EXPLORING HEURISTICS FOR THE VEHICLE ROUTING PROBLEM WITH SPLIT DELIVERIES AND TIME WINDOWS

DISSERTATION

Marcus E. McNabb, Major, USAF

AFIT-ENS-DS-14-S-19

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-DS-14-S-19

EXPLORING HEURISTICS FOR THE VEHICLE ROUTING PROBLEM WITH SPLIT DELIVERIES AND TIME WINDOWS

DISSERTATION

Presented to the Faculty

Department of Operational Sciences

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 Doctor of Philosophy

Marcus E. McNabb, B.S., M.S.

Major, USAF

September 2014

AFIT-ENS-DS-14-S-19

EXPLORING HEURISTICS FOR THE VEHICLE ROUTING PROBLEM WITH SPLIT DELIVERIES AND TIME WINDOWS

DISSERTATION

Marcus E. McNabb, B.S., M.S.

Major, USAF

Approved:

//signed//	15 Aug 2014
Jeffery D. Weir, Ph.D. (Chairman)	Date
//signed//	15 Aug 2014
Raymond R. Hill, Ph.D. (Member)	Date
//signed//	15 Aug 2014
Shane N. Hall, Lt Col, USAF, Ph.D. (Member)	Date
//signed//	15 Aug 2014
Kenneth M. Hopkinson, Ph.D. (Member)	Date

Accepted:

Adedeji B. Badiru, Ph.D. Dean, Graduate School of Engineering and Management	Date

AFIT-ENS-DS-14-S-19

Abstract

This dissertation investigates the Vehicle Routing Problem with Split Deliveries and Time Windows. This problem assumes a depot of homogeneous vehicles and set of customers with deterministic demands requiring delivery. Split deliveries allow multiple visits to a customer and time windows restrict the time during which a delivery can be made. Several construction and local search heuristics are tested to determine their relative usefulness in generating solutions for this problem. This research shows a particular subset of the local search operators is particularly influential on solution quality and run time. Conversely, the construction heuristics tested do not significantly impact either. Several problem features are also investigated to determine their impact. Of the features explored, the ratio of customer demand to vehicle ratio revealed a significant impact on solution quality and influence on the effectiveness of the heuristics tested. Finally, this research introduces an ant colony metaheuristic coupled with a local search heuristic embedded within a dynamic program seeking to solve a Military Inventory Routing Problem with multiple-customer routes, stochastic supply, and deterministic demand. Also proposed is a suite of test problems for the Military Inventory Routing Problem.

Dedicated to my wife for never wavering in her love and support

for me throughout this trying endeavor.

Acknowledgments

I owe many thanks to my advisor for his insight, knowledge, and patience in leading me on this journey. I will be forever indebted to him for his extensive help and support through this long and arduous process. I would also like to thank the rest of my committee and the other faculty members and students who provided valuable assistance in the production of this research. I feel like I have advanced academically, professionally, and personally over the past three years, and that advancement is all due to the quality of the people with whom I interacted while at AFIT.

Table of Contents

Page

List of Figures

Page

List of Tables

Page

I. Background

1.1 Introduction

This dissertation focuses on using heuristics to solve the vehicle routing problem with split deliveries and time windows. This chapter will give an overview of the problem and its military application as well as the problem formulations and the research questions this dissertation seeks to answer. Chapter II focuses on the use of local search operators on the problem while Chapter III focuses on the use of various construction heuristics as well as the impact of problem structure on the solution techniques and quality. Chapter IV applies the results of Chapters II and III to a military inventory routing problem. Finally, Chapter V discusses the original contributions of this work as well as some potential areas for future research.

1.2 Vehicle Routing Problem

In its simplest form, the capacitated vehicle routing problem (VRP) is represented as a depot with some supply of a commodity, a fleet of homogenous vehicles capable of carrying some finite capacity of that commodity, a set of destination points commonly referred to as customers—each with a demand for that commodity, and a cost associated with transporting the commodity between each customer. In some of the simplest instances of the VRP, the vehicles are not necessarily capacitated. However, any realistic implementation and all interesting applications are capacitated because solving a VRP using vehicles with unlimited capacity is effectively equivalent to solving a traveling

salesman problem (TSP). Therefore in the remainder of this document, the VRP refers to a capacitated VRP.

1.3 Military Application

Although varying objectives are used for individual problem instances, the most common objective is to minimize the total cost such that each customer's demand is satisfied. At its core, this describes many of the problems facing United States Transportation Command (USTRANSCOM) and Air Mobility Command (AMC). For example, the intra-theater routing problem considers the problem of transporting cargo loads from some number of ports of debarkation (POD) to their final destinations. In general, solutions must specify the transport mode for each requirement, the route for each requirement, and the time period of departure from each delivery node. Note departure time is preferred to arrival time because in the case of time windows the time spent at each node may include a waiting period in addition to a service time.

Hartlage [1] investigated aspects of this problem and developed an ant colony algorithm for solving the resource constrained shortest path problem. Lambert [2] studied the inter-theater airlift problem and developed a tabu search methodology to solve this aspect. Clapp [3] studied the intra-theater problem but focused on minimizing the number of vehicles. Hafich [4] solved a VRP for an intra-theater problem but his algorithm imposes the limitation that vehicles are only allowed to visit a single destination before returning to the depot. This is not a comprehensive review of the studies dedicated to military applications of the VRP but rather offers a compact

viewpoint on the current state of military research and indicates the need for further work in this area.

McCormack [5] defines another interesting military application in the form of a military inventory routing problem. The inventory routing problem is a combination of the VRP with inventory management in which the customers' demands evolve over time and there exists some optimal fulfillment strategy. The military version uses stochastic supply to account for the risk of destruction of military vehicles. However, McCormack uses direct delivery as opposed to a true routing schema, indicating the need to integrate VRP methods into his methodology.

1.4 Mathematical Formulations

1.4.1 Vehicle Routing Problem

To express the problem mathematically, consider a graph $G = (N, E)$ with vertex set $N = \{0,1,...,n\}$ and edge set $E = \{(i, j) : i, j \in N, i \neq j\}$ with $c_{ij} \geq 0$ being the cost to traverse an edge. Define vertex 0 to be the depot, meaning each vehicle must start and end at vertex 0. The set $N\backslash\{0\}$ defines the customers. Also, define *m* as the size of the vehicle fleet and *c* as the capacity of each vehicle. Associated with each vertex in the set $N \setminus \{0\}$ is some non-negative demand, q_i. Then, assuming symmetric costs on the edges, i.e., $c_{ij} = c_{ji}$ for every *i,j* pair, the formulation may be expressed as [6]:

Minimize: $\sum_{i \in N \setminus \{n\}} \sum_{j>i} c_{ij} x_{ij}$

Subject to:

$$\sum_{h<i} x_{hi} + \sum_{j>i} x_{ij} = 2 \quad \forall i \in N \setminus \{0\} \tag{1}$$

$$\sum_{j \in N \setminus \{0\}} x_{0j} = 2m \tag{2}$$

$$\sum_{i \in S} \sum_{\substack{h<i \\ h \notin S}} x_{hi} + \sum_{i \in S} \sum_{\substack{j>i \\ j \notin S}} x_{ij} \geq 2r(S) \quad \forall S \subseteq N \setminus \{0\}, S \neq \varnothing \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in N \setminus \{0\}, i < j \tag{4}$$

$$x_{0j} \in \{0,1,2\} \quad \forall j \in N \setminus \{0\} \tag{5}$$

The objective function minimizes the total routing cost where x_{ij} = 1 if the edge (i, j) belongs to the optimal solution and 0 otherwise. Constraint set (1) states that each customer is visited exactly once. Constraint (2) states that each vehicle makes a single trip. Constraint set (3) enforces both the connectivity of the solution (i.e., no disconnected subtours) and the vehicle capacity requirements where *r(S)* is the minimum number of vehicles required to service the subset *S* of customers. Constraint set (4) enforces the binary nature of edge traversal (i.e., an edge is used or it is not), with Constraint set (5) allowing for an exception in the case of an out-and-back (i.e., a vehicle visits a single customer and returns to the depot). These constraints are optimized in the sense that none are redundant with each other or unnecessary. Also, this research effort will not explore the problem of finding *r(S)*, the minimum number of vehicles required for each subset *S*. See [3] or [6] for more details on this problem.

1.4.2 Vehicle Routing Problem with Split Deliveries and Time Windows

This research will center on the vehicle routing problem with split deliveries and time windows (SDVRPTW). These characteristics are explained in detail in Chapter II. The SDVRPTW is explicitly defined by Belfiore et al. [7] and Ho and Haugland [8],

amongst others. This section uses a mixture of their formulations. First, the notation for the problem is defined. The set of customers is defined by $N = \{0, 1, 2, \ldots, n\}$, where the 0^{th} customer is defined as the depot. Each customer has a location and therefore a distance/travel time to every other customer and the depot, where the distance between customer $i \in N$ and customer $j \in N$ is defined as d_{ij}. Also associated with each customer $i \in N$ is a demand, q_i, service time, s_i, and a time window for service, (e_i, l_i), where e_i (l_i) is the earliest (latest) time service may begin. The fleet of vehicles is denoted by $V = \{1, 2, \ldots, m\}$, with each vehicle subject to a capacity, c (i.e., a homogenous fleet). For completeness, $q_0 = 0$, $s_0 = 0$, $e_0 = 0$, and $l_0 = \text{M}$, where M is an appropriate big-M value (e.g., $\text{M} = \max(d_{jh}|j,h \in N) * c/\min(q_i|i \in N)$). Another big-M value, M_{ij}, is used below in constraint (8). Each M_{ij} is constraint-specific and both Belfiore et al. [7] and Ho and Haugland [8] suggest $M_{ij} = l_i + d_{ij} - e_j$. Vehicle routes must start and end at the depot and each vehicle may be used, at most, once.

The decision variables for the problem are:

$x_{ijk} =$

$$\begin{cases} 1 & \text{if vehicle } k \in V \text{ travels directly from customer } i \in N \text{ to customer } j \in N \\ 0 & \text{otherwise} \end{cases}$$

$b_{ik} =$ time at which vehicle $k \in V$ begins service for customer $i \in N$

$y_{ik} =$ fraction of q_i, demand of customer $i \in N$, fulfilled by vehicle $k \in V$

Then, the formulation may be expressed as:

Minimize: $\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} d_{ij}x_{ijk}$

Subject to:

$$\sum_{j=1}^{n} x_{0jk} = 1 \quad \forall k \in V \tag{6}$$

$$x_{iik} = 0 \quad \forall i \in N \setminus \{0\}, k \in V \tag{7}$$

$$\sum_{i=0}^{n} x_{ipk} - \sum_{j=0}^{n} x_{pjk} = 0 \quad \forall p \in N, k \in V \tag{8}$$

$$\sum_{k=1}^{m} y_{ik} = 1 \quad \forall i \in N \setminus \{0\} \tag{9}$$

$$\sum_{i=1}^{n} q_i y_{ik} \le c \quad \forall k \in V \tag{10}$$

$$y_{ik} \le \sum_{j=0}^{n} x_{jik} \quad \forall i \in N \setminus \{0\}, k \in V \tag{11}$$

$$b_{ik} + s_i + t_{ij} - M_{ij}(1 - x_{ijk}) \le b_{jk} \quad \forall i, j \in N \setminus \{0\}, k \in V \tag{12}$$

$$e_i \le b_{ik} \le l_i \quad \forall i \in N \setminus \{0\}, k \in V \tag{13}$$

$$y_{ik} \ge 0 \quad \forall i \in N \setminus \{0\}, k \in V \tag{14}$$

$$b_{ik} \ge 0 \quad \forall i \in N \setminus \{0\}, k \in V \tag{15}$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j \in N, k \in V \tag{16}$$

The objective of the model is to minimize total travel distance. Constraint set (6) restricts the maximum fleet size to *m*. Constraint set (7) does not allow for a route to travel to the same customer on consecutive stops. Note this restriction is not enforced at the depot. Therefore, the first two constraint sets allow for "dummy" routes, meaning all vehicles are forced to leave the depot but some may return directly to the depot. In effect, this allows for a solution with no more than *m* vehicles. Constraint set (8) guarantees if a vehicle arrives at a customer, it will depart the customer. Constraint set (9) ensures all customer demands are fulfilled. Constraint set (10) ensures the capacity of each vehicle is not exceeded. Constraint set (11) indicates demand for customer $i \in N$ may only be fulfilled by vehicle $k \in V$ if vehicle *k* visits customer *i*. Constraint set (12)

enforces a minimum starting time for customers and guarantees no subtours. Constraint set (13) ensures time window constraints are satisfied. Constraint sets (14) and (15) are non-negativity constraints on the y_{ik} and b_{ik} decision variables. Constraint set (16) is a binary constraint on the x_{ijk} decision variables. This dissertation will not delve deeply into the formulation because in such a form the constraints are not useful for analyzing problem characteristics. Furthermore, the nature of the additional constraints is such that modeling them mathematically does not necessarily lend insight into the heuristic process, which this research will show represents the most promising and effective methods for generating solutions for the VRP.

1.5 Research Questions

As the next chapter shows, the literature surrounding the SDVRPTW is incomplete. In fact, very little work exists on the SDVRPTW, particularly concerning heuristics. This research effort will seek to answer several questions. First, which local search (LS) operators are most appropriate for this problem? The answer must account for solution cost, the number of vehicles required by the solution, and the run time of the algorithm. Second, should the choice vary depending on problem structure and characteristics? Is there a single "good" LS operator or set of operators that work well for all problem types? Third, given a strong LS, how does the construction phase and quality of the initial solution impact the overall result in terms of both solution quality and run time? Finally, the results of these questions are used to formulate a metaheuristic for solving the SDVRPTW which is then applied to a military instance of the inventory

routing problem. The literature review in the next chapter will illustrate the gaps in literature these questions seek to fill.

II. Literature Review

2.1 Introduction

Chapter I gave an overview of the problem and laid out the research questions this dissertation seeks to answer. This chapter gives a thorough overview of the current state of literature for the aspects covered by this dissertation, namely the use of ACO and other metaheuristics and local search operators in generating heuristic solutions for the VRP and some of its variants. This review shows that the research questions are not currently answered by any available sources.

2.2 Vehicle Routing Problem Variants

Chapter 1 gives a mathematical model representing the VRP. But even this formulation is a simplistic view of the vehicle routing problem. Most examples of transportation problems found in practice, including the military applications discussed in Chapter 1, have some combination of complicating factors. Split deliveries and time windows are two of the more common extensions to the VRP.

2.2.1 Time windows

In a VRP with time windows (VRPTW), any customer may have a time window associated with it. For example, when delivering a product to a store, that delivery must be made during the store's business hours so the store is able to receive the shipment. Typical formulations allow a delivery vehicle to arrive at a customer prior to the beginning of the time window but delivery of the commodity is not allowed until the

Table 1: VRP Literature Review

	Time Windows VRP	Split Delivery VRP	ACO	LS	Experimental Design/Analysis	Military Application
Archetti and Speranza (2008)		●		●		
Archetti and Speranza (2012)	●	●				
Archetti et al. (2006)		●		●		
Archetti et al. (2008)		●		●		
Belfiore et al. (2008)	●	●				
Campos et al. (2006)	●	●				
Chen et al. (2007)		●		●		
Cordeau et al. (2001)	●			●		
Cordeau et al. (2002)	●			●		
Dror and Trudeau (1989)		●		●		
El-Sherbeny (2010)	●		●	●		
Frizzell and Griffin (1995)	●	●				
Gendreau et al. (2002)			●	●		
Gulczynski et al. (2010)		●				
Ho et al. (2008)				●		
Kallehauge (2007)	●					
Laporte (2007)			●	●		
Renaud et al. (1996)				●		
Soeanu et al. (2011)		●				
Tan et al. (2001)	●					
Vacca and Salani (2009)	●	●				

beginning of the time window. This time window characteristic is intrinsic to virtually all problems of military application.

Table 1 gives an overview of the background and the sources are detailed further here. Kallehauge [9] gives a review of exact methods for the VRPTW, focusing mainly on path-formulation and decomposition methods. Cordeau et al. [10] give a synopsis of the major efforts done for VRPTW through 2002, showing heuristic methods are the most

promising avenue for large-scale VRPTW and concluding hybrid heuristics offer the best chances for meaningful progress. El-Sheberny [11] gives a more recent overview of methods for the VRPTW, also noting heuristics tend to outperform exact methods on problems of realistic size in that heuristics often deliver near-optimal solutions using significantly less computing time. El-Sheberny also notes most heuristics are problem-specific and are only useful on the problem for which they were developed. Tan et al. [12] implemented simulated annealing, tabu search and genetic algorithm approaches for the VRPTW, achieving some best-to-date results. They conclude a LS involving the λ-interchange operator is the cornerstone for the more complex heuristic implementations. Strength is also given to this argument by the fact that nearly all of the approaches surveyed here relied upon a LS. See [11] for a formal mathematical model of the VRPTW.

2.2.2 Split Delivery

In the most basic instance of the VRP defined in Chapter 1, only a single visit to each customer is allowed. This assumes a single vehicle is able to carry all of the customers' demands and the constraint forces the vehicle to do so. In cases where this assumption is met, incorporating this characteristic may make sense because visiting a customer multiple times is not only inefficient but is generally not desirable for the customer. However, the general USTRANSCOM problem does not fit this assumption. The demands of the customer may (and often will) exceed the capacity of any single vehicle, thus requiring multiple vehicles to visit the customer. Even in cases where the

demands fit onto a single vehicle, lower cost solutions may be possible by allowing split deliveries. This variant is known as the split delivery VRP (SDVRP).

Archetti and Speranza [13] offer a survey on the SDVRP in which they show the optimal solution to the SDVRP is no worse than the optimal solution to the analogous VRP (i.e., the same problem with split deliveries not allowed). Archetti and Speranza also survey the most popular techniques developed at the time: a LS algorithm developed by Dror and Trudeau [14], a tabu search developed by Archetti et al. [15], and an optimization-based approach developed by Archetti et al. [16]. The survey shows, in general, the optimization approach outperforms the tabu search which in turn outperforms the LS algorithm. Chen et al. [17] also outperform tabu search using a hybrid integer programming/LS approach. Gulczynski et al. [18] use a hybrid mixed integer program and record-to-record travel to solve the SDVRP with minimum delivery amounts. See [19] for a formal mathematical model of the SDVRP.

2.2.3 Split Delivery and Time Windows

Ho and Haugland [8] also use a tabu search to generate solutions for the split delivery VRP with time windows (SDVRPTW) while Belfiore et al. [7] use a scatter search. Favaretto et al. [20] adapt the approach of Gambardella et al. [21] for use in a VRP with multiple time windows and multiple customer visits but do not incorporate LS into their procedure. This algorithm could be considered a solution method for SDVRPTW by restricting the problem to single time windows and considering multiple customer visits as a split delivery. Frizzell and Giffin [22] use a novel construction heuristic along with exchange and relocate operators to solve the SDVRPTW with grid

distances. Vacca and Salani [23] use Dantzig-Wolfe decomposition and a branch-and-price algorithm to solve the VRPTW with discrete split delivery, meaning customer demands consist of discrete items that cannot be split. Campos et al. [24] use a genetic algorithm to solve the SDVRPTW. See [25] for a more comprehensive review of the SDVRPTW, including efforts toward exact solutions.

2.3 Algorithms and Methods

Researchers have applied various methods to solve the VRP, ranging from exact algorithms to heuristic methods. This research effort will focus on heuristic methods because the VRP is NP-hard and in practice even the most sophisticated exact algorithms can only handle relatively small instances (<100 nodes) [26].

2.3.1 Evolutionary Algorithms

Evolutionary algorithms are well-known and explored to a great extent in the literature. As a representative example, consider the genetic algorithm given by Baker and Ayechew [27] for solving a VRP. Their algorithm follows the general guidelines for a genetic algorithm in that some initial population of solutions is generated. "Good" solutions are then cross-pollinated and mutated to generate new, and hopefully better, solutions. This process continues until either a time or iteration count is reached or the current solution (or solution set) is deemed "good enough" according to a pre-defined stopping rule.

To generate the initial solutions, Baker and Ayechew use a combination of random solutions with "sorted" solutions. In the sorted solutions, customers are sorted using some metric (e.g., polar angle from origin or nearest-neighbor groupings). Given this initial population, two solutions are chosen at random and the "better" solution is chosen as the first parent. The second parent is obtained via the same process. Offspring are generated using a 2-point crossover technique and consequently mutated by randomly swapping the values of two genes. In terms of the VRP, this means two random customers traded servicing vehicles. Next, the two children generated either replace the parents in the population of solutions (if they are "better" solutions according to a fitness function) or discarded if they do not meet the criteria for entrance into the solution population.

However, these algorithms are only tractable on simpler instances of the VRP. Evolutionary algorithms are commonly outperformed by tabu search and simulated annealing [27] [28] in terms of solution quality for the VRP. In general, genetic algorithms are most successful in problems with relatively relaxed constraints. However, adding the constraints necessary for the VRP variants discussed earlier means introducing new constraints and therefore tightening the restrictions on possible solutions.

2.3.2 Tabu Search

Tabu search is a very popular method for solving a variety of problems, including the VRP and its variants. Tabu search is regarded as a metaheuristic because it is more of an idea for a search process than a specific algorithm. Tabu search is concerned with a particular set of attributes held by solutions. The search process then uses these attributes

as a way of selecting new solutions with certain attributes being “tabu,” meaning these attributes are either forced into or not allowed into the solution for some number of iterations.

As an example, consider the tabu search metaheuristic put forth by Brandao [29]. This metaheuristic uses a combination of nearest-neighbor and insertion techniques to produce an initial solution. Given this solution, the algorithm then executes a series of insert and swap moves. With an insert move, a customer is moved from one route to another, or possibly another location within the same route. The swap move exchanges the locations of two customers from different routes. In keeping with the tabu idea, the algorithm imposes a restriction stating if a customer is removed from a route via insertion or swap, the customer cannot return to that route for some number of iterations. The algorithm also has aspiration criteria based on improving the best known objective function evaluation. Brandao claims tabu methods represent the best known heuristics for the VRP, and his tabu metaheuristic yielded the best known results to the test problems on which he conducted his research.

2.3.3 Local Search/Improvement Algorithms

A LS takes a current solution, s_0, and tries to either reach a better solution or build a neighborhood of solutions about s_0, that neighborhood being some or all solutions that can be reached from s_0 through the application of a particular operator [26]. Elements of a neighborhood are referred to as neighbors. Local improvement operators are restricted to only accept improving solutions, while conditions within a LS may allow a sequence of non-improving solutions in hope of escaping a local optimum. The term LS operator

refers to a single instance of a LS method and neighborhood while the term LS refers to the entire LS heuristic which may include multiple operators.

Given the neighborhood structure of a chosen LS operator, one must decide how to explore the neighborhood. The two most common approaches are best improvement and first improvement. In best improvement, the entire neighborhood is explored and the neighbor with the greatest improvement is selected as the new solution. In first improvement, the first improving neighbor is selected as the improving solution. These two methods promote solution quality and speed, respectively. Another approach seeking to balance these two qualities is a *k*-neighbor approach in which *k* improving neighbors are identified and then a selection from this subset is made.

For the VRP, these LS operators can be classified as either intra-route or inter-route operators. Intra-route operators examine single routes and are often taken from literature on the TSP because a single route in the VRP is analogous to that problem. The most common intra-route operators are implementations of the λ-interchange operator in which the location of λ nodes are moved or exchanged within a route [30]. Conversely, inter-route operators involve multiple routes. Common methods involve moving or exchanging nodes between multiple routes.

In the case of the VRP variants considered above, neighborhood searches may become more difficult. As constraints are added, the probability of finding a feasible neighbor decreases. Therefore, finding improving feasible neighbors requires more computing time [31]. Three approaches are available to address this issue. The first method is to simply check the feasibility of solutions and only allow feasible improving solutions. This may require checking feasibility for many neighbors depending on the

specifications of the search. Second, an infeasible solution may be accepted and a repair operator applied to return the solution to feasibility. In this case, assuming the new solution is not allowed to be the same as the starting solution for this neighborhood, bounds do not generally exist on the run time or solution quality of the repair operator. Third, infeasible solutions may be allowed with a penalty in the objective function. The downside is the LS may generate many infeasible solutions.

Despite these issues, nearly every successful VRP algorithm incorporates at least some element of LS. Most algorithms consist of two phases: construction, in which some initial solution (perhaps infeasible) is built, and improvement, in which the neighborhood of the initial solution is explored in search of improving (and generally feasible) solutions. The methods and time devoted to each phase vary greatly amongst algorithms.

2.3.3.1 LS for the Vehicle Routing Problem

The LS algorithms listed in Table 2 and detailed below represent the most promising and widely used techniques from literature. The most common LS operators for the VRP are edge-exchange operators [31]. In general, a *k*-opt involves exchanging *k* edges with another disjoint set of *k* edges. Practical applications usually involve either the 2-opt or 3-opt operators as a *k*-opt with $k > 3$ is often computationally impractical. Or [32] developed what has become a popular variant of 3-opt called the Or-opt operator in which the 3-opt operator used must preserve the orientation of the routes. Similarly, Potvin and Rousseau [33] introduced the 2-opt* operator in which the specific 2-opt operator used does not alter the orientation of the routes. The authors also show a

Table 2: Local Search Literature Review

	Time Windows VRP	Split Delivery VRP	ACO	LS	Experimental Design/Analysis	Military Application
Aleman (2009)		●	●	●	●	
Archetti (2008)		●		●		
Bent and Hentenryck (2004)	●			●		
Braysy (2002)	●			●		
Braysy (2003)	●			●		
Braysy and Gendreau (2005)	●			●		
Braysy et al. (2002)	●			●		
Derigs et al. (2010)		●		●		
Glover (1992)				●		
Hashimoto et al. (2008)	●			●		
Ho and Haugland (2004)	●	●		●		
Ho et al. (2008)				●		
Ibaraki et al. (2005)	●			●		
Kilby et al. (1997)	●			●		
Kytojoki et al. (2007)				●		
Li and Lim (2002)	●			●		
Liu and Shen (1998)	●			●		
Or (1976)				●		
Osman (1993)				●		
Potvin and Rousseau (1995)	●			●		
Renaud et al. (1996)				●		
Savelsbergh (1985)	●			●		
Savelsbergh (1990)	●			●		
Soeanu et al. (2011)		●				
Solomon et al. (1988)	●			●		
Taillard et al. (1997)	●			●		
Thompson and Psaraftis (1993)				●		
Van Breedam (1994)	●			●	●	

combination of the Or-opt and 2-opt* operators is particularly effective. Each of these *k*-opt operators can be applied as either an inter-route or intra-route operator.

Glover [34] uses ejection chains to solve the VRP. In this method, a set of *k* nodes are cyclically exchanged, meaning node 1 replaces node 2, node 2 replaces node 3,

and so on with the k^{th} node inserted into the empty position left by removing node 1. The closely related method of cyclic transfers, introduced to the VRP by Thompson and Psaraftis [35], are also regularly used. Kilby et al. [36] use the 2-opt, relocate, exchange, and cross methods to solve a VRP while Kytojoki et al. [37] use the 2-opt, Or-opt, and 3-opt intra-route operators and the exchange, relocate, 2-opt*, and Cross Exchange inter-route operators.

2.3.3.2 LS for the Vehicle Routing Problem with Time Windows

Savelsbergh [38] combines the edge exchange concept with the methods introduced by Or [32] to produce 2-interchanges and Or-interchanges. Savelsbergh later refines these operators [39] and introduces the inter-route operators relocate, exchange, and cross [40]. The relocate operator moves a customer from one route to another. The exchange operator is a node exchange in which customers from separate routes are swapped. The cross operator attempts to fix routes such that no two routes cross over each other. Solomon et al. [41] use 2-opt, 3-opt, and Or-opt operators to solve a VRPTW. Osman [30] officially defines the λ-interchange operators in which subsets of customers no larger than λ from two routes are exchanged. This method differs from the normal edge exchange in that the size of the two subsets is not restricted to equality. Van Breedam [42] uses four operators closely related to the λ-interchange concepts. Gendreau et al. [43] define an algorithm called GENIUS in which constructed solutions are improved by inserting new customers into a route in a particular manner. Taillard et al. [44] define the Cross Exchange operator, a two-edge exchange in which some arbitrary number of consecutive customers is swapped between two routes. Braysy [45]

uses modified versions of the Or-opt and Cross Exchange operators as well as IRP (not to be confused with the Inventory Routing Problem discussed in Chapter 5), which constructs neighborhoods based on a distance metric between customers, and O-opt, which constructs all possible routes given a specified subset of customers. Braysy et al. [46] use injection trees – an extension of ejection chains, GENICROSS – a hybrid of the GENIUS algorithm and Cross Exchange operator, and two methods based on the Cross Exchange operator. Bent and Van Hentenryck [47] use the traditional two-exchange, Or-exchange, relocation, crossover, and exchange operators to define neighborhoods. Braysy [48] uses the Or-opt operator and a special operator based on ejection chains. Hashimoto et al. [49] use 2-opt*, Cross Exchange, and Or-opt operators. Ibaraki et al. [50] use the Cross Exchange, 2-opt*, Or-opt, and ejection chain operators. Li and Lim use the relocate, exchange, and rearrange operators, where rearrange is an intra-route implementation of *k*-opt with a variable parameter *k*. Liu and Shen [51] use a generalization of the λ-interchange operator.

2.3.3.3 LS for the Split Delivery Vehicle Routing Problem

Aleman [52] uses the popular relocate (which he calls shift) and exchange operators along with a shift* operator in which a single delivery is split if a vehicle capacity constraint is violated. Derigs et al. [53] implement 2-opt*, exchange, and relocate operators. Archetti et al. [54] use integer programming to explore promising regions of the solution space identified by a tabu search heuristic. Chen et al. [17] use a record-to-record travel operator.

2.3.3.4 LS for the Split Delivery Vehicle Routing Problem with Time Windows

Decidedly less attention is devoted to problems with the combination of these characteristics. Ho and Haugland [8] adapt the relocate, exchange, and 2-opt* operators to the SDVRPTW and develop a new operator, relocate-split, which splits delivery from one node and combines the deliveries from a node whose deliveries are currently split into a single delivery on one of the routes currently servicing that node. Belfiore et al. [7] employ relocation, insertion, and route addition operators as well as a novel demand reallocation operator.

2.3.4 Ant Colony Optimization

First introduced by Dorigo [55], ant colony optimization (ACO) works by iteratively constructing a series of solutions [56]. Each ant is an instance of a solution construction. Ants probabilistically add components to their individual solutions until reaching a complete solution. The addition of components is based upon heuristic and pheromone information about the problem. In the case of a VRP, the heuristic information consists of the edge costs (e.g., cost or time to transit a commodity over a given edge). The pheromone information is gleaned from previous solutions. More specifically, each edge is initialized with the same amount of pheromone. As a portfolio of solutions is built, a local pheromone update decreases the pheromone on those edges used in building a solution while a global pheromone update deposits additional pheromone onto the "good" edges. In general, a good edge is one included in what is deemed a good solution. Good solutions are typically defined as either the "best so far" or "best of iteration" solution. In either case, one can also define either a single best

solution or identify some “k-best” solutions. The exact implementation details depend on the algorithm and application, but Stutzle and Hoos [57] give empirical evidence an elitist strategy (i.e., only allowing some best solution (or set of solutions) to perform global updates) yields improved algorithm performance in both solution time and quality.

By choosing various values for parameters (e.g., evaporation rates, pheromone deposit rates, min/max pheromone amounts), the algorithm can be tuned to solve a number of different problems. One of the greatest advantages of an ACO algorithm is the ability to balance exploration versus exploitation. For example, a high evaporation rate combined with a high pheromone deposit rate strongly encourages the ants to use edges known to exist in good solutions. This process is known as exploitation and tends to encourage solutions “near” existing solutions in that these newly generated solutions are more likely to share components with the current portfolio of solutions. Conversely, weak evaporation and pheromone deposit rates encourage ants toward exploration and are more likely to find solutions with less in common than those solutions already known. A high-quality ACO algorithm strikes a balance between these two aspects of the algorithm. This balance is also not static amongst problems because some problems are more amenable to exploration while exploitation leads to better solutions in others.

These rates are also not necessarily static within an algorithm as they can also be dynamically adjusted as the algorithm proceeds. For example, an algorithm may yield a good solution. One would then want to encourage exploitation for several iterations to try to find a neighborhood of solutions about that original good solution in hopes of reaching a better solution or a local optimum. However, after some number of iterations the algorithm may no longer find improving solutions (e.g., the algorithm is stuck at a

Table 3: Ant Colony Literature Review

	Time Windows VRP	Split Delivery VRP	ACO	LS	Experimental Design/Analysis	Military Application
Bell and McMullen (2004)			●	●		
Bullnheimer et al. (1999)			●	●		
Ding et al. (2012)	●		●	●		
Dorigo (1992)			●			
Dorigo and Gambardella (1997)			●	●		
Dorigo and Stutzle (2010)			●	●		
Favaretto et al. (2007)	●	●	●	●		
Gambardella et al. (1999)	●		●			
Gutjahr (2002)			●			
Mazzeo and Loiseau (2004)			●	●		
Pellegrini et al. (2006)			●	●	●	
Rajappa (2012)		●	●	●		
Reimann et al. (2004)			●	●		
Ridge and Kudenko (2007)			●		●	
Sodsoon and Changyom (2011)	●		●	●		
Stutzle (1998)			●	●	●	
Stutzle and Dorigo (2002)			●			
Stutzle and Hoos (1996)			●	●	●	
Stutzle and Hoos (1997)			●	●		
Stutzle and Hoos (2000)			●	●		
Wang and Yu (2010)			●			
Xia et al. (2011)			●	●		
Yi and Kumar (2006)		●	●			
Yu et al. (2009)			●	●		
Yu et al. (2011)	●		●	●		
Yucenur and Demirel (2011)			●			
Zhang and Wang (2012)			●	●		

local optimum). At this point, the algorithm may alter the pheromone information along with the evaporation and/or pheromone deposit rates to encourage the ants to leave this

neighborhood in hopes of finding a better solution not sharing many components with the current portfolio of solutions.

A literature review of ACO algorithms, summarized in Table 3, reveals the application of a LS greatly enhances the performance of many ACO implementations [56]. This procedure is generally implemented after an ant has constructed a complete solution, at which time a LS attempts to improve this solution. This coupling tends to work well because ACO algorithms perform a rather coarse-grained search meaning a solution is generally amenable to improvement via a LS. Meanwhile the primary issue with a LS is the generation of a starting solution. Therefore, the combination of these two methods tends to yield excellent results.

The literature also shows ACO algorithms are competitive with other metaheuristics [58]. For example, Yu et al. found their implementation of an ACO for a VRP produced higher quality solutions but with a slightly higher cost in computation time than other known metaheuristics such as tabu search and simulated annealing [59]. The ACO implemented by Dorigo and Gambardella [60] compared favorably against state-of-the-art evolutionary algorithms. Furthermore, Gutjahr [61] proved ACO will converge in the limit to the optimal solution given no lower bound on the pheromone levels.

2.3.4.1 MAX-MIN Ant System

Stutzle and Hoos [62] introduce a variant of ACO called the Max-Min Ant System (MMAS) and show it outperforms Dorigo's original ACO implementation in a test set of symmetric and asymmetric TSPs and quadratic assignment problems [58]. The

primary change in this variant is the inclusion of explicit upper and lower bounds on the pheromones for each edge. This characteristic helps the algorithm to avoid early stagnation at a sub-optimal solution. MMAS also initializes all pheromones to the maximum pheromone amount. Experimentation shows this initialization produces higher quality solutions in the early runs and allows the algorithm to converge more quickly than other initialization procedures. MMAS also introduces a trail smoothing mechanism in which, if the search stagnates, all pheromones are updated by some proportion of the difference between the maximum pheromone and their current value. This strategy acts to reset the pheromones, but instead of resetting all pheromones to some arbitrary value (e.g., maximum pheromone limit) this method allows the algorithm to retain a portion of the current knowledge.

MMAS also employs an elitist strategy, allowing only the best solution (either best-so-far or best-of-iteration) to perform global pheromone updates [63]. Finally, Stutzle and Hoos [62] incorporate a 2-opt LS operator into MMAS and show empirically this implementation improves performance. Performance is further improved by employing an elitist strategy with respect to the LS, allowing only the ant with the current best solution to perform a LS. The specific upper and lower pheromone bounds for MMAS are determined in a problem-specific nature, depending on the average heuristic value (e.g., edge length) of the problem [57]. Stutzle and Dorigo [64] also extend Gutjahr's work [61] and show MMAS will converge to the optimal solution in the limit.

Stutzle [63] initially suggests parameter settings for MMAS based on experimentation of selected TSPs, concluding the number of ants used should be on the same order as the size of the problem (e.g., number of nodes). Stutzle and Hoos [57]

expand on these recommendations, adding further detail to the experiments and recommending a best-of-iteration elitism strategy. Pellegrini et al. [65] expand upon this foundation, discussing the specific impacts of the parameters on the solutions generated and giving a formula for determining the evaporation rate depending on the desired number of runs and the size of the problem. Ridge and Kudenko [66] further investigate the MMAS and recommend optimal parameter settings depending on problem size and standard deviation.

2.3.4.2 ACO applied to the Vehicle Routing Problem

Bullnheimer et al. [67] first adapted the ACO for use in solving a VRP using a 2-opt LS operator and candidate lists for route selection. The use of candidate lists entails a pre-processing phase in which the *k* closest customers to every customer are listed and customers on this list are the only customers eligible for selection from a given node. The reasoning behind these lists is to avoid complicating selection procedures by considering customers very far away from the current customer, and therefore not consider edges highly unlikely to be included in a good solution. Bell and McMullen [68] adapt an ACO for the VRP, including a 2-opt operator and candidate lists. The authors also speculate, based upon limited data, multiple ant colonies with independent pheromone matrices are more effective than a single colony, particularly on larger problems. Reimann et al. [69] propose a decomposition method for a large-scale VRP and then use ACO to solve the smaller subproblems. The authors incorporate the inter-route swap LS operator and then apply an intra-route 2-opt operator. Mazzeo and Loiseau [70] implement an ACO with candidate lists and 2-opt and Or-opt operators. Yu

et al. [59] introduce an improved ACO in which a mutation – essentially a specific implementation of the λ-interchange LS operator – is introduced in addition to the normal 2-opt operator. Zhang and Wang [71] implement an ACO in conjunction with a nearest-neighbor heuristic and a 2-opt operator. Wang and Yu [72] introduce an improved MMAS in which the authors use feedback mechanisms in the later runs to improve the ants' ability to explore the solution space. Xia et al. [73] use MMAS with 2-opt and relocate operators to solve a VRP. The authors also use pre-scheduled changes in the parameters based on the number of runs to improve the algorithm's performance.

2.3.4.3 ACO applied to the Vehicle Routing Problem with Time Windows

Gambardella et al. [21] use ACO to solve a multi-objective VRP. The authors effectively solve the objectives lexicographically, first minimizing the number of vehicles and then minimizing total travel time for the given number of vehicles. Gambardella et al. also incorporate a LS based on the Cross Exchange procedure. Ding et al. [74] use MMAS in concert with 2-opt and Or-opt operators as well as a disaster operator that randomly perturbs the pheromone matrix in an attempt to broaden the search space. Sodsoon and Changyom [75] adapt MMAS to the VRPTW, incorporating relocate, Or-opt, and 2-opt operators. Yu et al. (2011) [76] adapted an ACO/LS hybrid for VRPTW.

2.3.4.4 ACO applied to the Split Delivery Vehicle Routing Problem

Very little research exists into applying ACO to the other VRP variants under consideration here. Rajappa [19] uses an ACO to solve the SDVRP but does not

incorporate a LS. This literature review did not return any instances of an ACO metaheuristic being used to solve the SDVRPTW.

2.4 Gaps in Literature

As shown in this literature review, little research exists for the SDVRPTW, particularly in the area of heuristics. The next three chapters will add to the body of knowledge of the SDVRPTW by exploring the research questions described in the previous chapter. In particular, this research effort will seek to empirically measure performance of several LS operators for the SDVRPTW. This research will also examine the impact of the construction phase and problem structure. Finally, these results from the SDVRPTW are applied to a military instance of the inventory routing problem.

III. Testing Local Search Move Operators on the Vehicle Routing Problem with Split Deliveries and Time Windows

3.1 Introduction

The vehicle routing problem (VRP) is an important transportation problem seeking an optimal solution for constructing delivery routes given a depot, a fleet of vehicles and some number of geographically dispersed customers, each having a demand that must be fulfilled. The problem also incorporates characteristics such as travel times and/or distances as well as side constraints such as a maximum vehicle load. This problem is important due to both its widespread application and its complexity in solving. See [26] for a more thorough review of the VRP. The literature addresses several extensions of this problem, including variants having delivery time windows associated with customers (VRPTW) and variants allowing split deliveries to customers (SDVRP). The problem extension including both of these variations has received less attention in the literature. This research sheds further light on this problem, which is important because the addition of these two features more accurately represents important real-world applications of the VRP. Furthermore, the problem and methods used to approach the problem may differ significantly in the presence of these additional characteristics, implying the need for research expressly dedicated to these variants. Specifically, this chapter analyzes the effects of combinations of local search (LS) move operators commonly used on the VRP and its variants to empirically determine the combination best suited to generating good solutions for the VRP with split deliveries and time windows (SDVRPTW) within an ant colony optimization (ACO) metaheuristic and is

organized as follows. Section 3.2 presents background on the problem and provides a literature review, Section 3.3 describes the test problems and experimental design for the computational results presented in Section 3.4, and, finally, Section 3.5 concludes with findings and areas for future research.

3.2 Background

This section will cover the relevant literature for the SDVRPTW with a brief overview on LS operators and the ACO metaheuristic. This section will also discuss these heuristics as applied to the VRP, focusing specifically on applications involving the VRPTW, SDVRP, or SDVRPTW.

3.2.1 LS for SDVRPTW

Archetti and Speranza [25] offer a concise review of existing work for the SDVRP. They cover both heuristic and exact methods employed thus far, emphasizing the improvements in solutions to various test problems seen when comparing traditional VRP solutions without split deliveries to solutions allowing split deliveries. This research will focus in particular on the applications of LS operators from these research efforts.

Feillet et al. [77] use a branch-and-price algorithm to solve examples of the SDVRPTW exactly. However, like the VRP and many of its variants, the SDVRPTW is NP-hard [7] and exact solutions are difficult to come by, generally requiring extremely long computation times. Frizzell and Giffin [22] first introduce LS to the SDVRPTW,

pairing two operators–moving a customer to a new route or swapping customers between routes–with a look-ahead construction heuristic. They employ the LS on problems using grid network distances. Ho and Haugland [8] use a tabu search to tackle the SDVRPTW, employing the following LS operators: Relocate - moves a customer to new route; Relocate-split - splits a customer's load and moves those two loads to new routes; Exchange - trades a pair of customers on separate routes; and 2-opt* - exchanges the last *m* customers from one route with the last *n* customers of another route. Campos et al. [24] adapt the Clarke-Wright savings algorithm to the SDVRPTW to develop an initial solution and then use a genetic algorithm to improve this initial solution. Belfiore et al. [7] use scatter search to generate solutions for the SDVRPTW.

Many LS operators are employed in approximating solutions for the VRP and its variants. Some of the most popular or promising operators are now discussed. As seen above, Ho and Haugland [8] successfully utilize four LS operators (Relocate, Relocate-split, Exchange, and 2-opt*) on the SDVRPTW. In addition to these operators, one question this research will address is how well LS operators from the VRPTW and SDVRP variants extend to the SDVRPTW. Dror and Trudeau [14], generally regarded as the first to investigate the SDVRP, introduce the 2-split-interchange LS operator, which is also the basis for the Relocate-split operator described above. Aleman et al. [78] introduce a Shift* operator for the SDVRP. The Shift* operator is similar to the Exchange operator described above except it allows for a partial shift of one of the customers. Derigs et al. [53] introduce a series of LS operators specific to the SDVRP, including Combine, Relocate, and another operator similar to the Relocate-Split LS operator; additionally, the authors introduce the concept of combining a split delivery and

introducing a new route for this delivery. Braysy and Gendreau [31] detail many LS operators used to generate solutions for the VRPTW, including 2-opt*, Or-opt, and Cross Exchange. These three LS operators prove very popular and effective in the VRP literature (see [73], [71], [59], [21], [74], [75], and [37]). Each of these methods is described in further detail in Section 3.3. For further details, see [25] for the SDVRP and [26] and [31] for the VRPTW.

3.2.2 LS Performance Analysis

None of the LS implementations on the SDVRPTW discussed above make any explicit argument for why a particular LS operator is chosen. None tested the LS operators to show the one (or several) chosen was the best choice for the problem. Rather, LS operators are most likely chosen based on successful implementations on other variants of the VRP.

Others have undertaken the task of comparing the performances of LS operators for several variants of the VRP and related problems, but none have specifically investigated the SDVRPTW. Stutzle [63] investigates the effects of several LS operators on the traveling salesman problem, the quadratic assignment problem, and the flow shop problem when paired with an ACO metaheuristic. Van Breedam [42] analyzes the effectiveness of several LS operators, paired with several different solution construction heuristics, for the VRPTW and the pickup and delivery problem. Braysy and Gendreau [31] further analyze LS operators when applied to the VRPTW. Derigs et al. [53] investigate the effects of LS operators on the SDVRP. However, this literature review

revealed no work done to investigate the effects of LS operators when applied to the SDVRPTW.

3.2.3 Metaheuristics

Testing the performance of the LS operators requires combining these LS operators with a construction heuristic into a metaheuristic. This research effort uses an ACO metaheuristic. This metaheuristic is chosen for two reasons: first, it is successfully implemented on the VRP and several of its variations (see [21], [74], [75], and [68]); and second, it is studied less extensively than other metaheuristics such as tabu search (see [26] and [11]). The ACO metaheuristic was first introduced by Dorigo [55]. The ACO metaheuristic iteratively constructs a series of solutions [56] where each ant provides an instance of a solution construction. Ants probabilistically add components to their individual solutions until reaching a complete solution. The addition of components is based on heuristic and pheromone information about the problem. In the case of a VRP, the heuristic information consists of the edge costs (e.g., cost or time to transit a commodity over a given edge). The pheromone information is gleaned from previous solutions. More specifically, each edge is initialized with the same amount of pheromone. As a portfolio of solutions is built, a local pheromone update decreases the pheromone on those edges used in building a solution while a global pheromone update deposits additional pheromone onto the "good" edges. In general, a "good" edge is one included in what is deemed a high-quality solution (e.g., "global best" or "iteration best" solution). The local pheromone update encourages exploration of new solutions while the global update encourages exploitation of high-quality solutions.

A literature review of ACO algorithms reveals the application of LS greatly enhances the performance of many ACO implementations [56]. The LS is generally implemented after an ant has constructed a complete solution, at which time the LS attempts to improve this solution. This coupling tends to work well because ACO algorithms perform a rather coarse-grained search meaning a solution is generally amenable to improvement via LS. Meanwhile the primary issue with LS is the generation of a starting solution. Therefore, the combination of these two methods tends to yield excellent results and make the ACO metaheuristic a good candidate for the constructive phase when paired with a LS.

This research uses the MAX-MIN Ant System (MMAS), an implementation of an ACO metaheuristic introduced by Stutzle and Hoos [62]. They show MMAS outperforms Dorigo's original ACO implementation in a test set on symmetric and asymmetric TSPs and quadratic assignment problems [58]. The ACO metaheuristic, and in particular MMAS, has also proven capable and competitive in the context of the VRP (see [21], [74], [75], and [68]). The primary change in the MMAS variant compared to other ACO metaheuristics is the inclusion of explicit upper and lower bounds on the pheromone levels for each edge. This characteristic helps the algorithm avoid early stagnation at a sub-optimal solution. This implementation allows for split deliveries in the following manner: a customer is selected for addition to a route in the standard ACO manner. If the entire demand fits onto the vehicle, it is added. If only part of the demand fits onto the vehicle, the maximum delivery amount is added to the vehicle and the customer's demand is updated to reflect the remaining unfilled demand. Since the sole purpose of the MMAS metaheuristic in this research is to provide initial solutions for the

LS, any further discussion of the MMAS details is beyond the scope of this paper; see [63] for further details on the MMAS metaheuristic.

Given the descriptions of the relevant work from literature as it relates to the problem of interest, the SDVRPTW, the next section discusses in greater detail the experimental design and the implementation of the ACO metaheuristic, as well as details on the problem sets used.

3.3 Method

The experimental design consists primarily of 93 experiments that form the backbone for the results in Section 3.5. This section describes how and why these 93 experiments are conducted. This section also describes the test problems used in the experimentation.

3.3.1 Experimental Design

This research investigates the use of eight LS operators, in combinations of up to three, paired with an MMAS metaheuristic. This yields 93 different configurations of LS operators: 1 configuration with no LS, 8 configurations with one LS operator, 28 configurations with two LS operators, and 56 configurations with three LS operators. The eight LS operators were chosen due to either their widespread use in finding good solutions for the VRP and its variants (as is the case for 1, 3, 4, and 5) and/or because of promising results on the SDVRP (2, 6, 7, and 8) or SDVRPTW (1, 3, and 8). The eight LS operators are listed and described below. In this case, a delivery refers to a route

visiting a customer and making a non-empty delivery since referring to customer visits, in the context of split deliveries, is too vague. Furthermore, each of the LS operators must return a feasible solution in terms of time windows, vehicle capacity, and customer demand. Let R_a denote the a^{th} route in the solution and g_i denote the i^{th} delivery on a given route.

1. Relocate:

Two deliveries, g_i, $g_j \in R_a$, are selected and g_i is removed from its original position and inserted following g_j. Figure 1 depicts g_j as occurring after g_i in the initial solution, but it may occur either before or after g_i.

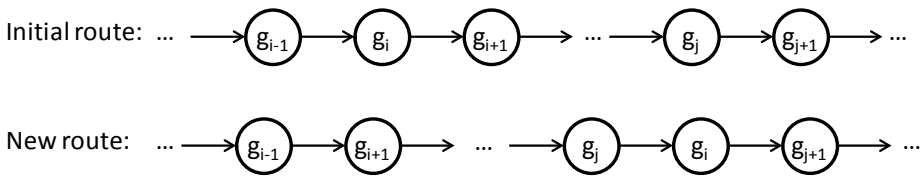


Figure 1: Relocate operator

2. Split-to-single:

A pair of deliveries, $g_i \in R_a$ and $g_j \in R_b$, is chosen such that both belong to a single customer. These two deliveries, g_i and g_j, are combined and a new route is created that satisfies this delivery (i.e., the new route departs the depot, makes the new delivery, g_1, and returns to the depot). See Figure 2.

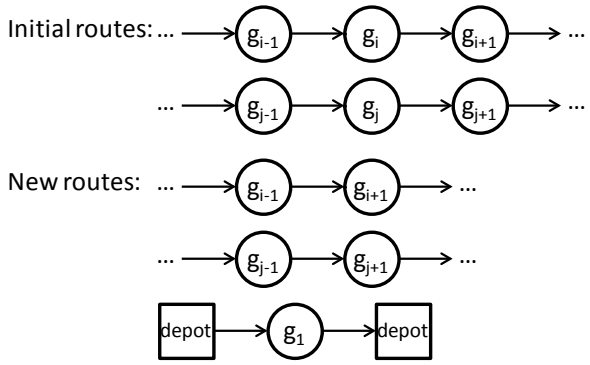


Figure 2: Split-to-single operator

3. 2-opt*:

Two deliveries, $g_i \in R_a$ and $g_j \in R_b$ ($a \neq b$), are chosen. Then, the edges connecting g_i to g_{i+1} and g_j to g_{j+1} are removed. Two new edges are added adjoining g_i with g_{j+1} and g_j with g_{i+1}. See Figure 3.

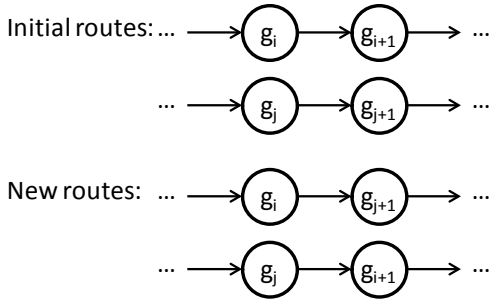


Figure 3: 2-opt* operator

4. Or-opt:

Three deliveries, g_i, $g_{i+\delta} \in R_a$ ($\delta \geq 2$) and $g_j \in R_b$ ($a \neq b$), are chosen. Then, the sequence of deliveries beginning with g_{i+1} and ending with $g_{i+\delta-1}$ is removed from

R_a. An edge is then added to R_a such that g_i and $g_{i+\delta}$ are now consecutive deliveries. The removed segment is then inserted into R_b such that g_j precedes g_{i+1} and $g_{i+\delta-1}$ precedes g_{j+1}. See Figure 4.

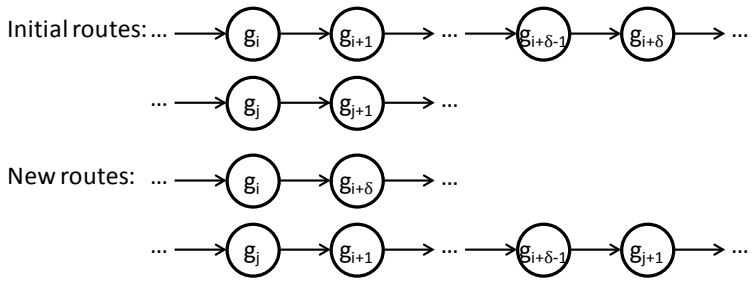


Figure 4: Or-opt operator

5. Cross Exchange:

Four deliveries, g_i, $g_{i+\delta} \in R_a$ and g_j, $g_{j+\varepsilon} \in R_b$ ($a \neq b$; $\delta, \varepsilon \geq 2$), are chosen. The sequence of deliveries beginning with g_{i+1} and ending with $g_{i+\delta-1}$ is removed from R_a. Similarly, the sequence of deliveries beginning with g_{j+1} and ending with $g_{j+\varepsilon-1}$ is removed from R_b. Four new edges are then added connecting the following pairs of deliveries: g_i to g_{j+1}, g_j to g_{i+1}, $g_{i+\delta-1}$ to $g_{j+\varepsilon}$, and $g_{j+\varepsilon-1}$ to $g_{i+\delta}$. See Figure 5.

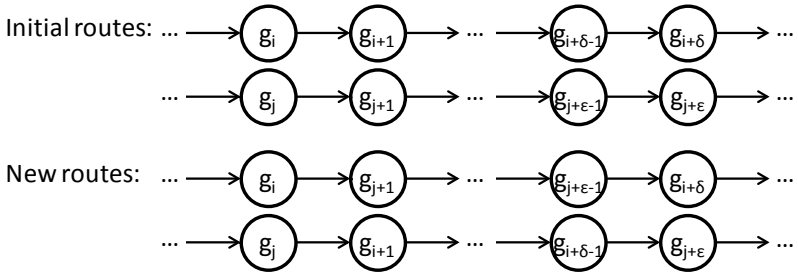


Figure 5: Cross Exchange operator

6. 2-split-interchange:

A delivery, $g_i \in R_c$, and a pair of routes, R_a and R_b ($a \neq c$, $b \neq c$, $a \neq b$), are chosen such that neither route has the capacity for g_i. The delivery, g_i, is then split between the two routes, R_a and R_b, such that the maximum amount possible is transferred to R_a and the remainder to R_b. Each split delivery is inserted in the first feasible location after departing the depot on its new route. See Figure 6.

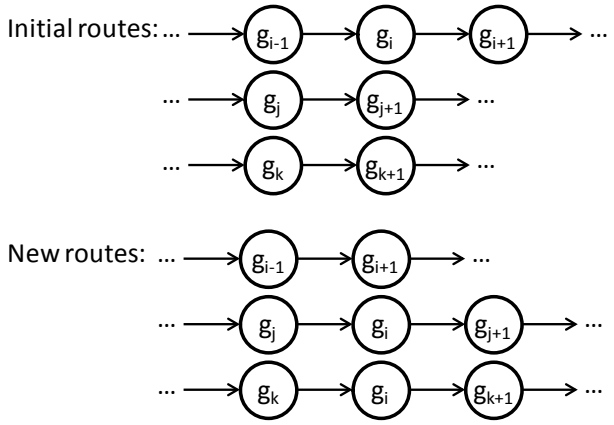


Figure 6: 2-split-interchange operator

7. Combine:

A pair of deliveries, $g_i \in R_a$ and $g_j \in R_b$, is chosen such that both belong to a single customer. The deliveries are then combined in one of the two existing deliveries, with g_i being the first choice. See Figure 7, where the case with the deliveries combined into g_i is illustrated.

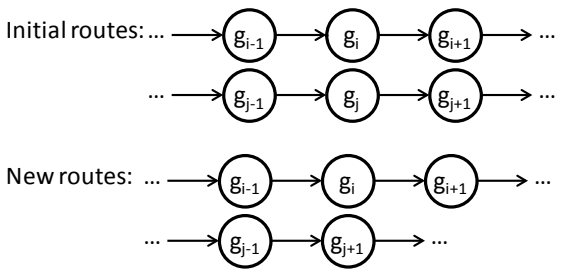


Figure 7: Combine operator

8. Shift*:

A pair of deliveries, $g_i \in R_a$ and $g_j \in R_b$ ($a \neq b$), is chosen such that the vehicle servicing R_b has the capacity for g_i but the vehicle servicing R_a lacks the capacity for g_j. Then, g_i is inserted into R_b at the first feasible location after departing the depot. Then g_j is split such that a partial delivery remains on R_b and a partial delivery is inserted into R_a. More explicitly, the partial delivery randomly chooses to capacitate one of the routes, meaning either the maximum possible amount remains on R_a or the maximum possible amount moves to R_b. Figure 8 illustrates this process with the g_i inserted into R_b after g_{j+1} and the partial delivery of g_j inserted into R_a after g_{i+1}, but this precedence relation is not necessary. Both deliveries are simply inserted into the first location yielding a feasible and improving solution.

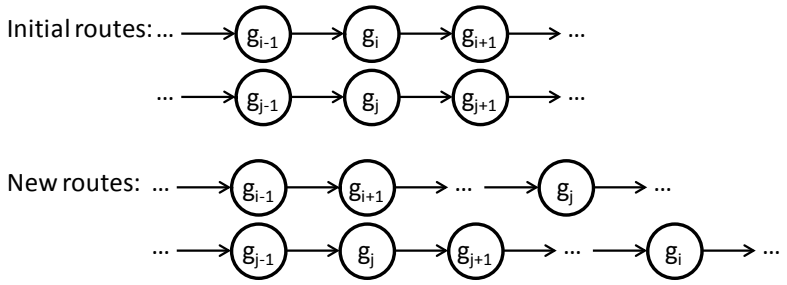


Figure 8: Shift* operator

Pseudocode for metaheuristic:

-Initialize parameters

-For i = 1:iteration_count

  Construction phase
  --For i = 1:ant_count
    ---Construct $\sqrt{n}$ solutions using MMAS
    ---Save solution with lowest cost

  Improvement Phase
  --Apply LS operator(s) until a local optimum is reached

  --Update global best solution
  --Global pheromone update

-Return global best solution

Figure 9: Pseudocode for metaheuristic

The pseudocode for the overall metaheuristic is shown in Figure 9 and is a standard application of MMAS. The specifics involving the LS phase are more complicated and are therefore not detailed here. For further details or the MATLAB code used, please contact the author. The LS operator (or operators) is applied to the best

solution from the current MMAS iteration, call it γ_b. The first LS operator is then applied to γ_b and the first improving solution is accepted and becomes the new γ_b solution. The same LS operator then attempts to find another improving solution, continuing this iterative process until it cannot improve the current solution, meaning the final solution is a local optimum for that specific LS operator. In configurations with a single LS operator, this solution is returned to the metaheuristic. In configurations with multiple LS operators, the LS operators are applied in the order (1-8) shown above. The first LS operator iteratively repeats in the manner described above until it reaches a local optimum. When the first LS operator reaches a local optimum, that solution becomes the starting point for the second LS operator, which then runs in the iterative manner described above. This process repeats once more for configurations with three LS operators. In the configurations with two or three LS operators, if the final solution generated by the last LS operator used is an improvement over γ_b, then this iterative process begins anew with the first LS operator. This process repeats until none of the LS operators are able to improve γ_b, thus guaranteeing the solution returned from the LS is a local optimum for all LS operators applied. This local optimum is returned to the metaheuristic. This process is depicted in Figure 10. One final note about the implementation: if a LS operator creates a route that visits the same customer more than once, the deliveries are combined into the earliest delivery.

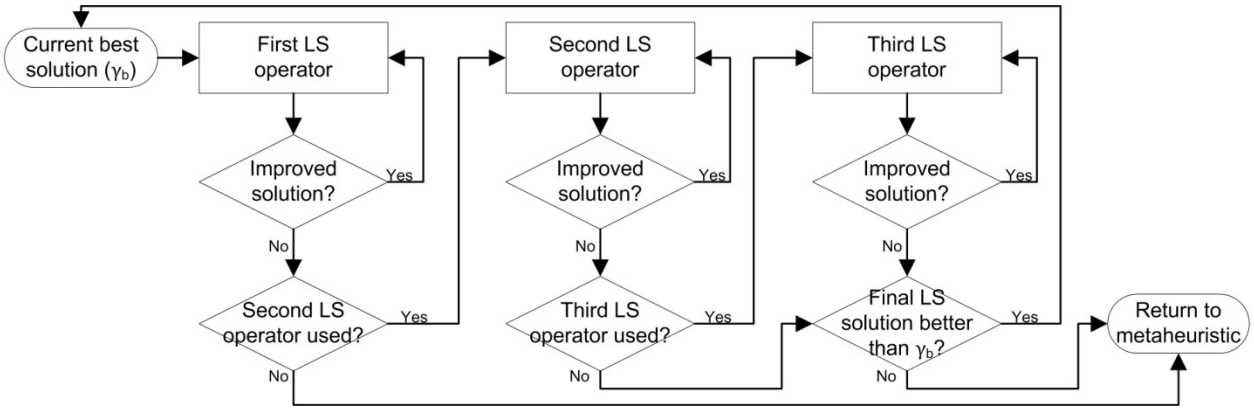


Figure 10: LS Implementation

Including the ordering of the LS operators in the experimental design requires 401 solution instances on each of the 12 problems described in Section 3.3.2 below, while a fixed-order schema requires 93 solution instances for each problem. Therefore, a fixed-order for the LS operators is chosen because of these run-time concerns. To determine the ordering, the 28 possible combinations of pairs of LS operators are applied to each problem twice–once using each permutation. A comparison of these results (see Appendix A for data) showed no more than a 2% variation in solution cost between any pairs of metaheuristics using permutations of the same LS operators. However, solution run time varied much more, with one pair having a second permutation requiring twice the run time of the first permutation (735 seconds vs. 1462 seconds). Since the variability between pairs of solution costs is negligible, these pair-wise comparisons are used to order the LS operators as shown above (1-8) with the goal of minimizing run-time.

Using the nomenclature given by Stutzle [63], our MMAS implementation includes $\alpha = 1$ and $\beta = 2$, meaning the heuristic information is considered twice as important as the pheromone information [58]. Upper and lower pheromone bounds were determined using Stutzle's rule [63] with a $p_{best} = 0.95$ [58]. The implementation uses a candidate list of size $n/5$ (n = number of non-depot customers) as suggested by Bell and McMullen [68]. Pheromone deposits are performed using the global best solution [64] with a deposit rate of n/cost where cost is the total cost of the best solution found so far [72]. Pheromone evaporation rate, ρ, is defined according to the equation set forth by Pellegrini et al. [65]. Finally, the number of ants is $\sqrt{n}$ as initial experiments indicated this number of ants struck an acceptable balance between the solution quality and run-time of the construction phase of the metaheuristic.

3.3.2 Problems

The test problems used are a subset of Solomon's classic set of VRPTW test problems [79]. Specifically, the set Solomon refers to as R1–problems R101-R112–is used for the majority of the analysis. This test set consists of 12 problems wherein the customers are randomly dispersed. Also, the vehicles are given a fairly small capacity relative to the demands with each vehicle able to handle the complete demand for approximately 5-10 customers. For a more in-depth look at these problems, see Solomon's original text [79].

For the purposes of this analysis, this restricted data set is used in an attempt to control as many variables as possible. The remainder of Solomon's test set varies the distributions used to generate customer locations and/or the vehicle load capacity. If

these variations were to induce changes into the algorithmic performance, then those problem characteristics would obscure any conclusions drawn on the performance of the algorithms themselves.

Given the experimental design and problems described above, the following section will delve into the results of these methods as applied to Solomon's test problems. On a final note, Solomon's test set was developed for the VRPTW. To transform into a SDVRPTW, the constraint stating a customer may only be visited once is simply removed. For detailed formulations of the SDVRPTW, see [7] or [8].

3.4 Results

This section presents computational and statistical results for the problem sets and experimental design presented in Section 3.3. Specifically, the section begins with some initial observations followed by an application of a hierarchical clustering technique to the data and other statistical analyses. This section also discusses how the results from the initial 93 experiments described in Section 3.3 extend to the remaining 44 problems of Solomon's test set (i.e., sets R2, C1, C2, RC1, and RC2).

3.4.1 Initial Observations

The results are characterized by three important qualities: the solution cost, the number of vehicles required, and the solution run time (shown here in seconds). The solution cost and the number of vehicles required exhibit a strong correlation as evidenced by a 0.83 correlation coefficient. This means good solutions tend to use fewer

routes. Previous research on the SDVRP supports this conclusion. Dror and Trudeau [14] first postulated this correlation and subsequent research [53], [80], [81], and [82] supports this hypothesis. Given this correlation, the results in this research focus on solution cost vs. run time.

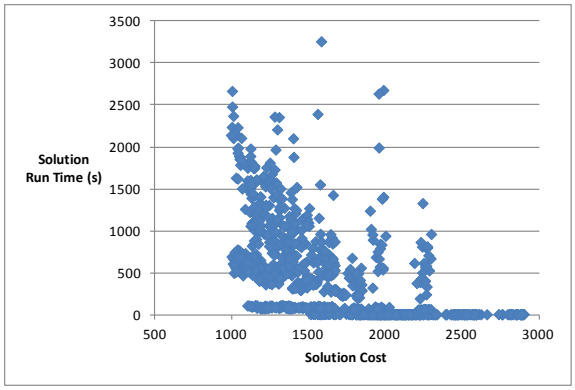


Figure 11: Raw results

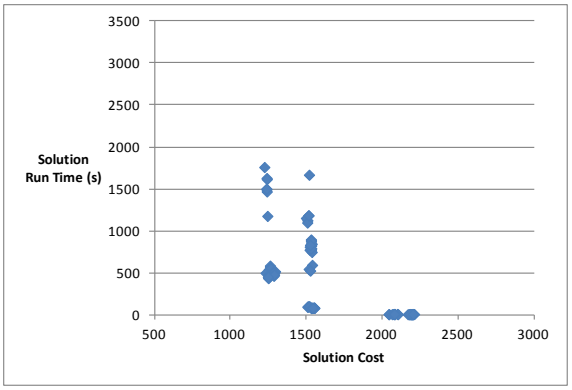


Figure 12: Average results

The heuristics were tested in MATLAB on a Windows 7 x64 PC with an Intel Xeon CPU E5-1650 @ 3.2GHz with 32 GB of RAM. The raw results are shown in Figure 11 with 93 LS configurations (henceforth referred to as configurations) implemented on 12 problems. At first glance there appears to be no pattern in this data. However, looking at the average result for a given configuration across the 12 problems, a pattern does emerge. Figure 12 shows the average of the results from the 12 problems for each of the 93 configurations (see Appendix B for data). In this grouping, clear distinctions amongst the configurations emerge. Instinctively, the first question is: "Do the configurations that implement more LS operators outperform those with fewer LS operators?" Figure 13 again depicts the average results, but distinguishes amongst the

configurations based on the number of LS operators used. Note “LS x3” means the data points use three LS operators, “LS x2” data points use two LS operators, “LS x1” data points use one LS operator, and “no LS” is the case of the MMAS metaheuristic with no LS. However, this view exhibits no discernible relationship between either cost or run time with the number of LS operators used, with configurations employing various numbers of LS operators scattered throughout the data points.

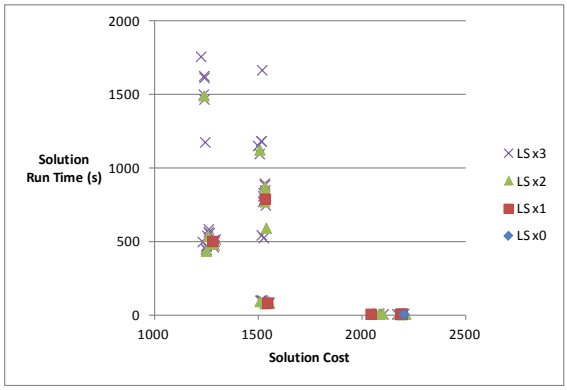


Figure 13: Average results colored by number of LS operators employed

3.4.2 Clusters

An initial glance at Figure 12 hints the data are not random, but rather are separable into clusters based on performance. To further explore this hypothesis, both cost and run time are normalized to account for the differences in scaling between these two values. For each data set, the minimum value in that set is subtracted from each data point in the set; this difference is then divided by the difference between the maximum and minimum values. This re-scales each of the data sets onto the [0,1] interval such that the minimum value in each set corresponds to 0 and the maximum to 1. Then, using a

hierarchical clustering algorithm [83] in MATLAB [84], the averaged data set is divided into k clusters. The similarity scores for the groupings of the data using a shortest Euclidean distance measure are shown in Table 4.

Table 4: Similarity scores for hierarchical clustering

Similarity Scores	# Clusters
0.088	8
0.118	7
0.166	6
0.225	5
0.242	4
0.258	3
0.275	2
0.497	1

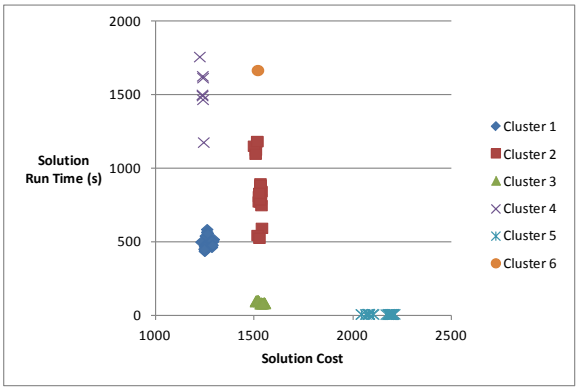


Figure 14: Six clusters

The clustering algorithm starts with a cluster for each data point and then iteratively joins clusters–joining one cluster at a time to another existing cluster–until the data are all in a single cluster. A large difference in similarity scores indicates the clusters joined together at that particular step are more different than two clusters joined with a smaller similarity gap [83]. The similarity scores in Table 4 suggest the data have natural divisions at either six or seven clusters due to the differences in the similarity scores. For example, going from six clusters to five requires joining clusters with a difference in similarity scores of 0.059 whereas the difference between five clusters and four is only 0.017. This means the clusters joined when eliminating a fifth cluster are more similar than those joined when a sixth is removed. Therefore, representing the data

with five clusters requires joining two less similar clusters compared to the clusters joined in previous steps. Seven clusters is also a possibility for a natural division because the clusters joined to remove the seventh cluster are also quite dissimilar. One may also argue the same for two clusters, but two clusters do not yield any meaningful insights. The scores in Table 4 are truncated to only show up to eight clusters. The scores for clusters 9 through 93 never significantly differ, with differences in similarity scores over this range all less than 0.02.

Consider the six clusters in Figure 14. The LS configurations comprising each of the clusters in this figure are shown in a table in Appendix C with a "1" indicating the LS operator for that column was used by the configuration of that row and a "0" otherwise. This table reveals a striking pattern as it relates to the clusters. Cluster 1 consists of every configuration that includes Or-opt but not Cross Exchange. Cluster 2 is the inverse of Cluster 1, that is, every configuration that uses Cross Exchange but not Or-opt, with one exception being that the configuration using Relocate, Cross Exchange, and 2-split-interchange is its own cluster due to its significantly longer run times than the data points in Cluster 2. Cluster 3 consists of those configurations that use 2-opt* but neither Or-opt nor Cross Exchange. Cluster 4 consists of those configurations that use both Cross Exchange and Or-opt. Finally, configurations in Cluster 5 do not use 2-opt*, Or-opt, or Cross Exchange.

Dividing the data set into seven clusters presents a similar picture, with the seventh cluster consisting of the lowermost point from Cluster 3, which is the configuration using 2-opt*, Or-opt, and Cross Exchange. However, taking into account

both similarity scores in Table 4 and the composition of the clusters, a division of the data into six clusters appears to be the best way to cluster the LS configurations.

Based on the six clusters shown in Figure 14, the optimal methods -in terms of both solution cost and run time - are those in Clusters 1, 4, and 5. The averaged data actually show Configuration 49–using Relocate, Or-opt, and Cross Exchange–of Cluster 4 lies on the Pareto front, but, compared with the configuration in Cluster 1 with the lowest solution cost - Configuration 44 - this configuration in Cluster 4 represents a 0.5% improvement in cost with the penalty of more than tripling the run time. Due to this unfavorable trade-off, combined with the fact that the difference in solution costs of the two configurations is statistically negligible, this configuration is not considered as a candidate for a good metaheuristic.

Furthermore, Cluster 5 consists of those configurations that do not use 2-opt*, Or-opt, or Cross Exchange. As the data in Table 5 show, this cluster, while Pareto optimal, offers very poor solutions with solution costs approximately 70% worse than the best solution costs seen in Figure 12. A comparison of the means in JMP version 10.0 using Dunnett's Method [85] with Configuration 1 (MMAS with no LS) as the control group reveals the other configurations in Cluster 5 are not statistically different with respect to solution cost than the control group. However, this same method also shows each of these methods is statistically the same as the control group with regards to solution time. This means these methods do not substantially improve the solution cost generated by the MMAS construction phase of the metaheuristic, but they also do not require longer run times. This is most likely due to the nature of the problems and the construction process in the MMAS metaheuristic. For example, consider the Relocate operator. The time

windows constraint restricts a customer's movement within a route while maintaining feasibility in this regard. Therefore, the nature of the problem is likely preventing this operator from contributing any substantial improvements. Also consider the Combine operator. In the construction process, customers are only split if the route cannot carry the customer's entire requirement. Therefore, the Combine operator alone is hampered given this construction process because split loads cannot possibly be combined onto one of the routes since the load was originally split out of necessity.

This research effort uses greedy implementations of the LS operators, meaning only improving solutions are accepted. Others (e.g., [1]) successfully combine some of these particular operators by allowing for non-improving moves, but this research effort does not consider that scenario. Given this particular construction process and implementation of the LS operators, any configuration that does not include at least one of 2-opt*, Or-opt, or Cross Exchange (i.e., the configurations in Cluster 5) is statistically no different than an MMAS metaheuristic, and as evidenced by Figure 14, the results given by Cluster 5 are quite poor with respect to solution cost compared to Clusters 1-4.

This leaves Clusters 1 and 3. The configurations in Cluster 1 have an average cost of 1269.1 with an average run time of 504.6 seconds. Cluster 3 averages 1534.9 and 88.7 seconds for cost and run time, respectively. In terms of cost and run time, these solutions are both non-dominated and are therefore Pareto optimal. Cluster 1 represents, on average, a 17% improvement in solution cost but with a 469% increase in run time in comparison to Cluster 3. One should note for Cluster 1, the run times are still of the order of a few minutes. However, given the results of this experiment, a 2-opt* LS operator should be employed if good solutions with fast run times are desired.

Table 5: Results averaged by cluster

Cluster	Avg Solution Cost	Avg Run Time (s)
1	1269.1	504.6
2	1524.7	863.0
3	1534.9	88.7
4	1237.8	1521.7
5	2147.0	9.0
6	1518.3	1668.5

Conversely, Or-opt should be employed if solution cost is valued more highly than run time. However, the other six LS operators investigated should not be used without combining them with at least 2-opt* or Or-opt, and Cross Exchange should not be used at all as Or-opt achieves similar results in terms of cost with much better performance in run time.

Of the non-dominated clusters, Cluster 1 is the focus of this research because, in most applications for the VRP, solution cost outweighs computational speed. Cluster 1 offers better performance than Clusters 3 or 5 in terms of solution quality. Isolating Cluster 1 and again applying the hierarchical clustering technique yields Figure 15. The similarity scores are shown in Table 6. Data are appended to show only up to six clusters because, of the remaining similarity scores, consecutive groupings all differ by less than 0.02.

Table 6: Similarity scores for sub-clusters of Cluster 1

Similarity Scores	Number of Clusters
0.129	5
0.156	4
0.387	3
0.400	2
0.446	1

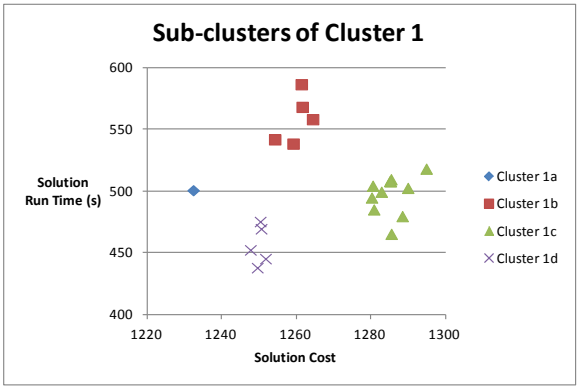


Figure 15: Sub-clusters of Cluster 1

The configurations and their accompanying LS operators are shown in Table 7. Just as before, the clustering revolves around the inclusion/exclusion of a subset of the total number of LS operators tested. Recall each of the configurations in Cluster 1 uses Or-opt and none use Cross Exchange. Given that, the inclusion of 2-opt* and Relocate drive the location within the cluster. More specifically, notice Cluster 1c contains Configuration 5, the configuration using only Or-opt. It also contains each of the configurations that do not use the Relocate or 2-opt* operators. Therefore, compared to Configuration 5, using the Combine, 2-split-interchange, Split-to-single, or Shift* operators in combination with Or-opt do not significantly affect the algorithm's performance in terms of either solution cost or run-time. Furthermore, every configuration within Cluster 1c is dominated by Cluster 1d. Next, note all of the configurations in Cluster 1b use Or-opt in conjunction with Relocate, but none use 2-opt*. Again, this cluster is dominated by Cluster 1d as well as Cluster 1a. The configurations in Clusters 1a and 1d use both Or-opt and 2-opt*. Cluster 1a also uses

Table 7: Composition of Cluster 1 sub-clusters

LS1: Relocate
LS2: Split-to-single
LS3: 2-opt*
LS4: Or-opt
LS5: Cross Exchange
LS6: 2-split-interchange
LS7: Combine
LS8: Shift*

Cluster 1a

onfiguratic	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
44	1	0	1	1	0	0	0	0

Cluster 1b

onfiguratic	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
12	1	0	0	1	0	0	0	0
39	1	1	0	1	0	0	0	0
50	1	0	0	1	0	1	0	0
51	1	0	0	1	0	0	1	0
52	1	0	0	1	0	0	0	1

Cluster 1c

Configuratic	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
5	0	0	0	1	0	0	0	0
18	0	1	0	1	0	0	0	0
29	0	0	0	1	0	1	0	0
30	0	0	0	1	0	0	1	0
31	0	0	0	1	0	0	0	1
65	0	1	0	1	0	1	0	0
66	0	1	0	1	0	0	1	0
67	0	1	0	1	0	0	0	1
87	0	0	0	1	0	1	1	0
88	0	0	0	1	0	1	0	1
89	0	0	0	1	0	0	1	1

Cluster 1d

Configuratic	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
23	0	0	1	1	0	0	0	0
59	0	1	1	1	0	0	0	0
75	0	0	1	1	0	1	0	0
76	0	0	1	1	0	0	1	0
77	0	0	1	1	0	0	0	1

Relocate, while Cluster 1d consists of the remaining 5 configurations that use 2-opt* and Or-opt.

From the results shown in Table 8, comparing Cluster 1b with Cluster 1c shows that including Relocate, but not 2-opt*, with Or-opt decreases the solution cost by an average of 2% but increase run time by 12%. The use of Relocate with 2-opt* and Or-opt, the lone configuration in Cluster 1a, improves the solution cost even further, with an average improvement of 4% but with less than 1% increase in run time. Furthermore, Cluster 1d shows including 2-opt* but not Relocate with Or-opt decreases both solution cost and run time–average improvements of 3% and 8%, respectively–relative to Cluster 1c. Therefore, the inclusion of 2-opt* with Or-opt is critical to further improving the quality of the solution in terms of both cost and run-time. Comparing the two non-dominated solutions, the inclusion of Relocate with 2-opt* and Or-opt–Cluster 1a–creates a non-dominated solution compared to Cluster 1d, decreasing the cost by an average of over 1% but with an average increase in run time of nearly 10%.

Table 8: Results averaged by sub-cluster

Sub-cluster	Avg Solution Cost	Avg Run Time (s)
1a	1232.4	500.6
1b	1260.2	558.5
1c	1285.4	497.6
1d	1250.0	456.0

Note the three most important LS operators as identified by clustering analysis are not specific to split delivery problems, but rather are borrowed directly from the VRPTW literature. Therefore, the data show implementing a LS operator with specific split delivery features does not improve the solution quality. This observation is unlikely an artifact of the problem set because Ho and Haugland [8] show splitting deliveries is effective in reducing cost for these problems. Note this does not mean splitting is not valuable in general because the construction phase used in this research incorporates splitting. Rather the data suggest, given a construction phase with a splitting option, LS operators taken from VRPTW solution implementations are more effective than operators that explicitly seek to again split deliveries.

3.4.3 Individual Problems

The results above depend upon the average result for a configuration across 12 problems that differ in customer time windows. While implementing a hierarchical clustering technique on each individual problem does not yield the same clusters as hypothesized in the previous section, for each of the problems the clustering presented represents a satisfactory view of the data. Figure 16 and Figure 17 show the clustering presented above applied to results from two individual problems. The data for the

remaining 10 problems can be seen in Appendix D. Figure 16 shows the solutions for R106 and the clusters to which each solution belongs according to the average clusters. This plot is representative of each of the problems in that the individual problems generally agree with the conclusions drawn from the averaged data. The few existing discrepancies are not large enough to dispute the overall results. For example, results for problem R112, shown in Figure 17, disagrees most with the average clusters out of these 12 problems, but the basic premises still remain; specifically, even for problem R112, Clusters 1 and 3 dominate Clusters 2 and 6 while Clusters 4 and 5 represent unsatisfactory trade-offs between solution cost and run time.

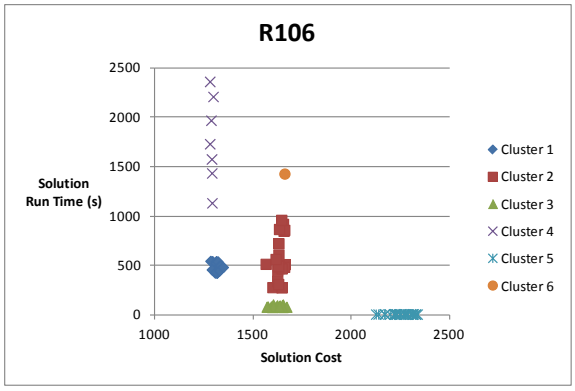


Figure 16: R106 denoted with average clusters

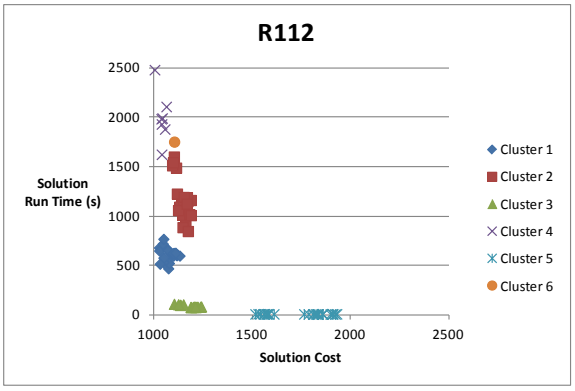


Figure 17: R112 denoted with average clusters

3.4.4 Statistical Analysis

Other statistical methods reinforce the conclusions made via clustering. Using JMP version 10.0 with the raw data as shown in Figure 11, a least squares regression fitting a third degree polynomial (i.e., up to three way interactions) with $\alpha = 0.05$ shows 2-opt*, Or-opt, Cross Exchange, and all of the 2 and 3-way interactions using only those

LS operators are the most significant factors impacting the solution cost. The other significant factors all involve Relocate in combination with other factors, meaning the Relocate operator is important to include if using multiple LS operators. Second, a partition of the data set based upon the solution cost with a validation portion of 0.1 reveals these same four LS operators–Relocate, 2-opt*, Or-opt, and Cross Exchange–are again the most significant factors in producing a statistically meaningful partition of the data. The Combine operator is shown as statistically significant, but only if 2-opt* and Or-opt are not used.

A regression fitting a third degree polynomial with respect to run time indicates the use of Or-opt or Cross Exchange significantly increases the run time as evidenced by the fact that these two individual treatments are the most significant factors and all of the significant factors involve at least one of these two operators with the exception of the treatment using only 2-opt*. However, the parameter estimates for Or-opt and Cross Exchange are an order of magnitude greater than that of 2-opt*. The results also indicate a significant negative two-way interaction between the 2-opt* and both Or-opt and Cross Exchange, meaning the inclusion of 2-opt* with one (or both) of these operators significantly reduces run time compared to using only Or-opt or Cross Exchange (or Or-opt and Cross Exchange). Furthermore, a partition of the data, with a validation portion of 0.15, reinforces the assertion that Or-opt and Cross Exchange are the most impactful LS operators in terms of run time. The mean run time for all instances is 505 seconds, while the mean run time using both Or-opt and Cross Exchange is 1521 seconds. The mean run times for instances using neither is 39 seconds while the mean run time for those instances using at most one of the two is 708 seconds. The partition also shows

using the Relocate operator in combination with either Or-opt or Cross Exchange (but not both) significantly increases run time.

These analyses reinforce the results from the clustering analyses, namely that 2-opt*, Or-opt, and Cross Exchange are the most impactful LS operators in terms of both solution cost and run time. It also reinforces the idea of a tradeoff between solution cost and run time because the LS operators that significantly decrease solution cost are the same operators that increase run time.

3.4.5 Replicates

Experimental results were replicated for a limited number of configurations. Practical restrictions with regards to run time prohibit running replications for the entire data set. The idea of Tongarlak et al. [86] of experimenting at selected points of the design space is utilized. Specifically, the configurations in Cluster 1 are replicated 10 additional times (11 runs total). Subsequently, joint confidence intervals for both solution cost and run time are constructed based on the non-parametric sign test. The intervals each have a 98.8% confidence level, yielding a 97.7% joint confidence level. These joint confidence intervals are plotted in Figure 18 against the 71 configurations not in Cluster 1. Based upon the distance between the clusters relative to the width of the confidence intervals, the above results do not appear to be statistically anomalous, but rather seem representative of the true performance of the algorithms on the given problem set.

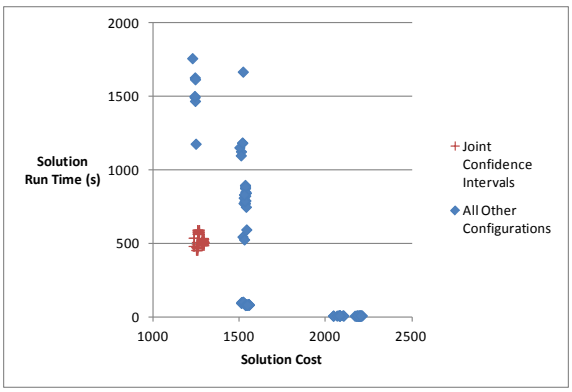


Figure 18: Cluster 1 confidence intervals

3.4.6 Additional Test Problems

Within each cluster is a basis for the cluster; that is, the simplest configuration containing the LS operators necessary for inclusion in the cluster. Those configurations are listed below for each cluster:

Cluster 1: Or-opt

Cluster 2: Cross Exchange

Cluster 3: Or-opt & Cross Exchange

Cluster 4: 2-opt*

Cluster 5: No LS

Each of these five configurations was tested against the 44 remaining problems of Solomon's test set [79] along with just these configurations for problem set R1 for reference. Results are shown in Figure 19. As in the original experiment, results are averaged across all of the problems in the data set for each configuration. Again, practical restrictions with respect to run time disallow testing all 93 configurations on the entire data set. Cluster 6 is omitted; recall this cluster consisted on the lone configuration

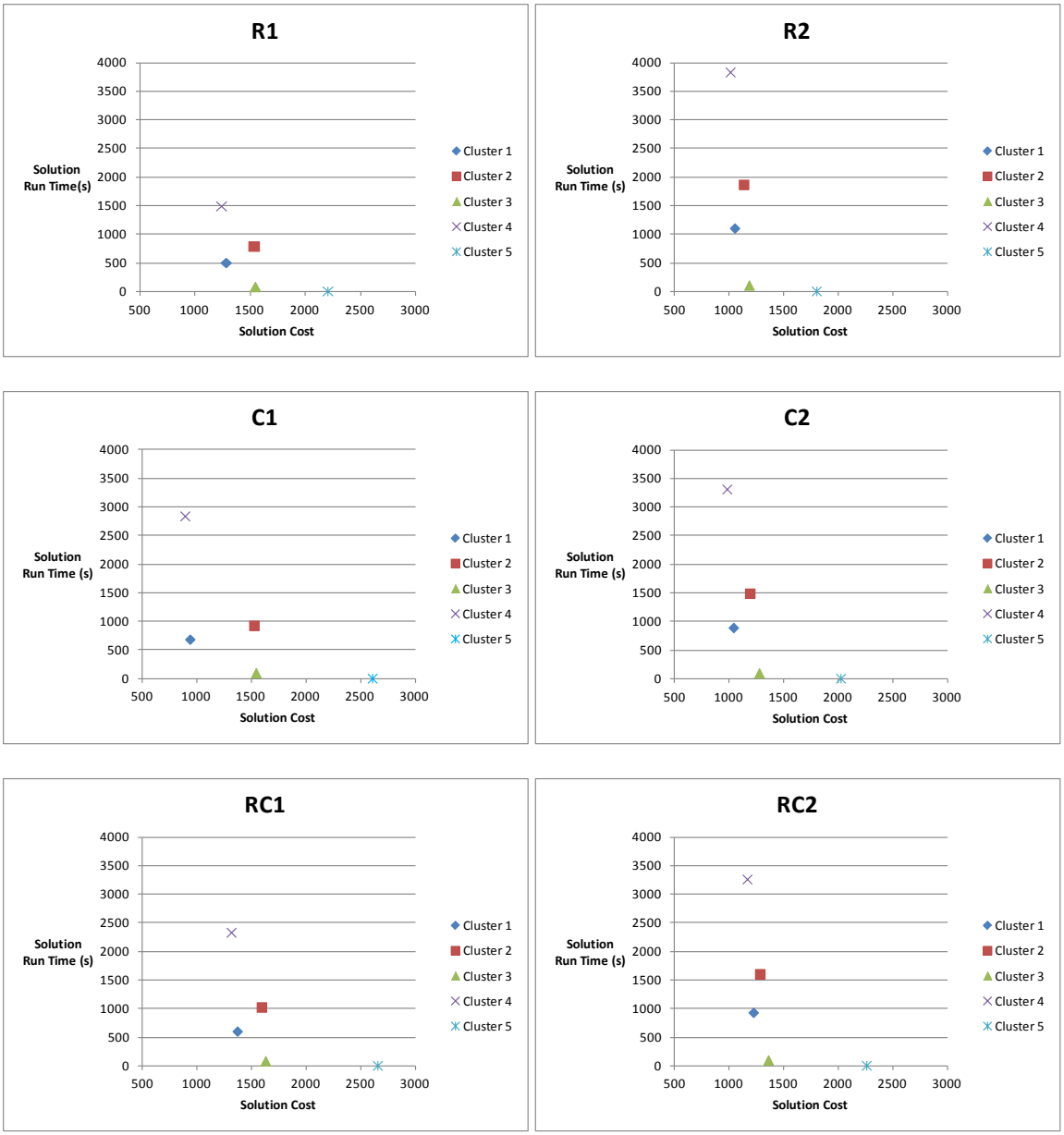


Figure 19: LS performance on additional problem sets

employing Relocate, Cross Exchange, and 2-split-interchange. The results detailed in the previous section strongly suggest this configuration will not match the performance of Cluster 2, let alone any of the non-dominated clusters.

While the performance of the algorithm does exhibit a dependence on the problem characteristics as evidenced by fluctuations in average solution cost and run time, the plots for each of the different problem types exhibit the same basic pattern as the plot for the R1 problem set. In particular, the metaheuristics incorporating Cross Exchange (Clusters 2 and 6) are Pareto-dominated by the metaheuristic that uses Or-opt (Cluster 1). Also, the metaheuristic with both Or-opt and Cross Exchange (Cluster 4) slightly outperforms the Or-opt metaheuristic (Cluster 1) in each of the cases in terms of solution cost, but always at a great expense in run time. Conversely, in terms of solution cost, the 2-opt* metaheuristic (Cluster 3) greatly outperforms the metaheuristic that does not incorporate LS (Cluster 5) with a relatively small increase in run time.

Based upon the results shown, clustered or random-clustered customer locations do not significantly impact the relative performance of the LS configurations. Similarly, extending the planning horizon from vehicles capable of servicing 5-10 customers to vehicles capable of servicing in excess of 30 customers has little impact on the relative performance of the LS configurations.

Next, a representative sample set is tested–similar to the above approach–of each of the sub-clusters of Cluster 1 for the remainder of Solomon's test problems. The representative configurations are:

Cluster 1a: Relocate, 2-opt*, & Or-opt

Cluster 1b: Relocate & Or-opt

Cluster 1c: Or-opt

Cluster 1d: 2-opt* & Or-opt

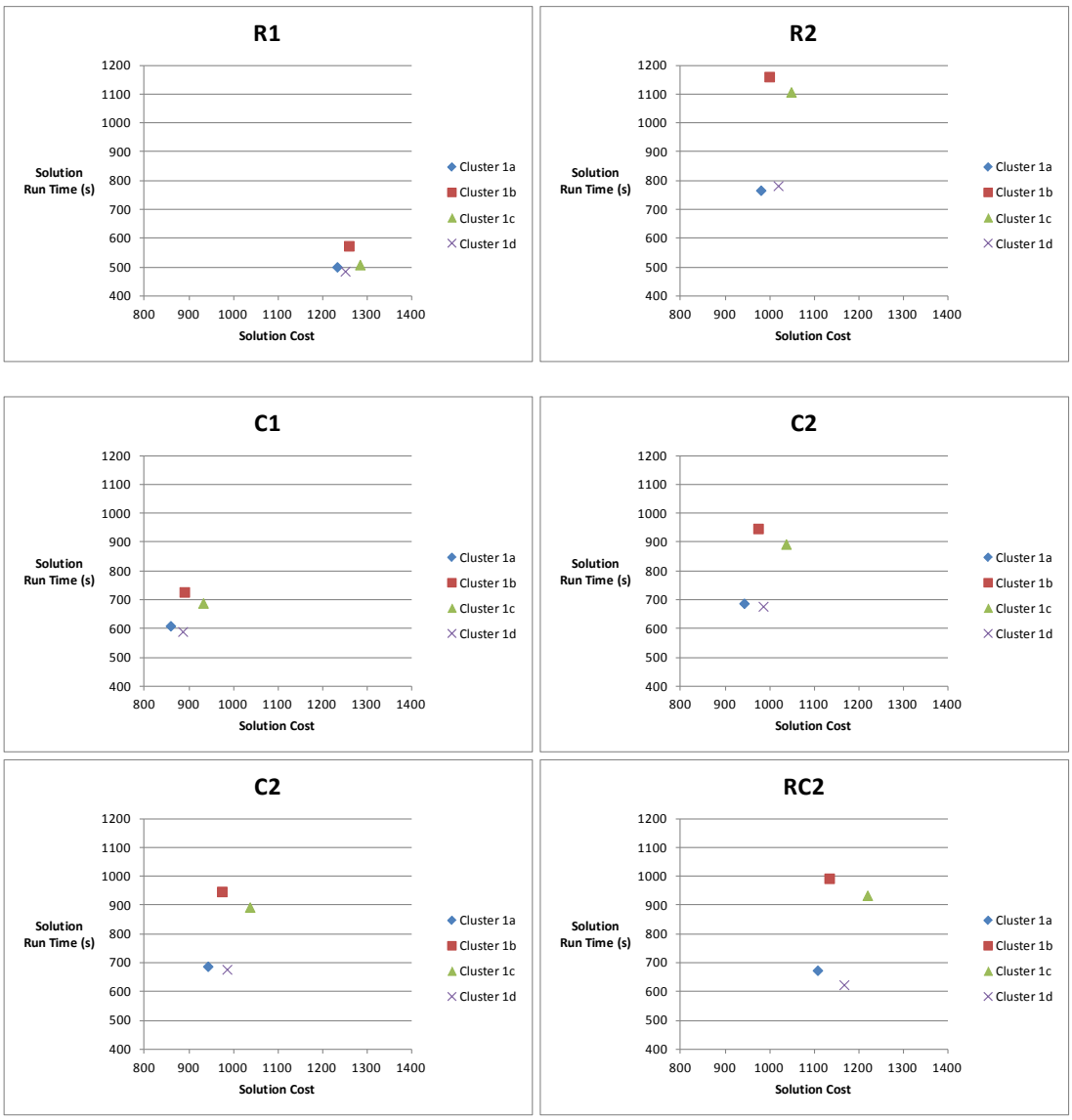


Figure 20: Sub-clusters of Cluster 1 on additional problem sets

Results are shown in Figure 20 and are again the average of the results of each of the individual problems within a test set. For these problems with varying vehicle load capacities and distributions of customer locations, the inclusion of 2-opt* with Or-opt improves both solution cost and run time for all of the problems. Also, including Relocate as a third LS operator improves solution cost on all of the problems, but with

varying impacts on run time. The R2 problem set is the exception where this triplet is the lone optimal solution within Cluster 1, dominating the other four data points. Based upon these samples, the conclusions from the sub-clusters of Cluster 1 on problem set R1– namely, including 2-opt* with Or-opt improves both solution cost and run time and including Relocate as a third LS operator improves solution cost but with an increase in run time–appear to extend to problem sets R2, C1, C2, RC1, and RC2.

This section presented several computational and statistical results on the effectiveness of LS operators applied to the VRPTW. These results are summarized in Section 3.5.

3.5 Conclusions

This chapter shows through computational and statistical results a metaheuristic consisting of an MMAS constructive process paired with a LS performs significantly different depending on the LS operators used. Unless an application has an extreme focus on run time or solution cost, Or-opt or 2-opt* appear to be the ideal LS operators to employ on the SDVRPTW with Or-opt finding higher quality solutions (i.e., lower cost) and 2-opt* requiring less run time.

Future research efforts should further explore some of the details of this analysis. In particular, this research bases the implementation and parameters of the MMAS metaheuristic and LS operators on previous research efforts found in the literature. Further experimentation focusing on these aspects may yield better implementations for each. Also, this research chooses to focus on the cluster of metaheuristics using Or-opt

(Cluster 1) because it balances the trade-off between solution cost and run time. However, three other clusters contain at least one configuration not Pareto-dominated by Cluster 1, and hence, these clusters may deserve more attention if a specific application values cost or run time differently. Similarly, run time concerns dictated only 5 of the 93 configurations are tested on the remainder of Solomon's data set (problem sets R2, C1, C2, RC1, and RC2). Further experimentation on these problem sets is needed in order to develop a more robust empirical data set from which to draw conclusions. Finally, this chapter uses MMAS as the lone construction heuristic, leaving open the research question of how the quality of the initial solution may impact LS performance and the clusters formed.

IV. Examining the Effects of Construction Heuristics and Problem Structure on Solution Quality of the Vehicle Routing Problem with Split Deliveries and Time Windows

4.1 Introduction

The vehicle routing problem (VRP) is an important transportation problem seeking an optimal solution for constructing delivery routes given a depot, a fleet of vehicles and some number of geographically dispersed customers, each having a demand that must be fulfilled. The problem also incorporates characteristics such as travel times and/or distances as well as side constraints such as a maximum vehicle load. This problem is important due to both its widespread application and its complexity in solving. See [26] for a more thorough review of the VRP. The literature addresses several extensions of this problem, including variants having delivery time windows associated with customers (VRPTW) and variants allowing split deliveries to customers (SDVRP). The problem extension including both of these variations has received less attention in the literature. This research sheds further light on this problem, which is important because the addition of these two features more accurately represents important real-world applications of the VRP. Furthermore, the problem and methods used to approach the problem may differ significantly in the presence of these additional characteristics, implying the need for research expressly dedicated to these variants.

Chapter III investigated local search (LS) move operators applied to the vehicle routing problem with split deliveries and time windows (SDVRPTW), revealing strong conclusions about which LS operators are best suited to this problem. In this research,

several features of the SDVRPTW are investigated to determine the influence of these features on the overall solution. This chapter is organized as follows: Section 4.2 gives a literature review of relevant topics. Section 4.3 details the experimental design for and results of testing construction heuristics while Section 4.4 shows how problem features such as split loads and customer demands impact solution quality. Finally, Section 4.5 finishes with conclusions and areas for future work.

4.2 Literature Review

This section covers the relevant literature for the SDVRPTW with a brief overview on LS operators and metaheuristics. This section also discusses these heuristics as they have been applied to the VRP, focusing specifically on applications involving the VRPTW, SDVRP, or SDVRPTW.

4.2.1 LS Move Operators

Chapter III empirically shows the relocate, 2-opt*, and Or-opt LS operators are a good combination for the SDVRPTW, offering near-best solutions compared to other LS operator combinations while also having acceptable run times. Ho and Haugland [8] introduce relocate and 2-opt* for the SDVRPTW. Chapter III adapts Or-opt [32] to the SDVRPTW.

The LS operators are listed and described below. In this case, a delivery refers to a vehicle on a route visiting a customer and making a non-empty delivery. Furthermore, each of the LS operators must return a feasible solution in terms of time windows, vehicle

capacity, and customer demand. Let R_a denote the a^{th} route in the solution and g_i denote the i^{th} delivery on a given route.

1. Relocate:

Two deliveries, g_i, $g_j \in R_a$, are selected and g_i is removed from its original position and inserted following g_j. Figure 21 depicts g_j as occurring after g_i in the initial solution, but it may occur either before or after g_i.

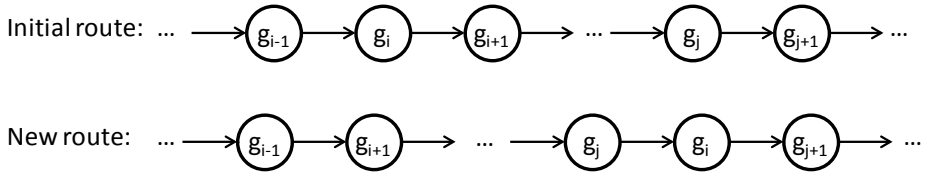


Figure 21: Relocate operator

2. 2-opt*:

Two deliveries, $g_i \in R_a$ and $g_j \in R_b$ ($a \neq b$), are chosen. Then, the edges connecting g_i to g_{i+1} and g_j to g_{j+1} are removed. Two new edges are added adjoining g_i with g_{j+1} and g_j with g_{i+1}. See Figure 22.

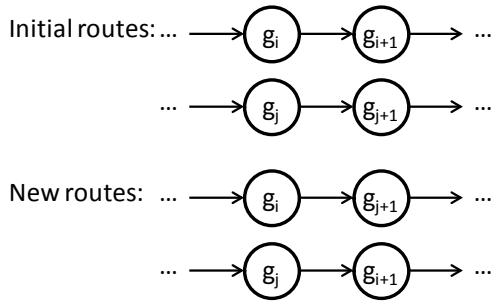


Figure 22: 2-opt* operator

3. Or-opt:

Three deliveries, g_i, $g_{i+\delta} \in R_a$ ($\delta \geq 2$) and $g_j \in R_b$ ($a \neq b$), are chosen. Then, the sequence of deliveries beginning with g_{i+1} and ending with $g_{i+\delta-1}$ is removed from R_a. An edge is then added to R_a such that g_i and $g_{i+\delta}$ are now consecutive deliveries. The removed segment is then inserted into R_b such that g_j precedes g_{i+1} and $g_{i+\delta-1}$ precedes g_{j+1}. See Figure 23.

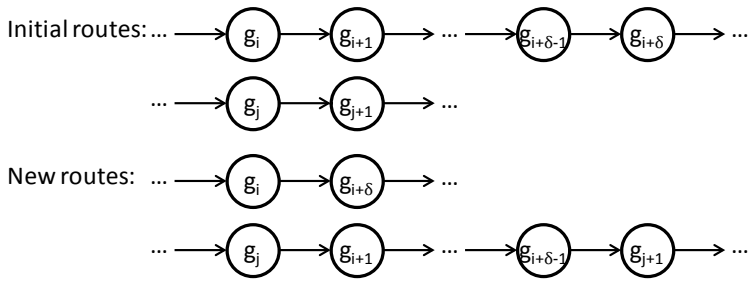


Figure 23: Or-opt operator

4.2.2 Metaheuristics

Testing the performance of the LS operators requires combining these LS operators with a construction heuristic to form a metaheuristic. For this research effort, three basic construction heuristics are chosen: Ant Colony Optimization (ACO), Greedy Randomized Adaptive Search Procedure (GRASP), and Probabilistic Nearest Neighbor (PNN). These three are chosen because they broadly cover the range of available metaheuristics. The ACO heuristic has a learning feature and attempts to build solutions similar to previous good solutions. The implementations of GRASP and PNN used here do not have learning features. Unlike ACO in which the pheromone matrix adapts over time, each iteration of these two construction heuristics has the same inputs (e.g., static distance matrix) every time it is used. These three construction heuristics cover the full range of construction heuristics available in the sense they cover the spectrum of learning and non-learning heuristics as well as the spectrum of exploration versus exploitation. These two basic characteristics define the main thrust of any construction heuristic. With proper choices for parameters the multiple implementations of these construction heuristics, any construction heuristic will fall somewhere within the spectrum covered by these construction heuristics. Note there are other metaheuristics (e.g., tabu search, variable neighborhood search) not covered here. These metaheuristics tend to focus primarily on the local search, which was the purview of Chapter III.

4.2.2.1 Ant Colony Optimization

The ACO heuristic was first introduced by Dorigo [55]. The ACO heuristic iteratively constructs a series of solutions [56] where each ant provides an instance of a

solution construction. Ants probabilistically add components to their individual solutions until reaching a complete solution. The addition of components is based on heuristic and pheromone information about the problem. In the case of a VRP, the heuristic information consists of the edge costs (e.g., cost or time to transit a commodity over a given edge). The pheromone information is gleaned from previous solutions. More specifically, each edge is initialized with the same amount of pheromone. As a portfolio of solutions is built, a local pheromone update decreases the pheromone on certain edges while a global pheromone update deposits additional pheromone onto the "good" edges. In general, a "good" edge is one included in what is deemed a high-quality solution (e.g., "global best" or "iteration best" solution). The pheromone deposit and evaporation rates, as well as the parameters controlling the balance of influence between the heuristic and pheromone matrices, determine how the ACO balances exploration versus exploitation.

This research uses the MAX-MIN Ant System (MMAS), an implementation of an ACO heuristic introduced by Stutzle and Hoos [62]. They show MMAS outperforms Dorigo's original ACO implementation in a test set on symmetric and asymmetric TSPs and quadratic assignment problems [58]. The primary change in the MMAS variant compared to other ACO implementations is the inclusion of explicit upper and lower bounds on the pheromone levels for each edge. This characteristic helps the algorithm avoid early stagnation at a sub-optimal solution.

4.2.2.2 Greedy Randomized Adaptive Search Procedure

Resende and Ribeiro [87] offer a concise review of another popular construction method: GRASP. The GRASP heuristic functions similarly to the ACO in that it

constructs a solution by iteratively adding edges. Where it differs is in the edge selection procedure. The GRASP procedure entails composing a restricted candidate list (RCL). Given a parameter $\alpha \in [0,1]$, an edge is included in the RCL if it is less than the product of α and the difference between the closest and farthest feasible neighbors. Choosing the level for α allows the implementation to vary from completely greedy ($\alpha=0$) to completely random ($\alpha=1$). Some popular implementations of GRASP (e.g., Reactive GRASP and path-relinking [87]) are more complicated and use learning mechanisms on top of this simple construction method. However, as discussed above, this simplistic implementation of GRASP is specifically chosen because it does not have a learning feature with the intent of determining the usefulness of the learning feature of ACO by comparing it to metaheuristics without such a feature.

4.2.2.3 Probabilistic Nearest Neighbor

The PNN construction heuristic is the most simplistic of the heuristics used here. This implementation defines an empirical distribution of all feasible neighbors based on the linear distance from the current location. This distribution then guides the edge selection process, resulting in edge selections where closer neighbors are more likely to be selected than those neighbors farther away. The implementation of the PNN construction heuristic in this research borrows the idea of a candidate list from the ACO, using a list of size $n/5$ where n is the total number of customers.

Now that the heuristic methods are defined, the next section lays out the experimental design by which the construction heuristics are tested.

4.3 Construction Heuristics

This section describes the process of testing the construction heuristics. Shown first are the experimental design and results of tuning the ACO metaheuristic followed by the experimental design and results of testing the different construction heuristics discussed in the previous section.

4.3.1 Optimization of ACO metaheuristic

Chapter III uses ACO as the sole construction technique. Therefore, the first aspect investigated here is the tuning of the ACO metaheuristic in hopes of achieving better solutions than those found in the previous work. Research identified nine parameters within the ACO metaheuristic of particular interest to this problem. Of those, the following three parameters are not included in the experimental design for two reasons. First, including all nine parameters would yield either an extremely time-consuming experiment (e.g., full factorial) or an experiment with weak conclusions (e.g., fractional factorial). Second, initial results indicated setting the parameters to values identified in previous research efforts yielded good solutions.

4.3.1.1 Experimental Design

Therefore, this research uses the values identified by these previous efforts as discussed below.

1. Size of the candidate list: A candidate list is an explicit restriction on the number of feasible neighbors for each customer. Bell and McMullen [68] researched this parameter for the VRPTW and recommend $n/5$. Some others

[70][69][67] recommend *n*/4; however, as seen below, other parameters are in further need of testing. Therefore, this experiment uses *n*/5.

2. Upper Pheromone Bound: Stutzle [63] quantitatively develops the upper pheromone bound used in this experiment. It is $\frac{1}{1-\rho} * \frac{1}{s_{opt}}$ where s_{opt} is the optimal solution and ρ is defined below. Since obtaining s_{opt} is non-trivial, the algorithm runs one iteration of the MMAS metaheuristic without pheromones and uses the best solution value obtained as an approximation for s_{opt}.
3. Pheromone deposit rate: Wang and Yu [72] propose a pheromone deposit rate of *n* divided by the cost of the global best solution. Others—e.g., [75], [69], [88], and [20]—propose various rates representing some fraction of the cost of the global best solution. However, this deposit rate is highly correlated to the distance between the upper and lower pheromone bounds. The relative size of the pheromone deposit is dependent on how many deposits are required to reach the upper bound from the current pheromone level. Therefore, this research fixes the deposit rate at *n*/cost as suggested by Wang and Yu while focusing experimentation on the pheromone lower bound.

Results from Chapter III indicate a combination of the relocate, 2-opt*, and Or-opt operators is the preferred LS operator combination, but permutations were not investigated. Here, the six permutations are investigated. Results indicate the only "bad" designs are those that use Or-opt ahead of 2-opt* because these designs had run times

50% greater than those that use 2-opt* ahead of Or-opt with negligible differences in solution quality. Therefore, the LS operator order is fixed with 2-opt* first, followed by Or-opt, and finally Relocate.

The following six parameters were either less investigated in prior literature or results from various sources conflicted. Therefore, the following are included in the experiment. Also discussed here is the rationale for choosing the levels for each of the factors.

1. Number of ants. Recommendations varied amongst sources, ranging from *n* (see [62] and [67]) to *n*/10 (see [21] and [89]). This range of values is used in this experiment.
2. Pheromone evaporation, ρ. Evaporation rates also varied amongst sources, ranging from 0.5 (see [72] and [19]) to 0.99 (see [90]). Most researchers—e.g., [20], [21], [58], [59], [67], [75], and [89]—conclude a value for ρ between 0.5 and 0.9 is appropriate, so those bounds are used here.
3. Pheromone deposit. Sources conflict on whether a global or iteration best solution is preferred as a basis on which edges to deposit pheromone. Some (see [64] and [70]) argue for use of the global-best solution while other sources such as [58] prefer an iteration-best solution. Both are tested in this experiment along with linear combinations of the two—e.g., deposit half of the pheromone from both a global-best and an iteration-best solution on the appropriate edges.
4. p_{best}. This parameter governs the difference between the minimum and maximum pheromone values. More specifically, the maximum pheromone

value is fixed and the minimum pheromone value is then determined by the formula given by Stutzle and Hoos [57]. This formula is dependent on p_{best}. In a separate work, Stutzle and Hoos [58] use 0.95 but provide no rationale. In general, this parameter is not well investigated throughout the literature. This experiment uses a range of 0.1 to 0.9 for p_{best}.

5. α/β. In the ACO construction, the next delivery on a route is chosen by the following rule: p(j = next delivery | current location = i) = $\frac{(d_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{\forall k \in f}\left((d_{ik})^{\alpha}(\eta_{ik})^{\beta}\right)}$

   where f is the set of all feasible deliveries and d and η are heuristic and pheromone matrices, respectively, with the entry in the i^{th} row and j^{th} column corresponding to the edge connecting the i^{th} and j^{th} customers [55]. The values of α and β govern the relative importance of the heuristic and pheromone matrices, respectively. Results of testing these values vary greatly amongst sources, with recommendations for an α/β ratio ranging from 1/1 (see [62]) to 1/20 (see [20]). However, most research (e.g., [19], [57], [58], [67], [74], and [75]) uses a ratio lying between 1/1 and 1/3. Therefore this test sets α = 1 while β varies from 1 to 3.

6. q_0. In addition to MMAS, Ant Colony System (ACS) is the other popular variation on ACO [60]. One of its primary differences from the original ACO is the introduction of the parameter q_0 where $0 \leq q_0 \leq 1$. Under ACS, a random number, q, between 0 and 1 is drawn. If $q < q_0$, then a greedy selection for the next customer is made while constructing a route. Otherwise, the empirical selection procedure shown above is used. A value of $q_0 = 0.9$ is widely used

(see [20], [21], and [68]). This experiment varies the parameter between 0 and 0.9 to investigate its impact of inclusion on an MMAS.

Another contribution of the ACS heuristic is the introduction of a parameter denoted here as φ. This parameter governs a second pheromone update called the local update where a pheromone edge is evaporated by φ every time it is selected for inclusion in a solution. This encourages further exploration amongst the ants. This parameter is not included in the initial experiment but is used in later experiments in Section 4.3.2.

4.3.1.2 Results

All experiments in this paper were conducted using MATLAB on a Windows 7 x64 PC with an Intel Xeon CPU E5-1650 @ 3.2GHz with 32 GB of RAM. In this section, a full 3-level factorial design with 2 replications using the six parameters discussed above tested on Solomon's R101 problem [79] indicates the factors shown in Table 9 are significantly influential on solution cost as indicated by their p-values with Type I error (traditionally called α, but not to be confused with the α governing the pheromone update) controlled at 0.05. Based on this information, a series of smaller experiments focusing on this subset of data were crafted. However, these experiments, even when tested over the same range of parameters, yielded vastly different results. Some experiments showed no influential factors beyond the intercept estimate. Others indicated first order terms initially shown as not influential are now influential. Overall, the results from subsequent experiments did not align in a cogent manner with the results of the original experiment.

As an example, one such follow-on experiment is described in detail here. First, note in Table 9 the vast majority of the explanatory power lies in a combination of the first order effects and a single two-way interaction—more specifically, the first four terms listed in the table. This is evidenced by p-values all <0.01. Based on this observation, a 28 run design was built using the Custom Design feature in the JMP statistical software to estimate first order effects and two-way interactions as well as the pure quadratic terms. The six most significant factors and their p-values from this experiment are shown in Table 10. These results show two significant findings: first, none of the factors are significant at the 0.05 level for Type I error. Second, even the marginally significant factors shown here do not align very well with the results from the initial experiment. Only the first order term for β and interaction between ρ and Pheromone Deposit appear in both tables and their p-values in the second experiment indicate it is unlikely either significantly impact solution quality.

Table 9: Full Factorial ACO Results

Term	p-value
q_0	<.0001
Number of Ants	<.0001
ρ x Pheromone Deposit	0.0001
β	0.0064
p_{best} x q_0 x β x Pheromone Deposit	0.0335
p_{best} x q_0 x ρ x β x Number of Ants	0.0391

Table 10: Follow-on Experiment

Term	p-value
q_0^2	0.0527
Number of Ants x Pheromone Deposit	0.0772
q_0 x Pheromone Deposit	0.0903
ρ x Pheromone Deposit	0.1898
ρ^2	0.2138
β	0.2356

Even though some factors are statistically significant in individual experiments, these conflicting results likely indicate none of the factors exhibit a consistently strong influence. In other words, an influential parameter for one problem instance may not be significant for another instance. Overall, this indicates the choice of ACO parameters is

not highly influential on the solution quality for a general SDVRPTW. In turn, this may indicate the choice of construction heuristic is not of great importance.

4.3.2 Testing Different Construction Heuristics

In order to determine if the choice of construction heuristic is important, several construction algorithms were tested.

4.3.2.1 Experimental Design

The following seven algorithms were chosen. See Section 4.2.2 for supporting rationale. Note the size of the candidate list, the upper pheromone bound, and the pheromone deposit rate remain fixed as described above for the two ACO metaheuristics and each of the seven metaheuristics runs for 50 total iterations.

1. ACO1 (tuned for exploration)

   $p_{best} = 0.05$, $q_0 = 0.5$, $\rho = 0.7$, $\beta = 2$, pheromone deposit = iteration-best solution, $\varphi = 0.2$

2. ACO2 (tuned for exploitation)

   $p_{best} = 0.95$, $q_0 = 0.9$, $\rho = 0.5$, $\beta = 5$, pheromone deposit = global-best solution, $\varphi = 0$

3. PNN
4. GRASP1 - $\alpha=0$ (greedy construction procedure)
5. GRASP2 - $\alpha=0.35$
6. GRASP3 - $\alpha=0.65$
7. GRASP4 - $\alpha=1$ (random construction procedure)

4.3.2.2 Results

This section describes the results from testing the construction heuristics described in the previous section. The test set is Solomon's R1, C1, and RC1 sets [79]. These sets were chosen over the R2, C2, and RC2 data sets because Ho and Haugland showed splitting deliveries is more valuable when the customer demands are closer to the vehicle capacity [8].

A Wilcoxon Ranked Sum nonparametric test for matched samples indicates no statistical differences in the means for the number of vehicles used when comparing each metaheuristic. Therefore, the remainder of the results in this research focuses on the solution cost and run time. The results given in Figure 24 are the averages across the indicated problem sets.

The ACO2 metaheuristic—the ACO tuned for exploitation—and the GRASP1 metaheuristic—the greedy construction heuristic—are the only two metaheuristics Pareto optimal for any of the problem sets. The results of the Wilcoxon Ranked Sum nonparametric tests for each data set comparing these two metaheuristics is shown in Table 11 with an alpha = 0.05. Note the greedy algorithm, GRASP1, is statistically faster than the ACO in every situation. The ACO delivers higher quality solutions in the random problems (R1) but neither metaheuristic is statistically better than the other for the clustered (C1) nor random-clustered (RC1) customer sets. When comparing all of the problems simultaneously, the one-tailed version of the test concludes ACO2 is statistically better than GRASP1 with respect to solution cost but the two-tailed test is not statistically significant. The practical significance is likely negligible in this case because

the mean difference in solution costs of 1.8 in favor of ACO2 represents <1% improvement for ACO2 over GRASP1.

The data in Figure 25 and Table 12 are duplicates of Figure 24 and Table 11, again with five replicates, except the problems are Ho and Haugland's first augmented data set with l = 0.01 and u = 0.50 [8]. Note ACO2 is again among the best performers, but GRASP4—the random construction heuristic—is now slightly better in terms of both solution cost and run time in the combined data set.

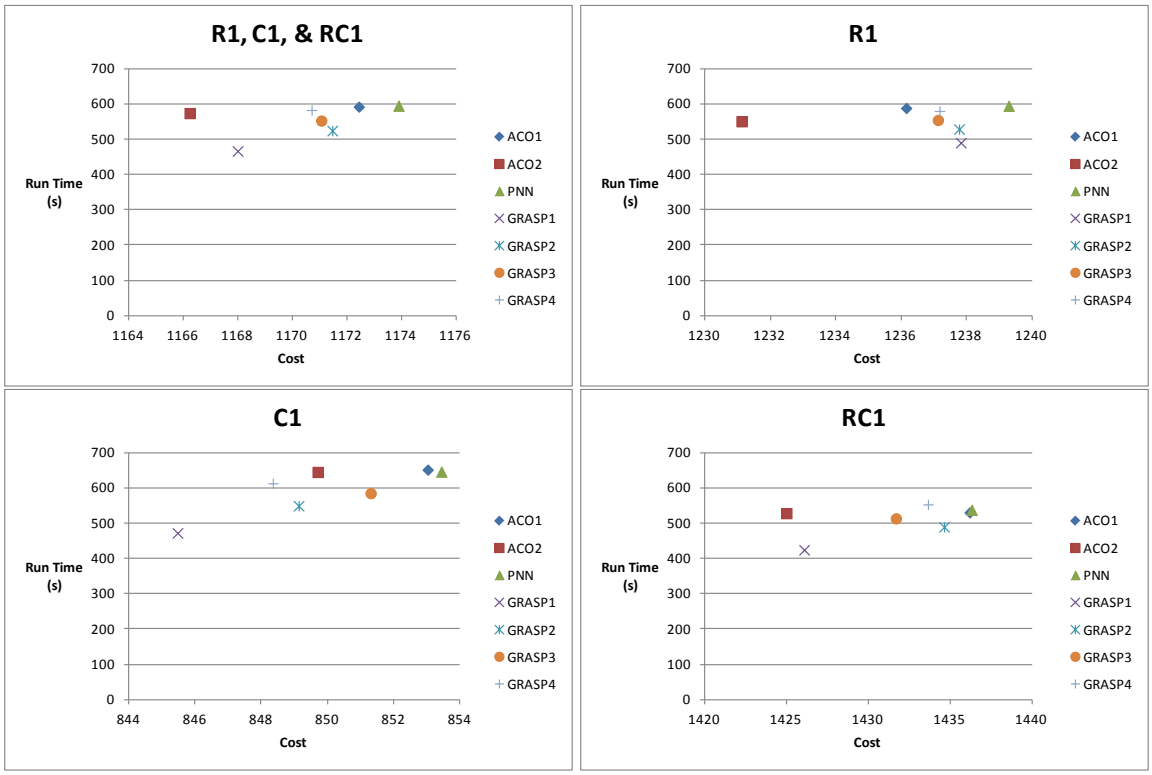


Figure 24: Construction Heuristic Results for Solomon's problems

Table 11: Wilcoxon Ranked Sum Tests for Solomon's Problems

R_C_RC1		R1		C1		RC1	
Cost		**Cost**		**Cost**		**Cost**	
	GRASP1-ACO2		GRASP1-ACO2		GRASP1-ACO2		GRASP1-ACO2
Test Statistic S	1462	Test Statistic S	870	Test Statistic S	-72	Test Statistic S	20
Prob>|S|	0.0821	Prob>|S|	0.0009	Prob>|S|	0.4223	Prob>|S|	0.8951
Prob>S	0.0411	Prob>S	0.0005	Prob>S	0.7889	Prob>S	0.4475
Prob<S	0.9589	Prob<S	0.9995	Prob<S	0.2111	Prob<S	0.5525
Run Time		**Run Time**		**Run Time**		**Run Time**	
	GRASP1-ACO2		GRASP1-ACO2		GRASP1-ACO2		GRASP1-ACO2
Test Statistic S	-10557	Test Statistic S	-1820	Test Statistic S	-1035	Test Statistic S	-814
Prob>|S|	<.0001	Prob>|S|	<.0001	Prob>|S|	<.0001	Prob>|S|	<.0001
Prob>S	1	Prob>S	1	Prob>S	1	Prob>S	1
Prob<S	<.0001	Prob<S	<.0001	Prob<S	<.0001	Prob<S	<.0001

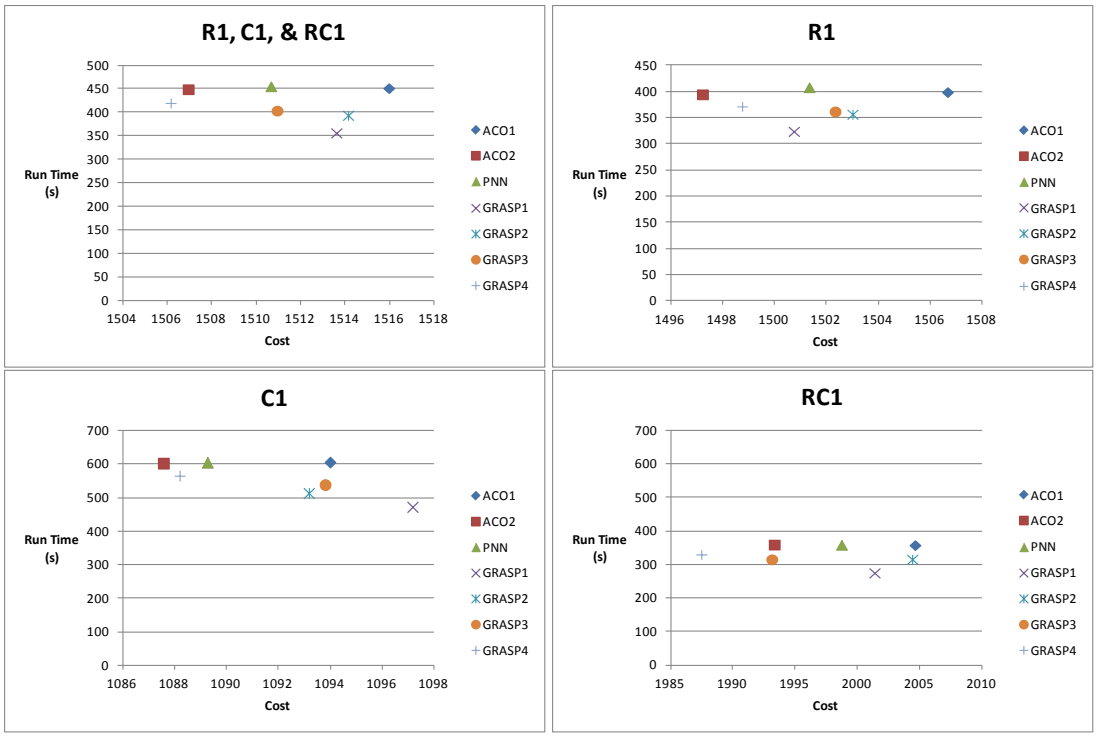


Figure 25: Construction Heuristic Results for Ho and Haugland's first augmented problem set

Table 12: Wilcoxon Ranked Sum Tests for Ho and Haugland's first augmented problem set (statistically significant results highlighted)

R_C_RC1

Cost

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	4335	-270	-5181
Prob>|S|	<.0001	0.7888	<.0001
Prob>S	<.0001	0.6056	1
Prob<S	1	0.3944	<.0001

Run Time

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	-10585	-9697	10585
Prob>|S|	<.0001	<.0001	<.0001
Prob>S	1	1	<.0001
Prob<S	<.0001	<.0001	1

R1

Cost

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	412	220	-382
Prob>|S|	0.1305	0.4227	0.1615
Prob>S	0.0653	0.2113	0.9193
Prob<S	0.9347	0.7887	0.0807

Run Time

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	-1830	-1562	1830
Prob>|S|	<.0001	<.0001	<.0001
Prob>S	1	1	<.0001
Prob<S	<.0001	<.0001	1

C1

Cost

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	771	64	-821
Prob>|S|	<.0001	0.7134	<.0001
Prob>S	<.0001	0.3567	1
Prob<S	1	0.6433	<.0001

Run Time

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	-1035	-1011	1035
Prob>|S|	<.0001	<.0001	<.0001
Prob>S	1	1	<.0001
Prob<S	<.0001	<.0001	1

RC1

Cost

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	278	-264	-466
Prob>|S|	0.0608	0.0756	0.001
Prob>S	0.0304	0.9622	0.9995
Prob<S	0.9696	0.0378	0.0005

Run Time

	GRASP1-ACO2	GRASP4-ACO2	GRASP4-GRASP1
Test Statistic S	-820	-730	820
Prob>|S|	<.0001	<.0001	<.0001
Prob>S	1	1	<.0001
Prob<S	<.0001	<.0001	1

4.3.2.3 Comparing with Previously Published Results

A comparison with previously published results for the SDVRPTW shows ACO2 compares favorably. Table 13 shows the results for ACO2 for Solomon's original problems as well as Ho and Haugland's [8] augmented data sets. Also shown are the results for the same problems given by Ho and Haugland using a tabu search metaheuristic [8] and by Belfiore et al. using a scatter search metaheuristic [7], except Belfiore et al. did not publish results for Solomon's original problem. The ACO2 metaheuristic is competitive, beating both of the other metaheuristics in terms of solution quality for R1 and RC1 of the unmodified data set and C1 of Ho and Haugland's first and second augmented data sets. However, ACO2 does not perform as well on the last two

data sets in which the customer demand is very high relative to the vehicle capacity. This phenomenon is explored further in the next section.

Table 13: Comparison of Algorithms

		R1		C1		RC1	
l,u	Algorithm	Cost	Vehicles	Cost	Vehicles	Cost	Vehicles
unmodified	ACO2	1231.12	14.42	849.71	10.02	1424.98	14.23
	HH	1247.17	13.50	833.76	10.00	1431.94	13.38
	BY	--	--	--	--	--	--
0.01,0.50	ACO2	1497.25	19.82	1087.59	13.38	1993.36	22.53
	HH	1471.49	18.25	1182.12	12.22	1965.05	20.13
	BY	1471.49	18.25	1160.74	12.22	1941.25	21.00
0.02,1.00	ACO2	2395.12	38.47	1883.97	25.51	3489.38	44.58
	HH	2291.46	35.00	2168.57	22.22	3339.20	40.00
	BY	2291.46	35.00	2009.37	24.00	3339.20	40.00
0.50,1.00	ACO2	4132.00	71.08	4160.52	62.85	5732.65	73.67
	HH	4040.67	67.00	3979.78	61.00	5453.10	70.00
	BY	4035.84	69.50	3975.49	60.75	5231.85	73.75
0.70,1.00	ACO2	4700.61	84.25	5198.25	79.89	6310.38	85.50
	HH	4581.54	79.00	4962.28	77.00	6095.20	81.00
	BY	4464.85	82.75	4950.81	76.88	6013.92	82.50

4.4 Problem Structure

Comparing the ACO2 results to the tabu search of Ho and Haugland reveals an interesting difference in solution characteristics. The ACO2 algorithm does not incorporate splitting as heavily as the tabu search. As Table 14 shows, this difference increases as the average customer demand increases. Therefore, the next two subsections detail modifications to ACO2 meant to introduce more split deliveries. Within each section, the effects of changes in customer demands are also explored.

Table 14: Average number of customer deliveries

	R1		C1		RC1	
l,u	HH	ACO2	HH	ACO2	HH	ACO2
0.01,0.50	103	100	102	100	106	100
0.02,1.00	116	107	117	105	119	106
0.50,1.00	167	142	125	124	163	140
0.70,1.00	180	154	171	156	176	154

4.4.1 More splits during construction phase

The constructive phase of the ACO2 algorithm is modified in the following way to attempt to introduce more split deliveries. In the original construction phase of ACO2, when the next customer to be added to a route is chosen, its load is only split if the vehicle does not possess the capacity to fulfill that customer's entire demand. In the modified version of ACO2, denoted ACO2-split, after the next customer is chosen, a random uniform number, x_1, on the interval [0, 1] is drawn. If $x_1<y$, where y is the desired percentage of forced splits, then ACO2-split draws a second random uniform number, x_2, on the interval [1, demand-1] representing the amount of demand added to the current route. If $x_1>y$, the entire demand (or as much as possible if the demand exceeds vehicle capacity) is added to the route.

Results are shown in Table 15. The algorithm that forces splits 25% of the time (i.e., y=0.25) is the only algorithm able to outperform the original algorithm (see the unmodified and third augmented data sets). However, the improvement is miniscule, and the results from the last two augmented data sets, which is where improvement is most necessary relative to the results from Table 13, is significantly worse. Furthermore, the data in Table 16 show the forced splitting modification does not result in any appreciable

gain in the number of split deliveries. It actually uses significantly fewer splits in a couple of the instances (e.g., the fourth augmented R1 problem set with 25% and 100% splitting).

4.4.2 More splits during local search phase

Another option to address the lack of splitting in the ACO2 solutions is to use another LS operator that will introduce new split loads for customers. Chapter III investigates two such techniques, two split interchange and split*. While both of these operators are shown to be inferior to the combination of the 2-opt*, Or-opt, and Relocate operators, they are re-introduced here for the sake of this investigation. Specifically, Table 17 shows the results of the ACO2 algorithm with three combinations of LS operators: ACO2 is the original algorithm with the 2-opt*, Or-opt, and Relocate LS operators; ACO2a uses the two split interchange, 2-opt*, and Or-opt LS operators; and ACO2b uses the 2-opt*, Or-opt, and split*. The order of LS operators is based on the research found in Chapter III. These runs were conducted independently of the runs in Table 13, so while the ACO2 algorithms are identical, the results are not exactly the same.

The results in Table 17 show the original ACO2 algorithm outperforms ACO2a and ACO2b in terms of solution cost on each of the first three data sets (i.e., Solomon's original data sets [79] and the first two augmented data sets from Ho and Haugland [8]). The third augmented data set has mixed results with ACO2 still superior for the clustered and random-clustered data sets but ACO2b having a lower solution cost for the random

Table 15: Forcing splits during construction

l,u	Forced Split %	R1 Cost	R1 Vehicles
unmodified	0	1231.1	14.4
	10	1238.0	14.5
	25	1229.5	14.3
	100	1271.7	14.6
0.01,0.50	0	1497.2	19.8
	10	1501.1	19.9
	25	1501.3	19.8
	100	1542.3	20.3
0.02,1.00	0	2395.1	38.5
	10	2399.3	38.6
	25	2394.9	38.8
	100	2425.0	38.3
0.50,1.00	0	4132.0	71.1
	10	4205.8	71.8
	25	4197.0	73.8
	100	4306.6	78.0
0.70,1.00	0	4700.6	84.3
	10	4931.6	85.1
	25	4876.7	87.3
	100	4905.3	93.2

Table 16: Average number of customer deliveries for R1 problems

	Forced Split Percentage			
l,u	0%	10%	25%	100%
0.01,0.50	100	100	100	102
0.02,1.00	107	106	105	107
0.50,1.00	142	142	140	131
0.70,1.00	154	155	149	126

data set. For the fourth augmented data set, ACO2b is superior to the other two algorithms in all three instances. This means the introduction of a splitting LS operator is only beneficial in the case of extremely high load factors as is the case for the fourth augmented data set.

However, even on the latter two augmented data sets, ACO2, while worse than ACO2b, still performs quite similarly with the differences between the two algorithms being less than 1% in each case. Therefore, these results reinforce the conclusion from Chapter III stating the combination of 2-opt*, Or-opt, and Relocate LS operators represents a strong LS search for the SDVRPTW.

Furthermore, while ACO2b slightly outperforms ACO2 on the fourth augmented data set, that performance is not the result of more splitting as anticipated. In fact, as Table 18 shows, ACO2b uses an average of one less delivery in two of the instances and the same number of deliveries in a third. While the final solutions from ACO2b do not use more splits, the splitting option in the LS phase may give the LS more flexibility in finding intermediate solutions given the large customer demands.

Table 19 shows the average vehicle utilization for each of the ACO2, ACO2a, and ACO2b algorithms. For each problem in the set, the load for each vehicle is averaged. Then, those averages for each problem are averaged again, yielding an overall average vehicle utilization rate for each problem set. These data show as the customer demand ratio increases, vehicle loading becomes a dominant problem feature. The solutions for Solomon's unmodified problem set achieve results on par with the results of Ho and Haugland (see Table 13) with relatively poor vehicle utilization rates. On average, vehicles are only approximately two-thirds full. However, on Ho and Haugland's

Table 17: Using LS operators that split loads (lowest cost solutions highlighted)

	R1			C1			RC1		
Algorithm	Cost	Vehicles	Run Time (m)	Cost	Vehicles	Run Time (m)	Cost	Vehicles	Run Time (m)
ACO2	1230.7	14.3	7.9	848.6	10.1	9.3	1428.3	14.3	7.5
ACO2a	1249.1	14.4	7.6	870.6	10.0	9.0	1460.2	14.3	6.9
ACO2b	1247.7	14.5	7.5	866.1	10.0	9.6	1456.9	14.4	7.2
ACO2	1492.0	19.8	5.2	1085.2	13.0	8.7	1986.6	22.3	4.8
ACO2a	1518.2	19.9	5.1	1096.9	13.3	8.3	2028.4	22.3	4.6
ACO2b	1522.3	19.8	6.3	1093.9	13.3	9.5	2023.4	22.4	5.7
ACO2	2395.2	38.2	4.3	1868.9	25.2	6.6	3454.6	44.3	3.9
ACO2a	2409.9	38.3	4.8	1879.9	25.4	6.7	3501.1	44.4	4.6
ACO2b	2403.5	38.3	5.9	1871.8	25.4	8.1	3464.1	44.3	5.0
ACO2	4091.4	71.0	4.2	4093.6	62.7	3.6	5547.2	74.0	4.1
ACO2a	4087.8	71.0	7.2	4120.1	62.6	5.6	5557.8	73.4	6.8
ACO2b	4084.7	71.2	7.1	4100.8	62.7	5.0	5577.1	73.9	6.0
ACO2	4700.6	84.3	4.1	5198.3	79.9	5.2	6310.4	85.5	4.3
ACO2a	4696.9	83.8	7.5	5213.0	80.0	9.3	6295.1	85.1	7.8
ACO2b	4678.8	84.0	6.6	5176.1	80.0	8.4	6266.8	85.3	5.8

Table 18: Average number of customer deliveries

	R1		C1		RC1	
l,u	ACO2	ACO2b	ACO2	ACO2b	ACO2	ACO2b
0.01,0.50	100	100	100	102	100	103
0.02,1.00	107	107	105	115	106	107
0.50,1.00	142	141	124	131	140	142
0.70,1.00	154	154	156	155	154	153

augmented problems, most rates are above 90%. This indicates efficiently packing the vehicle is the key to good solutions in the presence of high customer demand ratios whereas other problem features—likely the time windows—drive good solutions for problems with lower customer demand ratios.

Table 19: Vehicle utilization percentages

l,u	Algorithm	R1	C1	RC1
unmodified	ACO2	0.63	0.63	0.63
	ACO2a	0.64	0.64	0.64
	ACO2b	0.52	0.83	0.63
0.01,0.50	ACO2	0.93	0.93	0.92
	ACO2a	0.91	0.92	0.92
	ACO2b	0.89	0.85	0.92
0.02,1.00	ACO2	0.93	0.93	0.93
	ACO2a	0.93	0.93	0.93
	ACO2b	0.92	0.88	0.93
0.50,1.00	ACO2	0.95	0.95	0.95
	ACO2a	0.96	0.96	0.96
	ACO2b	0.94	0.96	0.95
0.70,1.00	ACO2	0.95	0.95	0.96
	ACO2a	0.96	0.95	0.96
	ACO2b	0.96	0.95	0.96

4.5 Conclusions

This research shed light on various aspects of the SDVRPTW. The results presented indicate the construction technique used is not of significant impact to the final solution. Therefore, when using a strong LS, the most appropriate construction technique is one that runs quickly. This allows the algorithm to spend more time in the LS phase, shown in Chapter III to be highly impactful. To that end, ACO appears to be a robust construction method, performing well compared to a tabu search [8] and a scatter search [7] on most problems. However, the ACO is weakest in cases where the customer demands are large relative to the vehicle capacity because the ACO, in combination with

these particular LS operators, do not appear to take advantage of splitting enough. In those cases, the tabu or scatter searches may be more appropriate. This performance is not necessarily indicative of the potential of ACO for the SDVRPTW, but rather reflects upon the implementation used here. In particular, this implementation does not take advantage of the splitting option to the same degree as the tabu search. This difference warrants further investigation.

This research also presents several opportunities for future work. First, further experimentation on the ACO parameters may yield more solid results and indicate a better choice of parameters chosen here. Second, ACO was chosen as a representative for learning algorithms. A comparison of ACO with other learning algorithms such as Reactive GRASP or genetic algorithms is warranted. Ultimately however, the results shown here indicate the most logical direction for future research is further investigation on the implementation of the LS. Comparing the results here with those from Chapter III suggest the LS is far more influential than the construction heuristic on overall solution quality and is therefore offers the best chance for significant improvements in solution quality.

V. Application of Techniques from the Vehicle Routing Problem with Split Deliveries and Time Windows to the Military Inventory Routing Problem with Multiple-customer Routes

5.1 Introduction

The inventory routing problem (IRP) is a combination of the vehicle routing problem (VRP) and inventory management. McCormack [5] proposes a military inventory routing problem (MILIRP) in which the delivery vehicles operate in a hostile environment and the risk of loss of vehicles must be taken into account. This is a novel approach to the IRP untouched by the larger research community. McCormack proposes a direct delivery model of vehicle routing. This research will expand McCormack's work by incorporating a vehicle routing metaheuristic that allows for multiple customers per route.

5.2 Literature Review

An IRP solution identifies which customers are scheduled for a delivery during the current time period, how much supply will be carried to each of those customers, routes for each vehicle, and a delivery schedule for each route [91]. Formulations of the IRP vary and can be defined by its characteristics. Coelho et al. list seven such characteristics by which IRP instances may be classified [91]:

- Time horizon. The time horizon considered by the problem may be finite or infinite.

- Structure. The problem may contain one depot and one customer (one-to-one) one depot and many customers (one-to-many), or many depots and many customers (many-to-many).
- Routing. Possibilities are direct in the case of only direct deliveries, multiple in the case that vehicles visit multiple customers, or continuous in the case that no central depot exists.
- Inventory policy. The policy governing the customers' inventory levels may be a maximum level in which the customer has a maximum capacity that may not be exceeded or an order-up-to level in which a delivery to a customer always contains the quantity required to fill the customer's inventory.
- Inventory decisions. In some cases, inventories may be allowed to go into the negative, resulting in a shortage. Shortages may be modeled as back-orders, in which case the shortage is fulfilled in a later time period, or lost sales, in which case the shortage is not filled. Alternatively, the inventory may be restricted to be non-negative. In this case, if the inventory reaches zero, a direct delivery is made to the customer at the expense of a cost penalty.
- Fleet composition. The vehicle fleet may be homogeneous or heterogeneous.
- Fleet size. The fleet may consist of a single vehicle, multiple vehicles, or be unconstrained.

Coelho et al. [91] also use the demand properties to distinguish between problem types. Specifically, if the demand is known *a priori* for each time period, then it is

deterministic. Alternatively, the demand may be unknown, in which case it is generally modeled using a probability distribution.

McCormack [5] defines the MILIRP with the following characteristics: finite time horizon, one-to-many structure, direct delivery routing, maximum level inventory policy, non-negative inventory decisions, homogeneous fleet composition, and multiple vehicles fleet size. However, the second nomenclature offered by Coelho et al. does not adequately describe the MILIRP. The MILIRP models the customer demand deterministically but Coelho et al. [91] assume a deterministic supply whereas the MILIRP assumes stochastic supply. This is because the MILIRP takes hostile threats to the vehicles into account, meaning a vehicle dispatched on a route may reach none, some, or all of the customers. Therefore, the supply is stochastic based upon the probability of success associated with a vehicle's route. McCormack is the first to entertain the notion of stochastic supply within the context of an IRP. Mu et al. [92] consider vehicle breakdowns for a VRP but only allot one spare vehicle because breakdowns rarely occur.

Kleywegt et al. [93] offer a dynamic programming approximation scheme for the IRP with stochastic demand that serves as a model for McCormack's work. One weakness to their algorithm is all of the routing problems are solved in a pre-processing stage, meaning all combinations of deliveries must be solved. Then, within the dynamic program, the optimal set of routes for any set of inputs is already given. This approach is not feasible for larger problems due to the fact that the VRP is NP-hard [70]. Kleywegt et al. alleviate this concern by restricting routes to no more than three customers. This research extends the work of McCormack [5] and Kleywegt et al. [93] by using a multiple customer routing concept (vs. direct deliveries for McCormack) wherein the

routing solutions are generated during the problem solving process (vs. during the preprocessing stage for Kleywegt et al.) with no explicit limit on the number of customers per route. See [91] for a detailed review of the IRP and associated research.

5.3 Pertinent Review of McCormack's Model

To model the risk to the vehicles, McCormack [5] proposes the imposition of a hex grid onto the geography of the problem wherein each hex is assigned a threat level (high or low). The threat map is assumed static within each time period but may change between time periods. Transitions between hexes are assigned a probability based on the threat level of the current hex and the hex into which the vehicle is moving. McCormack then uses Dijkstra's shortest path algorithm [94] to construct paths between each customer and the depot in which the shortest path is the path with the maximum probability of survival. McCormack then employs a dynamic programming approach to solve the MILIRP with direct deliveries. These are the pertinent details of the MILIRP necessary to develop a routing metaheuristic to embed within the MILIRP. See [5] for more details on McCormack's formulation and direct delivery solution method for the MILIRP.

5.4 Routing metaheuristic for the MILIRP

This research incorporates a routing metaheuristic to change the delivery model from direct to multiple. The metaheuristic used is an ant colony optimization (ACO) metaheuristic based on the results from Chapters III and IV. In those chapters, the

authors showed the Max-Min Ant System variant of ACO coupled with 2-opt*, Or-opt, and Relocate local search operators offers strong solutions to the VRP with time windows and split deliveries (SDVRPTW). This algorithm is now adapted to the MILIRP. Following in the nomenclature of McCormack [5], customers are combat outposts (COP), the depot is a brigade support battalion (BSB), and the vehicles are cargo unmanned aerial systems (CUAS).

First, a subset of COPs is defined for whom split deliveries are not allowed. This consideration is a military one where the risk of loss of the CUAS is substantial. Therefore, based on the threat map described above, a threat vector for the set of COPs designates each COP as residing in either a high or low threat environment. Those COPs in a high threat environment will not accept a split delivery, while those COPs in a low threat environment will accept a split delivery. Hence, the threat vector is an *n*-dimensional vector, where *n* is the number of COPs and the i^{th} element is 1(0) if the i^{th} COP is in a high (low) threat environment.

The inputs to the routing algorithm are a distance matrix, a probability matrix, a threat vector, the number of CUAS allotted, the current inventory level of each COP, and a theta vector. The distance matrix is a matrix with the distance from each node (COPs and BSB) to every other node. These distances are the physical distances associated with the path of highest survivability between two nodes. The probability matrix denotes the probability of survival for each of the paths in the distance matrix. The threat vector is described above. The number of CUAS is bounded above by some maximum number of available crews or vehicles. The current inventory levels of the COPs are the levels of

inventory at the current time step. The theta vector defines the parameters for the value curve, described in greater detail below.

Note several usual inputs to the vehicle routing problem with time windows (VRPTW), such as demand, service times, and time windows, are missing from the list of inputs used here. Service times are fixed at zero because the model emulated uses a drop system. In this particular research, the time windows are defined as the entire current time interval, *t*, in the interest of run time concerns and simplicity.

This algorithm also introduces several aspects not typically present on traditional instantiations of the VRP. First, this algorithm introduces a distance limit because the CUAS have a limited range. Therefore, in both the solution construction and improvement phases, each route is restricted by this limit.

The biggest difference between the MILIRP and a traditional VRP is the demands are not fixed in the MILIRP. In a traditional VRP, the demands are fixed and a feasible solution must satisfy those demands. However, in addition to solving a routing problem, the routing portion of this algorithm must also determine what the demand of each COP should be. This differs from the concept of stochastic demand because the demand for each COP is fixed within the current time step, *t*, but the set of deliveries chosen for that time step may satisfy all, part, or none of this demand.

Within the dynamic program, a set of thetas, $[\theta_1, \theta_2, \theta_3]$, is developed iteratively. This set of three coefficients defines a quadratic function as $f(x) = \theta_1 + \theta_2 x + \theta_3 x^2$ which in turn defines the value of deliveries to the COPs. The thetas are constant between the COPs but may change between time steps. The input to the quadratic function is a COP inventory level and the output is the value of that inventory level. In general, the

quadratic function will be either a strictly increasing function or an increasing function on some interval [0, x] where $x <$ COP capacity. In this case, the function reaches a maximum at inventory level equal to x and transitions to decreasing on the interval [x, COP capacity]. The reason for this second scenario is because while delivering any amount to a COP may be beneficial when viewed as an individual delivery, it may be detrimental in the overall scheme because of the risk incurred with deliveries. More specifically, the risk may outweigh the reward of making a small delivery to a well-stocked COP, yielding a decrease in the overall value of the solution when including that delivery.

Given this set of thetas, the concept of the value of a delivery, route, and solution can be defined. The value of a delivery is simply the delta between the current value and the future value (i.e., the value of that COP's inventory if the delivery is made). The value of a route is simply the sum of the values of each delivery. The value of a total solution is slightly more complicated because, in the case of a split delivery, the values are not additive because the value curve is non-linear. Therefore, the total delivery amount to each COP from all routes is calculated and then the future value of each COP's inventory is calculated in the same fashion as for a single delivery. The value of the total solution is then the sum of these future COP inventory values.

Now, given these thetas and the definition of the values above, an initial demand is defined. This initial demand is calculated based on the delta between a current inventory value and inventory level at which the maximum value is achieved. If the delta is negative—meaning the COP's inventory level is above the level at which the maximum value occurs—then the delta is defined as zero. These deltas are then divided

by the sum of all the deltas to give a demand for each COP that is a proportion of the total demand. This proportional demand is then multiplied by the product of the number of vehicles and vehicle capacity to define an initial demand. The demands are then rounded down and each demand is checked against the COP's maximum capacity. If any demand fulfillment would exceed the COP's maximum capacity, then that COP's demand is set to the difference between the current inventory and COP capacity.

The dynamic program also requires routing solutions for any number of CUAS up to the maximum bound given as an input parameter to the routing subroutine. Therefore, the routing algorithm returns solutions for one CUAS, two CUAS, and so forth. A zero CUAS solution is defined as having a value of zero.

The goal of the routing portion is to return a solution with both a high value and secure routes. The algorithm uses a lexicographic ordering of these two priorities with solution value being more important than routing security. Therefore, the ACO metaheuristic is modified thusly: first, solutions are constructed based on value. When the ants choose a COP to add to the current route, high value COPs (i.e., those lower in inventory) are more attractive than low value or higher inventory COPs. Compare this to a traditional ACO in which a physical distance metric constitutes the heuristic information and geographically closer COPs are more attractive. This algorithm uses these values as the heuristic information and combines this with a pheromone matrix in the same manner as a traditional ACO metaheuristic.

Next, the LS attempts to improve the value of the solution. This implementation of the ACO does not explicitly restrict the number of vehicles. Instead, it constructs as many routes as necessary to satisfy all of the COPs' demands and then chooses the

highest value routes. The initial solution is partitioned into two sets: a set of "good" routes and a set of "bad" routes. The good routes are simply the first *m* routes where *m* is the number of CUAS allotted in this particular solution step. The 2-opt* and Or-opt operators then attempt to swap deliveries between the two partitions in an attempt to increase the overall value of the good routes. The total vehicle loads on the bad routes are ignored here to avoid excluding a valuable delivery only because the delivery or deliveries it is replacing in the good route do not fit onto the bad route.

In this phase, the Relocate operator cannot increase the value of the solution because it is an intra-route operator but it is included in the current LS phase because it is possible the 2-opt* and Or-opt operators may be able to further increase the value of the solution by allowing the Relocate operator to rearrange deliveries within a route. These LS operators are run iteratively and in a greedy fashion until the total value of the solution is at a local optimum. The traditional implementations of the LS operators are then applied to only the good routes portion of the solution in order to improve the security of those good routes. This is accomplished by using the probability matrix as an input instead of the distance matrix. Therefore, the "shorter" routes are those with higher probabilities of survival. See Chapter III for details on the LS implementation.

The metaheuristic iterates through these steps for some number of pre-defined iterations. The solution given as an output is for the initial demand as defined above. The next step is to alter this demand in an attempt to allow for a solution of greater value. Therefore, the initial demand is altered in the following way: a COP is randomly chosen and its demand is randomly incremented or decremented by one unit. Again, no demands

are allowed to be negative and the sum of a COP's demand and its current inventory must not exceed the COP's capacity. The ACO metaheuristic is then applied to this new

Pseudocode for routing subroutine
1: For i = 1 : number of CUAS allotted
1: Define initial demand
2: Define initial solution with value = 0
3: While counter < consecutive non-improving solutions limit
1: Implement ACO metaheuristic
1: Construct solutions based on value
2: Improve solutions based on value
3: Improve solutions based on probabilities of survival
2: Compare solution to previous best
1: If higher value, accept new solution and reset counter to 0
2: If not higher value, increment counter
3: Alter demand

Figure 26: Routing metaheuristic pseudocode

demand. If the total value of the solution returned by the routing metaheuristic is higher than that of the previous solution, or if the total value is equal to that of the previous solution but requires fewer vehicles, this solution is accepted and the demand alteration step is repeated. Otherwise, another COP is selected and its demand is randomly perturbed. This method is similar to that of Kleywegt et al. [93] except they use a best improvement schema. In other words, they explore each possible demand change and choose the change yielding the greatest improvement in value, repeating until no further improvement is possible. The size of the problem under investigation here yields this method highly impractical so instead a first improvement schema is implemented in which the first improving solution is accepted. This demand alteration loop continues until some pre-defined bound on non-improving iterations is reached, after which the

highest value solution found so far is accepted. See Figure 26 for the pseudocode for the routing metaheuristic.

5.5 Developing test problems

Testing this algorithm required the generation of a set of test problems. Real-world examples are not available in great enough numbers to sufficiently test the algorithm's performance and no current test set incorporates the threat map aspect. Furthermore, a real-world threat map presents security concerns. Therefore, a new set of notional test problems is generated. This new test set is modeled after the well-known and oft-used set of test problems for the VRPTW generated by Solomon [79]. The test set also expands upon McCormack's [5] use of a hex grid. The hexes are fixed at a size of two for the test set, meaning the distance from the center of a hex to the middle point on an edge is two.

In his test set, Solomon [79] uses customer sets with random, clustered, and random-clustered geographical orientations. Random orientation means customers are randomly scattered throughout the area of interest. In the clustered set, subsets of customers are grouped together. The random-clustered set is a mixture of these two, with some customers grouped together and others randomly spread throughout the area of interest. This test set emulates this geographical relationship with three customer sets, one of each type, with one minor change. In the problem of interest, customers who are very close to the depot are easily resupplied using ground transportation. Therefore, all of the customers in this test set are a minimum of two hexes away from the depot. The

clustered data set groups the customers into four groups while the random-clustered set uses two groups. The number of customers is fixed at 36 because this is the approximate size of the real-world problems of interest. Similarly, the vehicle capacity is fixed at 8000 lbs with delivery increments of 500 lbs because this also reasonably approximates the real-world situation. Customer demands are random on the interval of [4000, 8000], meaning the test problems assume all of the customers' inventories are at least half full at the beginning of the problem.

The main thrust of the problem generation is in the generation of the threat maps. Using Solomon's ideas as a basis, three types of threat maps are developed: random, clustered, and random-clustered. Five distinct instantiations of each threat map are developed and used in conjunction with each of the instantiations of the customer locations, yielding a total of 45 test problems. The final parameter is in deciding the number of high threat hexes. In this problem set, the distance from the center of a hex to the center of any side is two kilometers with a 26x26 hex grid for a total of 676 hexes. Given this size, subject matter experts indicate approximately 10% of the hexes, or 68 hexes, should be high threat. For the random maps, this is accomplished simply by choosing 68 of the hexes to be high threat. For the clustered set, seven hexes are randomly chosen as high threat. Then, each adjacent hex (hexes one step from the original) is assigned a threat level with an 80% chance of being high threat. Each semi-adjacent hex (hexes two steps from the original) is assigned a threat level with a 40% chance of being high threat. To produce the desired parameter of 68 high threat hexes, if the number is too small, then an appropriate number of adjacent and semi-adjacent hexes are chosen to augment the original data with the adjacent hexes having twice the

likelihood of selection compared with the semi-adjacent hexes. Similarly, if the number of high threat hexes is greater than 68, then the appropriate number of adjacent or semi-adjacent hexes are converted to low threat with the semi-adjacent hexes now having twice the likelihood of being converted. Combining these two for the random-clustered data set, the clustered hexes are assigned as above but with only three seed hexes as opposed to seven. Of the remaining low threat hexes, the appropriate number are randomly assigned as high threat such that the total number of high threat hexes is 68. For each of these data sets, the parameters are easily adjusted, allowing for generation of new threat maps to account for varying conditions such as an overall higher or lower threat level or for stronger or weaker clustering of the threats. See Appendix E for the associated data.

5.6 Results

Results are presented for a test problem with simulated inputs from the dynamic program. The customer data is the random set and the threat data is the C1 set from the test problems from Appendix E. These data sets are shown graphically in Figure 27 with customers as black dots, the depot as the green dot, and high threat hexes as red dots. In this example, the theta vector is [0,2000,-2] implying the quadratic function $f(x) = 0 + 2000x - 2x^2$ and the maximum number of vehicles is 6. The limit on non-improving iterations is used as a parameter for comparative results. Initial inventories for the customers are a random vector of values between 2000 and 8000. These initial inventories are developed once and held constant between the instances described below.

The probability of a successful transition between low threat hexes is 0.999, the probability of a successful transition between low and high threat hexes is 0.994, and the probability of a successful transition between high threat hexes is 0.99. Other parameters include a range of 494 kilometers for the vehicles and a vehicle speed of 148 kilometers per hour.

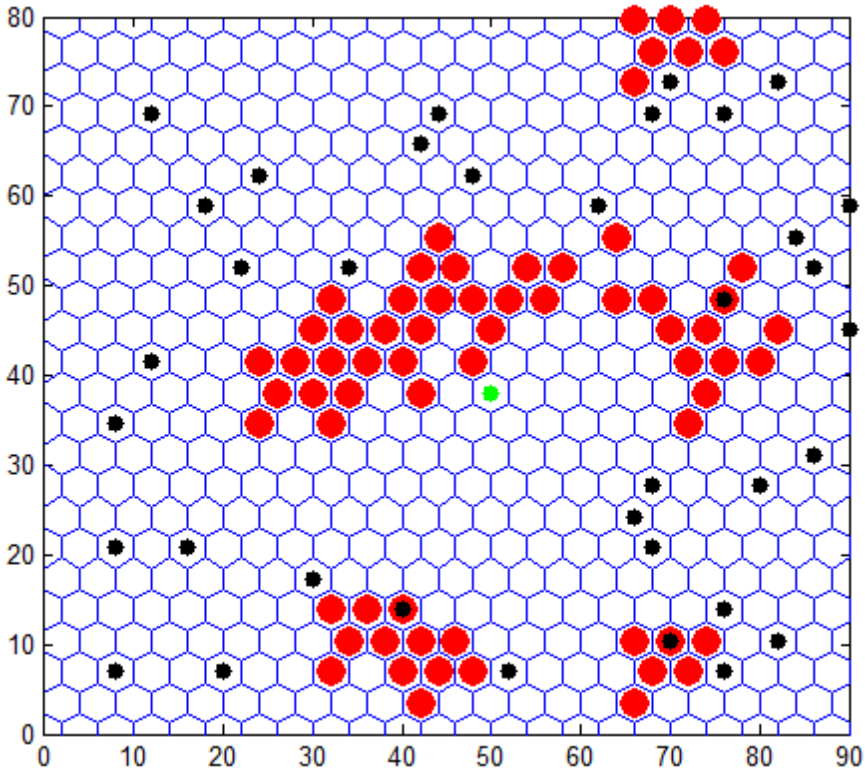


Figure 27: Random customers with clustered threats

Average solutions for three replications for the problem are shown in Table 20. The limit on non-improving iterations is indicated in the table. In the cases with Or-opt, the local search phases consist of the 2-opt*, Or-opt, and Relocate operators while the cases without Or-opt use the 2-opt* and Relocate operators. While not a large enough

Table 20: Routing algorithm results

Or-opt included?	Limit on number of non-improving iterations							
	5				15			
	Total solution time (s)	Number of vehicles	Value	Avg probability of survival for each vehicle	Total solution time (s)	Number of vehicles	Value	Avg probability of survival for each vehicle
No	50.2	1	250.3	0.78	179.6	1	250.0	0.78
		2	407.9	0.84		2	408.0	0.86
		3	506.8	0.90		3	513.4	0.87
		4	552.6	0.91		4	558.3	0.90
		5	557.5	0.90		5	568.4	0.90
		6	470.1	0.91		6	521.0	0.91
	Total solution time (s)	Number of vehicles	Value	Avg probability of survival for each vehicle	Total solution time (s)	Number of vehicles	Value	Avg probability of survival for each vehicle
Yes	74.7	1	250.0	0.78	237.1	1	251.0	0.78
		2	407.4	0.86		2	407.5	0.84
		3	508.8	0.89		3	516.1	0.90
		4	558.2	0.89		4	565.4	0.89
		5	568.9	0.90		5	566.8	0.90
		6	467.4	0.91		6	506.7	0.92

sample from which to draw definitive conclusions, these results contain some interesting observations. Based on the results from Chapter III, the inclusion of Or-opt is expected to influence solution quality positively and run time negatively. However, only one of these expectations is met for this particular problem. The solution quality in terms of both value and probability of survival is nearly identical irrespective of the inclusion of Or-opt. The run time increases as expected. The results are also solved using two limits on the number of non-improving iterations—5 and 15 as indicated in Table 20. The case with a higher limit requires more run time as expected but again the solution quality in terms of both value and survivability are quite similar regardless of the limit with the exception of the six vehicle case. When allotted six vehicles, the higher limit runs are able to find better solutions in terms of value in both the cases where Or-opt is and is not used. This may indicate a higher limit is useful for larger number of vehicles. Therefore,

a dynamic limit that increases as the number of vehicles increases may yield better results.

Overall, these results indicate two things: first, the viability of this method is shown because the routing algorithm is able to solve the problem as expected. Second, the unexpected solution characteristics point to the need for a more detailed analysis into both the method and the parameters used in this experiment. Furthermore, these solution characteristics may hint at an inherently different problem structure for the MILIRP compared to the SDVRPTW.

5.7 Conclusions

This research introduces multiple-customer routing to the MILIRP with results presented for a test problem. This novel approach allows for more flexibility than McCormack's [5] method. The next phase of research will involve subjecting the algorithm to the test problems to determine its effectiveness. Since no previous solutions exist for comparison, the goal is to field an algorithm that produces good solutions, as judged by subject matter experts, within a reasonable time frame. Beyond that, this method is merely an initial attempt and improvement is likely possible. Adapting more complex solution methods used on the IRP as documented in [91] will likely yield superior results for the MILIRP.

The addition of time windows to the problem may require significant changes to the routing algorithm. In the current implementation, the time windows for all customers cover the entire period. In effect, this means time windows are not incorporated into the

current method. In a future effort incorporating this aspect, the time windows will be randomly generated within each period. This randomness is preferred for the MILIRP because unpredictability is important for security purposes. A customer in the MILIRP does not want to be forced into a predictable pattern of deliveries.

If time windows are in effect, it may be necessary to place the customer(s) being removed from the good route onto a new route entirely within the 2-opt* and Or-opt operators during the value LS phase. This is necessary because the algorithm should not exclude an improving solution simply because the customer(s) being removed from the good route does not fit into the bad route. However, if time windows are simply ignored on the bad route, then it may be infeasible. This is not an issue for a bad route, but in later iterations the algorithm may try to move part of this infeasible route back into a good route, thereby yielding an infeasible solution.

Furthermore, the traditional VRPTW assumes a waiting time adds to the time of the route but not the cost (e.g., it doesn't cost a truck to sit in a parking lot other than the lost time). However, in the case of the MILIRP with CUAS as the vehicles, loitering does add to the cost of the solution because it decreases the range of the CUAS. This waiting time must be accounted for in the cost of the solution.

This research also uses a lexicographic ordering of route value and route security. Furthermore, physical distance is not accounted for except in limiting the total range of each vehicle. It is not used as an objective. The application of more sophisticated multi-criteria optimization techniques may yield solutions that better balance these competing objectives.

VI. Conclusions

6.1 Original Contributions

This dissertation contains several original contributions. First, the rigorous testing of LS operators for the SDVRPTW is unprecedented. Results indicate the LS operators from VRPTW literature tend to outperform the operators developed for the SDVRP. Specifically, this work showed, of the eight LS operators tested, the inclusion of 2-opt*, Or-opt, and Cross Exchange impacted performance in terms of solution quality and algorithm run-time far more than the other five operators. This research also concludes the choice of LS operator is far more important than the number of operators used. Furthermore, the customers' physical dispersion (i.e., random vs. clustered vs. random-clustered) does not affect these results.

This research also tested several construction heuristics, concluding the choice of construction heuristic is of minimal significance to the overall solution quality. This research also shed some light on how the ratio of customer demand to vehicle capacity affects solution results. First, the results show LS operators which split loads are only beneficial in the cases of extremely high customer demands relative to the vehicle capacity. Second, this research indicates high quality solutions for the SDVRPTW are possible with relatively poor vehicle utilization rates if the ratio of customer demand to vehicle capacity is relatively low. However, as the ratio increases, high quality solutions utilize vehicles more efficiently. The results of this research also indicate forcing more splits during the construction phase is not beneficial to the overall solution.

This research also makes several contributions to advance the state of research on the IRP, specifically to the MILIRP instance of the IRP. The MILIRP is the only known IRP variant with stochastic supply. This research introduces multiple-customer routes to the problem. Previous work used only direct deliveries. The introduction of multiple-customer routes required substantial modifications, most notably the incorporation of the concept of the value of a delivery. Unlike the VRP, the IRP does not generally have set customer demands. Rather, a certain value is acquired from delivering a particular amount of supply to a customer. This research introduced a method that not only constructs routes but also determines the customer demands in such a fashion as to attempt to maximize the value of the total solution. This research also introduced the idea of a "partial" split delivery problem in which certain customers do not allow split delivery. Finally, a set of test problems is proposed for the MILIRP. This set of test problems varies the customer locations as well as the threat locations, each being random, clustered, or random-clustered.

6.2 Future Work

Study of the SDVRPTW is relatively sparse, particularly in the area of heuristics. While this research offers many insights into this particular problem, much work remains. Future research efforts should further explore some of the details of this analysis. In the study of LS operators, this research chooses to focus on the cluster of metaheuristics using Or-opt (Cluster 1) because it strikes a balance between solution cost and run time. However, three other clusters contain at least one configuration not Pareto-dominated by

Cluster 1, and hence, these clusters may deserve more attention if a specific application values cost or run time differently. Similarly, run time concerns dictated only 5 of the 93 configurations are tested on the remainder of Solomon's data set (problem sets R2, C1, C2, RC1, and RC2) [79]. Further experimentation on these problem sets as well as the augmented data set proposed by Ho and Haugland [8] is needed in order to develop a more robust empirical data set from which to draw conclusions.

This research also uses a "first-improving" implementation for all of the LS operators. Future research should study the impact of other schemes on the solution quality and run-time. This work also allows each LS operator to reach a local minima before moving on to the next operator. A more complex integration of the operators may yield promising results.

The routing heuristic proposed for the MILIRP is also a first effort. The goal is a working algorithm by which future work can be judged. The SDVRPTW algorithms of Ho and Haugland [8] and Belfiore et al. [7] outperform the ACO algorithm proposed by this research in many of the cases tested, perhaps indicating ACO is not a good choice as a metaheuristic and further improvement is likely possible.

Furthermore, time windows are basically excluded in this instance of the MILIRP. Future work should seek to fully incorporate time windows and study the effect of the characteristics of the time windows (e.g., length of the time windows and relationships between time windows in terms of the location within the period). The stopping criterion for this heuristic is simply some number of non-improving iterations. Future work should seek to explore this criterion in hopes of developing a more robust stopping criterion.

In addition, value and survivability are handled in lexicographic order. Future work should incorporate more sophisticated multi-objective optimization procedures in hopes of finding a more robust solution set.

Appendix A: Pairwise comparisons of LS operators

LS1: Relocate
LS2: Split-to-single
LS3: 2-opt*
LS4: Or-opt
LS5: Cross Exchange
LS6: Two split interchange
LS7: Combine
LS8: Shift*

Configuration	Permutation 1		Permutation 2		Difference (expressed as (P2-P1)/P1)	
	Cost	Run Time	Cost	Run Time	Cost	Run Time
LS1 & LS2	2080.65	8.39	2066.94	9.06	-0.01	0.08
LS1 & LS3	1508.27	95.26	1511.29	104.04	0.00	0.09
LS1 & LS4	1259.22	538.29	1257.30	609.87	0.00	0.13
LS1 & LS5	1507.78	1126.25	1513.48	1262.19	0.00	0.12
LS1 & LS6	2044.06	8.63	2092.99	9.48	0.02	0.10
LS1 & LS7	2081.09	8.48	2053.31	8.46	-0.01	0.00
LS1 & LS8	2099.18	10.98	2075.27	13.16	-0.01	0.20
LS2 & LS3	1531.80	80.95	1536.49	81.11	0.00	0.00
LS2 & LS4	1280.30	494.66	1281.39	482.80	0.00	-0.02
LS2 & LS5	1532.22	824.67	1529.20	796.18	0.00	-0.03
LS2 & LS6	2222.56	7.85	2204.22	8.74	-0.01	0.11
LS2 & LS7	2194.00	7.51	2190.13	8.35	0.00	0.11
LS2 & LS8	2198.26	9.64	2188.16	10.02	0.00	0.04
LS3 & LS4	1249.60	437.90	1253.44	628.16	0.00	0.43
LS3 & LS5	1538.58	595.05	1539.65	933.69	0.00	0.57
LS3 & LS6	1548.99	82.12	1537.56	103.24	-0.01	0.26
LS3 & LS7	1546.25	81.68	1542.06	80.09	0.00	-0.02
LS3 & LS8	1536.96	82.30	1534.85	87.03	0.00	0.06
LS4 & LS5	1238.33	1495.08	1240.87	1734.12	0.00	0.16
LS4 & LS6	1287.45	485.25	1287.41	604.03	0.00	0.24
LS4 & LS7	1277.52	494.28	1283.51	483.15	0.00	-0.02
LS4 & LS8	1280.88	485.02	1284.06	492.1302	0.00	0.01
LS5 & LS6	1525.96	782.81	1520.47	1000.67	0.00	0.28
LS5 & LS7	1529.71	735.47	1529.29	1461.76	0.00	0.99
LS5 & LS8	1536.25	844.67	1541.18	785.0086	0.00	-0.07
LS6 & LS7	2158.94	7.84	2208.09	8.11	0.02	0.03
LS6 & LS8	2204.43	9.91	2177.33	10.14	-0.01	0.02
LS7 & LS8	2210.12	9.78	2193.87	9.86	-0.01	0.01

Appendix B: Average data for problem set R1

Cross-reference Appendix C for details on each configuration.

Configuration	Avg Cost	Avg Run Time
1	2202.05	7.91
2	2042.42	8.77
3	2192.78	7.83
4	1545.48	85.19
5	1280.59	504.21
6	1533.53	792.19
7	2186.39	8.40
8	2191.22	7.93
9	2193.10	10.08
10	2080.65	8.39
11	1508.27	95.26
12	1259.22	538.29
13	1507.78	1126.25
14	2044.06	8.63
15	2081.09	8.48
16	2099.18	10.98
17	1531.80	80.95
18	1280.30	494.66
19	1532.22	824.67
20	2200.00	7.95
21	2177.14	7.66
22	2198.26	9.64
23	1249.60	437.90
24	1538.58	595.05
25	1537.53	82.87
26	1553.84	83.12
27	1536.96	82.30
28	1238.33	1495.08
29	1285.18	507.92
30	1282.90	499.34
31	1280.88	485.02
32	1530.67	773.12
33	1531.39	876.15
34	1536.25	844.67
35	2186.61	7.82
36	2204.43	9.91
37	2210.12	9.78
38	1524.90	95.85
39	1254.29	541.72
40	1498.38	1153.67
41	2064.51	8.83
42	2103.05	8.53
43	2079.03	10.53
44	1232.39	500.56
45	1506.66	1099.23
46	1514.28	101.54
47	1509.48	100.81
48	1519.14	103.32
49	1224.19	1760.29
50	1261.67	568.00
51	1261.43	586.33
52	1264.47	558.04
53	1518.31	1668.52
54	1514.63	1184.67
55	1514.76	1185.95
56	2179.59	7.72
57	2081.69	10.97
58	2074.48	10.71
59	1251.80	445.22
60	1525.86	527.17
61	1543.26	82.88
62	1551.72	83.90
63	1542.92	82.64
64	1240.38	1469.51
65	1285.36	507.80
66	1290.01	502.45
67	1288.52	479.61
68	1523.48	811.11
69	1532.94	851.59
70	1529.20	788.66
71	2168.66	7.93
72	2202.22	10.36
73	2181.99	10.27
74	1244.11	1178.26
75	1250.63	469.38
76	1250.39	475.22
77	1247.75	452.30
78	1516.46	546.25
79	1522.52	774.72
80	1525.27	832.79
81	1550.83	84.12
82	1550.95	87.89
83	1536.38	86.76
84	1240.60	1616.90
85	1239.24	1629.16
86	1237.42	1502.84
87	1285.50	509.48
88	1285.57	465.27
89	1294.96	518.00
90	1531.00	897.79
91	1536.20	749.41
92	1531.55	887.27
93	2196.03	7.64

Appendix C: Six cluster composition

LS1: Relocate
LS2: Split-to-single
LS3: 2-opt*
LS4: Or-opt
LS5: Cross Exchange
LS6: 2-split-interchange
LS7: Combine
LS8: Shift*

Cluster 1

Configuration	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
5	0	0	0	1	0	0	0	0
12	1	0	0	1	0	0	0	0
18	0	1	0	1	0	0	0	0
23	0	0	1	1	0	0	0	0
29	0	0	0	1	0	1	0	0
30	0	0	0	1	0	0	1	0
31	0	0	0	1	0	0	0	1
39	1	1	0	1	0	0	0	0
44	1	0	1	1	0	0	0	0
50	1	0	0	1	0	1	0	0
51	1	0	0	1	0	0	1	0
52	1	0	0	1	0	0	0	1
59	0	1	1	1	0	0	0	0
65	0	1	0	1	0	1	0	0
66	0	1	0	1	0	0	1	0
67	0	1	0	1	0	0	0	1
75	0	0	1	1	0	1	0	0
76	0	0	1	1	0	0	1	0
77	0	0	1	1	0	0	0	1
87	0	0	0	1	0	1	1	0
88	0	0	0	1	0	1	0	1
89	0	0	0	1	0	0	1	1

Cluster 2

Configuration	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
6	0	0	0	0	1	0	0	0
13	1	0	0	0	1	0	0	0
19	0	1	0	0	1	0	0	0
24	0	0	1	0	1	0	0	0
32	0	0	0	0	1	1	0	0
33	0	0	0	0	1	0	1	0
34	0	0	0	0	1	0	0	1
40	1	1	0	0	1	0	0	0
45	1	0	1	0	1	0	0	0
54	1	0	0	0	1	0	1	0
55	1	0	0	0	1	0	0	1
60	0	1	1	0	1	0	0	0
68	0	1	0	0	1	1	0	0
69	0	1	0	0	1	0	1	0
70	0	1	0	0	1	0	0	1
78	0	0	1	0	1	1	0	0
79	0	0	1	0	1	0	1	0
80	0	0	1	0	1	0	0	1
90	0	0	0	0	1	1	1	0
91	0	0	0	0	1	1	0	1
92	0	0	0	0	1	0	1	1

Cluster 6

Configuration	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
47	1	0	0	0	1	1	0	0

Cluster 3

Configuration	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
4	0	0	1	0	0	0	0	0
11	1	0	1	0	0	0	0	0
17	0	1	1	0	0	0	0	0
25	0	0	1	0	0	1	0	0
26	0	0	1	0	0	0	1	0
27	0	0	1	0	0	0	0	1
38	1	1	1	0	0	0	0	0
46	1	0	1	0	0	1	0	0
47	1	0	1	0	0	0	1	0
48	1	0	1	0	0	0	0	1
61	0	1	1	0	0	1	0	0
62	0	1	1	0	0	0	1	0
63	0	1	1	0	0	0	0	1
81	0	0	1	0	0	1	1	0
82	0	0	1	0	0	1	0	1
83	0	0	1	0	0	0	1	1

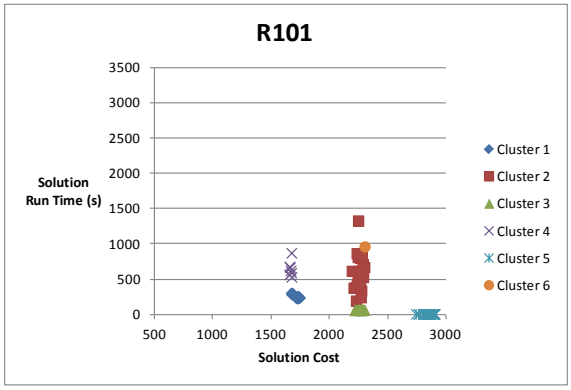
Cluster 4

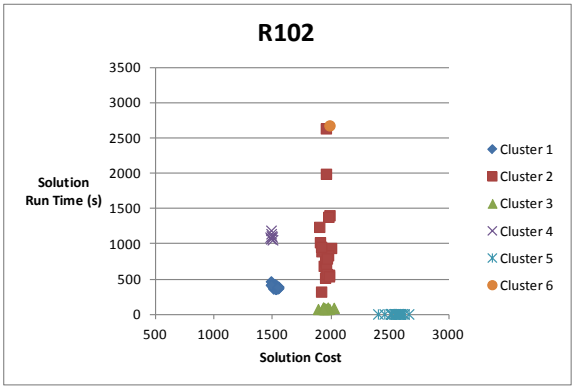
Configuration	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
28	0	0	0	1	1	0	0	0
49	1	0	0	1	1	0	0	0
64	0	1	0	1	1	0	0	0
74	0	0	1	1	1	0	0	0
84	0	0	0	1	1	1	0	0
85	0	0	0	1	1	0	1	0
86	0	0	0	1	1	0	0	1

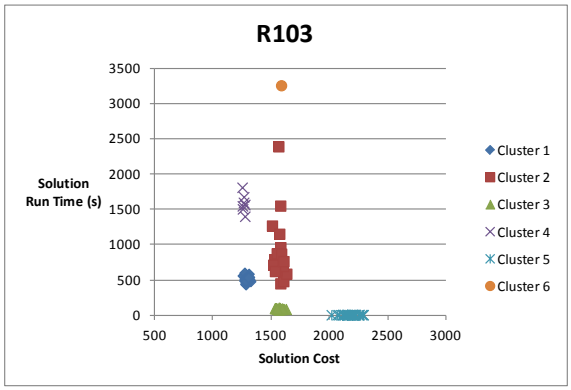
Cluster 5

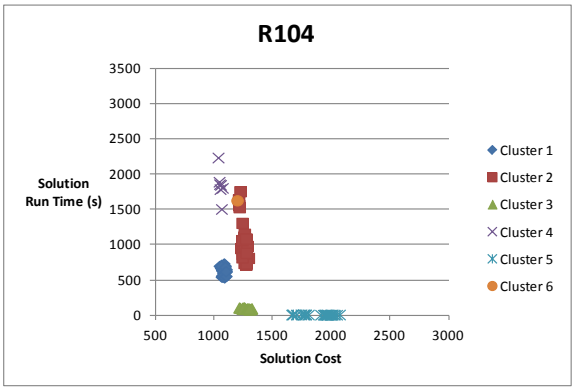
Configuration	LS1	LS2	LS3	LS4	LS5	LS6	LS7	LS8
1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	1
10	1	1	0	0	0	0	0	0
14	1	0	0	0	0	1	0	0
15	1	0	0	0	0	0	1	0
16	1	0	0	0	0	0	0	1
20	0	1	0	0	0	1	0	0
21	0	1	0	0	0	0	1	0
22	0	1	0	0	0	0	0	1
35	0	0	0	0	0	1	1	0
36	0	0	0	0	0	1	0	1
37	0	0	0	0	0	0	1	1
41	1	1	0	0	0	1	0	0
42	1	1	0	0	0	0	1	0
43	1	1	0	0	0	0	0	1
56	1	0	0	0	0	1	1	0
57	1	0	0	0	0	1	0	1
58	1	0	0	0	0	0	1	1
71	0	1	0	0	0	1	1	0
72	0	1	0	0	0	1	0	1
73	0	1	0	0	0	0	1	1
93	0	0	0	0	0	1	1	1

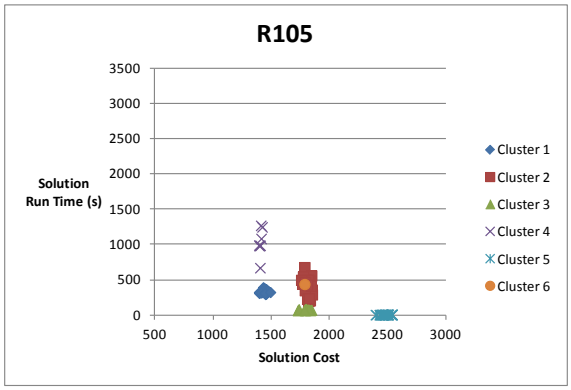
Appendix D: Individual results for problem set R1

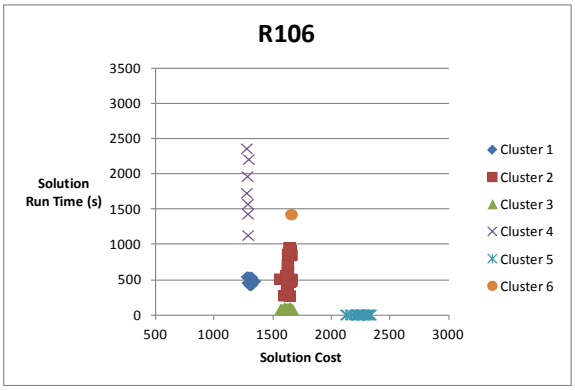


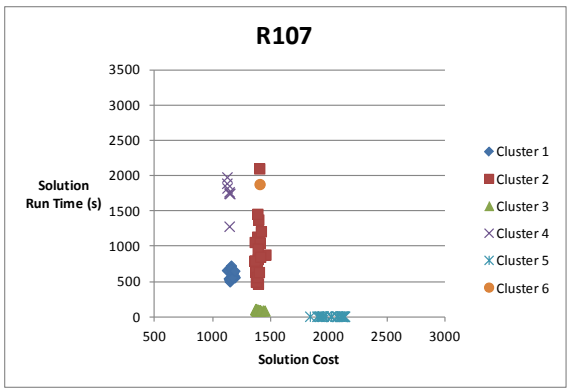


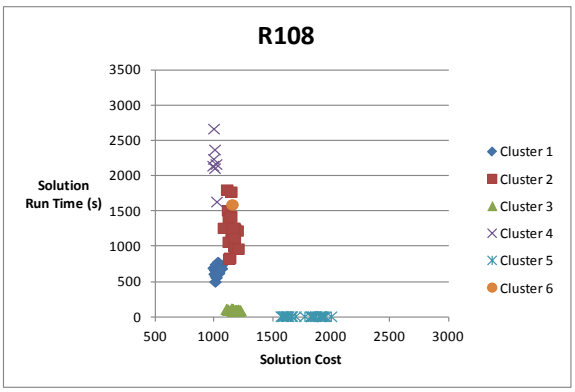


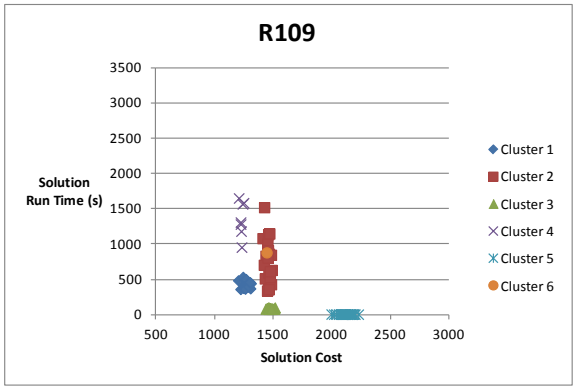


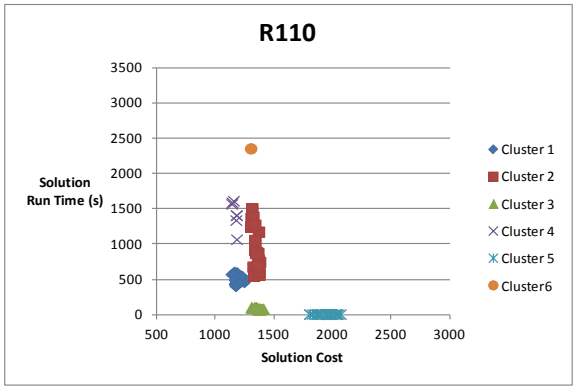


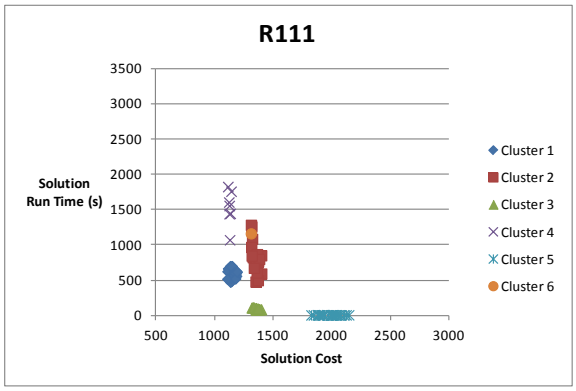


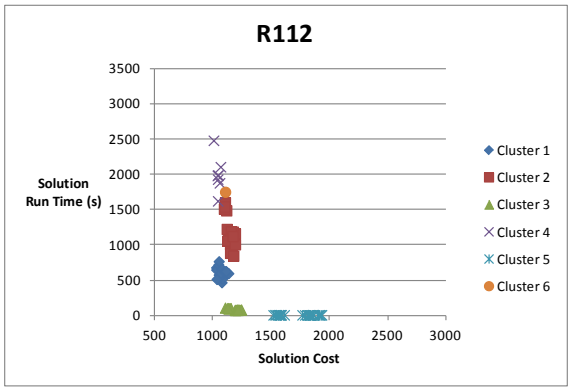

R107
Solution Run Time (s)
Solution Cost
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6


R108
Solution Run Time (s)
Solution Cost
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6


R109
Solution Run Time (s)
Solution Cost
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6


R110
Solution Run Time (s)
Solution Cost
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster6


R111
Solution Run Time (s)
Solution Cost
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6


R112
Solution Run Time (s)
Solution Cost
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6

Appendix E: Test Problems

Customer 37 is the depot for each of the following test problems. A problem instance is constructed by pairing a set of customer coordinates with a threat map and the customer demands.

Customer coordinates and initial inventory levels:

Random customer coordinates

Customer	X Coord	Y Coord
1	88	45
2	76	48
3	68	28
4	8	21
5	64	24
6	84	55
7	84	31
8	84	52
9	76	69
10	48	62
11	68	21
12	12	42
13	20	52
14	80	10
15	20	7
16	28	17
17	32	52
18	8	7
19	12	69
20	76	14
21	52	7
22	16	21
23	60	59
24	40	66
25	8	35
26	24	62
27	40	14
28	44	69
29	16	59
30	88	59
31	68	69
32	76	7
33	80	28
34	68	10
35	80	73
36	68	73
37	48	38

Clustered customer coordinates

Customer	X Coord	Y Coord
1	12	9
2	19	12
3	26	17
4	17	15
5	9	19
6	23	6
7	14	16
8	17	20
9	19	6
10	21	18
11	7	55
12	12	61
13	16	58
14	9	68
15	14	68
16	11	77
17	16	75
18	19	75
19	59	9
20	64	14
21	65	9
22	72	9
23	75	9
24	74	15
25	59	20
26	64	5
27	68	17
28	57	61
29	54	68
30	59	68
31	65	64
32	70	71
33	54	78
34	59	75
35	64	74
36	66	77
37	48	38

Random-clustered customer coordinates

Customer	X coord	Y coord
1	12	9
2	19	12
3	26	17
4	17	15
5	9	19
6	23	6
7	14	16
8	17	20
9	19	6
10	57	61
11	54	68
12	59	68
13	65	64
14	70	71
15	54	78
16	59	75
17	64	74
18	66	77
19	21	19
20	65	41
21	66	10
22	28	8
23	32	60
24	13	33
25	31	48
26	49	14
27	41	70
28	61	66
29	20	68
30	64	34
31	69	22
32	33	29
33	32	10
34	60	57
35	15	43
36	12	72
37	48	38

Customer	Initial Inventory Level
1	7500
2	6000
3	6000
4	8000
5	4000
6	4000
7	5000
8	4000
9	8000
10	7500
11	7500
12	5000
13	7000
14	5000
15	7000
16	4000
17	4000
18	6000
19	6500
20	8000
21	5000
22	6000
23	5500
24	8000
25	8000
26	5500
27	8000
28	6500
29	5000
30	7500
31	7500
32	4000
33	8000
34	4000
35	6500
36	5000
37	4500

Random threats:

R1

X coord	Y Coord
0	0
0	27.71281
2	51.96152
2	86.60254
4	6.928203
4	13.85641
6	17.32051
8	20.78461
8	55.42563
8	62.35383
8	76.21024
10	38.10512
12	69.28203
14	65.81793
14	79.67434
18	45.03332
18	58.88973
18	79.67434
20	34.64102
20	48.49742
20	83.13844
22	24.24871
22	31.17691
24	69.28203
24	76.21024
26	10.3923
28	34.64102
28	83.13844
30	10.3923
32	55.42563
32	62.35383
38	51.96152
38	58.88973
40	6.928203
40	55.42563
40	76.21024
46	51.96152
48	6.928203
48	62.35383
54	51.96152
54	58.88973
56	41.56922
58	24.24871
60	6.928203
60	13.85641
62	17.32051
62	38.10512
62	79.67434
66	45.03332
68	0
68	62.35383
72	6.928203
72	55.42563
74	79.67434
78	65.81793
80	69.28203
82	24.24871
82	31.17691
86	86.60254
88	6.928203
90	10.3923
90	38.10512
90	51.96152
90	58.88973
92	62.35383
96	13.85641
98	65.81793
102	24.24871

R2

X coord	Y Coord
0	27.71281
0	76.21024
4	55.42563
4	62.35383
4	69.28203
8	34.64102
10	65.81793
10	86.60254
14	10.3923
16	0
16	20.78461
16	69.28203
22	65.81793
22	86.60254
26	65.81793
28	20.78461
28	62.35383
30	51.96152
30	58.88973
32	76.21024
34	45.03332
36	69.28203
38	17.32051
38	38.10512
38	79.67434
44	0
46	72.74613
48	0
48	83.13844
50	79.67434
54	10.3923
54	45.03332
54	65.81793
54	72.74613
58	3.464102
58	58.88973
58	65.81793
60	6.928203
62	17.32051
62	24.24871
64	0
66	38.10512
66	51.96152
66	65.81793
68	34.64102
70	86.60254
74	65.81793
74	86.60254
76	0
76	62.35383
78	45.03332
78	51.96152
80	34.64102
82	3.464102
86	10.3923
86	72.74613
88	0
90	17.32051
90	38.10512
90	86.60254
92	20.78461
92	34.64102
94	65.81793
98	51.96152
98	58.88973
102	17.32051
102	45.03332
102	65.81793

R3

X coord	Y Coord
2	45.03332
4	6.928203
4	13.85641
6	31.17691
10	17.32051
12	20.78461
14	24.24871
14	72.74613
18	45.03332
18	79.67434
20	48.49742
20	76.21024
22	3.464102
22	31.17691
22	65.81793
24	13.85641
24	34.64102
34	17.32051
34	24.24871
34	45.03332
34	65.81793
34	86.60254
36	0
36	76.21024
40	55.42563
42	51.96152
42	79.67434
44	76.21024
44	83.13844
46	17.32051
46	24.24871
46	72.74613
52	41.56922
56	13.85641
56	55.42563
60	0
60	48.49742
62	65.81793
66	65.81793
68	69.28203
70	17.32051
70	31.17691
70	45.03332
70	58.88973
72	6.928203
72	13.85641
74	45.03332
74	51.96152
76	27.71281
76	83.13844
78	51.96152
78	65.81793
78	72.74613
80	20.78461
80	34.64102
80	41.56922
84	76.21024
86	17.32051
86	79.67434
88	0
88	48.49742
90	51.96152
92	41.56922
92	48.49742
94	3.464102
94	72.74613
96	27.71281
96	69.28203

R4

X coord	Y Coord
0	6.928203
0	27.71281
4	69.28203
6	24.24871
6	38.10512
6	58.88973
6	86.60254
8	41.56922
10	45.03332
10	51.96152
12	13.85641
12	41.56922
14	38.10512
18	51.96152
20	0
20	20.78461
20	48.49742
22	31.17691
24	34.64102
26	24.24871
26	72.74613
30	51.96152
32	34.64102
34	24.24871
36	55.42563
40	13.85641
44	27.71281
44	62.35383
44	69.28203
44	83.13844
50	10.3923
50	38.10512
52	6.928203
52	13.85641
56	0
56	6.928203
56	13.85641
60	20.78461
60	41.56922
64	0
64	6.928203
64	69.28203
66	58.88973
66	72.74613
68	0
68	6.928203
70	65.81793
72	62.35383
74	24.24871
74	45.03332
76	76.21024
78	24.24871
82	3.464102
82	31.17691
82	45.03332
84	27.71281
84	76.21024
86	3.464102
88	34.64102
90	45.03332
90	65.81793
92	76.21024
94	24.24871
96	34.64102
96	48.49742
98	31.17691
100	6.928203
102	79.67434

R5

X coord	Y Coord
4	48.49742
6	65.81793
6	86.60254
10	58.88973
10	72.74613
10	79.67434
12	41.56922
12	48.49742
12	76.21024
16	34.64102
16	69.28203
18	65.81793
20	55.42563
22	72.74613
22	79.67434
24	6.928203
26	86.60254
34	38.10512
36	55.42563
36	62.35383
38	86.60254
40	6.928203
40	13.85641
40	27.71281
40	41.56922
40	69.28203
42	65.81793
44	27.71281
44	41.56922
46	31.17691
46	86.60254
48	76.21024
52	20.78461
52	55.42563
52	69.28203
54	58.88973
56	62.35383
60	34.64102
62	31.17691
64	20.78461
64	55.42563
68	62.35383
70	24.24871
70	45.03332
70	79.67434
72	62.35383
76	20.78461
76	55.42563
78	17.32051
78	51.96152
80	27.71281
80	76.21024
82	3.464102
82	45.03332
84	34.64102
84	76.21024
88	13.85641
88	69.28203
90	38.10512
90	51.96152
94	3.464102
94	79.67434
96	6.928203
96	62.35383
98	65.81793
100	69.28203
100	76.21024
102	58.88973

Clustered threats:

C1

X coord	Y Coord
24	34.64102
24	38.10512
24	41.56922
30	38.10512
30	41.56922
30	45.03332
32	6.928203
32	10.3923
32	13.85641
32	34.64102
32	38.10512
32	41.56922
32	45.03332
32	48.49742
38	10.3923
38	13.85641
38	41.56922
38	45.03332
40	3.464102
40	6.928203
40	10.3923
40	13.85641
40	38.10512
40	41.56922
40	45.03332
40	48.49742
40	51.96152
46	6.928203
46	10.3923
46	48.49742
46	51.96152
46	55.42563
48	6.928203
48	41.56922
48	45.03332
48	48.49742
54	48.49742
54	51.96152
56	48.49742
56	51.96152
62	55.42563
64	3.464102
64	10.3923
64	48.49742
64	72.74613
64	79.67434
70	6.928203
70	10.3923
70	45.03332
70	48.49742
70	76.21024
70	79.67434
72	6.928203
72	10.3923
72	38.10512
72	41.56922
72	45.03332
72	48.49742
72	76.21024
72	79.67434
78	41.56922
78	48.49742
78	51.96152
78	76.21024
80	41.56922
80	45.03332
96	58.88973
96	69.28203

C2

X coord	Y Coord
0	72.74613
0	76.21024
0	79.67434
6	69.28203
6	72.74613
6	76.21024
6	79.67434
8	69.28203
8	72.74613
14	72.74613
14	76.21024
46	45.03332
46	48.49742
48	45.03332
48	48.49742
48	51.96152
54	41.56922
54	45.03332
54	48.49742
54	51.96152
56	45.03332
56	48.49742
56	51.96152
62	65.81793
64	62.35383
64	69.28203
64	72.74613
70	69.28203
70	72.74613
72	58.88973
72	62.35383
72	65.81793
72	69.28203
72	72.74613
78	3.464102
78	6.928203
78	10.3923
78	58.88973
78	62.35383
78	65.81793
78	69.28203
78	72.74613
80	3.464102
80	6.928203
80	10.3923
80	48.49742
80	51.96152
80	55.42563
80	58.88973
80	62.35383
80	65.81793
80	69.28203
80	72.74613
80	76.21024
86	3.464102
86	6.928203
86	10.3923
86	13.85641
86	55.42563
86	65.81793
86	69.28203
86	72.74613
88	3.464102
88	51.96152
88	55.42563
88	58.88973
94	3.464102

C3

X coord	Y Coord
0	38.10512
0	41.56922
6	38.10512
6	41.56922
8	31.17691
8	34.64102
8	38.10512
8	41.56922
8	48.49742
14	34.64102
14	38.10512
14	41.56922
14	45.03332
14	48.49742
14	51.96152
14	55.42563
16	34.64102
16	41.56922
16	45.03332
16	48.49742
16	51.96152
22	45.03332
22	48.49742
22	51.96152
32	41.56922
38	38.10512
38	45.03332
40	38.10512
40	41.56922
40	45.03332
46	13.85641
46	34.64102
46	38.10512
46	41.56922
48	10.3923
48	13.85641
48	41.56922
54	0
54	3.464102
54	6.928203
54	10.3923
54	13.85641
54	17.32051
56	0
56	3.464102
56	6.928203
56	10.3923
56	13.85641
56	17.32051
62	0
62	3.464102
62	17.32051
64	0
64	3.464102
70	3.464102
70	6.928203
70	31.17691
72	24.24871
72	27.71281
72	31.17691
78	24.24871
78	27.71281
78	31.17691
78	34.64102
80	24.24871
80	27.71281
80	31.17691
80	34.64102

C4

X coord	Y Coord
0	0
0	3.464102
0	6.928203
0	41.56922
0	45.03332
6	3.464102
6	6.928203
6	38.10512
6	41.56922
6	45.03332
6	48.49742
8	0
8	41.56922
8	45.03332
14	45.03332
30	6.928203
30	13.85641
32	3.464102
32	6.928203
32	10.3923
38	3.464102
38	6.928203
38	10.3923
38	13.85641
38	17.32051
40	6.928203
40	10.3923
46	13.85641
48	31.17691
48	34.64102
54	34.64102
54	38.10512
54	79.67434
56	31.17691
56	34.64102
56	38.10512
56	55.42563
56	76.21024
56	79.67434
62	31.17691
62	34.64102
62	38.10512
62	41.56922
62	48.49742
62	51.96152
62	76.21024
62	79.67434
64	31.17691
64	38.10512
64	45.03332
64	48.49742
64	51.96152
64	55.42563
64	58.88973
70	51.96152
70	55.42563
72	6.928203
72	48.49742
72	55.42563
78	6.928203
78	10.3923
78	13.85641
78	17.32051
80	6.928203
80	10.3923
86	6.928203
86	10.3923
86	13.85641

C5

X coord	Y Coord
24	31.17691
24	34.64102
24	38.10512
24	58.88973
24	62.35383
24	69.28203
24	72.74613
24	76.21024
30	31.17691
30	34.64102
30	38.10512
30	55.42563
30	58.88973
30	62.35383
30	65.81793
30	69.28203
30	72.74613
30	76.21024
30	79.67434
32	27.71281
32	31.17691
32	34.64102
32	55.42563
32	58.88973
32	69.28203
32	72.74613
32	76.21024
32	79.67434
38	34.64102
38	38.10512
38	55.42563
38	58.88973
38	62.35383
38	65.81793
38	72.74613
38	76.21024
38	79.67434
40	34.64102
40	38.10512
40	51.96152
40	55.42563
40	58.88973
40	62.35383
40	69.28203
40	76.21024
46	58.88973
46	62.35383
48	24.24871
48	27.71281
48	55.42563
48	58.88973
48	62.35383
54	31.17691
54	34.64102
56	24.24871
56	27.71281
56	31.17691
62	27.71281
62	31.17691
64	24.24871
64	27.71281
64	31.17691
72	3.464102
78	0
78	3.464102
80	0
86	0
86	3.464102

Random-clustered threats:

RC1

X coord	Y Coord
2	51.96152
4	20.78461
4	48.49742
6	17.32051
6	31.17691
6	45.03332
6	65.81793
8	13.85641
8	20.78461
10	17.32051
10	24.24871
10	79.67434
12	13.85641
12	20.78461
14	17.32051
14	38.10512
16	13.85641
18	10.3923
18	17.32051
18	24.24871
20	55.42563
22	10.3923
22	17.32051
24	41.56922
26	31.17691
26	72.74613
32	48.49742
38	45.03332
40	69.28203
42	31.17691
44	48.49742
44	55.42563
46	31.17691
50	58.88973
54	38.10512
54	45.03332
54	51.96152
54	58.88973
56	20.78461
58	45.03332
58	51.96152
58	58.88973
60	20.78461
60	27.71281
60	41.56922
62	51.96152
62	31.17691
64	13.85641
64	20.78461
64	27.71281
64	20.78461
66	24.24871
68	13.85641
68	20.78461
70	17.32051
70	24.24871
70	79.67434
72	20.78461
74	58.88973
82	45.03332
82	51.96152
84	34.64102
84	27.71281
88	41.56922
90	58.88973
92	0
94	3.464102
94	79.67434

RC2

X coord	Y Coord
4	27.71281
4	69.28203
6	45.03332
6	38.10512
8	41.56922
10	51.96152
14	51.96152
18	31.17691
20	48.49742
22	31.17691
24	20.78461
26	24.24871
28	13.85641
30	31.17691
32	13.85641
34	24.24871
36	6.928203
36	13.85641
36	69.28203
36	76.21024
38	10.3923
38	17.32051
38	24.24871
38	79.67434
40	6.928203
40	13.85641
40	76.21024
42	3.464102
42	10.3923
42	51.96152
42	79.67434
44	69.28203
44	76.21024
46	79.67434
48	34.64102
48	76.21024
50	17.32051
50	38.10512
50	79.67434
54	72.74613
56	69.28203
56	6.928203
58	38.10512
60	69.28203
62	65.81793
64	69.28203
64	76.21024
66	51.96152
66	72.74613
66	58.88973
68	69.28203
68	76.21024
68	0
70	72.74613
72	20.78461
72	76.21024
72	62.35383
74	65.81793
74	72.74613
78	72.74613
82	10.3923
84	69.28203
88	76.21024
92	6.928203
92	13.85641
94	65.81793
96	48.49742
102	79.67434

RC3

X coord	Y Coord
8	20.78461
8	27.71281
12	13.85641
14	79.67434
16	6.928203
16	27.71281
18	72.74613
18	79.67434
24	76.21024
26	79.67434
28	41.56922
30	17.32051
32	62.35383
34	17.32051
34	24.24871
36	6.928203
36	34.64102
36	62.35383
36	69.28203
38	65.81793
38	72.74613
40	69.28203
42	65.81793
42	79.67434
44	69.28203
44	76.21024
46	65.81793
46	72.74613
48	0
48	6.928203
48	48.49742
48	69.28203
48	76.21024
50	79.67434
52	6.928203
52	76.21024
54	3.464102
54	17.32051
54	45.03332
56	0
56	6.928203
56	55.42563
56	13.85641
58	3.464102
58	10.3923
60	6.928203
62	3.464102
62	72.74613
64	48.49742
64	69.28203
66	3.464102
66	17.32051
66	24.24871
66	72.74613
68	6.928203
70	24.24871
70	58.88973
70	31.17691
72	48.49742
74	72.74613
78	31.17691
80	62.35383
82	65.81793
88	41.56922
90	38.10512
90	79.67434
92	41.56922
94	3.464102

RC4

X coord	Y Coord
0	13.85641
0	62.35383
0	69.28203
0	27.71281
2	58.88973
4	6.928203
4	20.78461
4	34.64102
4	69.28203
6	17.32051
6	65.81793
6	38.10512
8	13.85641
8	20.78461
8	62.35383
8	69.28203
10	17.32051
10	58.88973
10	65.81793
10	45.03332
12	20.78461
12	48.49742
14	10.3923
14	17.32051
18	17.32051
18	24.24871
18	31.17691
28	69.28203
32	76.21024
34	38.10512
34	65.81793
36	34.64102
38	58.88973
42	38.10512
44	6.928203
44	27.71281
44	83.13844
50	10.3923
54	3.464102
62	24.24871
64	41.56922
64	0
66	38.10512
66	45.03332
66	65.81793
68	41.56922
68	48.49742
68	0
70	45.03332
72	20.78461
72	48.49742
74	45.03332
74	51.96152
74	79.67434
74	45.03332
78	51.96152
84	0
84	41.56922
84	69.28203
86	58.88973
86	65.81793
86	3.464102
88	48.49742
90	65.81793
92	48.49742
94	45.03332
94	24.24871
96	48.49742

RC5

X coord	Y Coord
0	76.21024
2	51.96152
6	31.17691
10	65.81793
12	0
16	69.28203
18	24.24871
18	65.81793
20	62.35383
22	45.03332
26	65.81793
28	76.21024
28	20.78461
28	62.35383
30	10.3923
30	17.32051
40	34.64102
40	41.56922
42	38.10512
42	45.03332
44	34.64102
44	41.56922
44	48.49742
46	38.10512
48	13.85641
48	34.64102
48	41.56922
48	48.49742
48	69.28203
48	83.13844
50	38.10512
50	45.03332
54	38.10512
54	65.81793
58	65.81793
58	3.464102
60	0
60	6.928203
60	41.56922
60	62.35383
60	69.28203
62	10.3923
62	24.24871
62	65.81793
64	0
64	6.928203
64	62.35383
64	69.28203
66	3.464102
66	10.3923
66	24.24871
66	51.96152
66	72.74613
68	0
68	62.35383
68	76.21024
70	65.81793
70	72.74613
74	58.88973
76	0
82	38.10512
82	65.81793
84	55.42563
86	10.3923
90	38.10512
92	41.56922
94	24.24871
102	17.32051

Bibliography

[1] R. Hartlage, "Rough-Cut Capacity Planning in Multimodal Freight Transportation Networks," AFIT, Wright Patterson AFB, OH, 2012.

[2] G. Lambert, "A Tabu Search Approach to the Strategic Airlift Problem," University of Texas, Austin, 2004.

[3] B. Clapp, "Vehicle Minimization for the Multimodal Pickup and Delivery Problem with Time Windows," AFIT, Wright-Patterson AFB, OH, 2013.

[4] M. Hafich, "A Mixed Integer Programming Model for Improving Theater Distribution Force Flow Analysis," AFIT, Wright-Patterson AFB, OH, 2013.

[5] I. M. McCormack, "The Military Inventory Routing Problem with Direct Delivery," Air Force Institute of Technology, Wright-Patterson AFB, OH, 2014.

[6] P. Toth and D. Vigo, "An Overview of Vehicle Routing Problems," in *The Vehicle Routing Problem*, Philadelphia, SIAM, 2002, pp. 1-26.

[7] P. Belfiore, H. Tsugunobu and Y. Yoshizaki, "Scatter Search for Vehicle Routing Problem with Time Windows and Split Deliveries," in *Vehicle Routing Problem*, T. Caric and H. Gold, Eds., Vienna, I-Tech, 2008, p. 142.

[8] S. Ho and D. Haugland, "A tabu search heuristic for the vehicle routing problem with time windows and split deliveries," *Computers & Operations Research*, vol. 31, pp. 1947-1964, 2004.

[9] B. Kallehauge, "Formulations and exact algorithms for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 35, pp. 2307-2330, 2008.

[10] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon and F. Soumis, "VRP with Time Windows," in *The Vehicle Routing Problem*, P. Toth and D. Vigo, Eds., Philadelphia, SIAM, 2002, pp. 157-193.

[11] N. A. El-Sherbeny, "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods," *Journal of King Saud University*, vol. 22,

pp. 123-131, 2010.

[12] K. C. Tan, L. H. Lee, Q. L. Zhu and K. Ou, "Heuristic methods for vehicle routing problem with time windows," *Artificial Intelligence in Engineering,* vol. 15, pp. 281-295, 2001.

[13] C. Archetti and M. G. Speranza, "The Split Delivery Vehicle Routing Problem: A Survey," in *The Vehicle Routing Problem,* B. Golden, S. Raghavan and E. Wasil, Eds., Berlin, Springer, 2008, pp. 103-122.

[14] M. Dror and P. Trudeau, "Savings by Split Delivery Routing," *Transportation Science,* vol. 23, pp. 141-145, 1989.

[15] C. Archetti, M. Speranza and A. Hertz, "A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem," *Transportation Science,* vol. 40, no. 1, pp. 64-73, 2006.

[16] C. Archetti, S. M. Grazia and M. W. Savelsbergh, "An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem," *Transportation Science,* vol. 42, no. 1, pp. 22-31, 2008.

[17] S. Chen, B. Golden and E. Wasil, "The Split Delivery Vehicle Routing Problem: Applications, Algorithms, Test Problems, and Computational Results," *Networks,* vol. 49, no. 4, pp. 318-329, 2007.

[18] D. Gulczynski, B. Golden and E. Wasil, "The split delivery vehicle routing problem with minimum delivery amounts," *Transporation Research Part E,* vol. 46, pp. 612-626, 2010.

[19] G. P. Rajappa, "Solving Combinatorial Optimization Problems Using Genetic Algorithms and Ant Colony Optimization," University of Tennessee, Knoxville, 2012.

[20] D. Favaretto, E. Moretti and P. Pellegrini, "Ant colony system for a VRP with multiple time windows and multiple visits," *Journal of Interdisciplinary Mathematics,* vol. 10, no. 2, pp. 263-284, 2007.

[21] L. M. Gambardella, E. Taillard and G. Agazzi, "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows," in *New*

Ideas in Optimization, 1999.

[22] P. Frizzell and J. Giffin, "The Split Delivery Vehicle Scheduling Problem with Time Windows and Grid Network Distances," *Computers & Operations Research*, vol. 22, no. 6, pp. 655-667, 1995.

[23] I. Vacca and M. Salani, "The Vehicle Routing Problem with Discrete Split Delivery and Time Windows," Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 2009.

[24] G. G. Campos, H. T. Y. Yoshizaki and P. P. Belfiore, "Genetic algorithms and parallel computing for the vehicle routing problem with time windows and split deliveries," *Management and Production*, vol. 13, no. 2, 2006.

[25] C. Archetti and M. Speranza, "Vehicle routing problems with split deliveries," *International Transactions in Operational Research*, vol. 19, pp. 3-22, 2012.

[26] G. Laporte, "What You Should Know about the Vehicle Routing Problem," *Naval Research Logistics*, vol. 54, no. 8, pp. 811-819, 2007.

[27] B. M. Baker and M. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers and Operations Research*, vol. 30, pp. 787-800, 2003.

[28] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers and Operations Research*, vol. 31, no. 12, pp. 1985-2002, 2004.

[29] J. Brandao, "Metaheuristic for the Vehicle Routing Problem with Time Windows," in *Meta-Heuristics*, S. Voss, S. Martello, I. H. Osman and C. Roucairol, Eds., Springer US, 1999, pp. 19-36.

[30] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of Operations Research*, vol. 41, no. 4, pp. 421-452, 1993.

[31] O. Braysy and M. Gendreau, "Vehicle Routing Problem, Part I: Route Construction and Local Search Algorithms," *Transportation Science*, vol. 39, no. 1, pp. 104-118, 2005.

[32] I. Or, "Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking," Northwestern University, Evanston, IL, 1976.

[33] J.-Y. Potvin and J.-M. Rousseau, "An Exchange Heuristic for Routeing Problems with Time Windows," *Journal of the Operational Research Society,* vol. 46, no. 12, pp. 1433-1446, 1995.

[34] F. Glover, "New ejection chain and alternating path methods for traveling salesman problems," *Computer Science and Operations Research,* vol. 449, 1992.

[35] P. M. Thomspon and H. N. Psaraftis, "Cyclic Transfer Algorithms for Multivehicle Routing and Scheduling Problems," *Operations Research,* vol. 41, no. 5, pp. 935-946, 1993.

[36] P. Kilby, P. Prosser and P. Shaw, "Guided Local Search for the Vehicle Routing Problem," in *2nd International Conference on Metaheuristics*, Sophia-Antipolis, France, 1997.

[37] J. Kytojoki, T. Nuortio, O. Braysy and M. Gendreau, "An efficient variable neighborhood search heuristic for very large scale vehicle routing problems," *Computers & Operations Research,* vol. 34, pp. 2743-2757, 2007.

[38] M. Savelsbergh, "Local Search in Routing Problems with Time Windows," *Annals of Operations Research,* vol. 4, pp. 285-305, 1985/6.

[39] M. Savelsbergh, "An efficient implementation of local search algorithms for constrained routing problems," *European Journal of Operational Research,* vol. 47, pp. 75-85, 1990.

[40] M. W. Savelsbergh, "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration," *ORSA Journal on Computing,* vol. 4, no. 2, pp. 146-154, 1992.

[41] M. M. Solomon, E. K. Baker and J. R. Schaffer, "Vehicle Routing and Scheduling Problems with Time Window Constraints: Efficient Implementations of Solution Improvement Procedures," in *Vehicle Routing: Methods and Studies,* B. Golden and A. Assad, Eds., North-Holland, Elsevier, 1988, pp. 85-105.

[42] A. Van Breedam, "An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a selection of problems with Vehicle-related, Customer-related, and Time-related Constraints," University of Antwerp, Antwerp, Belgium, 1994.

[43] M. Gendreau, A. Hertz and G. Laporte, "New Insertion and Postoptimization Procedures for the Traveling Salesman Problem," *Operations Research,* vol. 40, no. 6, pp. 1086-1094, 1992.

[44] E. Taillard, P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin, "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows," *Transportation Science,* vol. 31, no. 2, pp. 170-186, 1997.

[45] O. Braysy, "A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows," *INFORMS Journal on Computing,* vol. 15, no. 4, pp. 347-368, 2003.

[46] O. Braysy, G. Hasle and W. Dullaert, "A multi-start local search algorithm for the vehicle routing problem with time windows," *European Journal of Operational Research,* vol. 159, pp. 586-605, 2004.

[47] R. Bent and P. Van Hentenryck, "A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows," *Transportation Science,* vol. 38, no. 4, pp. 515-530, 2004.

[48] O. Braysy, "Fast Local Searches for the Vehicle Routing Problem with Time Windows," *INFOR: Information Systems and Operational Research,* vol. 40, pp. 319-330, 2002.

[49] H. Hashimoto, M. Yagiura and T. Ibaraki, "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows," *Discrete Optimization,* vol. 5, pp. 434-456, 2008.

[50] T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno and M. Yagiura, "Effective Local Search Algorithms for Routing and Scheduling Problems with General Time-Window Constraints," *Transportation Science,* vol. 39, no. 2, pp. 206-232, 2005.

[51] F.-H. F. Liu and S.-Y. Shen, "A route-neighborhood-based metaheuristic for vehicle routing problem with time windows," *European Journal of Operational*

Research, vol. 118, pp. 485-504, 1999.

[52] R. E. Aleman, "A Guided Neighborhood Search Applied to the Split Delivery Vehicle Routing Problem," Wright State University, Dayton, OH, 2009.

[53] U. Derigs, B. Li and U. Vogel, "Local search-based metaheuristics for the split delivery vehicle routing problem," *Journal of the Operational Research Society,* vol. 61, pp. 1356-1364, 2010.

[54] C. Archetti, M. Savelsbergh and M. Speranza, "An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem," *Transportation Science,* vol. 42, no. 1, pp. 22-31, 2008.

[55] M. Dorigo, "Optimization, Learning, and Natural Algorithms," Politecnico di Milano, Milan, Italy, 1992.

[56] M. Dorigo and T. Stutzle, "Ant Colony Optimization: Overview and Recent Advances," in *Handbook of Metaheuristics,* Springer US, 2010, pp. 227-263.

[57] T. Stutzle and H. Hoos, "Improving the Ant System: A Detailed Report on the MAX-MIN Ant System," Technical University of Darmstadt, Darmstadt, Germany, 1996.

[58] T. Stutzle and H. H. Hoos, "MAX-MIN Ant System," *Future Generation Computer Systems,* vol. 16, pp. 889-914, 2000.

[59] B. Yu, Z.-Z. Yang and B. Yao, "An improved ant colony optimization for vehicle routing problem," *European Journal of Operations Research,* vol. 196, pp. 171-176, 2009.

[60] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation,* vol. 1, no. 1, pp. 53-66, 1997.

[61] W. J. Gutjahr, "ACO Algorithms with Guaranteed Convergence to the Optimal Solution," *Information processing letters,* vol. 82, no. 3, pp. 145-153, 2002.

[62] T. Stutzle and H. Hoos, "MAX-MIN ant system and local search for the traveling salesman problem," in *IEEE International Conference on Evolutionary*

Computation, 1997.

[63] T. Stutzle, "Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications," Technical University of Darmstadt, Darmstadt, Germany, 1998.

[64] T. Stutzle and M. Dorigo, "A Short Convergence Proof for a Class of Ant Colony Optimization Algorithms," *IEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 358-365, 2002.

[65] P. Pelligrini, D. Favaretto and E. Moretti, "On MAX-MIN Ant System's Parameters," in *Ant Colony and Swarm Intelligence*, M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli and T. Stutzle, Eds., Heidelberg, Springer, 2006, pp. 203-214.

[66] E. Ridge and D. Kudenko, "Tuning the Performance of the MMAS Heuristic," in *Engineering Stochastic Local Search Algorithms: Designing, Implementing and Analyzing Effective Heuristics*, T. Stutzle, M. Birattari and H. H. Hoos, Eds., Heidelberg, Germany, Springer, 2007, pp. 46-60.

[67] B. Bullnheimer, R. F. Hartl and C. Strauss, "An improved Ant System algorithm for the Vehicle Routing Problem," *Annals of Operations Research*, vol. 89, pp. 319-328, 1999.

[68] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, pp. 41-48, 2004.

[69] M. Reimann, K. Doerner and R. F. Hartl, "D-Ants: Savings Based Ants divide and conquer the vehicle routing problem," *Computers & Operations Research*, vol. 31, pp. 563-591, 2004.

[70] S. Mazzeo and I. Loiseau, "An Ant Colony Algorithm for the Capacitated Vehicle Routing," *Electronic Notes in Discrete Mathematics*, vol. 18, pp. 181-186, 2004.

[71] X. Zhang and J.-q. Wang, "Hybrid Ant Algorithm and Applications for Vehicle Routing Problem," *Physics Procedia*, vol. 25, pp. 1892-1899, 2012.

[72] G.-s. Wang and Y.-x. Yu, "An improved ant colony algorithm for VRP problem," in *Third International Symposium on Intelligent Information Technology and*

Security Informatics, Jinggangshan, 2010.

[73] L. Xia, Y. Chao and L. Xia, "Optimization of Vehicle Routing Problem Based on Max-Min Ant System with Parameter Adaptation," in *Seventh International Conference on Computational Intelligence and Security*, Hainan, 2011.

[74] Q. Ding, X. Hu, L. Sun and Y. Wang, "An improved ant colony optimization and its application to vehicle routing problem with time windows," *Neurocomputing,* vol. 98, pp. 101-107, 2012.

[75] S. Sodsoon and P. Changyom, "Max-Min Ant System (MMAS) for Vehicle Routing Problem with Time Windows," *KKU Engineering Journal,* vol. 38, no. 3, pp. 313-323, 2011.

[76] B. Yu, Z. Yang and B. Yao, "A hybrid algorithm for vehicle routing problem with time windows," *Expert Systems with Applications,* vol. 38, pp. 435-441, 2011.

[77] D. Feillet, P. Dejax, M. Gendreau and C. Gueguen, "Vehicle Routing with Time Windows and Split Deliveries," Laboratoire d'Informatique d'Avignon, Universite d'Avignon, Avignon, France, 2002.

[78] R. E. Aleman, X. Zhang and R. R. Hill, "An adaptive memory algorithm for the split delivery routing problem," *Journal of Heuristics,* vol. 16, no. 3, pp. 441-473, 2010.

[79] M. M. Solomon, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Operations Research,* vol. 35, no. 2, pp. 254-265, 1987.

[80] M. Boudia, C. Prins and M. Reghioui, "An effective memetic algorithm with population management for the split delivery vehicle routing problem," in *Hybrid Metaheuristics, Lecture Notes in Computer Science 4771*, T. Bartz-Beielstein, M. J. B. Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph and M. Sampels, Eds., Berlin, Springer, 2007, pp. 16-30.

[81] E. Mota, V. Campos and A. Corberan, "A new metaheuristic for the vehicle routing problem with split demands," in *Lecture Notes in Computer Science 4446*, Berlin, Springer, 2007, pp. 121-129.

[82] J. H. I. Wilck and T. M. Cavalier, "A construction heuristic for the split delivery vehicle routing problem," *American Journal of Operations Research,* no. 2, pp. 153-162, 2012.

[83] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, Second ed., New York: Wiley-Interscience, 2001, pp. 550-556.

[84] MathWorks, "Hierarchical Clustering," [Online]. Available: http://www.mathworks.com/help/stats/hierarchical-clustering.html. [Accessed 21 November 2013].

[85] C. W. Dunnett, "A Multiple Comparison Procedure for Comparing Several Treatments with a Control," *Journal of the American Statistical Association,* vol. 50, no. 272, pp. 1096-1121, 1955.

[86] M. H. Tongarlak, B. E. Ankenman and B. L. Nelson, "Relative Error Stochastic Kriging," in *Winter Simulation Conference*, 2011.

[87] M. G. Resende and C. C. Ribeiro, "Greedy Randomized Adaptive Search Procedures," AT&T Labs Research Technical Report, 2001.

[88] A. E. Rizzoli, F. Oliverio, R. Montemanni and L. M. Gambardella, "Ant Colony Optimisation for vehicle routing problems: from theory to applications," *Galleria Rassegna Bimestrale Di Cultura,* vol. 9, no. 1, pp. 1-50, 2004.

[89] S. R. Balsiero, I. Loiseau and J. Ramonet, "An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows," *Computers & Operations Research,* vol. 38, pp. 954-966, 2011.

[90] B. Yu and Z. Z. Yang, "An ant colony optimization model: The period vehicle routing problem with time windows," *Transportation Research Part E,* vol. 47, pp. 166-181, 2011.

[91] L. C. Coelho, J.-F. Cordeau and G. Laporte, "Thirty Years of Inventory Routing," Interuniversity Research Centre on Enterprise Networks, Logistics, and Transportation, Montreal, 2012.

[92] Q. Mu, Z. Fu, J. Lysgaard and R. Eglese, "Disruption management of the vehicle routing problem with vehicle breakdown," *Journal of the Operational Research Society,* vol. 62, pp. 742-749, 2011.

[93] A. J. Kleywegt, V. S. Nori and M. W. P. Savelsbergh, "Dynamic Programming Approximations for a Stochastic Inventory Routing Problem," *Transportation Science,* vol. 38, no. 1, pp. 42-70, 2004.

[94] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik,* vol. 1, no. 1, pp. 269-271, 1959.

REPORT DOCUMENTATION PAGE		*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE (DD–MM–YYYY) 18-09-2014	2. REPORT TYPE Dissertation	3. DATES COVERED *(From — To)* September 2011 – September 2014
4. TITLE AND SUBTITLE Exploring Heuristics for the Vehicle Routing Problem with Split Deliveries and Time Windows		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) McNabb, Marcus E., Maj, USAF		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENS-DS-14-S-19
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for Public Release; Distribution Unlimited.

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
This dissertation investigates the Vehicle Routing Problem with Split Deliveries and Time Windows. This problem assumes a depot of homogeneous vehicles and set of customers with deterministic demands requiring delivery. Split deliveries allow multiple visits to a customer and time windows restrict the time during which a delivery can be made. Several construction and local search heuristics are tested to determine their relative usefulness in generating solutions for this problem. This research shows a particular subset of the local search operators is particularly influential on solution quality and run time. Conversely, the construction heuristics tested do not significantly impact either. Several problem features are also investigated to determine their impact. Of the features explored, the ratio of customer demand to vehicle ratio revealed a significant impact on solution quality and influence on the effectiveness of the heuristics tested. Finally, this research introduces an ant colony metaheuristic coupled with a local search heuristic embedded within a dynamic program seeking to solve a Military Inventory Routing Problem with multiple-customer routes, stochastic supply, and deterministic demand. Also proposed is a suite of test problems for the Military Inventory Routing Problem.

15. SUBJECT TERMS
Heuristics, Ant Colony Optimization, Local Search/Improvement, Vehicle Routing Problem

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dr. Jeffery D. Weir, AFIT/ENS
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	143	19b. TELEPHONE NUMBER (Include Area Code) (937) 255-3636 x4523 jeffery.weir@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18