reached with a threshold of 25 jump points.

Table 4 – Jump Point Thresholds for JFB

Jump Point Threshold	R1	C1	RC1	R2	C2	RC2	Total	% Over Best
1	1279.43	845.39	1439.83	1017.80	600.08	1264.76	6447.29	3.61%
5	1271.39	842.20	1431.39	1010.43	598.53	1237.03	6390.97	2.70%
10	1269.77	841.11	1427.81	1003.87	598.53	1199.49	6340.58	1.89%
15	1265.75	841.11	1425.02	997.78	598.53	1186.48	6314.67	1.48%
20	1265.75	841.11	1421.64	994.18	598.53	1169.18	6290.39	1.09%
25	1262.55	834.82	1421.64	988.88	598.53	1169.18	6275.60	0.85%
30	1260.94	834.82	1421.64	982.68	598.53	1165.70	6264.31	0.67%
all	1252.69	833.97	1419.90	976.26	598.53	1141.41	6222.76	

Table 5 contains the same data for the JGB algorithm. For the JGB algorithm, we are within 1% of the best with a threshold of 15 jump points.

Table 5 – Jump Point Thresholds for JGB

Jump Point Threshold	R1	C1	RC1	R2	C2	RC2	Total	% Over Best
1	1273.49	834.28	1444.23	1018.93	595.27	1226.34	6392.54	2.87%
5	1259.86	833.19	1435.39	1004.8	593.16	1203.08	6329.48	1.85%
10	1259.24	833.19	1426.77	999.51	593.16	1176.37	6288.24	1.19%
15	1256.59	833.19	1426.77	986.36	593.16	1175.17	6271.24	0.91%
20	1254.13	833.19	1422.21	986.36	593.16	1171.72	6260.77	0.75%
25	1253.6	833.19	1418.24	985.54	593.16	1168.81	6252.54	0.61%
30	1252.07	833.19	1418.24	984.14	593.16	1166.95	6247.75	0.54%
all	1244.96	833.19	1417.27	973.76	593.16	1152.05	6214.39	

Table 6 contains the same data for the JTS algorithm. For the JTS algorithm, we are within 1% of the best with a threshold of 25 jump points. However, the JTS algorithm takes much longer to explore each jump point than either of the other algorithms. Since we believe the UAV problems will have a minimum of 100 customers,

we would like the average total solution time for the 100 customer problems to be under one minute on a Pentium II 400 MHz processor.

Table 6 – Jump Point Thresholds for JTS

Jump Point Threshold	R1	C1	RC1	R2	C2	RC2	Total	% Over Best
1	1239.26	832.91	1410.55	999.48	593.74	1186.38	6262.32	2.68%
5	1231.15	832.91	1404.57	980.38	593.16	1146.76	6188.93	1.48%
10	1230.27	832.91	1399.79	976.17	593.16	1146.76	6179.06	1.32%
15	1228.53	832.77	1397.84	971.15	593.16	1146.76	6170.21	1.17%
20	1228.53	832.77	1396.09	966.74	593.16	1146.76	6164.05	1.07%
25	1227.45	832.77	1396.09	958.96	593.16	1138.09	6146.52	0.78%
30	1227.45	832.77	1396.09	958.74	593.16	1138.09	6146.30	0.78%
all	1222.86	832.77	1389.86	950.36	593.16	1109.69	6098.70	

For the JTS algorithm, the number of iterations performed on each jump point and the number of jump points explored determine the total solution time of the algorithm. These factors also provide the balance between intensification and diversification. Table 7 contains the average distances for three iteration and threshold combinations with the desired average solution time.

Table 7 – Threshold / Iteration Combinations for JTS

Jump Point Threshold	Iterations	R1	C1	RC1	R2	C2	RC2	Total
10	100	1230.27	832.91	1399.79	976.17	593.16	1146.76	6179.06
15	75	1230.82	832.77	1401.31	972.88	593.16	1151.72	6182.65
20	50	1234.61	832.77	1406.10	972.48	593.16	1148.66	6187.78

Although there is no statistical difference between the different combinations, our intuition leads us to choose a threshold of 10 jump points and a base of 100 iterations.

4.5 Heuristic Analysis

To compare the performance of our algorithms, we solved the Solomon problem sets with each algorithm attempting to minimize distance. The distances produced by the algorithms are not normally distributed, so we use non-parametric techniques for comparison. Table 8 contains the minimum distance found for each problem by each algorithm.

We use the sign test to compare our algorithms. We perform a pair-wise comparison of the algorithms two at a time and count the number of occurrences where algorithm one is better than algorithm two. We determine the probability of this count based on a binomial distribution with 50% probability. If the two algorithms are equal in efficiency, we would expect each algorithm to have an equal chance of producing the better answer.

With the exception of the JTS algorithm, none of the algorithms proved statistically superior to the others. The JTS algorithm proved statistically superior to the JFB and JGB algorithms with p-values of less than 0.0001 supporting our hypothesis that exploring the jump points beyond the first local optimal improves the algorithm.

Table 8 – Minimum Distances Found by All Algorithms

Problem	JFB	JGB	JTS	RTS - best	RTS - one
r101	1712.64	1704.81	1678.37	1707.18	1734.13
r102	1535.42	1529.47	1485.33	1500.62	1482.81
r103	1289.28	1296.38	1266.67	1247.28	1240.34
r104	1068.62	1056.90	1032.15	1066.83	1085.20
r105	1433.37	1433.89	1404.25	1446.50	1389.22
r106	1315.17	1296.50	1291.17	1304.70	1332.21
r107	1145.48	1159.27	1123.80	1147.27	1106.34
r108	1049.11	1030.68	992.83	1063.37	1063.37
r109	1259.32	1228.25	1215.47	1276.89	1271.51
r110	1171.88	1170.23	1153.36	1162.44	1212.26
r111	1137.59	1130.52	1107.82	1135.11	1097.74
r112	1032.75	1042.18	1012.07	1001.88	1058.33
R1 Average	1262.55	1256.59	1230.27	1255.01	1256.12
c101	828.93	828.93	828.93	828.93	828.93
c102	828.93	828.93	828.93	936.72	928.01
c103	839.57	831.86	829.27	829.27	836.67
c104	871.34	864.37	864.37	859.39	843.65
c105	828.93	828.93	828.93	828.93	828.93
c106	828.93	828.93	828.93	828.93	862.26
c107	828.93	828.93	828.93	828.93	828.93
c108	828.93	828.93	828.93	848.93	828.93
c109	828.93	828.93	828.93	828.93	828.93
C1 Average	834.82	833.19	832.91	846.55	846.14
rc101	1694.74	1694.75	1693.47	1726.92	1690.05
rc102	1515.44	1507.35	1500.68	1520.47	1522.35
rc103	1384.37	1384.71	1362.87	1331.54	1364.30
rc104	1221.98	1238.79	1216.71	1181.71	1220.52
rc105	1604.82	1612.19	1558.09	1611.83	1598.35
rc106	1435.26	1428.87	1419.05	1461.27	1443.11
rc107	1303.18	1334.34	1280.41	1270.37	1376.36
rc108	1213.36	1213.16	1167.04	1233.66	1197.25
RC1 Average	1421.64	1426.77	1399.79	1417.22	1426.54
r201	1291.00	1281.64	1324.45	1298.40	1329.24
r202	1120.05	1140.40	1127.76	1133.52	1108.41
r203	969.90	976.44	967.56	1032.61	947.55
r204	830.48	817.28	810.44	803.14	839.18
r205	1126.92	1134.40	1073.52	1046.21	1098.40
r206	983.72	991.80	996.07	990.65	972.62
r207	905.39	904.94	894.59	884.83	896.74
r208	784.72	770.42	749.85	748.28	784.46
r209	1010.87	986.97	969.93	952.30	947.99
r210	991.23	979.51	966.67	981.90	1050.54
r211	863.43	866.17	857.06	899.51	832.97
R2 Average	988.88	986.36	976.17	979.21	982.55
c201	591.55	591.55	591.55	591.55	591.55
c202	591.55	591.55	591.55	591.55	591.55
c203	628.81	617.61	617.61	591.17	591.17
c204	622.34	590.59	590.59	590.59	878.62
c205	588.87	588.87	588.87	588.87	588.87
c206	588.49	588.49	588.49	588.49	588.49
c207	588.28	588.28	588.28	588.28	588.28
c208	588.32	588.32	588.32	588.32	588.32
C2 Average	598.53	593.16	593.16	589.85	625.86
rc201	1449.02	1480.89	1409.22	1578.30	1504.80
rc202	1327.54	1305.57	1305.57	1383.43	1248.23
rc203	1107.86	1142.96	1102.81	1131.34	1182.34
rc204	880.26	917.30	876.59	913.57	913.57
rc205	1342.63	1316.62	1315.15	1433.18	1437.76
rc206	1194.53	1218.82	1172.22	1289.79	1187.08
rc207	1106.25	1134.32	1123.77	1193.56	1065.96
rc208	945.37	884.84	868.76	868.76	894.58
RC2 Average	1169.18	1175.17	1146.76	1223.99	1179.29
Overall Average	1054.58	1053.37	1037.77	1058.91	1059.83

As expected, the JFB and JGB algorithms are both statistically faster in solve time with p-values of less than 0.0001. Table 9 contains the average CPU time to find the best solution and the average total solution CPU time (in seconds on a Pentium II 400 MHz processor) for all algorithms. All algorithms are coded using Java 1.2.

Table 9 – Average Solve Times for All Algorithms

Average Time (secs)	JFB	JGB	JTS	RTS - best	RTS – one
Best Found	19.50	19.58	20.32	19.54	12.33
Total	26.40	33.79	56.49	55.54	32.33

The JTS proved statistically superior to both RTS algorithms with p-values of 0.044 for RTS-best and 0.036 for RTS-one. This supports our hypothesis that multiple starting points are an effective diversification and intensification strategy for tabu search.

We did not find any statistical difference in best found or total run times between the JTS and the RTS-best algorithm. The RTS-one algorithm is statistically faster than the JTS algorithm, as RTS-one does not generate multiple jump points.

4.6 Comparison to O'Rourke's Reactive Tabu Search Algorithm

Admittedly, our RTS algorithm lacks any complex or sophisticated search strategies. We compare our results to those produced by O'Rourke and Ryer's RTS algorithm which incorporates a reactive penalty scheme as well as a reactive tabu list length (O'Rourke 1999, Ryer 1999). Their algorithm is written in Java and executed on a Pentium II 400MHz processor just as our JTS algorithm.

Tables 10, 11, 12 contain the minimum distances found, the time the best solution was found, and total solve time for both O'Rourke and Ryer's RTS and our JTS

algorithms. The 25 and 50 customer problems are generated by using the first 25 and 50 customers of the 100 customer problems. Distances between customers are truncated to the tenths digit before each problem is solved. From the tables, it is clear JTS achieves the same quality of solutions with significant savings in computational effort.

The first column shows the problem number. Four columns a provided for each algorithm denoting the minimum distance found, number of vehicles used, the CPU time the best solution was found (in seconds), and the total CPU run time of the algorithm. O'Rourke and Ryer only provide an approximate average total time for each problem size. The final three columns show the percent difference for the JTS algorithm from O'Rourke and Ryer in terms of minimum distance, time best found, and total run time.

Table 10 – Comparison on Solomon 25 Customer Problems

25 Customer Problems	O'Rourke & Ryer				Jump Tabu Search				Percent Difference		
	Distance	Number of Vehicles	Time Best Found	Approx. Total Time	Distance	Number of Vehicles	Time Best Found	Total Time	Distance	Time Best Found	Total Time
r101	617.1	8.0	4	28	617.1	8.0	0.38	1.15	0.0%	-90.5%	-95.9%
r102	547.1	7.0	1	28	547.1	7.0	0.38	1.87	0.0%	-62.0%	-93.3%
r103	454.1	5.0	1	28	463.5	6.0	0.44	1.53	2.1%	-56.0%	-94.5%
r104	416.9	4.0	2	28	436.0	5.0	0.33	1.82	4.6%	-83.5%	-93.5%
r105	530.5	6.0	1	28	530.5	6.0	0.44	1.15	0.0%	-56.0%	-95.9%
r106	465.4	5.0	12	28	465.4	5.0	0.38	2.09	0.0%	-96.8%	-92.5%
r107	424.3	4.0	24	28	424.3	4.0	0.55	1.76	0.0%	-97.7%	-93.7%
r108	397.3	4.0	1	28	397.3	4.0	0.60	1.65	0.0%	-40.0%	-94.1%
r109	441.3	5.0	1	28	441.3	5.0	0.44	1.49	0.0%	-56.0%	-94.7%
r110	444.1	5.0	1	28	444.1	5.0	0.55	1.31	0.0%	-45.0%	-95.3%
r111	428.8	4.0	3	28	438.3	4.0	0.44	1.82	2.2%	-85.3%	-93.5%
r112	393.0	4.0	1	28	402.0	4.0	0.55	1.75	2.3%	-45.0%	-93.8%
R1 Average	463.3	5.1	4.3	28.0	467.2	5.3	0.5	1.6	0.9%	-67.8%	-94.2%
c101	191.3	3.0	0	28	191.3	3.0	0.11	1.16	0.0%	N/A	-95.9%
c102	190.3	3.0	1	28	190.3	3.0	0.55	1.97	0.0%	-45.0%	-93.0%
c103	190.3	3.0	1	28	190.3	3.0	0.55	1.82	0.0%	-45.0%	-93.5%
c104	186.9	3.0	8	28	190.0	3.0	0.61	1.65	1.7%	-92.4%	-94.1%
c105	191.3	3.0	1	28	191.3	3.0	0.33	1.21	0.0%	-67.0%	-95.7%
c106	191.3	3.0	1	28	191.3	3.0	0.16	1.21	0.0%	-84.0%	-95.7%
c107	191.3	3.0	0	28	191.3	3.0	0.38	1.26	0.0%	N/A	-95.5%
c108	191.3	3.0	4	28	191.3	3.0	0.55	1.37	0.0%	-86.3%	-95.1%
c109	191.3	3.0	2	28	191.3	3.0	0.33	1.60	0.0%	-83.5%	-94.3%
C1 Average	190.6	3.0	2.0	28.0	190.9	3.0	0.4	1.5	0.2%	-71.9%	-94.7%
rc101	461.1	4.0	2	28	461.1	4.0	0.44	1.21	0.0%	-78.0%	-95.7%
rc102	351.7	3.0	1	28	351.8	3.0	0.22	1.32	0.0%	-78.0%	-95.3%
rc103	332.8	3.0	2	28	332.8	3.0	0.33	1.43	0.0%	-83.5%	-94.9%
rc104	306.6	3.0	1	28	308.7	3.0	0.49	1.37	0.7%	-51.0%	-95.1%
rc105	411.2	4.0	1	28	416.1	4.0	0.05	1.21	1.2%	-95.0%	-95.7%
rc106	345.5	3.0	1	28	345.5	3.0	0.39	1.32	0.0%	-61.0%	-95.3%
rc107	298.3	3.0	2	28	298.3	3.0	0.55	1.43	0.0%	-72.5%	-94.9%
rc108	294.5	3.0	3	28	294.5	3.0	0.55	1.48	0.0%	-81.7%	-94.7%
RC1 Average	350.2	3.3	1.6	28.0	351.1	3.3	0.4	1.3	0.2%	-75.1%	-95.2%
Overall	347.5	3.9	2.9	28.0	349.5	4.0	0.4	1.5	0.5%	-71.0%	-94.7%

Table 11 – Comparison on Solomon 50 Customer Problems

50 Customer Problems	O'Rourke & Ryer				Jump Search				Percent Difference		
	Distance	Number of Vehicles	Time Best Found	Approx. Total Time	Distance	Number of Vehicles	Time Best Found	Total Time	Distance	Time Best Found	Total Time
r101	1043.8	12.0	9	100	1051.4	13.0	1.32	5.77	0.7%	-85.3%	-94.2%
r102	909.0	11.0	82	100	916.0	12.0	2.03	6.37	0.8%	-97.5%	-93.6%
r103	778.7	9.0	87	100	781.5	9.0	2.31	7.80	0.4%	-97.3%	-92.2%
r104	637.4	6.0	69	100	635.8	6.0	1.54	8.29	-0.3%	-97.8%	-91.7%
r105	901.6	9.0	16	100	916.9	10.0	1.54	6.81	1.7%	-90.4%	-93.2%
r106	793.0	8.0	99	100	798.5	8.0	2.31	6.54	0.7%	-97.7%	-93.5%
r107	711.1	7.0	79	100	723.1	7.0	0.66	6.76	1.7%	-99.2%	-93.2%
r108	617.7	6.0	78	100	630.2	6.0	2.19	13.73	2.0%	-97.2%	-86.3%
r109	786.7	8.0	61	100	814.9	9.0	2.03	6.75	3.6%	-96.7%	-93.3%
r110	707.8	7.0	84	100	697.0	7.0	2.04	7.20	-1.5%	-97.6%	-92.8%
r111	716.6	7.0	76	100	722.0	7.0	2.48	13.02	0.8%	-96.7%	-87.0%
r112	635.0	6.0	68	100	652.3	7.0	2.75	7.53	2.7%	-96.0%	-92.5%
R1 Average	769.9	8.0	67.3	100.0	778.3	8.4	1.9	8.0	1.1%	-95.8%	-92.0%
c101	362.4	5.0	3	100	362.4	5.0	1.75	5.65	0.0%	-41.7%	-94.4%
c102	361.4	5.0	9	100	361.4	5.0	2.15	5.72	0.0%	-76.1%	-94.3%
c103	361.4	5.0	87	100	361.4	5.0	2.14	6.59	0.0%	-97.5%	-93.4%
c104	382.8	5.0	79	100	364.9	5.0	2.85	6.20	-4.7%	-96.4%	-93.8%
c105	362.4	5.0	19	100	362.4	5.0	2.04	5.94	0.0%	-89.3%	-94.1%
c106	362.4	5.0	4	100	362.4	5.0	1.92	5.65	0.0%	-52.0%	-94.4%
c107	362.4	5.0	6	100	362.4	5.0	1.76	5.82	0.0%	-70.7%	-94.2%
c108	362.4	5.0	4	100	362.4	5.0	1.93	6.05	0.0%	-51.8%	-94.0%
c109	362.4	5.0	26	100	362.4	5.0	1.59	5.82	0.0%	-93.9%	-94.2%
C1 Average	364.4	5.0	26.3	100.0	362.5	5.0	2.0	5.9	-0.5%	-74.4%	-94.1%
rc101	946.8	8.0	60	100	948.9	8.0	1.98	6.10	0.2%	-96.7%	-93.9%
rc102	831.8	7.0	60	100	843.4	8.0	1.43	7.09	1.4%	-97.6%	-92.9%
rc103	710.9	6.0	94	100	779.6	7.0	2.14	5.77	9.7%	-97.7%	-94.2%
rc104	546.5	5.0	18	100	548.2	5.0	2.14	5.16	0.3%	-88.1%	-94.8%
rc105	855.3	8.0	4	100	859.8	8.0	0.87	10.32	0.5%	-78.3%	-89.7%
rc106	723.2	6.0	58	100	765.6	6.0	0.94	6.21	5.9%	-98.4%	-93.8%
rc107	644.4	6.0	36	100	652.5	6.0	1.43	5.55	1.3%	-96.0%	-94.5%
rc108	598.1	6.0	58	100	603.9	6.0	2.58	9.06	1.0%	-95.6%	-90.9%
RC1 Average	732.1	6.5	48.5	100.0	750.2	6.8	1.7	6.9	2.5%	-93.5%	-93.1%
Overall	633.6	6.7	49.4	100.0	641.5	6.9	1.9	7.1	1.0%	-88.5%	-92.9%

Table 12 – Comparison on Solomon 100 Customer Problems

100 Customer Problems	O'Rourke & Ryer				Jump Search				Percent Difference		
	Distance	Number of Vehicles	Time Best Found	Approx. Total Time	Distance	Number of Vehicles	Time Best Found	Total Time	Distance	Time Best Found	Total Time
r101	1676.2	20.0	414	550	1672.8	20.0	8.41	37.90	-0.2%	-98.0%	-93.1%
r102	1502.4	19.0	96	550	1493.9	19.0	9.61	41.64	-0.6%	-90.0%	-92.4%
r103	1265.0	15.0	228	550	1245.6	15.0	8.29	58.66	-1.5%	-96.4%	-89.3%
r104	1039.6	12.0	338	550	1038.7	12.0	11.92	63.05	-0.1%	-96.5%	-88.5%
r105	1399.4	16.0	378	550	1397.5	15.0	8.13	40.76	-0.1%	-97.8%	-92.6%
r106	1268.4	14.0	491	550	1286.3	13.0	9.33	34.93	1.4%	-98.1%	-93.6%
r107	1129.0	13.0	406	550	1086.8	11.0	9.61	38.56	-3.7%	-97.6%	-93.0%
r108	956.8	10.0	565	550	990.7	11.0	12.02	46.74	3.5%	-97.9%	-91.5%
r109	1181.0	14.0	311	550	1223.7	12.0	9.39	57.78	3.6%	-97.0%	-89.5%
r110	1133.2	13.0	328	550	1139.0	12.0	10.98	38.39	0.5%	-96.7%	-93.0%
r111	1077.3	12.0	491	550	1096.4	12.0	10.93	48.17	1.8%	-97.8%	-91.2%
r112	971.6	11.0	460	550	978.4	11.0	12.63	46.63	0.7%	-97.3%	-91.5%
R1 Average	1216.7	14.1	375.5	550.0	1220.8	13.6	10.1	46.1	0.4%	-96.7%	-91.6%
c101	827.3	10.0	43	550	827.3	10.0	6.86	28.56	0.0%	-84.0%	-94.8%
c102	827.3	10.0	253	550	827.3	10.0	7.58	25.70	0.0%	-97.0%	-95.3%
c103	828.9	10.0	535	550	827.4	10.0	9.73	25.44	-0.2%	-98.2%	-95.4%
c104	950.0	10.0	509	550	863.6	10.0	11.31	29.98	-9.1%	-97.8%	-94.5%
c105	827.3	10.0	65	550	827.3	10.0	0.77	27.02	0.0%	-98.8%	-95.1%
c106	827.3	10.0	55	550	827.3	10.0	8.24	26.09	0.0%	-85.0%	-95.3%
c107	827.3	10.0	210	550	827.3	10.0	2.80	28.12	0.0%	-98.7%	-94.9%
c108	827.3	10.0	321	550	827.3	10.0	8.13	24.22	0.0%	-97.5%	-95.6%
c109	853.3	10.0	463	550	827.3	10.0	6.21	26.64	-3.0%	-98.7%	-95.2%
C1 Average	844.0	10.0	272.7	550.0	831.3	10.0	6.8	26.9	-1.4%	-95.1%	-95.1%
rc101	1669.9	16.0	381	550	1688.2	16.0	6.86	39.21	1.1%	-98.2%	-92.9%
rc102	1498.4	15.0	419	550	1510.9	15.0	8.40	36.52	0.8%	-98.0%	-93.4%
rc103	1363.6	13.0	270	550	1320.7	12.0	10.60	37.74	-3.1%	-96.1%	-93.1%
rc104	1179.2	11.0	308	550	1206.8	11.0	10.76	30.75	2.3%	-96.5%	-94.4%
rc105	1557.4	15.0	473	550	1557.3	15.0	6.37	36.03	0.0%	-98.7%	-93.4%
rc106	1432.8	13.0	434	550	1415.2	13.0	7.58	53.72	-1.2%	-98.3%	-90.2%
rc107	1266.1	12.0	417	550	1269.7	12.0	9.99	54.37	0.3%	-97.6%	-90.1%
rc108	1175.1	12.0	475	550	1170.4	11.0	11.37	30.16	-0.4%	-97.6%	-94.5%
RC1 Average	1392.8	13.4	397.1	550.0	1392.4	13.1	9.0	39.8	0.0%	-97.6%	-92.8%
Overall	1149.6	12.6	349.6	550.0	1147.3	12.3	8.8	38.4	-0.3%	-96.5%	-93.0%

4.7 Comparison to Best Known Solutions for Solomon's MVRPTW Instances

More commonly, the Solomon instances are solved by minimizing vehicles first, then total distance. Table 13 contains the solutions produced by the JTS algorithm based on this criterion. The solutions are compared to the best known solutions for each of the problems. Minimum vehicles is achieved by attempting to solve each problem with the amount of vehicles used by the best known solution for that problem. If JTS is unable to find a solution that visits all of the customers with the minimum number of vehicles, the number of vehicles is increased by one until JTS can find a complete solution.

Columns 2 and 3 contain the minimum distance and number of vehicles for the best known solution. Column 4 contains a reference for the source of the best known solution. Columns 5 and 6 contain the minimum distance and number of vehicles found by JTS. The last two columns show the percent difference in minimum distance and number of vehicles for JTS over the best known solutions. The best known solutions to the C1 problem set are proven optimal solutions and were solved with distance between customers truncated to tenths digit before the problem is solved.

While it is difficult to compare CPU times across different programming languages and processors, the algorithms producing the best known solutions used significantly more CPU time than JTS. In some cases, the algorithms were executed multiple times with different random seeds. For JTS, the average CPU time to find the best solution is 31.3 seconds and the average total solve time is 48.8 seconds on a Pentium II 400 MHz processor.

Table 13 – Comparison to Best Known Solutions for Solomon Instances

Problem	Best Known		Source	JTS		Percentage from Best	
	Distance	NV		Distance	NV	Distance	NV
r101	1607.70	18	Desrochers *et al* 1992	1679.42	19	4.5%	5.6%
r102	1434.00	17	Desrochers *et al* 1992	1485.33	18	3.6%	5.9%
r103	1207.00	13	Thangiah *et al* 1994	1295.77	14	7.4%	7.7%
r104	1007.31	9	Shaw 1997	1100.88	10	9.3%	11.1%
r105	1377.10	14	Rochat and Taillard 1995	1449.30	14	5.2%	0.0%
r106	1252.03	12	Rochat and Taillard 1995	1357.88	12	8.5%	0.0%
r107	1104.66	10	Shaw 1997	1109.70	11	0.5%	10.0%
r108	963.99	9	Shaw 1997	994.10	10	3.1%	11.1%
r109	1205.96	11	Shaw 1997	1274.81	12	5.7%	9.1%
r110	1135.07	10	Shaw 1997	1171.37	11	3.2%	10.0%
r111	1096.73	10	Shaw 1997	1144.94	11	4.4%	10.0%
r112	953.63	10	Rochat and Taillard 1995	1022.16	10	7.2%	0.0%
R1 Average	1195.43	11.92		1257.14	12.67	5.16%	6.29%
c101	827.30	10	Desrochers *et al* 1992	828.93	10	0.2%	0.0%
c102	827.30	10	Desrochers *et al* 1992	828.93	10	0.2%	0.0%
c103	826.30	10	Kohl and Madsen 1997	829.27	10	0.4%	0.0%
c104	822.90	10	Kohl and Madsen 1997	864.37	10	5.0%	0.0%
c105	827.30	10	Kohl and Madsen 1997	828.93	10	0.2%	0.0%
c106	827.30	10	Desrochers *et al* 1992	828.93	10	0.2%	0.0%
c107	827.30	10	Desrochers *et al* 1992	828.93	10	0.2%	0.0%
c108	827.30	10	Desrochers *et al* 1992	828.93	10	0.2%	0.0%
c109	827.30	10	Kohl and Madsen 1997	828.93	10	0.2%	0.0%
C1 Average	826.70	10.00		832.91	10.00	0.75%	0.00%
rc101	1669.00	14	Thangiah *et al* 1994	1707.31	15	2.3%	7.1%
rc102	1554.75	12	Taillard *et al* 1997	1576.64	14	1.4%	16.7%
rc103	1110.00	11	Thangiah *et al* 1994	1356.07	12	22.2%	9.1%
rc104	1135.48	10	Shaw 1997	1216.71	11	7.2%	10.0%
rc105	1643.38	13	Taillard *et al* 1997	1569.86	15	-4.5%	15.4%
rc106	1448.26	11	Taillard *et al* 1997	1454.40	12	0.4%	9.1%
rc107	1230.48	11	Shaw 1997	1289.34	12	4.8%	9.1%
rc108	1139.82	10	Taillard *et al* 1997	1171.26	11	2.8%	10.0%
RC1 Average	1366.40	11.50		1417.70	12.75	3.75%	10.87%
r201	1254.09	4	Kilby *et al* 1997	1351.94	4	7.8%	0.0%
r202	1214.28	3	Taillard *et al* 1997	1127.77	4	-7.1%	33.3%
r203	948.74	3	Rochat and Taillard 1995	962.74	3	1.5%	0.0%
r204	867.33	2	Kilby *et al* 1997	815.69	3	-6.0%	50.0%
r205	998.72	3	Kilby *et al* 1997	1098.23	3	10.0%	0.0%
r206	833.00	3	Thangiah *et al* 1994	996.07	3	19.6%	0.0%
r207	814.78	3	Rochat and Taillard 1995	894.59	3	9.8%	0.0%
r208	738.60	2	Rochat and Taillard 1995	792.82	2	7.3%	0.0%
r209	855.00	3	Thangiah *et al* 1994	982.44	3	14.9%	0.0%
r210	963.37	3	Kilby *et al* 1997	1009.91	3	4.8%	0.0%
r211	923.80	2	Taillard *et al* 1997	857.06	3	-7.2%	50.0%
R2 Average	946.52	2.82		989.93	3.09	4.59%	9.68%
c201	591.56	3	Potvin and Bengio 1996	591.55	3	0.0%	0.0%
c202	591.56	3	Potvin and Bengio 1996	591.55	3	0.0%	0.0%
c203	591.17	3	Rochat and Taillard 1995	617.61	3	4.5%	0.0%
c204	590.60	3	Potvin and Bengio 1996	590.59	3	0.0%	0.0%
c205	588.88	3	Potvin and Bengio 1996	588.87	3	0.0%	0.0%
c206	588.49	3	Potvin and Bengio 1996	588.49	3	0.0%	0.0%
c207	588.29	3	Rochat and Taillard 1995	588.28	3	0.0%	0.0%
c208	588.32	3	Rochat and Taillard 1995	588.32	3	0.0%	0.0%
C2 Average	589.86	3.00		593.16	3.00	0.56%	0.00%
rc201	1406.94	4	Kilby *et al* 1997	1513.79	4	7.6%	0.0%
rc202	1162.80	4	Kilby *et al* 1997	1310.36	4	12.7%	0.0%
rc203	1068.07	3	Kilby *et al* 1997	1139.71	3	6.7%	0.0%
rc204	803.90	3	Kilby *et al* 1997	876.59	3	9.0%	0.0%
rc205	1302.42	4	Kilby *et al* 1997	1463.48	4	12.4%	0.0%
rc206	1156.26	3	Kilby *et al* 1997	1288.20	3	11.4%	0.0%
rc207	1075.25	3	Kilby *et al* 1997	1175.33	3	9.3%	0.0%
rc208	833.97	3	Rochat and Taillard 1995	868.76	3	4.2%	0.0%
RC2 Average	1101.20	3.38		1204.53	3.38	9.38%	0.00%
Overall	1004.35	7.10		1049.23	7.49	4.03%	4.47%

4.8 Multiple Depot Problems

The Solomon problem instances do not model such factors as multiple depots, a heterogeneous vehicle fleet, customer priorities, and time walls. These factors are modeled in our algorithm; however, as demonstrated in the previous section, these factors do not affect the quality of the solution when they are not present in the problem.

Cordeau *et al* (1997) develops 10 randomly generated MD VRPs to test their tabu search heuristic. These problems vary in terms of number and location of customers, number of depots, number of vehicles, maximum route length, and vehicle capacity. The objective is to find the shortest tour visiting all of the customers. Table 14 compares the distances found by the JTS algorithm to those reported by Cordeau *et al* (1997).

Table 14 – Comparison on Cordeau *et al* MD VRPs

Problem Number	Number of Vehicles	Number of Depots	Route Length	Vehicle Capacity	Number of Customers	Cordeau *et al* Best Distance	JTS Best Distance	Percent Difference
pr01	1	4	500	200	48	861.32	909.89	5.64%
pr02	2	4	480	195	96	1307.61	1386.11	6.00%
pr03	3	4	460	190	144	1806.60	1910.31	5.74%
pr04	4	4	440	185	192	2072.52	2198.60	6.08%
pr05	5	4	420	180	240	2385.77	2570.21	7.73%
pr06	6	4	400	175	288	2723.27	3000.13	10.17%
pr07	1	6	500	200	72	1089.56	1132.24	3.92%
pr08	2	6	475	190	144	1666.60	1802.66	8.16%
pr09	3	6	450	180	216	2153.10	2370.79	10.11%
pr10	4	6	425	170	288	2921.85	3159.48	8.13%
Average						1898.82	2044.04	7.65%

As seen in Table 14, JTS does an admirable job on these MD VRPs coming within 10% or better of the best known solution across all ten problem instances.

In addition, Cordeau et al (1997) compile benchmark problems from the literature and achieve new best known solutions to all but two of them. Table 15 compares the

distances found by JTS to the best known solutions. Problems 1 through 7 are found in Christofides and Eilon (1969). Problems 8 through 11 are found in Gillett and Johnson (1976). Problems 12 through 23 are found in Chao *et al* (1993).

Table 15 – Comparison on MD VRPs from Literature

Problem Number	Number of Vehicles	Number of Depots	Route Length	Vehicle Capacity	Number of Customers	Best Known Distance	JTS Best Distance	Percent Difference
p01	4	4	infinite	80	50	576.87	601.70	4.30%
p02	2	4	infinite	160	50	473.53	495.00	4.53%
p03	3	5	infinite	140	75	641.19	685.99	6.99%
p04	8	2	infinite	100	100	1001.59	1074.83	7.31%
p05	5	2	infinite	200	100	750.03	811.52	8.20%
p06	6	3	infinite	100	100	876.50	927.01	5.76%
p07	4	4	infinite	100	100	885.80	994.40	12.26%
p08	14	2	310	500	249	4437.68	4723.49	6.44%
p09	12	3	310	500	249	3900.22	4287.41	9.93%
p10	8	4	310	500	249	3663.02	4242.42	15.82%
p11	6	5	310	500	249	3554.18	4129.99	16.20%
p12	5	2	infinite	60	80	1318.95	1453.82	10.23%
p13	5	2	200	60	80	1318.95	1357.48	2.92%
p14	5	2	180	60	80	1360.12	1365.68	0.41%
p15	5	4	infinite	60	160	2505.42	2741.75	9.43%
p16	5	4	200	60	160	2572.23	2658.05	3.34%
p17	5	4	180	60	160	2709.09	2731.36	0.82%
p18	5	6	infinite	60	240	3702.85	4371.05	18.05%
p19	5	6	200	60	240	3827.06	3951.66	3.26%
p20	5	6	180	60	240	4058.07	4097.04	0.96%
p21	5	9	infinite	60	360	5474.84	6486.66	18.48%
p22	5	9	200	60	360	5702.16	6014.87	5.48%
p23	5	9	180	60	360	6095.46	6145.56	0.82%
Average						2669.82	2884.73	8.05%

JTS achieves solutions within 10% of the best known in a short amount of time. For JTS, the average CPU time to find the best solution is 74.4 seconds and the average total solve time is 170.3 seconds on a Pentium II 400 MHz processor.

For problems 12 through 23, there is a clear pattern for the performance of JTS. As the vehicle capacity gets smaller, the quality of the JTS solution improves. Since JTS maintains feasibility throughout the algorithm, we hypothesize that the performance of

JTS is improved on problems with a smaller feasible solution space, *i.e.* tightly constrained problems.

4.9 Conclusions

Our JTS algorithm provides quick, feasible, high-quality solutions. The algorithm handles many additional factors, such as, multiple depots, multiple heterogeneous vehicles, time windows, time walls, and priorities. These factors combined with the multiple parameter sets for solution generation make JTS a robust algorithm capable of solving many variations of the vehicle routing problem.

JTS does not seriously challenge the best known solutions for the Solomon or MDVRP benchmark problems. However, the results achieved by JTS are within 10% of the best known solutions and the solution time is significantly shorter. We recognize that we have not modeled the UAV problem with 100% accuracy. However, in practical UAV applications, solution speed is the most important attribute of the solution algorithm. Therefore, it is not cost-effective to spend excess time searching for a 99% solution to the model when it is quite likely that the 90% solution will suffice.

A 99% solution is probably not achievable by the JTS algorithm. Even when all jump points are explored, the solutions are only improved by approximately 2%. The algorithm finds the good quality 90% solutions quickly, but likely converges too quickly to achieve the remaining 10%. Additional diversification strategies are likely needed to achieve the additional solution quality improvement.

Chapter 5. Recommendations for Further Research

5.1 Modeling UAV Realism

Weather conditions have a significant impact upon UAV missions. This is the main area in which the algorithm needs to be improved. The JTS algorithm incorporates penalty factors for each segment. Weather data can be brought into the algorithm, processed and represented using these penalties. The algorithm would then consider weather factors in building solutions.

Terrain can also impact UAV missions. Terrain features can be incorporated into the algorithm in the same manner as weather. The speed of the algorithm must be kept in mind when adding these factors.

5.2 Tour Construction

The best direction for improving the tour construction portion of the algorithm would be to produce a more diverse set of jump points. Even when all of the jump points are explored, the algorithm does not challenge the best known solutions for the Solomon or MDVRP benchmark problems.

This improvement is likely to come from new construction heuristics or new parameter values. Of the two, new construction heuristics seem more promising. The genetic sectoring algorithm developed by Thangiah *et al* (1994) in combination with the Solomon heuristic could prove very effective.

5.3 Search Techniques

Another direction in which the algorithm may be improved is with more advanced local search techniques. There are many advanced tabu and local search techniques proposed by Shaw (1997) and Taillard *et al* (1997) that may improve the solution quality.

When making these improvements, the needs of the eventual user of the algorithm must be considered. Do they require a 90% solution in a very short amount of time or can they wait longer for a potentially better solution?

5.4 Extensions to VRP

Our algorithm models many VRP factors such as heterogeneous vehicles. However, we do not 'take advantage' of heterogeneous vehicles mainly because it is unclear how to best utilize a heterogeneous vehicle fleet. Is it more efficient to use larger vehicles before smaller ones, faster ones before slower ones? Much research remains in this area.

Bibliography

Battiti, Roberto. "Reactive Search: Toward Self-Tuning Heuristics" in *Modern Heuristic Search Methods*. Ed. V.J.Rayward-Smith, and others. Wiley & Sons, Inc., (1996).

Brandao, Jose and Alan Mercer. "A Tabu Search Algorithm for the Multi-Trip Vehicle Routing and Scheduling Problem," *European Journal of Operational Research*, 100: 180-191 (1997).

Carlton, William B. *A Tabu search to the General Vehicle Routing Problem.* Ph.D. dissertation. University of Texas, Austin TX, (1995).

Chao, I.M., B.L. Golden, and E.A. Wasil. "A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves Upon Best Known Solutions," *American Journal of Mathematical and Management Sciences*, 13:371-406 (1993).

Christofides, N. and S. Eilon. "An Algorithm for the Vehicle-Dispatching Problem," *Operations Research Quarterly*, 20:309-318 (1969).

Clarke, G. and J.W. Wright. "Scheduling Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, 12: 568-581 (1964).

Cordeau, J.F., M. Gendreau, and G. Laporte. "A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems," *Networks*, 30(2): 105 (1997).

Desrochers M., J. Desrosiers, and M.M. Solomon. "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows," *Operations Research*, 40: 342-354 (1992).

Gendreau, Michel, Alian Hertz and Gilbert Laporte. "A Tabu Search Heuristic for the Vehicle Routing Problem," *Management Science*, 40(10): 1276-1290 (1994).

Gendreau, Michel, Gilbert Laporte and Frederic Semet. "A Tabu Search Heuristic for the Undirected Selective Travelling Salesman Problem," *European Journal of Operational Research*, 106: 539-545 (1998).

Gillett, B.E. and J.G. Johnson. "Multi-Terminal Vehicle-Dispatch Algorithm," *Omega*, 4: 711-718 (1976).

Gillett, B.E. and L. Miller. "A Heuristic Algorithm for the Vehicle Dispatching Problem," *Operations Research*, 22: 340-349 (1974).

Glover, Fred "Tabu Search – Part I," *ORSA Journal on Computing*, 1: 190-206 (1989).

Glover, Fred "Tabu Search – Part II," *ORSA Journal on Computing*, 2: 4-32 (1990a).

Glover, Fred. "Tabu Search: A Tutorial," *Interfaces*, 20: 74-94 (July-August 1990b).

Glover, Fred and M. Laguna. *Tabu Search.* Boston: Kluwer Academic Publishers, (1997).

Harder, Robert. *A Java Universal Vehicle Router in Support of Routing Unmanned Aerial Vehicles.* MS thesis, AFIT/GOA/ENS/00M. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, (February 2000).

Hooker, J.N. "Testing Heuristics: We Have It All Wrong," *Journal of Heuristics*, 1: 33-42 (1995).

Kilby P., P. Prosser, and P. Shaw. "Guided Local Search for the Vehicle Routing Problem," In *Proceedings of the 2nd International Conference on Meta-heuristics*, (1997).

Kohl, N. and O. B. G. Madsen. "An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangian Relaxation," *Operations Research*, 45(3): 395 (1997).

Laporte, Gilbert. "The Traveling Salesman Problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, 59: 231-247 (1992a).

Laporte, Gilbert. "The Vehicle Routing Problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, 59: 345-358 (1992b).

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization.* John Wiley & Sons Ltd, (1985).

O'Rourke, Kevin P. *Dynamic Unmanned Aerial Vehicle (UAV) Routing With a Java-Encoded Reactive Tabu Search Metaheuristic.* MS thesis, AFIT/GOA/ENS/99M. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, (February 1999).

Parker, Gary R. and Ronald L. Rardin. "An Overview of Complexity Theory in Discrete Optimizations: Part I. Concepts," *IIE Transactions*, March: 3-10 (1982a).

Parker, Gary R. and Ronald L. Rardin. "An Overview of Complexity Theory in Discrete Optimizations: Part II. Results and Implications," *IIE Transactions*, June: 3-10 (1982b).

Potvin, J.-Y. and S. Bengio. "The Vehicle Routing Problem with Time Windows—Part II: Genetic Search," *ORSA Journal on Computing*, 8(2): 165 (1996).

Rochat, Y. and F. Semet. "A Tabu Search Approach for Delivering Pet Food and Flour in Switzerland," *Journal of Operations Research Society*, 45: 1223-1246 (1994).

Rochat, Y. and E. Taillard. "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing," *Journal of Heuristics*, 1(1): 147-167 (1995).

Rosenkrantz, D.J., R.E. Stearns, and P.M. Lewis II. "An Analysis of Several Heuristics for the Traveling Salesperson Problem," *SIAM Journal on Computing*, 6: 563-581 (1977).

Ryan, Joel L., T.G. Bailey, J.T. Moore, and W.B. Carlton. "Unmanned Aerial Vehicles (UAV) Route Selection Using Reactive Tabu Search," *Military Operations Research*, 4(3): 5-24 (1999).

Ryer, David M. *Implementation of the Metaheuristic Tabu Search in Route Selection for Mobility Analysis Support System.* MS thesis, AFIT/GOA/ENS/99M-07. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, (March 1999).

Semet, Frederic and Eric Taillard. "Solving Real-Life Vehicle Routing Problems Efficiently Using Tabu Search," *Annals of Operations Research*, 41: 469-488 (1993).

Shaw, P. A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems. APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK. (June 1997).

Sisson, Mark R. *Applying Tabu Heuristic to Wind Influenced, Minimum Risk, and Maximum Expected Coverage Routes.* MS thesis, AFIT/GOR/ENS/97M. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, (February 1997).

Solomon, Marius M. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Operations Research*, 35(2): 254-265 (1987).

Taillard, E., P. Badeau, M. Gendreau, F. Guertain, and J.-Y. Potvin. "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows," *Transportation Science*, 32(2) (1997).

Thangiah, S.R., I.H. Osman, and T. Sun. *Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time*

Windows. Technical Report UKC/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury, UK. (1994).

Tsubakitani, Shigeru. and James R. Evans. "An empirical study of a new metaheuristic for the traveling salesman problem," *European Journal of Operational Research*, 104: 113-128 (1998).

"USAF Unmanned Aerial Vehicle Battlelab homepage." Excerpt from unpublished article, n. pag. http://www.wg53.eglin.af.mil/battlelab/default.htm. (25 March 1999).