

ABSTRACT

Title of document: SOLVING THE INVENTORY SLACK
ROUTING PROBLEM FOR EMERGENCY
MEDICATION DISTRIBUTION

Adam Montjoy, Master of Science, 2010

Directed by: Associate Professor Jeffrey Herrmann
Department of Mechanical Engineering and
Institute for Systems Research

A bioterrorist attack, or natural disaster, would prompt an immediate government response in order to efficiently address the possible health effects of the population. Such a scenario would create a logistics problem of delivering medication (or other supplies) to makeshift dispensing centers in a short period of time and in high quantities while operating. These makeshift centers, or Points of Dispensing, require schedules of delivery that are robust against uncertainty. This inventory slack routing problem is a novel vehicle routing problem. The objective function is to maximize the slack in the schedule.

This thesis presents heuristic approaches that separate the problem into routing and scheduling. The routing problem is solved using a route first-cluster second method. The scheduling problem is solved using a heuristic and an improvement approach.

This thesis also presents a search approach that uses heuristics to search various neighborhoods in the solution space. These heuristics are chosen randomly based on probabilities that adapt during the search according to their performance.

The inventory slack routing problem is also formulated as a mixed-integer program and solved using a column generation procedure that utilizes simulated annealing to generate new vehicle schedules.

This thesis presents the results of testing these three approaches on a set of 432 instances that were generated from real-world data to evaluate solution quality and computational effort. The search approach outperformed the heuristic approach with a reasonable amount of computational effort. The column generation approach did not generate desirable vehicle schedules and therefore was not productive in solving the problem.

SOLVING THE INVENTORY SLACK ROUTING PROBLEM FOR
EMERGENCY MEDICATION DISTRIBUTION

by

Adam Montjoy

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2010

Advisory Committee:

Associate Professor Jeffrey W. Herrmann, Chair/Advisor

Associate Professor Linda C. Schmidt

Professor Michael C. Fu

© Copyright by
Adam Montjoy
2010

Dedication

To my parents and sisters.

Acknowledgments

First and foremost, I have many thanks for my advisor Dr. Jeffrey Herrmann. He has always been supportive, helpful, and with an infinite capacity of patience. I have much gratitude for Dr. Linda Schmidt and Dr. Michael Fu for serving on my committee and teaching great courses that I had the privilege to take. Also, Dr. Zhi-Long Chen's help with integer programming was very valuable and appreciated.

I would like to personally thank the folks at Montgomery County's Advanced Practice Center who have supported and provided much assistance with this project.

I was fortunate to share a workspace with a great group of students. Many thanks to Atul Thakur who has, on more than one occasion, taken time from his research to help me with mine.

In addition to dedication of this thesis, much acknowledgement is needed for family who helped me in all ways non-academic during this process.

Table of Contents

List of Tables	vi
List of Figures	vii
List of Abbreviations	viii
1 Introduction	1
1.1 Background	1
1.2 ISRP Problem Formulation	5
1.2.1 Notation	5
1.2.2 Formulation	6
2 Heuristic Approach	10
2.1 Routing	10
2.1.1 Route First	12
2.1.1.1 Nearest Neighbor	12
2.1.1.2 2-opt Exchange	12
2.1.2 Cluster Second	13
2.1.2.1 Improvement by Route Duration	13
2.1.2.2 Improvement by Total Demand	14
2.2 Scheduling	14
2.3 Improvement	16
3 Adaptive Large Neighborhood Search Approach	20
3.1 Removal Heuristics	20
3.1.1 Random Removal	20
3.1.2 Worst Removal	20
3.1.3 Related Removal	22
3.1.4 Longest Travel Time Removal	22
3.2 Insertion Heuristics	23
3.2.1 Random Insertion	23
3.2.2 Best Position of Vehicle with Lowest Travel Time Insertion	23
3.2.3 Best Position of Vehicle with Lowest Total Demand Insertion	23
3.2.4 Nearest Neighbor Insertion	24
3.3 Scheduling and Calculating Slack	24
3.4 Selecting Heuristics and Accepting Solutions	24
3.5 Parameter Selection	25
4 Column Generation with Simulated Annealing	26
4.1 MIP Formulation	26
4.2 Solution Approach	29
4.3 Solving the Subproblem	30
4.3.1 Scheduling Times and Quantities	30
4.4 Implementation	31
4.4.1 Formulation Changes	31
4.4.2 Parameter Selection	32

5	Computational Analysis	33
5.1	Creating Instances for Testing	33
5.2	Upper Bound	34
5.3	Heuristic Testing	35
5.3.1	Heuristic Computational Effort	35
5.3.2	Heuristic Performance	35
5.3.3	DVI Performance	36
5.3.4	Upper Bound Analysis	39
5.4	ALNS Testing	40
5.5	CG Testing	46
6	Conclusions	53
A	Example Instance	55
B	Pseudocode	56
C	ALNS Results	59
	Bibliography	76

List of Tables

1.1	Schedule for example.	9
1.2	Slack calculations for example.	9
2.1	Schedule for example with DVI.	19
2.2	Slack calculations for example with DVI.	19
5.1	Summary of Baseline Instances	34
5.2	Quality of solutions generated by each heuristic on each problem set.	37
5.3	Number of instances in each problem set that the nearest neighbor and 2-opt routing procedures generated the better solution.	38
5.4	Number of instances in which minimum slack does not occur on first wave.	39
5.5	Average improvement for each heuristic.	40
5.6	Quality of solutions generated by each heuristic with DVI on each problem set.	41
5.7	Average difference between upper bound and minimum slack for different sets of instances classified by number of vehicles and travel times.	42
5.8	Average solutions by baseline of best performing heuristic and ALNS with varying parameters.	43
5.9	Results for the CG procedure on the baseline instances.	47
5.10	CG attempts for R1.	48
5.11	CG attempts for R3.	48
5.12	Results for the CG procedure on the baseline instances with poorer starting columns.	49
5.13	CG attempts for R1 with poorer starting columns.	50
5.14	Results for the CG procedure on the baseline instances with 20 minute intervals.	51
5.15	CG attempts for R1 with 20 minute intervals.	52
A.1	Deliveries to the depot.	55
C.1	ALNS test for instance set R1.	60
C.2	ALNS test for instance set F1.	61
C.3	ALNS test for instance set C1.	62
C.4	ALNS test for instance set M1.	63
C.5	ALNS test for instance set R3.	64
C.6	ALNS test for instance set M2.	65
C.7	ALNS test for instance set C2.	66
C.8	ALNS test for instance set M3.	67
C.9	ALNS test for instance set V1.	68
C.10	ALNS test for instance set V2.	69
C.11	ALNS test for instance set V3.	70
C.12	ALNS test for instance set V4.	71
C.13	ALNS test for instance set V5.	72
C.14	ALNS test for instance set V11.	73
C.15	ALNS test for instance set V12.	74
C.16	ALNS test for instance set M4.	75

List of Figures

1.1	Cumulative delivery function from R1	7
2.1	Basic approach to obtaining a solution.	11
3.1	ALNS procedure.	21
4.1	CG Procedure.	28
5.1	Computational time for finding initial routes.	36
5.2	ALNS procedure with 2,000 iterations.	44
5.3	Average computational time in seconds per iteration for one complete run of the ALNS with 500 iterations.	45

List of Abbreviations

ALNS	Adaptive Large Neighborhood Search
CG	Column Generation
DVI	Delivery Volume Improvement
GUB	Generalized Upper Bound
IRP	Inventory Routing Problem
ISRP	Inventory Slack Routing Problem
LP	Linear Program
MIP	Mixed Integer Program
NN	Nearest Neighbor
NP	Non-Deterministic Polynomial-Time
RSS	Receipt, Storage, and Stage
VRP	Vehicle Routing Problem
PODs	Points of Dispensing

Chapter 1

Introduction

1.1 Background

Events in the last ten years have highlighted the increased need for emergency preparedness by government officials. Events such as the terrorist attacks on September 11, 2001, Hurricane Katrina, the 2008 earthquake in Chengdu, China, and the 2010 earthquake in Haiti have provided real world examples of communities that were ill-prepared for major disasters [7]. Thus, it is important for government officials to anticipate disasters and plan accordingly. Mathematical models and decision support tools can be used to support planning activities.

Some scenarios could require the quick and efficient distribution of medication to a large number of people. For instance, the widespread release of anthrax without government response in a metropolitan area could result in casualties equivalent to that of a small nuclear explosion [21]. In this scenario (and others involving mass vaccination against communicable diseases such as smallpox and influenza), it is logical to create Points of Dispensing (PODs) such that large populations can be given medication without having to travel to one central location. PODs may be setup in schools, recreation centers, churches, and other non-medical facilities. The medication to be distributed at these PODs must be delivered quickly from a central depot as soon as it arrives.

The proposed research is motivated by work with public health officials in the state of Maryland who must plan the logistics for distributing medication to the PODs from a central location. We consider the problem at the state and local levels (not the national

level). After the decision for mass dispensing is made, county public health departments will begin preparing to open multiple PODs simultaneously at a designated time. The state will request medication from the federal government, who will deliver an initial but limited supply of medication to a state receipt, storage, and stage (RSS) facility (which we call the “depot”). Contractors will deliver more medication to the depot, but the state will begin shipping medication from the depot to the PODs before all inventory arrives from the contractors. The deliveries to the depot arrive in batches that we call “waves.” Time will be needed to prepare the PODs. This will delay the opening of PODs for distribution until after the first wave has been delivered to the depot.

Poor medication distribution plans will delay the time that some PODs receive medication. This can delay the opening of these PODs, and some residents may not get their medication in a timely manner, which increases their risk of death or illness. Clearly, there are many uncertainties in medication distribution, including the timing of shipments to the depot, the time needed to load and unload vehicles, travel times, and the demand for medication at each POD. For this reason, planners need a robust plan. In particular, it is better if the plan calls for delivering medication to PODs much earlier than it is needed. This improves the likelihood that the PODs will open on-time, will not run out of medication during operations, and will dispense medication to the largest number of people in a timely manner.

Specifically, the problem addressed has some features of the inventory routing problem but also has some unique assumptions, constraints, and objectives. In this case, a set of PODs are served by a given set of vehicles delivering a quantity of one item during a short time span. Thus, the objective is not to minimize the cost or maximize the profit. Instead, the objective is to increase the time between a POD running out of supplies for

each delivery made from the depot. This value will be known as the slack.

Much research has been done to develop models to improve emergency preparedness planning. Hupert *et al.* [21] have presented a model to predict the hospital surge after a large-scale anthrax attack. The researchers emphasize the importance of timely antibiotic distribution, making logistics of delivery equally important. Similarly, much work has been done to create simulation methods and planning tools for PODS in makeshift locations such as school gymnasiums [1, 2, 22].

The operations of firefighters, emergency medical services, and police departments have motivated research into location models [4, 10, 14] and dynamic vehicle routing models [18, 25, 29]. However, these models are not relevant to the medication distribution problem, which is more closely related to the inventory routing problem [3, 8, 15, 24] and the production-distribution scheduling problem [11]. Still, the models used for those problem are also not directly relevant.

Planning humanitarian logistics is related to the Vehicle Routing Problem (VRP) and Inventory Routing Problem (IRP). These problems have been applied to a variety of commercial, military, and government applications. The following description of the VRP is by Toth and Vigo [27].

The VRP details the delivery of a set of goods to a set of customers by a set of vehicles. These goods are stored at a depot, or a set of depots, and are delivered by a road network. This road network is usually detailed using a graph with arcs representing roads and vertices as the sites and depots. The solution to the VRP specifies a route for each vehicle that begins and ends at the depot. Typical VRP problems have the following characteristics: customer locations, demands for the customers, time windows for the customers, loading/unloading times, and a set of available vehicles that can be used.

In many cases, it may not be possible to fully satisfy all of the customer demand, and priorities or penalty functions must be employed. With this, it is possible to formulate various objective functions to obtain a solution, including minimization of global transportation cost, minimization of vehicles used, balancing routes for travel times and load, and minimization of penalties. The VRP is a well-researched problem with many heuristics, mathematical programming, and search techniques available.

The VRP has many variations including the Inventory Routing Problem (IRP). The following description of the IRP is by Campbell *et al.* [9]. The IRP differs from the VRP because the the delivery company decides when and what quantity to deliver to customers, as long as they do not run out. The objective is minimization of cost over the planning horizon while preventing customers from running out of product. A single product is delivered from a single depot to a set of n customers over a specified time period. These customers are served by a homogenous fleet of V vehicles with a capacity of Q . A problem solution should answer three questions: when to serve a customer, how much to deliver, and which routes to follow?

Most solutions detailed in literature focus on short-term scenarios solved by mathematical programming techniques. There is a lack of basic heuristics for solving IRPs. The Inventory Slack Routing Problem (ISRP) that we present is similar to an IRP but has some unique assumptions. The main concern is to supply medication as quickly as possible, not to minimize cost. As Hupert *et al.* [21] emphasize, delaying the start of POD operations will significantly increase the number of people hospitalized. In addition, the limited availability of medication at the depot adds an additional constraint to the problem. Finally, because there is uncertainty in loading/unloading, travel times, and demand, it is necessary to have overall maximum slack to hedge against these uncertainties.

The objective of the ISRP is to maximize the minimum slack in order to develop a more robust plan.

1.2 ISRP Problem Formulation

In the ISRP, a set of vehicles must deliver material from a depot to a set of sites that will consume this material. Not all of the material is available at the depot at the beginning of the time frame. Instead, material will become available in waves, which are deliveries to the depot at different points in time. The sites will start operating at a designated time. Each site consumes material at a given rate, and this demand may vary from site to site. We consider deterministic opening times and dispensing rates. The vehicles must deliver enough material from the depot to the sites to satisfy the total demand over the time horizon. The following section details the notation to be used. Note that an example is provided in Appendix A.

Although, in theory, a vehicle could follow a different route each time it leaves the depot, and a site could be served by multiple vehicles, this makes supervising and performing the deliveries more complex in practice. We therefore assume that each and every site is assigned to exactly one vehicle, and each vehicle always follows the same route to visit the sites assigned to it.

1.2.1 Notation

t - Time in minutes

T_1 - Time, in minutes, that sites will begin dispensing

T_2 - Time, in minutes, that sites will end dispensing

$I(t)$ - Cumulative amount of material delivered to the depot between time 0 and t

V - Number of vehicles

C - Vehicle capacity in units of material

σ_v - Route assigned to vehicle v , $v = 1, \dots, V$

σ - Routes for all vehicles

n - Number of sites

L_k - Demand in units per minute for sites $k = 1, \dots, n$

p_k - Load (unload) time, in minutes, at sites $k = 1, \dots, n + 1$

c_{ij} - Time, in minutes, to travel from site i to j

r_v - Number of trips vehicle v makes

t_{vj} - Time of trip j for vehicle v

q_{vjk} - Amount delivered to site k by vehicle v on trip j

y_v - Total duration of a trip by vehicle v

1.2.2 Formulation

In the ISRP, $t = 0$ refers to the first instant that material is available at the depot, $t = T_1$ is the time that the sites begin operating, and $t = T_2$ is the time that the sites stop operating. There are n sites denoted by $k = 1, \dots, n$. The demand rate for sites is denoted as L_k material per time unit, which in this paper is minutes. Thus, site k has a total demand of $(T_2 - T_1)L_k$ units of material.

The depot, denoted by $k = n + 1$, receives material in multiple “waves” that arrive at different times. The times and quantities are known in advance and are used to determine the discontinuous, non-decreasing cumulative function $I(t)$. In our example, there are three waves. At $t = 0$, 48,000 units are delivered; at $t = 180$, 98,000 units are delivered; and at $t = 360$, 73,000 units are delivered. Figure 1.1 shows $I(t)$.

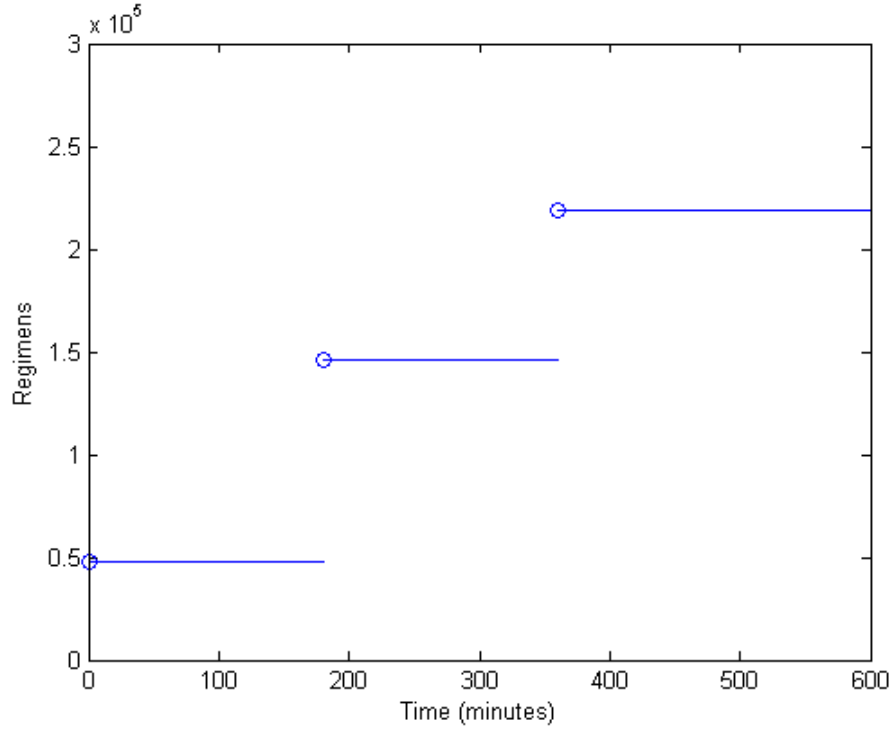


Figure 1.1: Cumulative delivery function from R1

The time to load or unload a vehicle at site k is given by p_k . The time to travel from site i to site j is c_{ij} . An instance will have V vehicles at the depot where vehicle v has a capacity of C units.

A solution specifies, for each vehicle, a route, the number of trips that it makes, the time to start each trip, and the quantity to deliver to each site on each trip. The following constraints must be satisfied for a solution to be feasible.

The quantity shipped from the depot cannot exceed the amount delivered to the depot:

$$\sum_{(a,b):t_{ab}\leq t_{vj}} \sum_{k\in\sigma_a} q_{abk} \leq I(t_{vj}) \quad v = 1, \dots, V; j = 1, \dots, r_v \quad (1.1)$$

A vehicle cannot begin a new route until it returns to the depot:

$$t_{vj} \geq t_{v,j-1} + y_v \quad v = 1, \dots, V; j = 2, \dots, r_v \quad (1.2)$$

All delivery quantities are non-negative. Each vehicle has a fixed capacity and can carry a maximum of C units, that is $\sum_{k \in \sigma_v} q_{vjk} \leq C$ for all $v = 1, \dots, V$ and $j = 1, \dots, r_v$. All route start times are non-negative such that $t_{vj} \geq 0$ for all $v = 1, \dots, V$ and $j = 1, \dots, r_v$. Each site must receive all required medication, that is $\sum_{j=1}^{r_v} q_{vjk} = (T_2 - T_1)L_k$ for $v = 1, \dots, V$ and $k \in \sigma_v$.

A feasible solution for our example is shown in Table 1.1. To evaluate a solution, we need to calculate its minimum slack. Let w_{vk} be the duration until vehicle v visits site k after it begins a trip. This is calculated as follows, where $[a]$ is the a -th site in route σ_v :

$$w_{vk} = p_{n+1} + c_{n+1,[1]} + p_{[1]} + c_{[1],[2]} + \dots + p_k \quad (1.3)$$

For a site $k \in \sigma_v$, let Q_{vjk} be the quantity delivered to site k by vehicle v on trips before trip j :

$$Q_{vjk} = \sum_{i=1}^{j-1} q_{vik} \quad (1.4)$$

Note that $Q_{v1k} = 0$. If, on trip j , the vehicle's delivery at site k were delayed, then the site would run out of inventory at time $T_1 + Q_{vjk}/L_k$.

The slack for site k on trip j can be found as follows:

$$s_{vjk} = T_1 + \frac{Q_{vjk}}{L_k} - (t_{vj} + w_{vk}) \quad (1.5)$$

The evaluation of a solution is the minimum slack over all vehicles, sites, and trips:

Table 1.1: Schedule for example.
Schedule for Vehicle 1

Site/Time	0	180	360
5	10,521	21,479	16,000
4	7,890	16,110	12,000

Schedule for Vehicle 2

Site/Time	0	180	360
3	13,151	26,849	20,000

Schedule for Vehicle 3

Site/Time	0	180	360
2	9,863	20,137	15,000
1	6,575	13,425	10,000

Table 1.2: Slack calculations for example.
Slack

Vehicle v	Site k	w_{jk}	Slack		
			s_{v1k}	s_{v2k}	s_{v3k}
1	5	45	555	507	595
	4	73	527	479	567
2	3	46	540	492	580
3	2	45	555	507	595
	1	74	526	478	566

$S = \min\{s_{vjk}\}$. Slack values for the example are given in Table 1.2.

Chapter 2

Heuristic Approach

The ISRP, like other versions of the VRP and IRP, is NP-hard, which makes it computationally expensive to obtain an exact solution. Therefore, it is of benefit to develop simple heuristics that can construct good feasible solutions. A solution is a schedule for each vehicle with a starting time to begin loading for each trip, specified sites to visit, and a quantity to bring to each site on that trip.

The overall approach can be seen in Figure 2.1. This approach constructs a solution by separating the ISRP into two subproblems: routing and scheduling. A combination of different routing techniques will be discussed in the following section. The scheduling subproblem is further separated into scheduling for each vehicle by using the routes that have been found.

2.1 Routing

The routing subproblem creates routes for each vehicle by assigning sites to each vehicle and determining the order in which they are visited. The ISRP differs from traditional VRP because the objective is not to minimize total travel time. Instead, it is desirable to create routes that are nearly the same duration so that the minimum slack is not too small.

Bramel and Simchi-Levi [6] present two categories for this type of routing: (1) route first-cluster second methods and (2) cluster first-route second methods. With a route first-cluster second method, a tour is created through all of the sites, and then the sites (and the route) are divided into a desired number of partitions. Gillett and Miller's [17] sweep algorithm is a popular example of the route first-cluster second approach [6]. One

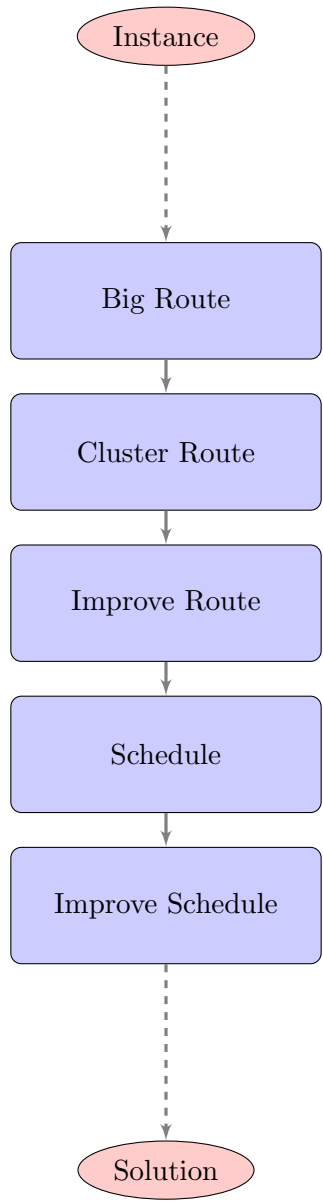


Figure 2.1: Basic approach to obtaining a solution.

major drawback for these methods is that vehicles may be poorly utilized since the routing is done first. Algorithms have also been developed for cluster first-route second methods. These methods devote more priority to the clustering phase. Because these methods tend to require more computational effort and we are interested in heuristic approaches at this time, we will consider a route first-cluster second method.

2.1.1 Route First

When routing, it is first necessary to create a “big route” that visits all of the sites. We consider two different methods that do not use X-Y coordinates. In many real world situations, the X-Y coordinates are not as important as the travel times between sites, and, in some situations, the X-Y coordinates may be unavailable.

2.1.1.1 Nearest Neighbor

The nearest neighbor (NN) technique generates a tour through all of the sites. The tour starts at the depot. The next site selected for the tour is the site that has the shortest travel time from the current site and has not already been visited. This is repeated until no sites remain. Once all of the sites have been visited, the tour ends with the depot.

2.1.1.2 2-opt Exchange

Given an initial tour, the 2-opt exchange systematically removes two edges in the tour and reconnects the vertices to obtain a tour of shorter length. This algorithm finds all pairs of edges that will decrease the tour length. Of all these pairs, the pair chosen is the one that will make the greatest decrease in travel time of the tour. The 2-opt algorithm is continued until no more improving pairs can be found [23]. Although many pairwise, or 2-opt, exchange implementations start with a randomly generated trip through the sites, we use the nearest neighbor algorithm to first generate a route because it is computationally inexpensive.

2.1.2 Cluster Second

Once a big route has been obtained, it is necessary to divide the sites among all of the vehicles available. The sites are first divided between vehicles as equally as possible.

In the example, there are five sites, three vehicles, and the big route using NN is found to be 6, 5, 4, 3, 2, 1, 6. The initial clusters will be as follows:

Vehicle 1: 6, 5, 4, 6

Vehicle 2: 6, 3, 2, 6

Vehicle 3: 6, 1, 6

It is important to note that this initial cluster ignores both the demand and the travel times. Thus, the durations (and demands) of the routes may vary widely, which can reduce the slack of any solution constructed from these clusters. Thus, we use an improvement algorithm to reduce the variation. We tested an improvement algorithm that considers the travel time and one that considers the total demand.

2.1.2.1 Improvement by Route Duration

Each cluster is assigned to a vehicle. The vehicles are sequenced by the position of their cluster in the big route. This improvement algorithm method strives to make the route durations as similar as possible by minimizing the range of route durations. This method begins by calculating each vehicle's route duration. In each iteration, the algorithm examines the vehicles with maximum and minimum travel times and considers moving sites at the beginning (or end) of one route to the previous (or next) vehicle's route. If the potential move decreases the range of route durations, then the routes are updated to reflect this change. This continues until no further improvement can be made.

Of course, this type of local search may not find the smallest possible range. The pseudocode can be seen in Appendix B. This pseudocode, and formulation in subsequent sections, require the following notation.

y_v is the route duration for vehicle v , which can be calculated by the following equation, where r_v denotes the number of sites on route σ_v and $[a]$ denotes the a -th site on route σ_v .

$$y_v = p_{n+1} + c_{n+1,[1]} + p_{[1]} + c_{[1],[2]} + \dots + p_{[r_v]} + c_{[r_v],n+1}$$

This cluster improvement algorithm will be applied to the example introduced in the previous section. The durations of the initial clusters are 90, 124, and 64 minutes. The initial range of durations is 60 minutes. The following clusters are the result, and the range of route durations is 1 minute.

Vehicle 1: 6, 5, 4, 6; route duration: 90 minutes

Vehicle 2: 6, 3, 6; route duration: 90 minutes

Vehicle 3: 6, 2, 1, 6; route duration: 91 minutes

2.1.2.2 Improvement by Total Demand

This improvement algorithm attempts to reduce the range of total demand of the sites on the routes but searches in the same way as the previous algorithm. That is, we replace y by D , where the total demand, D_v , can be calculated for vehicle v by the following equation.

$$D_v = (T_2 - T_1) \sum_{k \in \sigma_v} L_k$$

2.2 Scheduling

After constructing routes for the vehicles, it is necessary to schedule their deliveries. A schedule specifies the quantity to be delivered to each site as well as the time for the vehicle to begin loading for departure. Because we are interested in developing heuristics for the ISRP, we will allocate material to vehicles and schedule the deliveries of each vehicle using the following policies.

Let f_k be the relative demand of site k :

$$f_k = \frac{L_k}{\sum_{i=1}^n L_i}$$

For each vehicle, we create a cumulative material function $J_v(t)$ that describes the material available to be delivered by vehicle v at time t . Recall that $I(t)$ describes the total material received at the depot by time t .

Then, if σ_v is the route that vehicle v visits, $J_v(t) = I(t) \sum_{k \in \sigma_v} f_k$.

Once $J_v(t)$ has been established for vehicle v , it is then possible to determine how many trips the vehicle will take to service the sites assigned to it, what time each trip will start, and how much quantity to deliver each trip. Let y_v be the route duration for vehicle v to complete its route.

Vehicle v will begin loading for its first route as soon as the depot has stock. Vehicle v will carry as much material as it can at the first instance the depot has material, which is $J_v(0)$, without surpassing its capacity. When the vehicle returns at time y_v , if there is material still available for that vehicle, the vehicle will start loading at this time. Otherwise, the vehicle will wait until the next wave of deliveries to the depot. Once again, the vehicle will either carry all of the material allotted to it or the maximum capacity of the vehicle. This will continue until no more material is available for that vehicle. It is important to note that this method does not require the vehicle to be full to begin a trip. To do so would lead to vehicles sitting at the depot while material is available, which would reduce slack (unless the quantity available is small and the delay until the next wave is short).

The material on a vehicle will be divided between the sites that the vehicle visits using their respective proportions of the total demand.

The pseudocode for the scheduling algorithm is as follows. The following notation is used.

t_{vj} : The time, in minutes, at which vehicle v begins trip j

f_k : Relative demand of site k

g_k : Relative proportion of vehicle delivery allocated to site k

R_{vj} : Quantity delivered on trip j by vehicle v

q_{vjk} : Quantity delivered to site k assigned by v on trip j

C : Vehicle capacity (units of material)

schedule(σ)

1 **CALCULATE** f_1, \dots, f_n

2 **CALCULATE** y_1, \dots, y_V

```

3
4 FOR  $v = 1 : V$ 
5      $J_v(t) = I(t) \sum_{k \in \sigma_v} f_k$ 
6      $t_{v1} = 0$ 
7      $R_{v1} = \min\{J_v(0), C\}$ 
8
9      $\theta = 2$ 
10
11     REPEAT
12          $t_{v\theta} = t_{v,\theta-1} + y_v$ 
13
14         IF  $J_v(t_{v\theta}) - \sum_{j=1}^{\theta-1} R_{vj} == 0$ 
15              $t_{v,\theta} = \min t$  such that  $J_v(t) - \sum_{j=1}^{\theta-1} R_{vj} > 0$ 
16         END
17
18          $R_{v\theta} = \min\{C, J_v(t_{v\theta}) - \sum_{j=1}^{\theta-1} R_{vj}\}$ 
19
20          $\theta = \theta + 1$ 
21
22     UNTIL  $\sum_{j=1}^{\theta-1} R_{v,j} == (T_2 - T_1) \sum_{k \in \sigma_v} L_k$ 
23
24      $r_v = \theta - 1$ 
25     FOR  $j = 1 : r_v$ 
26         FOR  $k \in \sigma_v$ 
27              $g_k = \frac{f_k}{\sum_{i \in \sigma_v} f_i}$ 
28              $q_{vjk} = g_k R_{vj}$ 
29         END
30     END
31 END

```

This scheduling algorithm produces the schedules in Table 1.1 for our example. The first column denotes the sites visited by that vehicle and each remaining column denotes a trip taken by that vehicle. The first row denotes the starting time in minutes for each trip. The solution is evaluated in Table 1.2 with a minimum slack of 478 minutes.

2.3 Improvement

After a solution has been found, it may be possible to manipulate the quantities carried on each trip to increase the slack. Note that, as shown in Table 1.2, the slack at sites 2 and 1 in the second trip of Vehicle 3 are different. The slack at site 2 is larger than the slack at site 1 because Vehicle 3 visits that site before it visits site 1. If, in its first

trip, Vehicle 3 delivered more material to site 1 (and less to site 2), the slacks could be the same, which would increase the minimum slack.

Because the slack for a delivery depends upon the material delivered to that site on previous trips, the goal of the Delivery Volume Improvement (DVI) algorithm is to adjust the delivery quantities on one route in such a way that the slacks for all sites on the next route are the same. The algorithm starts by setting the delivery quantities for the first trip and then proceeds to the next trip. Note that the trip start times and site delivery times are given and not changed by this algorithm.

Consider a vehicle v making a delivery to site k at in second trip (so $j = 2$). Let D_{vjk} be the time that this delivery occurs. We would like the slack of every delivery on this trip to be equal to K , which determines the delivery quantity during the first trip:

$$K = T_1 + \frac{q_{v1k}}{L_k} - D_{vjk}$$

$$q_{v1k} = (K + D_{vjk} - T_1)L_k$$

We want to find the largest possible K that is feasible with respect to the total material that the vehicle delivers on that trip. Let R_{vj} be the total material that vehicle v delivers on trip j . This is given and is not changed by the algorithm. Because the total of the delivery quantities in the first trip must equal R_{v1} and the delivery times equal w_k , then we can determine K and the delivery quantities as follows:

$$K = T_1 + \frac{R_{v1}}{\sum_{i \in \sigma_v} L_i} - \frac{\sum_{i \in \sigma_v} w_i L_i}{\sum_{i \in \sigma_v} L_i}$$

$$q_{v1k} = \frac{L_k}{\sum_{i \in \sigma_v} L_i} (R_{v1} - \sum_{i \in \sigma_v} w_i L_i) + w_k L_k$$

For subsequent trips, it easy to show that letting the delivery quantities be proportional to the site demands will suffice. Of course, it is important not to deliver more that a site needs, which affects the delivery quantities of the last trips.

It is important to note that if the minimum slack occurs in the first trip for any vehicle, then the procedure will not be able to increase the minimum slack. However, the procedure may increase the slack for deliveries on subsequent trips.

In the DVI algorithm, let q' be the desired amount to deliver, and let DR_k be the remaining material needed at site k . Let s_v be the number of sites on route σ_v , and let $[i]$ be the i -th site on route σ_v .

```

DVI( $\sigma$ )
1 FOR  $v = 1 : V$ 
2   FOR  $k \in \sigma_v$ 
3      $DR_k = L_k(T_2 - T_1)$ 
4   END
5   FOR  $j = 1 : r_v$ 
6     FOR  $k = s_v, s_v - 1, \dots, 1$ 
7       IF  $R_{vj} < \sum_{i=1}^k DR_{[i]}$ 
8         IF  $j == 1$ 
9            $q' = \frac{L_{[k]}}{\sum_{i=1}^k L_{[i]}} (R_{vj} - \sum_{i=1}^k w_{[i]} L_{[i]}) + w_{[k]} L_{[k]}$ 
10          ELSE
11             $q' = \frac{L_{[k]}}{\sum_{i=1}^k L_{[i]}} R_{vj}$ 
12          END
13          IF  $q' < DR_{[k]}$ 
14             $q_{vj[k]} = q'$ 
15             $DR_{[k]} = DR_{[k]} - q'$ 
16          ELSE
17             $q_{vj[k]} = DR_{[k]}$ 
18             $DR_{[k]} = 0$ 
19          END
20           $R_{vj} = R_{vj} - q_{vj[k]}$ 
21        ELSE
22           $q_{vj[k]} = DR_{[k]}$ 
23           $DR_{[k]} = 0$ 
24           $R_{vj} = R_{vj} - q_{vj[k]}$ 
25        END
26      END
27    END
28  END
29 END

```

It is important to note that if there is not sufficient inventory to make the slacks of sites equal on the second trip, it is necessary to deliver zero quantity to sites at the beginning of the sequence and then performing the DVI procedure. This occurs when there is a site very close to the depot early in the sequence and sites very far away later in the sequence.

In the example previously introduced, the DVI algorithm updates the quantities and the slacks as shown in Tables 2.1 and 2.2. Note that, in the second and third trips, the slacks for sites 5 and 4 are the same and that the slacks for sites 2 and 1 are the same. (The agreement between vehicles 1 and 3 is a coincidence that reflects the similarity in

delivery times.) The new minimum slack is 492 minutes.

If the vehicle trips are coordinated (for example, if every vehicle makes one trip for each wave), then we can apply a different type of DVI algorithm to all of the vehicles simultaneously and shift material from one vehicle to another to make all of the slacks the same [20].

Because the approach in this paper schedules each vehicle separately, it may yield a solution in which the number of trips per vehicle varies. Therefore, the DVI algorithm used here considers only one vehicle at a time.

Table 2.1: Schedule for example with DVI.
Schedule for Vehicle 1

Site/Time	0	180	360
5	9,561	21,479	16,960
4	8,850	16,110	11,040

Schedule for Vehicle 2

Site/Time	0	180	360
3	13,151	26,849	20,000

Schedule for Vehicle 3

Site/Time	0	180	360
2	8,993	20,137	15,870
1	7,445	13,425	9,130

Table 2.2: Slack calculations for example with DVI.
Slack

Vehicle v	Site k	w_{jk}	Slack		
			s_{v1k}	s_{v2k}	s_{v3k}
1	5	45	555	495	583
	4	73	527	495	583
2	3	46	540	492	580
3	2	45	555	495	583
	1	74	526	495	583

Chapter 3

Adaptive Large Neighborhood Search Approach

This chapter introduces the approach of an Adaptive Large Neighborhood Search (ALNS). We will adapt the procedure used by Pisinger and Ropke [28] to solve vehicle routing problems to assign sites to vehicles and sequence those routes. We will then use the previously introduced scheduling algorithm to determine the trip start times and delivery quantities.

The ALNS begins with an initial solution and iteratively destroys and rebuilds the solution by randomly choosing and applying a number of quick heuristics which define neighborhoods of the search. Associated with each heuristic is a weight that determines its selection probability. With each iteration, the new solution is either accepted or rejected and the heuristic weights are updated according to their performance.

3.1 Removal Heuristics

Our ALNS uses four removal heuristics that are appropriate for the ISRP and have the ability to diversify the search. These removal heuristics take a complete sequence of sites for each vehicle, remove a specified number of sites, and output a partial solution.

3.1.1 Random Removal

One of the goals of the ALNS is to avoid exclusively searching around local critical points in favor of searching for a global best solution. This heuristic removes a fixed number of randomly selected sites.

3.1.2 Worst Removal

This heuristic removes sites that have the smallest slack on one or more of their trips. The minimum slack is calculated for each site over all visits, the sites are reordered from worst to best, and a site to remove is chosen based on a random number and a

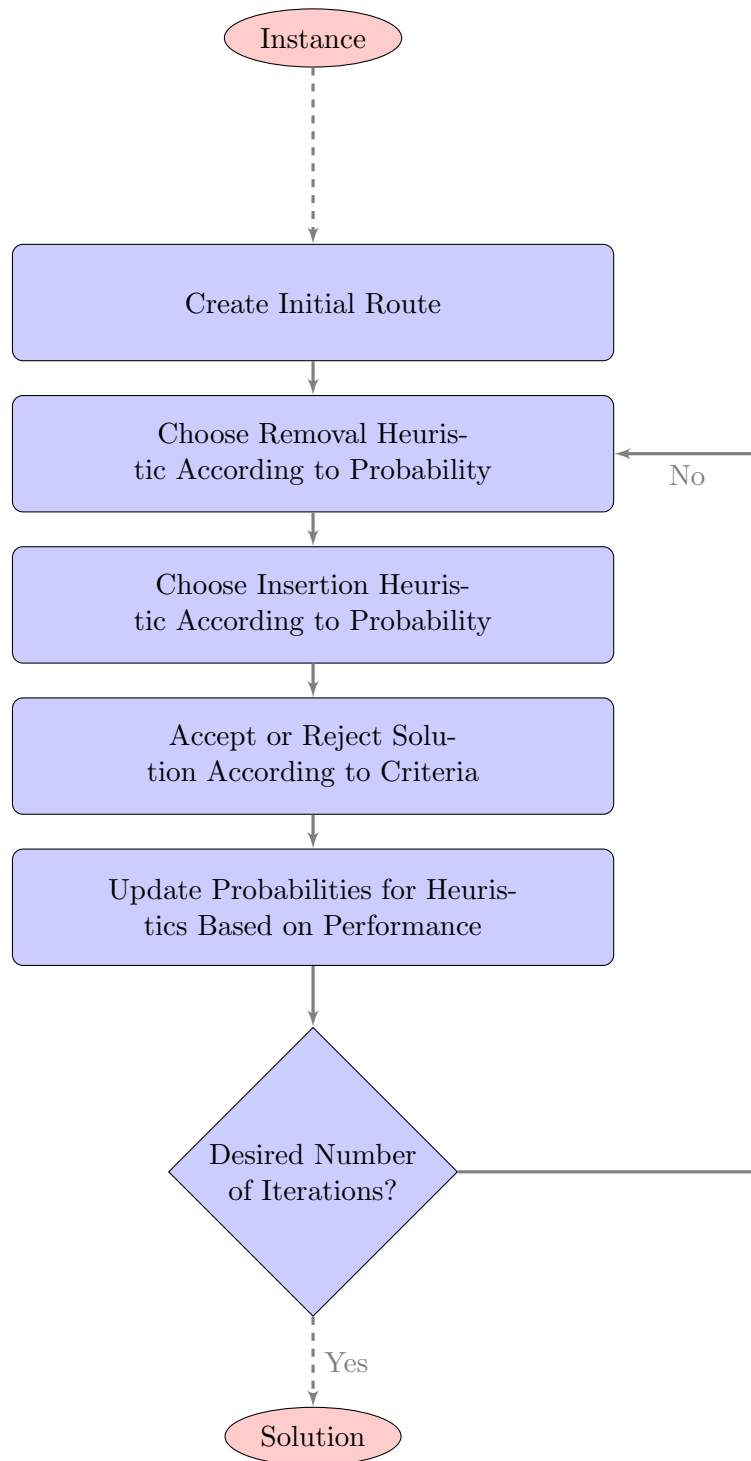


Figure 3.1: ALNS procedure.

parameter. A random number r is drawn from the uniform distribution over $(0,1]$ and the site relaxed is $\lceil r^p n \rceil$ ranked of worst to best slacks. The larger p is, the probability is higher that sites with the smallest slack will be chosen. This is done in order to avoid the same sites being removed repeatedly. This selection is repeated until a desired number of sites are removed.

3.1.3 Related Removal

The related removal heuristic was proposed by Shaw [26] as the sole heuristic for a large neighborhood search. The motivation for removing sites that are related to one another is that, when the removed sites are drastically different, often they are inserted in the same place. Removing similar sites provides better opportunities to generate a new solution. Sites are considered related if they are geographically close and have similar dispensing rates (in magnitude). The relatedness between sites i and j is defined as follows:

$$R(i, j) = \alpha c_{ij} + \delta |L_i - L_j| \quad (3.1)$$

The constants α and δ depend on the magnitudes of the travel times and dispensing rates, as well as their importance relative to the other. When R is smaller, the two sites are more related. This heuristic operates by first randomly selecting a site to remove. Then it randomly chooses another site, where the more related sites are more likely to be selected (the procedure is similar to that used in the Worst Removal heuristic). This continues until the desired number of sites have been removed.

3.1.4 Longest Travel Time Removal

Sites with large travel times tend to have smaller slacks. Thus, this heuristic aims to remove sites that have a long trip to deliver inventory. The heuristic is the similar to the Worst Removal heuristic but ranks the sites by their travel time before randomly selecting them.

3.2 Insertion Heuristics

The insertion heuristics take a partial solution and insert the removed sites to form a complete solution. When a partial solution is presented to an insertion heuristic, the first step is to ensure that each vehicle has at least one site to service. This rule could be implemented in the removal heuristics to prohibit removing a site if it is the only site on a vehicle's route.

3.2.1 Random Insertion

Similar to the random removal heuristic, this heuristic serves the purpose of diversifying the search. This heuristic iteratively places a removed site randomly within a vehicle's sequence on each pass until all sites have been placed.

3.2.2 Best Position of Vehicle with Lowest Travel Time Insertion

For each removed site, this heuristic identifies the vehicle with the smallest travel time, considers each possible position in its route, and inserts the site in the position that yields the highest minimum slack. Two different procedures are used to sequence the sites for consideration. In the first procedure, the sites first inserted were those to be first removed, which, with the worst and longest travel removal heuristics, entails inserting the "worst" sites first. The second procedure inserts the sites backwards such that the "worst" sites are inserted last. The motivation for this is that it may sometimes be of benefit to place difficult sites first while other times it may be best to insert them last.

3.2.3 Best Position of Vehicle with Lowest Total Demand Insertion

This method is similar to the previous insertion technique. It selects the vehicle with the lowest total demand. This method also has two different procedures for the order in which sites are inserted into the solution.

3.2.4 Nearest Neighbor Insertion

Upon each iteration for this heuristic, the relaxed site is placed before or after the site geographically closest already in the partial solution according to which yields the higher minimum slack for the vehicle. This move operation is motivated by influencing vehicles to visit sites closely located.

3.3 Scheduling and Calculating Slack

Scheduling is performed as discussed in the Sections 2.2 and 2.3. Upon each iteration, a schedule is assigned to the vehicle for the newly created route. Whenever a change to the route is performed and a slack measurement is needed, the scheduling procedure is performed on any temporary routes.

When a heuristic identifies a vehicle schedule to insert a site, it will decide placement by attempting all possible points in the route sequence, creating schedules for each possible placement, and choosing the one with the greatest minimum slack.

3.4 Selecting Heuristics and Accepting Solutions

Similar to Pisinger and Ropke’s [28] approach, our ALNS selects a removal heuristic and an insertion heuristic each iteration. A heuristic’s selection probability is proportional to its weight. Both selected heuristics are rewarded in three cases: (1) a new global best solution is found, (2) the new solution is better than the previous one and has not been accepted before, and (3) the new solution is not better than the previous one and it has not been accepted before. If rewarded, the heuristics’ observed weights are increased by 5 (in case 1), 3 (in case 2), or 1 (in case 3). The search process is divided into segments of 50 iterations. At the beginning of the segment, each heuristic has an observed weight of zero. At the end of any segment, the ALNS calculates new weights based on the weights from the previous segment and the observed weights.

Our ALNS uses a simulated annealing procedure to determine if a new solution is accepted. The probability to accept a solution x' (given a current solution x) is given by $\min\{1, e^{\frac{f(x')-f(x)}{T}}\}$ where $T > 0$ is the temperature, which is updated according to a

cooling rate such that $T = Tc$, where $0 < c < 1$. An ALNS can be adapted to utilize various acceptance methods such as simple rejection of poorer solutions or Tabu searches. We have chosen to use simulated annealing since it allows for further exploration of the solution space by allowing poorer solutions to be accepted. These poorer solutions can be intermediate steps in finding better solutions. This is a strategy for only focusing a search near a local optimum.

As introduced previously, a reward is given to a chosen heuristic based on its performance. Thus, it is necessary to store all accepted solutions to evaluate if a solution has not been found before.

3.5 Parameter Selection

The number of sites removed in each iteration was kept small because moving a small number of sites can significantly affect the minimum slack. We set the selection parameter as $p = 5$ for the Worst Removal and Longest Travel Time Removal heuristics in order to focus on the “worst” sites. The α and δ parameters of the Related Removal heuristic were chosen as 1 and 0.75 respectively to scale the magnitudes of the values and put more weight on travel time. The number of sites removed in each iteration was 4.

The starting temperature of the simulated annealing procedure was selected as 600 by observing the objective value of an initial solution and choosing by a desired probability for a relatively lesser value. The cooling rate was then tuned to have reasonable acceptance probabilities towards the end of the search. Two different cooling rates were used and will be discussed in the results.

Chapter 4

Column Generation with Simulated Annealing

With the heuristic and search approaches, the problem was separated into simpler optimization problems. It is necessary to attempt to solve the problem without separation to attain higher quality solutions. A column generation approach is appropriate since it makes it possible to dynamically create candidate schedules that can lead to a higher objective value.

4.1 MIP Formulation

The problem as described in the general formulation is not suitable to be solved as a Mixed-Integer Program (MIP). To reduce the number of variables, the operation period is transformed into T discrete time intervals of length Δ . Although time has been divided into intervals, slack is still calculated in time units. With the heuristic approach, a slack was calculated upon each delivery. With the MIP formulation, a slack value is assigned to each site at the end of a portion (λ) of the time intervals where $\lambda < T$.

A vehicle schedule is defined as ω within the set of all feasible schedules Ω . As stated before, a vehicle schedule contains a sequence of sites to visit, times to start each trip, as well as when to visit the sites, and the amount to deliver at each site. The same feasibility constraints apply to this problem. With a schedule ω , the following parameters are present.

a_t^ω - Amount of inventory taken from the depot in interval t

b_{it}^ω - Amount of inventory delivered to site i in interval t

The decision variables for the formulation are as follows.

x_ω - Binary variable for inclusion of schedule ω

y_{it} - Total amount of inventory delivered to site i in interval t

Q_{it} - Total amount of inventory delivered to site i in intervals 1 to t

z_t - Total amount of inventory taken from depot in interval t

S - Minimum slack for all sites in intervals 1 to λ

The MIP problem can be represented as follows.

$$\max S \quad (4.1)$$

$$\sum_{\omega \in \Omega} a_t^\omega x_\omega - z_t = 0 \text{ for } t = 1, \dots, T \quad (4.2)$$

$$\sum_{\omega \in \Omega} b_{it}^\omega x_\omega - y_{it} = 0 \text{ for } i = 1, \dots, n; t = 1, \dots, T \quad (4.3)$$

$$\sum_{\omega \in \Omega} x_\omega \leq V \quad (4.4)$$

$$Q_{it} - \sum_{j=1}^t y_{ij} = 0 \text{ for } i = 1, \dots, n; t = 1, \dots, T \quad (4.5)$$

$$\sum_{j=1}^t z_j \leq I(\Delta t) \text{ for } t = 1, \dots, T \quad (4.6)$$

$$\sum_{j=1}^T y_j = L_i(T_2 - T_1) \text{ for } i = 1, \dots, n \quad (4.7)$$

$$S - \frac{Q_{it}}{L_i} + \Delta t - T_1 \leq 0 \text{ for } i = 1, \dots, n; t = 1, \dots, \lambda \quad (4.8)$$

$$x_\omega \in \{0, 1\} \forall \omega \in \Omega \quad (4.9)$$

$$y_{it} \geq 0 \text{ for } i = 1, \dots, n; t = 1, \dots, T \quad (4.10)$$

$$z_t \geq 0 \text{ for } t = 1, \dots, T \quad (4.11)$$

Equation 4.1 refers to the objective function of maximizing the minimum slack. Constraint 4.2 defines the value of z_t . Similarly, 4.3 defines the value of y_{it} and 4.5 defines Q_{it} . Constraint 4.4 ensures that no more than the available number of vehicles is used. The amount taken from the depot does not exceed that available as guaranteed by constraint 4.6. Constraint 4.7 guarantees that all sites receive the required amount of inventory. The minimum slack is defined by constraint 4.8 which eliminates the need to define a slack value for each site and time interval. Standard bounds for all decision variables (positivity and binary qualities) are defined by 4.9, 4.10, and 4.11.

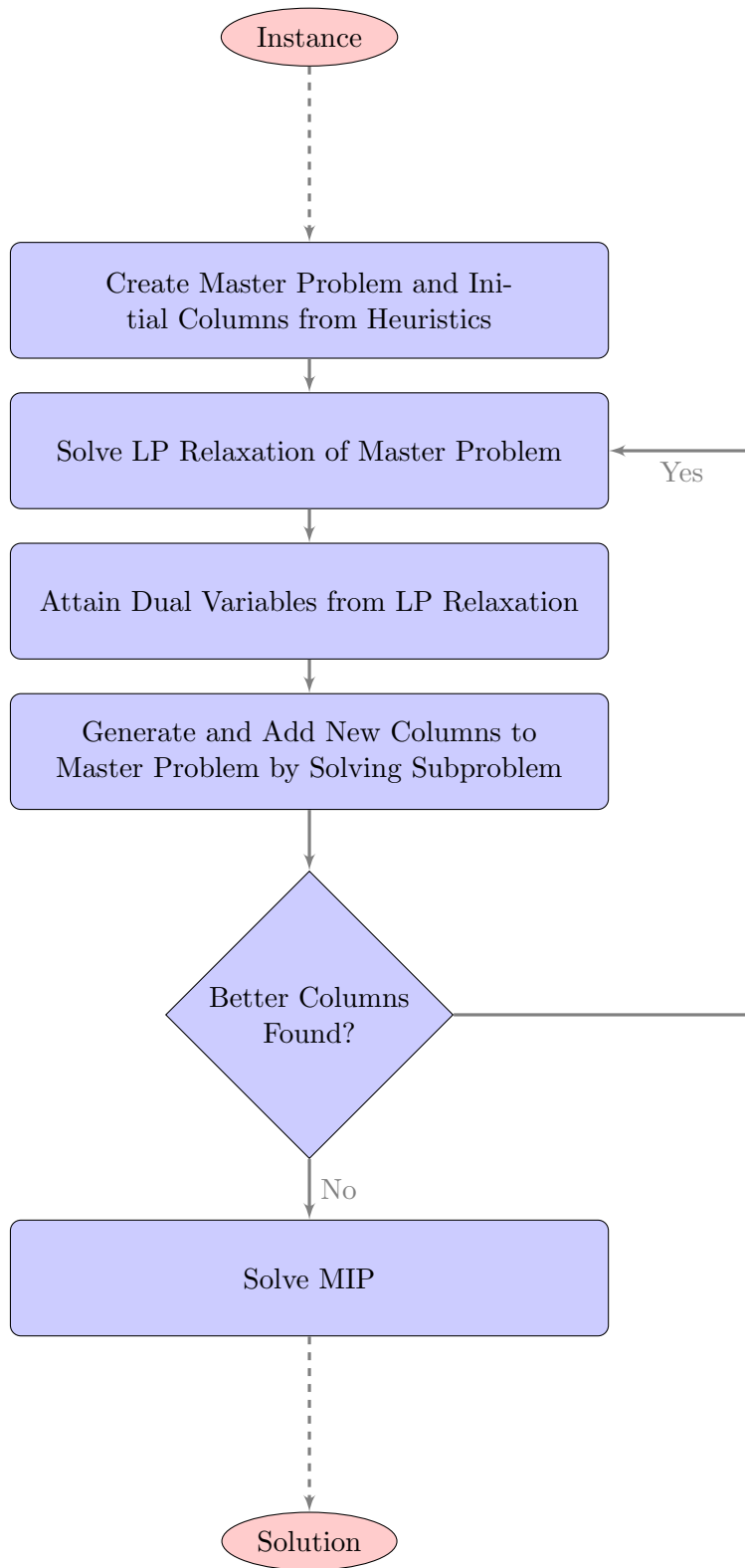


Figure 4.1: CG Procedure.

4.2 Solution Approach

Column generation (CG) approaches can be effective in solving a variety of Operations Research problems [12, 13] where the number of variables in a problem are very large or unknown. The set of all feasible schedules Ω is very large, thus it is necessary to add these at each step of the procedure.

The outline of this approach can be seen in Figure 4.1. Initial columns are created for the MIP by the heuristics introduced in Chapter 2. A linear programming relaxation of this master problem is solved. Dual variables are attained from the solution which are passed to a subproblem for generation of new schedules that will improve the solution of the master problem. This process is repeated until no schedules can be found that will improve the master problem. Then the master problem is solved as a MIP.

Solving the Master Problem involves relaxing the MIP such that $0 \leq x_\omega \leq 1$. Solving this Master Problem yield the dual variables π_t for constraint 4.2 and ρ_{it} for constraint 4.3. By using these dual variables, it is possible to add new schedules that will provide a better slack. By utilizing a subproblem and adding new columns to the Master Problem, the problem is iteratively resolved to add new columns until new ones can not be found. Once the column generation procedure has completed, the schedules that are not a part of the optimal solution with the linear programming relaxation are removed and the problem is solved as a MIP with a branch-and-bound procedure.

The subproblem of the column generation procedure finds schedules, which will become columns in the MIP, that have a positive value of $\sum_{t=1}^T \pi_t a_t^\omega + \sum_{i=1}^n \sum_{t=1}^T \rho_{it} b_{it}^\omega$. This research will utilize simulated annealing in order to generate schedules with a positive objective value. Any such schedules will be placed into the Master Problem as they will increase the slack value.

Since simulated annealing focuses on finding a satisfactory solution in a limited amount of time [16], it will be necessary to investigate techniques (such as dynamic programming) that find optimal schedules upon each iteration in the future. When the subproblem is called, it is repeated a number of times with the same dual variables in order to add multiple schedules.

4.3 Solving the Subproblem

Upon initialization of the subproblem, a site is added at random to the candidate solution. With each iteration of the search, a site is either added at random, removed from the candidate route at random, or two sites are swapped in the sequence visited by the vehicle. These procedures are picked with equal probability at random. In the case of a route with only one site, a site will be added.

4.3.1 Scheduling Times and Quantities

With each iteration of the simulated annealing procedure, a route is found and a delivery schedule must be assigned to it. A decision is made randomly to decide the behavior of the vehicle of when it leaves from the depot. The two behaviors are 1) leaving for the depot as soon as the vehicle returns from a delivery and reload inventory and 2) following the wave structure of the deliveries to the depot. If the behavior is the second, then a vehicle may wait at the depot until the next wave occurs or immediately start loading for the next trip if it has missed the beginning of a wave.

The LP for setting quantities for a given route is as follows. Let J be the number of times that the vehicle completes its route in the solution. Let σ be the set of sites visited in the route. Let $t(j, k)$ be the time interval in which the j -th delivery is made to site k , for $j = 1, \dots, J$ and $k \in \sigma$. Let b_{jk} be amount delivered during the j -th delivery to site k , for $j = 1, \dots, J$ and $k \in \sigma$. Let $t(j, n + 1)$ be the time interval in which the j -th loading occurs at the depot, for $j = 1, \dots, J$. Let $I_j = I(\Delta t(j, n + 1))$, which is the cumulative available at the depot at $t(j, n + 1)$. Let a_j be the amount loaded when the j -th loading is made at the depot.

$$\max \sum_{j=1}^J \pi_{t(j, n+1)} a_j + \sum_{j=1}^J \sum_{k \in \sigma} \rho_{kt(j, k)} b_{jk} \quad (4.12)$$

$$\sum_{j=1}^J b_{jk} = L_k(T_2 - T_1) \quad \forall k \in \sigma \quad (4.13)$$

$$\sum_{i=1}^j a_i \leq I_j \quad \text{for } j = 1, \dots, J \quad (4.14)$$

$$\sum_{k \in \sigma} b_{jk} = a_j \text{ for } j = 1, \dots, J \quad (4.15)$$

$$b_{jk} \geq 0 \text{ for } j = 1, \dots, J; \forall k \in \sigma \quad (4.16)$$

$$a_j \geq 0 \text{ for } j = 1, \dots, J \quad (4.17)$$

This LP is solved in order to maximize the objective sum with the dual variables from the Master Problem. With the attempt to find diverse schedules, two behaviors can be chosen for the scheduling of quantities, which are to 1) completely satisfy the demands of the sites assigned to a vehicle (Constraint 4.13) or 2) schedule deliveries only to maximize the objective and not exceed the quantities demanded by the sites (Constraint 4.13 as an equality).

The linear program solved to scheduled quantities has constraints to not exceed inventory available at the depot and to not exceed or to equal demand constraints of the sites, depending on the behavior chosen (Constraint 4.14). Also, a constraint is present to ensure that a vehicle delivers all quantity loaded before returning to the depot (Constraint 4.15).

4.4 Implementation

CPLEX was called from Matlab to solve the linear programs and mixed-integer programs. The linear programming and mixed-integer programming solvers were used with default parameters with the exception of probing for the mixed-integer procedure changed to very aggressive.

4.4.1 Formulation Changes

In order for CPLEX to find better dual variables, constraints 4.2 and 4.3 were changed from equality constraints to inequality constraints as \leq and \geq , respectively. Also, the upper bound in constraint 4.9 is removed. This is because a generalized upper bound (GUB) can affect the method in which the LP is solved in order to gain computation efficiency. Doing so can reduce the accuracy of the dual variables.

4.4.2 Parameter Selection

The number of iterations for each subproblem call, as well as the number of times the subproblem is tried, is chosen to quickly search the solution space for new columns. These values are set at 40 and 5, respectively. The starting temperature and cooling rate were set at 100 and .9, respectively.

For each instance, the number of time intervals was chosen so that Δ , the duration of a time interval, equals 40 minutes. λ , the number of time intervals used to evaluate the slack, was set at various levels, as described in the computational results. The difficulty here is to pick λ so that the procedure is productive. If $\lambda = T$, then S will always be zero because the end of the last time interval corresponds to the end of the dispensing, when there is no inventory left. However, if λ is too small, the evaluation of the minimum slack will become to inaccurate.

Chapter 5

Computational Analysis

We tested the solution approaches on a set of instances in order to determine which techniques generated the best solutions and evaluate their computational cost. All coding and testing was performed in Matlab, with use of functions from Matlog and CPLEX.

5.1 Creating Instances for Testing

To create the instances, 16 baseline instances were varied systematically to create 27 instances for each baseline for a total of 432 instances. As shown in Table 5.1, the number of sites ranged from 5 to 199. The data for site location, site demand, depot location, and vehicle capacities were obtained from three sources: mass dispensing plans from Montgomery County, Maryland; California PODs from an example provided in the online routing software Toursolver; and the classical vehicle routing problems from Christofides [5]. For the Maryland and California sites, which had street addresses, Toursolver and Google Maps were used to calculate travel times between the sites. We invented demand and wave delivery information to be similar to real world mass dispensing plans from Maryland. All of the instances have loading times of 15 minutes.

For a baseline instance, the parameters varied were the number of vehicles, the average travel time, and the average demand. (Changes to the average demand also required corresponding changes to the amount delivered to the depot in each wave, though we did not change the timing of the waves.) Other times were not modified because varying the travel times changes the loading/unloading times and the wave intervals relative to the travel times.

For each baseline instance, three values were set for the number of vehicles: the initial number V (shown in Table 5.1), $V - 0.2V$, and $V + 0.2V$. The last two values were rounded to the nearest integer. Large-demand instances were created by multiplying every site's demand by 3, and small-demand sites were created by dividing every site's

Baseline instance	Number of sites	Number of vehicles	Number of waves
R1	5	3	3
F1	9	5	4
C1	9	5	5
M1	10	5	5
R3	10	5	4
M2	15	8	3
C2	20	10	4
M3	50	25	5
V1	50	25	3
V2	75	38	5
V3	100	50	6
V4	150	75	7
V5	199	100	7
V11	120	60	5
V12	100	50	6
M4	189	71	4

Table 5.1: Summary of Baseline Instances

demand by 3.

Likewise, in the large-travel-time instances, all of the travel times were multiplied by 2; in the small-travel-time instances, all of the travel times were divided by 2.

Thus, for each baseline instance, we generated 27 instances by combining the three values for the number of vehicles, the three sets of demands, and the three sets of travel times.

5.2 Upper Bound

To find an upper bound on the optimal minimum slack, we relax the problem by ignoring the vehicle capacity and assuming that each site has a vehicle available to deliver material to that site at each wave. Thus, each site is visited once each wave, and the delivery at site k occurs $p_{n+1} + c_{n+1,k} + p_k$ time units after the wave is delivered to the depot.

We use a version of the DVI algorithm to assign delivery quantities to each site so that the slacks of the deliveries in the same wave are equal. The pseudocode for setting the delivery quantities is given in Appendix B. The slacks are calculated based on these

delivery quantities and times.

5.3 Heuristic Testing

The routing and scheduling heuristics were tested on all 432 instances. This section will present solution quality and computational effort.

5.3.1 Heuristic Computational Effort

As shown in Figure 5.1, the time required to generate solutions increased as the number of sites increased and when the 2-opt routing heuristic was used. Other characteristics of the instances did not affect the computational effort. The choice of clustering objective did not affect the computational effort.

Running the 2-opt heuristic generally added 10 to 20 percent to the computational effort. The notable exception to this was the problem set M4 (which is not included in Figure 5.1). For these instances, with the 2-opt heuristic, the average time required was nearly 23 seconds. In all of the other problem sets, the sites surround a central depot. In the problem set M4, however, the depot is located outside of the region in which the sites lie. Thus, it appears that the nearest neighbor heuristic generates a poor route, for the 2-opt procedure spends a great deal of effort to improve the route.

5.3.2 Heuristic Performance

The minimum slack in the best solutions found varied by problem set. Within a problem set, some combinations of number of vehicles, travel times, and demands had only solutions with low slack, while other combinations had solutions with much more slack.

To compare the routing and clustering heuristics, we determined the average minimum slack of the solutions within a problem set and counted the number of times that each routing and clustering combination generated the best solution found. The results, shown in Table 5.2, show that clustering by duration, in general, generated better solutions.

The nearest neighbor procedure and the 2-opt procedure perform equally well. Table 5.3 shows that, when used with the cluster by duration objective, using the nearest

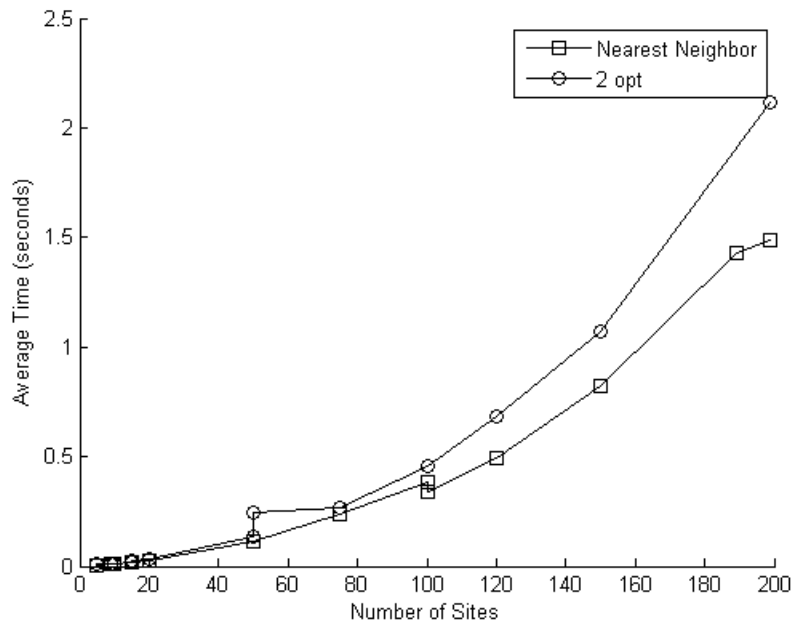


Figure 5.1: Computational time for finding initial routes.

neighbor procedure was slightly more likely to generate a better solution. In many cases, however, they generated equally good solutions. The performance of these heuristics was not affected by the relative length of the wave intervals. Changes in the demand often made no change in the quality of the solution, because the vehicles had sufficient capacity to carry the increase material. In some cases, increasing the site demands generated instances in which the vehicles needed more trips to deliver the material; this naturally reduced the slack and led to poor-quality solutions. The relative performance of the heuristics did not change however.

5.3.3 DVI Performance

The same body of instances were used to test the DVI procedure. The solutions improved were those from the heuristics previously introduced. Table 5.4 shows the number of instances that are candidates for DVI. These are the instances in which the minimum slack does not occur on the first trip of a vehicle. By only looking at DVI results for these instances, we can find the average improvement seen in Table 5.5. This table also shows the average over the problem sets. The two cluster by demand heuristics, which gave the

Routing Heuristic	Average minimum slack				Number of best solutions found			
	Nearest neighbor		2-opt		Nearest neighbor		2-opt	
Clustering objective	Duration	Demand	Duration	Demand	Duration	Demand	Duration	Demand
R1	452.67	433.78	449.52	447.89	24	18	3	3
F1	1255.04	1252.07	1263.48	1250.30	11	6	12	7
C1	394.85	384.85	394.85	384.85	23	7	23	7
M1	566.22	552.04	566.22	552.04	27	12	27	12
R3	1035.00	1038.00	1048.78	1041.11	9	9	15	3
M2	380.78	373.22	381.11	373.22	24	15	27	15
C2	120.67	110.74	103.81	100.04	21	19	24	18
M3	1189.33	1125.22	1177.56	1135.33	21	6	9	9
V1	590.67	574.89	599.67	593.22	6	0	12	12
V2	600.89	590.67	598.00	583.11	15	0	12	0
V3	473.11	459.89	470.56	450.78	18	0	12	3
V4	471.44	414.00	477.44	472.78	0	0	27	12
V5	470.67	462.22	481.67	474.67	9	3	18	9
V11	504.89	497.00	501.56	500.67	24	12	0	0
V12	492.67	476.78	491.56	474.89	24	15	18	3
M4	1254.07	1262.96	1232.04	1262.52	1	12	15	11
Average	640.81	625.52	639.86	631.09	16.1	8.4	15.9	7.8

Table 5.2: Quality of solutions generated by each heuristic on each problem set.

Baseline Instance	Nearest neighbor	Same	2-Opt
R1	24		3
F1	11		16
C1		27	
M1		27	
R3	12	9	6
R2		27	
M2		24	3
C2	3	18	6
M3	12	12	3
V1	6		21
V2	15		12
V3	15	3	9
V4			27
V5	9		18
V11	27		
V12	9	3	15
M4	13		14
Total	156	123	153
Percent	36.1%	28.5%	35.4%

Table 5.3: Number of instances in each problem set that the nearest neighbor and 2-opt routing procedures generated the better solution.

Routing Heuristic	Nearest neighbor		2-opt		
	Clustering objective	Duration	Demand	Duration	Demand
R1		27	27	27	27
F1		6	6	3	3
C1		9	9	9	9
M1		3	3	3	3
R3		27	27	27	27
M2		4	4	4	4
C2		1	1	1	1
M3		27	27	27	27
V1		0	0	0	0
V2		0	0	0	0
V3		0	0	0	0
V4		0	9	0	0
V5		0	0	0	0
V11		0	0	0	0
V12		3	3	3	3
M4		3	3	3	3

Table 5.4: Number of instances in which minimum slack does not occur on first wave.

worst results without DVI, showed the greatest improvement.

To analyze the overall solution quality for the four heuristics with DVI, we consider the solutions for all 432 instances, including those where the minimum slack occurs in a first trip and DVI was not used. Table 5.6 shows the average minimum slack for all heuristics. The two cluster by duration heuristics were similar and outperformed the cluster by demand heuristics. The same is true for the number of best solutions found. Neither routing heuristic dominated the other.

5.3.4 Upper Bound Analysis

We determined the upper bound for each instance and compared to the objective function values of the solutions found by the four heuristics with DVI. Table 5.7 compares the upper bounds to the results obtained from the Route by Nearest Neighbor and Cluster by Duration heuristic. This table presents the difference between the upper bound and heuristic solution for different groups of instances, classified by number of vehicles and travel times between sites and the depot. As the number of vehicles increases, the difference between the heuristic solution and the upper bound decreases. Also, as the travel

Routing Heuristic	Nearest neighbor		2-opt		Problem Set Averages
Clustering objective	Duration	Demand	Duration	Demand	
R1	20.30	24.67	22.63	20.15	21.94
F1	18.00	49.50	36.00	65.67	39.44
C1	28.78	37.11	28.78	37.11	32.94
M1	51.33	63.67	51.33	63.67	57.50
R3	21.11	27.00	17.78	22.89	22.19
M2	0.00	0.00	4.00	0.00	1.00
C2	52.00	52.00	92.00	92.00	72.00
M3	21.11	57.22	21.00	53.11	38.11
V1	-	-	-	-	-
V2	-	-	-	-	-
V3	-	-	-	-	-
V4	-	27.33	-	-	27.33
V5	-	-	-	-	-
V11	-	-	-	-	-
V12	2.00	26.00	2.00	26.00	14.00
M4	24.67	18.33	27.33	21.33	22.92
Average	23.93	34.80	30.29	40.19	31.76

Table 5.5: Average improvement for each heuristic.

times decrease, the difference decreases. From the results, it can be seen that, in many cases, the heuristics generate near-optimal solutions. For some instances, the gap between the upper bound and the heuristic solution is much larger because the heuristic generated a poor solution or the quality of the upper bound is poor.

5.4 ALNS Testing

The ALNS procedure was tested on all of the instances and compared to the heuristic solutions. All parameters for the search were defined in the ALNS section. All tests had a cooling rate of 0.8, with the exception of the specified column in Table 5.8 which had a cooling rate of 0.6. For purpose of succinct reporting, all instance solutions were divided by the upper bound objective value and averaged over an instance set as reported in Table 5.8. Each column in this table refers to a separate run of the ALNS (or heuristic alone) with various characteristics.

The column for the heuristic solution refers to the solution with the maximum

Routing Heuristic	Average minimum slack				Number of best solutions found			
	Nearest neighbor		2-opt		Nearest neighbor		2-opt	
Clustering objective	Duration	Demand	Duration	Demand	Duration	Demand	Duration	Demand
R1	472.96	458.44	472.15	468.04	18	18	17	9
F1	1259.04	1263.07	1267.48	1257.59	11	6	12	7
C1	404.44	397.22	404.44	397.22	23	7	23	7
M1	571.93	559.11	571.93	559.11	27	12	27	12
R3	1056.11	1065.00	1066.56	1064.00	18	15	12	6
M2	380.78	373.22	381.56	373.22	21	12	27	12
C2	122.59	112.67	107.22	103.44	21	19	24	18
M3	1210.44	1182.44	1198.56	1188.44	15	6	12	12
V1	590.67	574.89	599.67	593.22	6	0	12	12
V2	600.89	590.67	598.00	583.11	15	0	12	0
V3	473.11	459.89	470.56	450.78	18	0	12	3
V4	471.44	423.11	477.44	472.78	0	0	27	12
V5	470.67	462.22	481.67	474.67	9	3	18	9
V11	504.89	497.00	501.56	500.67	24	12	0	0
V12	492.89	479.67	491.78	477.78	24	15	18	3
M4	1256.81	1265.00	1235.07	1264.89	1	12	15	11
Average	646.23	635.23	645.35	639.31	15.7	8.6	16.8	8.3

Table 5.6: Quality of solutions generated by each heuristic with DVI on each problem set.

Number of Vehicles	Travel Time Size	Nearest neighbor		2-opt	
		Duration	Demand	Duration	Demand
Many	Large	37.81	44.81	41.63	46.31
Many	Average	18.19	28.00	20.75	27.00
Many	Small	15.63	20.44	16.44	19.44
Average	Large	51.94	65.88	47.50	57.31
Average	Average	28.31	39.88	24.44	36.44
Average	Small	22.13	27.19	19.50	26.38
Few	Large	147.52	167.77	165.83	153.90
Few	Average	62.54	80.75	57.75	71.39
Few	Small	48.50	56.88	46.63	55.19

Table 5.7: Average difference between upper bound and minimum slack for different sets of instances classified by number of vehicles and travel times.

objective value from the four variants of the routing and scheduling heuristic. For the ALNS searches, with the exception of the last one referenced in the table, a NN route was built and clustered by dividing the sites evenly among the vehicles. The last search, with a heuristic starting point, used the route with the best objective value from the routing and scheduling heuristics as the starting point.

For the search results with sufficient time to search the solution space, which can be seen by the ALNS with 2000 iterations, the heuristic solution is outperformed consistently. When performing the search with a short number of iterations and a faster cooling rate, it was possible to outperform the heuristic in all cases, with the exception of V2 and V12. A visual representation of one search with sufficient time to explore the solution space can be seen in Figure 5.2.

By starting with a heuristic solution, the ALNS was able to improve all solution values. However, many of the improvements appear to be subtle, while others make significant improvements (most notably with C2). With that being said, it is worth noting that if a heuristic is being used, given sufficient computational time is available, it would be of value to execute the ALNS to improve the route.

The most interesting trend in the results is that the best solution overall is found from the longest search. There are two possible reasons for this result: 1) the search performs best when starting with a poor solution since it may be a better starting point

Set of Instances	Heuristic	ALNS with 500 Iterations	ALNS with 1000 Iterations	ALNS with 2000 Iterations	ALNS with 500 Iterations (Faster Cooling)	ALNS with 500 Iterations (Heuristic Starting Point)
R1	0.958	0.962	0.962	0.962	0.962	0.962
F1	0.970	0.981	0.981	0.981	0.981	0.981
C1	0.843	0.911	0.910	0.911	0.911	0.911
M1	0.900	0.936	0.937	0.939	0.937	0.935
R3	0.972	0.982	0.982	0.983	0.982	0.982
M2	0.934	0.967	0.969	0.971	0.968	0.967
C2	0.647	0.880	0.888	0.904	0.883	0.875
M3	0.943	0.952	0.960	0.966	0.958	0.954
V1	0.941	0.940	0.947	0.957	0.944	0.946
V2	0.947	0.939	0.945	0.956	0.943	0.948
V3	0.925	0.930	0.937	0.950	0.936	0.935
V12	0.963	0.953	0.965	0.976	0.962	0.964
V11	0.968	0.966	0.970	0.976	0.969	0.971
V4	0.932	0.925	0.935	0.952	0.934	0.941
M4	0.946	0.955	0.958	0.963	0.956	0.957
V5	0.949	0.930	0.936	0.953	0.935	0.950

Table 5.8: Average solutions by baseline of best performing heuristic and ALNS with varying parameters.

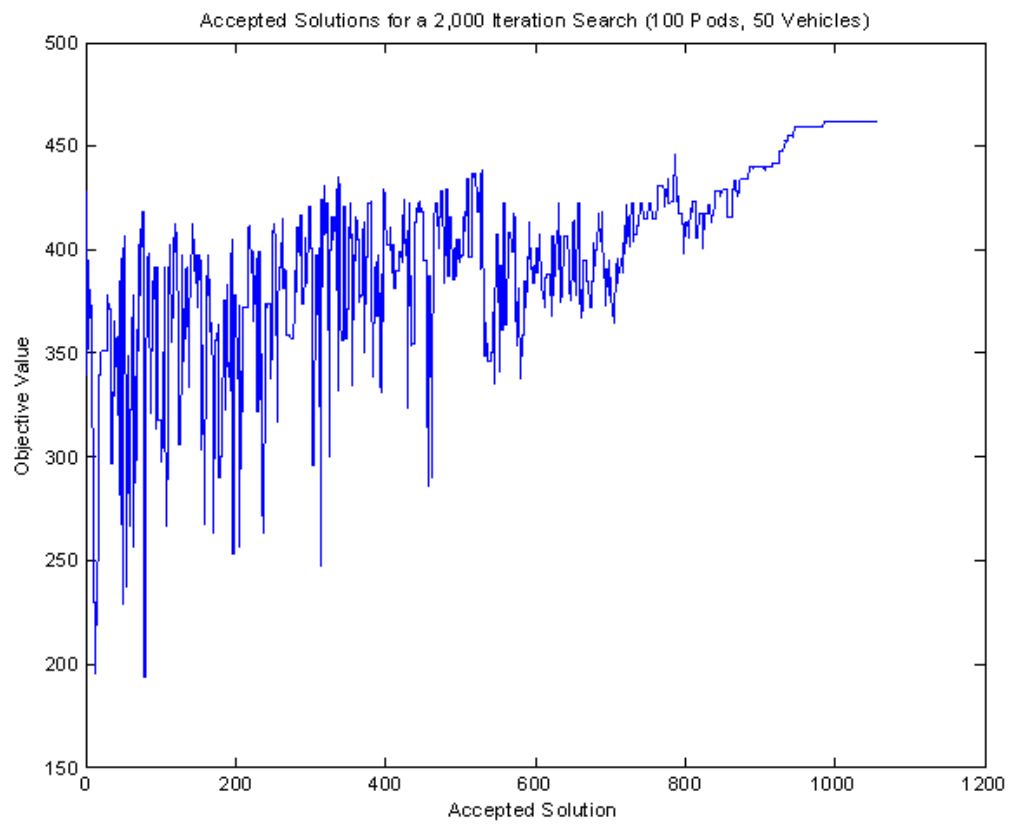


Figure 5.2: ALNS procedure with 2,000 iterations.

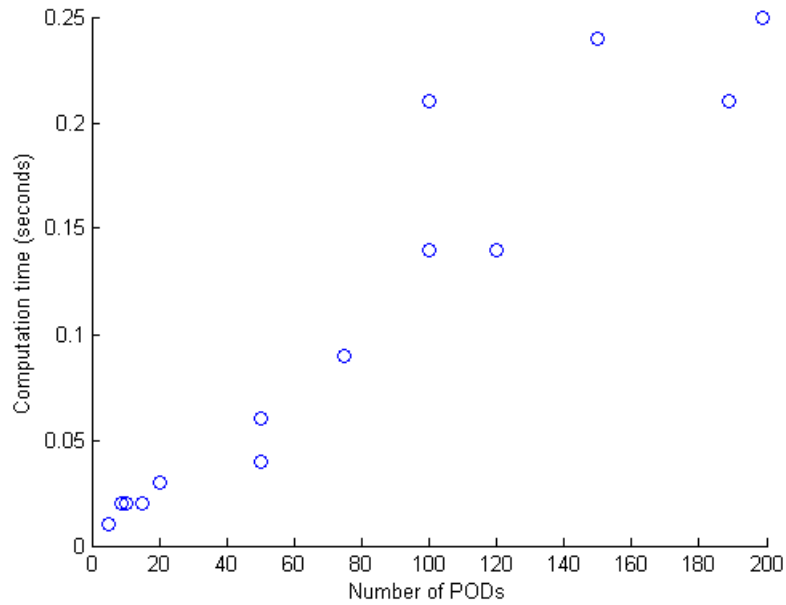


Figure 5.3: Average computational time in seconds per iteration for one complete run of the ALNS with 500 iterations.

for searching the solution space, or 2) even with a heuristic starting point, it is necessary to have a longer search until a good route has been found.

In Appendix C, extensive results are included for each instance set with the average solution value over 5 runs of the ALNS procedure, along with the confidence interval half-width. With smaller instances, much variation did not exist, which indicates that the optimal route was found. Generally, shorter searches resulted in more variation between final solution objective values.

The interval half-widths seem to be reasonably small (less than 10 minutes) in most cases. Some results present much variance which is most likely a result of a search not finding a good route due to the simulated annealing behavior.

Computation was performed on three different computers in parallel. Thus, to compare computational effort, the running time of the first search with 500 iterations were recorded. The average computational time per iteration and instance set are presented in Figure 5.3. The computational effort is presented as a function of the number of PODs, since this most greatly affected the time. As expected, the time increases with the number of PODs.

5.5 CG Testing

A smaller set of 16 instances (corresponding to the baseline instances) were used to test the CG procedure. As seen in Table 5.9, the results were not as productive as the heuristics or the search procedure. This procedure was not possible for instances with more than 20 PODs due to insufficient memory. All values are for one procedure attempt except for R1 and R3 which were attempted three times.

For instances F1, C1, and C2, by attempting the procedure with all possible number of examined intervals, no improvement could be found. Thus, the default number of examined intervals was chosen as the last interval that receives inventory in the initial heuristic columns. As seen by the computation time, the initial LP was solved, the subproblem was attempted, and no columns were found. With these instances, the dual information was not helpful in generating new columns to add to the master problem.

Two possible explanations for this behavior are that the solver did not accurately calculate the dual variables or the subproblem failed in finding new columns because no feasible columns exist that would improve the solution.

For instances M1 and M2, the CG procedure was successful in finding new columns, but they provided extremely small improvements to the objective value. For M1, the procedure had a long running time of nearly half an hour. For M2, it iterated a small number of times (as seen by the computation time of less than 1 minute). The number of examination intervals (λ) for the instances was chosen by attempting the procedure on all possibilities and selecting the value such that the procedure did not immediately terminate.

Further examination is required for R1 and R3 which showed improvement. The number of examined intervals was chosen such that an improvement to the objective value was reported. For R1, as seen in Table 5.10, the CG procedure was attempted three times. The first attempt did not find any columns and terminated naturally. The second and third attempt were terminated prematurely when no significant improvement was made to the objective value after 3,000 iterations. This can be seen by the long run times for these two attempts.

Instance	Initial IP Objective	Initial LP Objective	Final LP Objective	Final IP Objective	Initial Number of Columns	Final Number of Columns	Computation Time (Minutes)	Total Intervals (T)	Examined Intervals (λ)
R1	503.5	503.5	531.5	491.5	24	13.7	59.16	30	16
F1	1320	1360	1360	1320	40	10	0.03	72	16
C1	480	520	520	480	40	10	0.03	72	24
M1	600	600	600	600	40	10	29.65	54	13
R3	1085.3	1112.2	1128.1	955.8	40	35.3	45.76	51	21
M2	400	400	400	400	64	21	0.73	30	20
C2	180.8	180.8	180.8	180.8	80	10	0.12	80	10
M3	-	-	-	-	-	-	-	51	-
V1	-	-	-	-	-	-	-	90	-
V2	-	-	-	-	-	-	-	87	-
V3	-	-	-	-	-	-	-	87	-
V4	-	-	-	-	-	-	-	87	-
V5	-	-	-	-	-	-	-	87	-
V11	-	-	-	-	-	-	-	87	-
V12	-	-	-	-	-	-	-	87	-
M4	-	-	-	-	-	-	-	72	-

Table 5.9: Results for the CG procedure on the baseline instances.

CG Attempt	Columns in Final Basis	Final IP Objective	Computation Time (minutes)
1	14	491.5	8.34
2	13	491.5	84.67
3	14	491.5	84.46

Table 5.10: CG attempts for R1.

CG Attempt	Columns in Final Basis	Final IP Objective	Computation Time (minutes)
1	36	1072.2	45.72
2	38	723.0	45.68
3	32	1072.2	45.87

Table 5.11: CG attempts for R3.

Although improvement for the LP occurred, when the procedure finished and all columns not in the optimal basis were removed, the resulting IP solution was worse than the original IP solution. This indicates that the original heuristic columns were removed and that it was not possible to find a feasible set of columns among those generated with a better objective function value.

Similarly, the attempts for R3, seen in Table 5.11, yielded similar results. All three attempts were ended prematurely at 1,000 iterations with no significant improvement to the objective function value. Both R1 and R3 achieved the maximum LP relaxation objective within the first 5 minutes of the procedure.

The columns in the initial master problem were created using all of the heuristics with and without the DVI procedure. To test the impact of starting with a fewer number of initial columns with poorer quality, the study was repeated using only schedules generated by the routing by nearest neighbor, cluster by duration, and no DVI as seen below.

Similar behavior occurred in this scenario with the CG procedure increasing the objective value of the LP to that seen in Table 5.14 for R1 and R3, but not F1 and C1. In Table 5.13, we can see specific information for three replications of the procedure. (All other instances were attempted once.) For R1, two attempts (1 and 2) were terminated prematurely at 3,000 iterations without improvement of the objective function. R3 was

Instance	Initial	Initial	Initial	Final	Final	Initial	Final	Computation	Total	Examined
	Objective	LP	Objective	LP	Objective					
	IP	Objective	Objective	Objective	Objective	Number	of Columns	(Minutes)	Intervals	
R1	491.5	491.5	531.5	491.5	491.5	3	25.3	59.81	30	16
F1	1320	1320	1320	1320	1320	5	5	0.03	72	16
C1	480	480	480	480	480	5	5	0.03	72	24
M1	600	600	600	600	600	5	5	0.02	54	13
R3	1072.2	1072.2	1128.1	1072.2	1072.2	5	41	272.14	51	21
M2	400	400	400	400	400	8	46	4.19	30	20
C2	180.8	180.8	180.8	180.8	180.8	10	10	0.06	80	10
M3	-	-	-	-	-	-	-	-	-	-
V1	-	-	-	-	-	-	-	-	-	-
V2	-	-	-	-	-	-	-	-	-	-
V3	-	-	-	-	-	-	-	-	-	-
V4	-	-	-	-	-	-	-	-	-	-
V5	-	-	-	-	-	-	-	-	-	-
V11	-	-	-	-	-	-	-	-	-	-
V12	-	-	-	-	-	-	-	-	-	-

Table 5.12: Results for the CG procedure on the baseline instances with poorer starting columns.

CG Attempt	Columns in Final Basis	Final IP Objective	Computation Time (minutes)
1	20	491.5	96.44
2	20	491.5	66.12
3	36	491.5	16.86

Table 5.13: CG attempts for R1 with poorer starting columns.

attempted three times, but output is only available for one attempt. During the other two attempts, the procedure terminated because the constraint matrix grew too large and exceeded the memory.

The above results utilized time intervals of 40 minutes. It is necessary to select time interval length such that a pickup or delivery can not occur in the same interval to provide adequate precision with quantity allocation. To analyze the effect of smaller time intervals than 40 minutes, the study was repeated as shown below with 20 minute intervals. All heuristics are utilized for the initial columns.

Attempting the CG procedure on C2 exceeded the memory upon building the initial constraint matrix. The CG procedure terminated for R3 because the constraint matrix grew too large. Three replications were attempted for R1 with replications 1 and 2 terminating prematurely at 3,000 iterations with no change in the objective function. Using finer time intervals resulted in a large increase in computational time.

This CG procedure generated undesirable results. In some cases, no improvements were seen because the dual variables from the initial LP were not productive for the subproblem. In the cases where improvement was seen, the procedure required long run and made only infinitesimal improvements following the initial improvement. When the procedure terminated, the IP solution was worse than the initial solution.

These two issues with instances that can be improved give insight into future work. A subproblem is needed that can produce columns that are compatible with other columns such that they are an improvement and are suitable for the IP constraints (most notably with respect to available inventory in each interval).

Instance	Initial	Initial	Initial	Final	Final	Initial	Final	Computation	Total	Examined
	Objective	LP	Objective	LP	Objective					
R1	491.5	500.5	506.0	491.5	491.5	24	25.7	103.99	60	32
F1	1300	1340	1340	1300	1300	40	11	0.07	144	32
C1	480	500	500	480	480	40	5	0.07	144	48
M1	600	600	600	600	600	40	5	0.05	108	26
R3	-	-	-	-	-	-	-	-	-	-
M2	409.4	421.2	421.2	409.4	409.4	64	11	0.10	60	40
C2	-	-	-	-	-	-	-	-	-	-
M3	-	-	-	-	-	-	-	-	-	-
V1	-	-	-	-	-	-	-	-	-	-
V2	-	-	-	-	-	-	-	-	-	-
V3	-	-	-	-	-	-	-	-	-	-
V4	-	-	-	-	-	-	-	-	-	-
V5	-	-	-	-	-	-	-	-	-	-
V11	-	-	-	-	-	-	-	-	-	-
V12	-	-	-	-	-	-	-	-	-	-

Table 5.14: Results for the CG procedure on the baseline instances with 20 minute intervals.

CG Attempt	Columns in Final Basis	Final IP Objective	Computation Time (minutes)
1	28	491.5	115.43
2	23	491.5	119.07
3	26	491.5	77.46

Table 5.15: CG attempts for R1 with 20 minute intervals.

Chapter 6

Conclusions

This thesis presented a novel vehicle routing problem can be used for planning medication (or other inventory) distribution in case of an emergency. Three solution techniques were developed and tested on a large set of instances. Contributions from this project are seen below.

- The formulation of the inventory slack routing problem, a novel vehicle routing problem in which the objective is to maximize the slack in the delivery schedule. Such a problem can be used for responding to emergencies because it emphasizes building plans that are robust against uncertainty.
- A routing and scheduling heuristic was introduced as a quick and computationally efficient method for constructing solutions to this problem. The results of the heuristics were compared to an upper bound to see the quality of solutions generated.
- A search technique was introduced as an extension to the routing and scheduling heuristics to improve the routes in which the schedules are built.
- A framework for an integer programming approach to this problem was introduced.
- A body of instances similar to real-world emergency situations have been created for use by researchers interested in studying the ISRP.

In an overall comparison of the three techniques, the ALNS approach provided the best solutions and is executed with reasonable computational effort. Thus, this technique is currently the most valuable approach for solving the ISRP.

Future work includes implementing and extending this work. The routing and scheduling heuristics, as well as the search technique, should be implemented in a stand-alone program that can be used freely by public health emergency preparedness planners. This is important to distribute the work for the purpose of gaining feedback on performance of the heuristics, as well as promoting awareness of this vehicle routing problem.

It is necessary to further investigate the column generation approach and make improvements. With the current approach, examining the LP solely and relying on fractional schedules to make improvements towards a better IP solution is insufficient. Thus, a more sophisticated technique is needed to make improvements towards integral solutions upon each iteration. A branch-and-price approach should be first investigated towards this goal. It is also necessary to investigate the effects of discretization the problem.

Further techniques should be investigated, including simulation, to study the effects of uncertainty on solutions, as well as to find optimal solutions.

Appendix A

Example Instance

The following data is used for the examples described in the introduction. Note that there are three waves (deliveries to the depot), as shown in the following table.

$$T_1 = 600 \text{ minutes}, T_2 = 1200 \text{ minutes}$$

$$I(0) = 48,000 \text{ units}, I(180) = 146,000 \text{ units}, I(360) = 219,000 \text{ units}$$

$$V = 3 \text{ vehicles}$$

$$C = 112,000 \text{ units per vehicle}$$

$$n = 5 \text{ PODs}$$

$$L = (50 \ 75 \ 100 \ 60 \ 80) \text{ units per minute}$$

$$p = 15 \text{ minutes}$$

$$c = \begin{pmatrix} 0 & 14 & 39 & 29 & 26 & 17 \\ 14 & 0 & 34 & 24 & 21 & 15 \\ 39 & 34 & 0 & 17 & 16 & 30 \\ 29 & 24 & 17 & 0 & 13 & 17 \\ 26 & 21 & 16 & 13 & 0 & 15 \\ 17 & 15 & 30 & 17 & 15 & 0 \end{pmatrix} \text{ with all times in minutes}$$

Wave	1	2	3
Time (minutes)	0	180	360
Quantity	48,000	98,000	73,000

Table A.1: Deliveries to the depot.

Appendix B

Pseudocode

The pseudocode for improvement by duration algorithm is as follows. During implementation, with one iteration it is important to examine all maximum and minimum vehicles instead of the first encountered. In this code, the following notation is necessary: l_v denotes the first site assigned to vehicle v , and m_v denotes the last site assigned to vehicle v .

improvebyduration(σ)

```
1 REPEAT
2   CALCULATE  $y_1, \dots, y_V$ 
3    $F = \max\{y_1, \dots, y_V\} - \min\{y_1, \dots, y_V\}$ 
4
5    $v_{max} = \operatorname{argmax}\{y_1, \dots, y_V\}$ 
6    $v_{min} = \operatorname{argmin}\{y_1, \dots, y_V\}$ 
7
8    $\sigma'_{v_{max}-1} = \sigma_{v_{max}-1} \cup \{l_{v_{max}}\}$ 
9    $\sigma'_{v_{max}} = \sigma_{v_{max}} \setminus \{l_{v_{max}}\}$ 
10
11  CALCULATE  $y'_1, \dots, y'_V$ 
12
13  IF  $\max\{y'_1, \dots, y'_V\} - \min\{y'_1, \dots, y'_V\} < \max\{y_1, \dots, y_V\} - \min\{y_1, \dots, y_V\}$ 
14     $\sigma_{v_{max}-1} = \sigma'_{v_{max}-1}$ 
15     $\sigma_{v_{max}} = \sigma'_{v_{max}}$ 
16     $y_{v_{max}-1} = y'_{v_{max}-1}$ 
17     $y_{v_{max}} = y'_{v_{max}}$ 
18  END
19
20   $\sigma'_{v_{max}+1} = \{m_{v_{max}}\} \cup \sigma_{v_{max}+1}$ 
21   $\sigma'_{v_{max}} = \sigma_{v_{max}} \setminus \{m_{v_{max}}\}$ 
22
23  CALCULATE  $y'_1, \dots, y'_V$ 
24
25  IF  $\max\{y'_1, \dots, y'_V\} - \min\{y'_1, \dots, y'_V\} < \max\{y_1, \dots, y_V\} - \min\{y_1, \dots, y_V\}$ 
26     $\sigma_{v_{max}} = \sigma'_{v_{max}}$ 
27     $\sigma_{v_{max}+1} = \sigma'_{v_{max}+1}$ 
28     $y_{v_{max}} = y'_{v_{max}}$ 
29     $y_{v_{max}+1} = y'_{v_{max}+1}$ 
30  END
31
32   $\sigma'_{v_{min}} = \{m_{v_{min}-1}\} \cup \sigma_{v_{min}}$ 
```

```

33    $\sigma'_{v_{min}-1} = \sigma_{v_{min}-1} \setminus \{m_{v_{min}-1}\}$ 
34
35   CALCULATE  $y'_1, \dots, y'_V$ 
36
37   IF  $\max\{y'_1, \dots, y'_V\} - \min\{y'_1, \dots, y'_V\} < \max\{y_1, \dots, y_V\} - \min\{y_1, \dots, y_V\}$ 
38        $\sigma_{v_{min}-1} = \sigma'_{v_{min}-1}$ 
39        $\sigma_{v_{min}} = \sigma'_{v_{min}}$ 
40        $y_{v_{min}-1} = y'_{v_{min}-1}$ 
41        $y_{v_{min}} = y'_{v_{min}}$ 
42   END
43
44    $\sigma'_{v_{min}} = \sigma_{v_{min}} \cup \{l_{v_{min}+1}\}$ 
45    $\sigma'_{v_{min}+1} = \sigma_{v_{min}+1} \setminus \{l_{v_{min}+1}\}$ 
46
47   CALCULATE  $y'_1, \dots, y'_V$ 
48
49   IF  $\max\{y'_1, \dots, y'_V\} - \min\{y'_1, \dots, y'_V\} < \max\{y_1, \dots, y_V\} - \min\{y_1, \dots, y_V\}$ 
50        $\sigma_{v_{min}} = \sigma'_{v_{min}}$ 
51        $\sigma_{v_{min}+1} = \sigma'_{v_{min}+1}$ 
52        $y_{v_{min}} = y'_{v_{min}}$ 
53        $y_{v_{min}+1} = y'_{v_{min}+1}$ 
54   END
55
56    $F' = \max\{y_1, \dots, y_V\} - \min\{y_1, \dots, y_V\}$ 
57
58 UNTIL  $F == F'$ 

```

To determine the upper bound for an instance, we use the following pseudocode to determine delivery quantities. In this code, Y and m are introduced as values used to find the desired slack. These values are calculated in lines 1-13 of the pseudocode. W_j refers to the amount of inventory made available to the depot in wave j . The value t_j refers to the time that the wave delivery j is made to the depot. Lines 15-37 of the pseudocode calculate the quantities, q , for each vehicle. After determining these delivery quantities we determine the minimum slack of the deliveries.

upperbound

```

1  Renumber the sites so that  $w_1 \geq \dots \geq w_n$ 
2
3   $Y_0 = 0$ 
4  FOR  $h = 1, \dots, n - 1$ 
5       $m_h = \sum_{i=1}^h L_i$ 
6       $Y_h = Y_{h-1} + (w_h - w_{h+1})m_h$ 
7  END

```

```

8  $m_n = \sum_{i=1}^n L_i$ 
9  $Y_n = Y_{n-1} + (T_2 - w_1 - T_2 + w_n)m_n$ 
10 FOR  $h = n + 1, \dots, 2n - 1$ 
11      $m_h = \sum_{i=h-n+1}^n L_i$ 
12      $Y_h = Y_{h-1} + (w_{h-n} - w_{h-n+1})m_h$ 
13 END
14
15 FOR  $j = 1, \dots, r - 1$ 
16      $Q = W_1 + \dots + W_j$ 
17     FIND  $h$  such that  $Y_{h-1} \leq Q < Y_h$ 
18     IF  $h \leq n$ 
19          $K_j = T_1 - t_{j+1} - w_h + \frac{Q - Y_{h-1}}{m_h}$ 
20         FOR  $k = 1, \dots, h$ 
21              $q_{jk} = (K_j - T_1 + t_{j+1} + w_k)L_k - \sum_{i=1}^{j-1} q_{ik}$ 
22         END
23         FOR  $k = h + 1, \dots, n$ 
24              $q_{jk} = 0$ 
25         END
26     ELSE
27          $K_j = T_2 - t_{j+1} - w_{h-n} + \frac{Q - Y_{h-1}}{m_h}$ 
28         FOR  $k = 1, \dots, h - n$ 
29              $q_{jk} = L_k(T_2 - T_1) - \sum_{i=1}^{j-1} q_{ik}$ 
30         END
31         FOR  $k = h - n + 1, \dots, n$ 
32              $q_{jk} = (K_j - T_1 + t_{j+1} + w_k)L_k - \sum_{i=1}^{j-1} q_{ik}$ 
33         END
34     END
35 FOR  $k = 1, \dots, n$ 
36      $q_{rk} = L_k(T_2 - T_1) - \sum_{i=1}^{r-1} q_{ik}$ 
37 END

```

Appendix C

ALNS Results

The data provided is for ALNS testing with varied parameters. All parameters are as described in the computation section that are not varied in the tables. The instance within a set is specified by a three letter code where the first entry is Many, Average, or Few in regard to the number of vehicle, Large, Average, or Small in regard to the travel times between the depot and PODs, and Large, Average, or Small in regard to the size of demands.

For each search, an average of the maximum minimum slack over the 5 replications was taken and reported along with its 95% confidence half-width. Blank entries correspond to test runs that had no variance over the 5 replications.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	462	-	462	-	462	-	462	-	462	-	462	-
MLA	465	-	465	-	465	-	465	-	465	-	465	-
MLS	465	-	465	-	465	-	465	-	465	-	465	-
MAL	492	-	492	-	492	-	492	-	492	-	492	-
MAA	492	-	492	-	492	-	492	-	492	-	492	-
MAS	492	-	492	-	492	-	492	-	492	-	492	-
MSL	505	-	505	-	505	-	505	-	505	-	505	-
MSA	505	-	505	-	505	-	505	-	505	-	505	-
MSS	505	-	505	-	505	-	505	-	505	-	505	-
ALL	462	-	462	-	462	-	462	-	462	-	462	-
ALA	465	-	465	-	465	-	465	-	465	-	465	-
ALS	465	-	465	-	465	-	465	-	465	-	465	-
AAL	492	-	492	-	492	-	492	-	492	-	492	-
AAA	492	-	492	-	492	-	492	-	492	-	492	-
AAS	492	-	492	-	492	-	492	-	492	-	492	-
ASL	505	-	505	-	505	-	505	-	505	-	505	-
ASA	505	-	505	-	505	-	505	-	505	-	505	-
ASS	505	-	505	-	505	-	505	-	505	-	505	-
FLL	400	-	400	-	400	-	400	-	400	-	400	-
FLA	438	-	438	-	438	-	438	-	438	-	438	-
FLS	438	-	438	-	438	-	438	-	438	-	438	-
FAL	479	-	479	-	479	-	479	-	479	-	479	-
FAA	479	-	479	-	479	-	479	-	479	-	479	-
FAS	479	-	479	-	479	-	479	-	479	-	479	-
FSL	494	-	494	-	494	-	494	-	494	-	494	-
FSA	494	-	494	-	494	-	494	-	494	-	494	-
FSS	494	-	494	-	494	-	494	-	494	-	494	-

Table C.1: ALNS test for instance set R1.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	1248.0	-	1248.0	-	1248.0	-	1248.0	-	1248.0	-	1248.0	-
MLA	1248.0	-	1248.0	-	1248.0	-	1248.0	-	1248.0	-	1248.0	-
MLS	1248.0	-	1248.0	-	1248.0	-	1248.0	-	1248.0	-	1248.0	-
MAL	1328.0	-	1328.0	-	1328.0	-	1328.0	-	1328.0	-	1328.0	-
MAA	1328.0	-	1328.0	-	1328.0	-	1328.0	-	1328.0	-	1328.0	-
MAS	1327.4	1.2	1328.0	-	1328.0	-	1328.0	-	1328.0	-	1328.0	-
MSL	1361.8	0.4	1362.0	-	1362.0	-	1361.6	0.8	1362.0	-	1362.0	-
MSA	1362.0	-	1362.0	-	1362.0	-	1362.0	-	1362.0	-	1362.0	-
MSS	1362.0	-	1362.0	-	1362.0	-	1362.0	-	1362.0	-	1362.0	-
ALL	1234.6	2.9	1237.0	-	1237.0	-	1237.0	-	1237.0	-	1237.0	-
ALA	1234.6	2.9	1237.0	-	1237.0	-	1237.0	-	1237.0	-	1235.8	2.4
ALS	1237.0	-	1237.0	-	1237.0	-	1237.0	-	1237.0	-	1235.8	2.4
AAL	1315.4	1.2	1316.0	-	1316.0	-	1316.0	-	1316.0	-	1315.4	1.2
AAA	1315.4	1.2	1316.0	-	1316.0	-	1316.0	-	1316.0	-	1316.0	-
AAS	1316.0	-	1316.0	-	1316.0	-	1316.0	-	1316.0	-	1316.0	-
ASL	1356.0	-	1356.0	-	1356.0	-	1356.0	-	1356.0	-	1356.0	-
ASA	1356.0	-	1356.0	-	1356.0	-	1356.0	-	1356.0	-	1356.0	-
ASS	1356.0	-	1356.0	-	1356.0	-	1356.0	-	1356.0	-	1356.0	-
FLL	1123.4	3.1	1125.0	-	1125.0	-	1125.0	-	1125.0	-	1125.0	-
FLA	1181.2	3.4	1185.0	-	1184.4	1.2	1182.0	2.4	1182.0	2.4	1184.0	2.0
FLS	1183.4	2.0	1185.0	-	1184.4	1.2	1183.4	2.0	1183.4	2.0	1182.8	1.9
FAL	1279.6	0.8	1281.0	-	1281.0	-	1281.0	-	1281.0	-	1280.8	0.4
FAA	1280.2	0.4	1280.8	0.4	1281.0	0.4	1279.8	1.0	1279.8	1.0	1280.8	0.4
FAS	1280.0	1.1	1280.6	0.5	1281.0	0.5	1280.4	0.5	1280.4	0.5	1280.2	0.4
FSL	1330.0	0.6	1331.0	-	1331.0	-	1330.6	0.5	1330.6	0.5	1330.2	0.4
FSA	1330.8	0.4	1330.8	0.4	1331.0	0.4	1330.4	0.5	1330.4	0.5	1330.0	-
FSS	1330.6	0.5	1330.8	0.4	1331.0	0.4	1330.8	0.4	1330.8	0.4	1330.2	0.4

Table C.2: ALNS test for instance set F1.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-
MLA	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-
MLS	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-
MAL	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-
MAA	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-
MAS	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-
MSL	531.0	-	531.0	-	531.0	-	531.0	-	531.0	-	531.0	-
MSA	531.0	-	531.0	-	531.0	-	531.0	-	531.0	-	531.0	-
MSS	531.0	-	531.0	-	531.0	-	531.0	-	531.0	-	531.0	-
ALL	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-
ALA	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-
ALS	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-	345.0	-
AAL	481.4	0.8	482.2	1.0	482.6	0.8	482.2	1.0	482.2	1.0	481.4	0.8
AAA	482.2	1.0	483.0	-	482.6	0.8	482.2	1.0	482.2	1.0	481.8	1.0
AAS	481.4	0.8	481.8	1.0	483.0	-	481.4	0.8	481.4	0.8	482.2	1.0
ASL	518.0	-	518.0	-	518.0	-	518.0	-	518.0	-	518.0	-
ASA	518.0	-	518.0	-	518.0	-	518.0	-	518.0	-	518.0	-
ASS	518.0	-	518.0	-	518.0	-	518.0	-	518.0	-	518.0	-
FLL	320.0	-	309.6	20.4	320.0	-	319.0	2.0	319.0	2.0	320.0	-
FLA	328.8	0.4	329.0	-	329.0	-	328.8	0.4	328.8	0.4	329.0	-
FLS	329.0	-	329.0	-	329.0	-	329.0	-	329.0	-	329.0	-
FAL	455.6	2.9	458.0	-	458.0	-	458.0	-	458.0	-	458.0	-
FAA	455.6	2.9	458.0	-	458.0	-	458.0	-	458.0	-	458.0	-
FAS	456.8	2.4	458.0	-	458.0	-	455.4	2.7	455.4	2.7	458.0	-
FSL	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-	495.8	0.4
FSA	496.0	-	496.0	-	496.0	-	495.8	0.4	495.8	0.4	496.0	-
FSS	495.6	0.5	496.0	-	496.0	-	496.0	-	496.0	-	496.0	-

Table C.3: ALNS test for instance set C1.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	583.0	-	583.0	-	583.0	-	583.0	-	583.0	-	583.0	-
MLA	583.0	-	583.0	-	583.0	-	583.0	-	583.0	-	583.0	-
MLS	583.0	-	583.0	-	583.0	-	583.0	-	583.0	-	583.0	-
MAL	629.0	-	629.0	-	629.0	-	629.0	-	629.0	-	629.0	-
MAA	629.0	-	629.0	-	629.0	-	629.0	-	629.0	-	629.0	-
MAS	629.0	-	629.0	-	629.0	-	629.0	-	629.0	-	629.0	-
MSL	652.0	-	652.0	-	652.0	-	652.0	-	652.0	-	652.0	-
MSA	652.0	-	652.0	-	652.0	-	652.0	-	652.0	-	652.0	-
MSS	652.0	-	652.0	-	652.0	-	652.0	-	652.0	-	652.0	-
ALL	549.8	4.6	547.8	1.0	554.2	3.8	549.6	7.3	547.2	5.7	547.2	5.7
ALA	545.4	4.5	550.2	1.6	557.4	3.1	545.0	7.2	546.8	5.5	546.8	5.5
ALS	547.6	7.2	551.8	3.6	557.4	3.1	546.4	5.4	544.0	5.5	544.0	5.5
AAL	605.0	-	605.0	-	605.4	0.8	605.0	-	606.2	2.4	606.2	2.4
AAA	607.4	2.9	605.2	0.4	609.0	3.3	605.0	-	605.0	-	605.0	-
AAS	606.0	2.0	608.0	3.6	609.2	3.0	606.4	2.7	607.8	3.4	607.8	3.4
ASL	640.0	-	640.0	-	640.0	-	640.0	-	640.0	-	640.0	-
ASA	640.0	-	640.0	-	640.0	-	640.0	-	640.0	-	640.0	-
ASS	640.0	-	640.0	-	640.0	-	640.0	-	640.0	-	640.0	-
FLL	489.2	8.7	500.0	-	500.0	-	500.0	-	490.6	12.0	490.6	12.0
FLA	500.4	0.5	500.2	0.4	501.0	-	500.2	0.4	500.0	-	500.0	-
FLS	500.0	-	500.6	0.5	501.0	-	500.4	0.5	496.4	4.5	496.4	4.5
FAL	575.6	3.3	579.6	0.5	580.0	-	579.0	1.5	576.8	2.4	576.8	2.4
FAA	577.6	2.7	577.8	3.1	580.0	-	577.8	2.1	576.6	2.8	576.6	2.8
FAS	577.0	2.7	579.4	0.5	580.0	-	578.6	1.8	576.6	3.4	576.6	3.4
FSL	613.0	0.6	613.4	0.5	613.8	0.4	613.2	0.4	612.8	0.7	612.8	0.7
FSA	612.8	0.4	613.4	0.5	613.8	0.4	613.2	0.7	613.0	0.6	613.0	0.6
FSS	612.6	0.5	613.2	0.4	614.0	-	613.0	0.6	613.0	-	613.0	-

Table C.4: ALNS test for instance set M1.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	1062.0	-	1062.0	-	1062.0	-	1062.0	-	1062.0	-	1062.0	-
MLA	1062.0	-	1062.0	-	1062.0	-	1062.0	-	1062.0	-	1061.6	0.8
MLS	1061.6	0.8	1062.0	-	1062.0	-	1062.0	-	1062.0	-	1062.0	-
MAL	1091.8	0.4	1092.0	-	1092.0	-	1091.8	0.4	1091.8	0.4	1092.0	-
MAA	1091.6	0.5	1092.0	-	1092.0	-	1091.8	0.4	1091.8	0.4	1092.0	-
MAS	1091.6	0.5	1091.8	0.4	1092.0	-	1091.8	0.4	1091.8	0.4	1092.0	-
MSL	1106.0	-	1106.0	-	1106.0	-	1106.0	-	1106.0	-	1106.0	-
MSA	1106.0	-	1106.0	-	1106.0	-	1106.0	-	1106.0	-	1106.0	-
MSS	1106.0	-	1106.0	-	1106.0	-	1106.0	-	1106.0	-	1106.0	-
ALL	1066.6	0.8	1066.8	0.4	1067.0	-	1066.6	0.8	1066.6	0.8	1066.2	0.4
ALA	1067.0	-	1067.0	-	1067.0	-	1066.6	0.5	1066.6	0.5	1066.8	0.4
ALS	1065.8	1.1	1066.8	0.4	1067.0	-	1065.2	1.9	1065.2	1.9	1066.8	0.4
AAL	1091.6	0.5	1091.8	0.4	1092.0	-	1091.6	0.5	1091.6	0.5	1091.6	0.5
AAA	1091.4	0.5	1091.8	0.4	1092.0	-	1091.8	0.4	1091.8	0.4	1091.6	0.5
AAS	1091.2	0.4	1092.0	-	1092.0	-	1091.6	0.5	1091.6	0.5	1091.8	0.4
ASL	1102.8	0.4	1103.0	-	1103.0	-	1103.0	-	1103.0	-	1102.8	0.4
ASA	1102.8	0.4	1103.0	-	1103.0	-	1103.0	-	1103.0	-	1103.0	-
ASS	1102.8	0.4	1103.0	-	1103.0	-	1103.0	-	1103.0	-	1102.6	0.5
FLL	1056.0	0.9	1057.2	0.7	1057.6	0.5	1057.0	0.6	1057.0	0.6	1056.8	0.4
FLA	1056.6	1.0	1057.0	-	1057.0	-	1056.6	1.5	1056.6	1.5	1056.2	1.1
FLS	1055.8	1.6	1057.2	0.4	1057.4	0.5	1056.6	0.8	1056.6	0.8	1056.6	0.5
FAL	1080.6	0.8	1081.8	0.7	1083.0	-	1080.0	0.6	1080.0	0.6	1080.0	1.1
FAA	1080.2	1.1	1081.6	1.0	1082.4	0.8	1081.2	1.6	1081.2	1.6	1081.2	1.0
FAS	1081.2	1.8	1080.6	1.0	1081.8	1.4	1080.4	0.5	1080.4	0.5	1080.6	0.8
FSL	1092.8	0.4	1093.6	0.5	1094.0	-	1093.0	-	1093.0	-	1092.4	0.5
FSA	1092.8	0.7	1093.4	0.5	1094.0	-	1092.8	0.7	1092.8	0.7	1093.4	0.8
FSS	1092.6	0.5	1093.6	0.5	1093.6	0.5	1093.0	0.6	1093.0	0.6	1093.4	0.5

Table C.5: ALNS test for instance set R3.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	392.0	-	392.0	-	392.0	-	392.0	-	392.0	-	392.0	-
MLA	392.0	-	392.0	-	392.0	-	392.0	-	392.0	-	392.0	-
MLS	392.0	-	392.0	-	392.0	-	392.0	-	392.0	-	392.0	-
MAL	424.0	1.8	426.0	-	426.0	-	425.2	1.0	424.4	1.5	424.4	1.5
MAA	426.0	-	426.0	-	426.0	-	426.0	-	425.2	1.6	425.2	1.6
MAS	425.2	1.6	426.0	-	426.0	-	424.8	1.6	426.0	-	426.0	-
MSL	436.6	0.5	438.0	-	438.0	-	437.0	0.9	437.6	0.8	437.6	0.8
MSA	437.8	0.4	438.0	-	438.0	-	437.8	0.4	436.8	1.0	436.8	1.0
MSS	437.6	0.5	438.0	-	438.0	-	438.0	-	436.8	1.0	436.8	1.0
ALL	383.2	3.8	386.0	-	386.0	-	383.6	1.9	383.6	1.9	383.6	1.9
ALA	384.0	1.8	386.0	-	386.0	-	384.0	1.8	384.8	1.6	384.8	1.6
ALS	385.2	1.6	386.0	-	386.0	-	385.2	1.6	384.0	1.8	384.0	1.8
AAL	415.6	0.8	416.4	0.8	417.6	0.8	416.4	2.3	416.0	-	416.0	-
AAA	416.8	1.0	416.4	0.8	417.6	0.8	416.8	1.0	416.0	-	416.0	-
AAS	416.4	0.8	417.2	1.0	417.6	0.8	416.0	0.8	416.4	0.8	416.4	0.8
ASL	432.8	0.4	433.0	-	433.0	-	432.8	0.4	432.8	0.4	432.8	0.4
ASA	433.0	-	433.0	-	433.0	-	431.8	1.1	433.0	-	433.0	-
ASS	433.0	-	433.0	-	433.0	-	432.8	0.4	432.6	0.5	432.6	0.5
FLL	358.4	3.1	356.8	1.6	361.2	2.7	356.8	1.0	356.8	1.6	356.8	1.6
FLA	357.6	4.2	357.6	2.3	361.2	4.7	358.0	3.9	359.6	3.6	359.6	3.6
FLS	354.8	2.4	357.2	2.4	364.8	4.2	357.2	2.4	356.0	-	356.0	-
FAL	398.0	-	398.0	-	398.4	0.8	398.4	0.8	397.2	1.0	397.2	1.0
FAA	396.8	2.4	398.0	-	398.0	-	398.0	-	396.0	2.1	396.0	2.1
FAS	396.4	1.5	397.6	0.8	398.0	-	396.8	1.6	398.0	-	398.0	-
FSL	417.2	1.4	418.4	0.5	419.0	-	417.8	1.0	418.0	1.2	418.0	1.2
FSA	418.6	0.8	418.4	1.2	419.0	-	418.2	1.1	418.2	1.1	418.2	1.1
FSS	418.2	1.1	418.4	1.2	419.0	-	418.4	1.2	417.0	0.9	417.0	0.9

Table C.6: ALNS test for instance set M2.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	106.0	-	106.0	-	106.0	-	106.0	-	106.0	-	106.0	-
MLA	106.0	-	106.0	-	106.0	-	106.0	-	106.0	-	106.0	-
MLS	106.0	-	106.0	-	106.0	-	106.0	-	106.0	-	106.0	-
MAL	188.0	-	188.0	-	188.0	-	188.0	-	188.0	-	188.0	-
MAA	188.0	-	188.0	-	188.0	-	188.0	-	188.0	-	188.0	-
MAS	188.0	-	188.0	-	188.0	-	188.0	-	188.0	-	188.0	-
MSL	225.6	0.5	225.6	0.5	226.6	0.5	225.2	1.7	225.2	1.7	225.2	1.0
MSA	224.2	2.1	225.4	0.5	226.8	0.4	225.6	0.5	225.6	0.5	223.8	1.1
MSS	224.6	0.8	225.0	0.6	226.6	0.5	225.2	1.6	225.2	1.6	224.6	1.6
ALL	100.6	2.5	103.6	2.5	106.0	-	102.4	3.7	102.4	3.7	101.4	3.1
ALA	103.2	4.2	104.8	1.4	106.0	-	103.6	1.2	103.6	1.2	97.8	4.0
ALS	102.6	1.3	104.8	1.4	106.0	-	102.2	1.6	102.2	1.6	100.6	4.2
AAL	178.6	1.5	176.4	2.4	181.6	1.0	176.6	2.9	176.6	2.9	176.0	2.6
AAA	176.2	0.7	179.0	1.1	180.6	1.3	177.4	2.5	177.4	2.5	173.8	1.0
AAS	176.2	2.0	176.6	1.5	180.4	0.8	175.8	1.3	175.8	1.3	174.8	1.8
ASL	213.8	0.4	214.6	1.2	216.6	0.5	214.4	1.0	214.4	1.0	214.8	0.4
ASA	213.2	0.7	215.2	0.7	216.2	0.4	214.0	0.6	214.0	0.6	214.4	0.5
ASS	214.8	0.4	215.4	0.5	216.4	0.5	213.8	1.1	213.8	1.1	215.0	0.6
FLL	51.4	4.1	54.0	4.9	62.6	3.3	55.8	3.1	55.8	3.1	49.6	7.7
FLA	53.0	3.5	57.6	1.9	65.0	3.5	55.6	4.7	55.6	4.7	54.0	3.0
FLS	55.4	4.7	57.6	2.3	65.6	2.7	53.2	3.4	53.2	3.4	54.2	2.4
FAL	144.4	1.3	145.2	1.0	149.2	1.4	145.4	0.8	145.4	0.8	142.8	1.0
FAA	145.0	2.5	144.8	2.6	149.2	2.3	144.2	1.6	144.2	1.6	144.8	2.7
FAS	143.8	0.4	146.6	2.6	148.2	1.6	144.4	1.0	144.4	1.0	143.6	0.8
FSL	190.2	1.0	191.2	0.7	193.0	1.2	189.0	0.9	189.0	0.9	189.4	1.5
FSA	190.0	0.6	190.0	0.6	192.4	0.5	189.4	0.5	189.4	0.5	189.4	1.8
FSS	189.4	1.5	190.2	1.1	192.8	1.6	190.8	1.1	190.8	1.1	188.8	1.0

Table C.7: ALNS test for instance set C2.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	1189.0	8.3	1203.8	4.6	1213.8	4.6	1201.2	4.9	1196.4	4.9	1196.4	4.9
MLA	1198.0	6.7	1201.4	7.6	1210.8	2.2	1204.2	3.2	1184.6	13.7	1184.6	13.7
MLS	1199.2	4.0	1203.2	3.9	1209.8	2.4	1197.8	5.9	1197.4	4.5	1197.4	4.5
MAL	1283.0	1.2	1283.0	1.8	1285.0	-	1283.4	1.5	1282.4	0.8	1282.4	0.8
MAA	1283.2	1.6	1283.8	1.0	1285.0	-	1284.2	1.0	1282.8	1.6	1282.8	1.6
MAS	1281.8	1.9	1283.8	1.6	1285.0	-	1283.4	1.5	1282.4	0.8	1282.4	0.8
MSL	1304.6	0.5	1305.2	0.4	1306.8	0.4	1304.6	0.8	1304.4	0.5	1304.4	0.5
MSA	1305.0	0.6	1305.0	-	1307.0	-	1305.4	0.8	1304.6	0.5	1304.6	0.5
MSS	1305.2	0.4	1305.0	-	1307.0	-	1304.2	0.4	1304.4	0.5	1304.4	0.5
ALL	1164.2	13.8	1190.6	11.0	1203.4	4.7	1189.2	6.9	1172.0	-	1172.0	-
ALA	1161.8	4.8	1191.8	6.2	1203.0	6.2	1179.0	18.1	1176.2	4.8	1176.2	4.8
ALS	1165.0	10.8	1194.6	3.7	1202.4	3.7	1180.8	9.3	1172.8	1.6	1172.8	1.6
AAL	1282.2	1.4	1284.8	0.4	1285.0	-	1281.0	1.6	1280.8	1.0	1280.8	1.0
AAA	1281.4	1.9	1282.6	1.9	1284.6	0.8	1282.6	0.8	1281.0	0.6	1281.0	0.6
AAS	1280.4	0.8	1283.2	1.1	1285.0	-	1281.4	2.0	1281.0	1.1	1281.0	1.1
ASL	1301.4	0.5	1302.0	-	1302.8	0.4	1301.4	0.5	1300.8	0.7	1300.8	0.7
ASA	1300.8	0.7	1301.6	0.5	1303.0	0.6	1301.2	0.7	1301.0	0.9	1301.0	0.9
ASS	1301.2	0.4	1301.4	0.5	1303.2	0.4	1301.6	0.5	1300.8	0.4	1300.8	0.4
FLL	1103.6	5.2	1136.2	1.7	1153.2	3.2	1132.8	12.1	1114.8	15.9	1114.8	15.9
FLA	1106.8	12.7	1132.8	7.5	1152.6	6.9	1131.0	8.2	1110.6	5.1	1110.6	5.1
FLS	1100.2	5.1	1135.8	10.8	1156.0	5.4	1132.6	9.1	1114.2	9.1	1114.2	9.1
FAL	1216.0	5.8	1237.8	6.4	1257.0	7.4	1225.2	10.6	1219.4	2.7	1219.4	2.7
FAA	1216.8	7.9	1232.4	9.0	1256.4	4.1	1227.2	10.1	1220.4	5.1	1220.4	5.1
FAS	1218.4	8.1	1228.4	9.5	1253.4	8.6	1223.0	7.8	1219.0	2.9	1219.0	2.9
FSL	1289.4	0.5	1291.0	0.9	1293.6	0.8	1290.6	0.5	1289.0	0.9	1289.0	0.9
FSA	1290.2	1.9	1291.4	0.5	1293.8	0.7	1290.6	1.0	1289.8	1.6	1289.8	1.6
FSS	1289.8	1.4	1291.2	0.4	1294.2	0.7	1290.4	1.2	1289.4	1.0	1289.4	1.0

Table C.8: ALNS test for instance set M3.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	594.8	4.9	599.2	2.4	602.0	-	596.4	3.6	591.6	2.8		
MLA	590.2	5.4	599.6	1.9	602.0	-	595.6	4.7	587.6	1.8		
MLS	594.0	3.7	598.0	3.5	602.0	-	597.6	3.1	591.8	5.7		
MAL	632.8	2.6	636.8	1.1	639.6	0.5	633.2	1.6	633.2	2.5		
MAA	632.8	1.7	636.6	0.8	641.0	1.4	634.4	1.8	630.8	1.7		
MAS	633.6	2.2	636.0	1.6	640.8	0.4	634.0	2.0	631.8	1.0		
MSL	653.6	1.2	653.4	0.8	657.8	1.1	652.8	1.1	652.2	0.4		
MSA	653.0	0.9	655.6	1.2	657.8	0.7	653.4	0.5	653.0	0.6		
MSS	653.4	1.3	654.4	1.3	657.2	1.1	653.4	1.0	653.0	1.1		
ALL	567.6	2.9	576.4	6.8	583.4	1.2	575.2	3.9	580.2	0.4		
ALA	567.8	5.9	578.2	4.6	585.0	1.5	573.6	3.1	580.0	-		
ALS	567.6	4.7	573.2	5.1	582.2	1.8	570.2	2.4	580.0	-		
AAL	617.4	1.0	619.6	1.9	626.0	2.6	618.2	2.9	623.0	-		
AAA	616.0	1.1	619.4	1.9	629.2	1.6	618.0	2.2	623.0	-		
AAS	619.8	2.9	621.6	2.3	626.0	2.9	621.4	3.1	623.0	-		
ASL	642.6	0.8	644.0	1.6	649.2	1.3	645.0	2.0	649.0	-		
ASA	643.2	0.4	645.4	0.8	651.8	0.4	643.4	0.8	649.0	-		
ASS	644.0	2.6	645.0	0.6	651.2	1.1	644.0	2.6	649.0	-		
FLL	506.8	9.1	524.6	6.3	532.4	8.6	521.0	6.2	516.0	2.0		
FLA	504.2	7.6	523.6	4.5	532.6	5.0	510.4	10.6	516.0	2.0		
FLS	512.8	8.5	515.6	7.9	536.8	5.1	519.6	11.6	516.0	2.0		
FAL	579.4	6.1	577.2	3.6	587.6	2.4	579.4	5.7	583.2	3.1		
FAA	576.2	4.4	577.6	4.7	587.8	1.7	578.4	5.4	580.8	1.0		
FAS	578.6	3.3	580.0	2.7	588.0	2.5	577.8	3.2	581.0	1.2		
FSL	613.0	1.5	614.0	1.8	618.2	1.1	612.2	0.4	612.6	0.8		
FSA	613.2	1.1	614.0	1.1	617.2	1.1	613.2	1.6	612.0	-		
FSS	612.6	0.8	612.8	0.7	617.8	1.7	613.2	0.7	612.4	0.8		

Table C.9: ALNS test for instance set V1.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	590.4	2.4	597.6	4.6	603.0	-	593.8	3.2	590.4	1.8		
MLA	589.6	2.6	599.8	1.4	602.8	0.4	594.0	2.5	589.6	0.5		
MLS	590.4	1.6	594.0	2.8	602.6	0.8	592.0	4.2	590.4	1.8		
MAL	632.8	1.6	634.0	1.1	638.2	1.7	633.4	1.0	632.8	1.1		
MAA	632.4	1.4	633.4	1.3	638.0	1.1	631.6	1.0	632.4	0.5		
MAS	633.0	1.4	634.0	1.2	638.0	1.2	632.2	1.1	633.0	1.1		
MSL	654.2	0.5	653.8	0.4	656.0	0.9	653.6	0.5	654.2	0.4		
MSA	654.0	1.6	654.0	0.6	656.0	1.1	653.2	1.1	654.0	-		
MSS	654.4	0.5	653.2	0.4	656.8	0.7	653.2	1.0	654.4	0.8		
ALL	580.2	3.8	573.4	2.2	587.8	3.7	570.0	3.6	580.2	0.4		
ALA	580.0	6.0	576.0	3.5	583.4	3.7	570.4	4.7	580.0	-		
ALS	580.0	5.6	573.8	3.8	588.4	1.3	571.0	4.2	580.0	-		
AAL	620.0	3.1	616.0	2.1	623.2	2.3	618.4	3.5	620.0	-		
AAA	620.0	2.8	617.2	0.7	625.8	2.3	617.8	1.9	620.0	-		
AAS	620.4	0.7	618.2	0.7	623.8	2.6	616.8	1.7	620.4	0.8		
ASL	647.0	0.6	641.8	1.8	649.8	2.4	641.6	2.8	647.0	-		
ASA	647.0	1.4	642.6	0.8	649.4	1.8	641.2	1.0	647.0	-		
ASS	647.0	1.7	644.0	1.2	650.0	1.4	642.4	2.2	647.0	-		
FLL	525.0	7.9	530.0	2.6	537.6	5.4	510.2	8.3	525.0	-		
FLA	526.8	5.8	512.4	5.0	537.8	5.3	514.6	3.9	526.8	3.5		
FLS	527.6	5.4	510.8	6.6	533.4	6.8	515.4	8.3	527.6	5.1		
FAL	583.6	-	583.0	2.0	587.2	4.7	583.2	2.4	583.6	0.8		
FAA	583.0	1.2	583.4	2.7	588.0	3.0	582.0	-	583.0	-		
FAS	583.8	3.6	582.6	0.8	586.2	3.5	587.0	6.1	583.8	1.0		
FSL	621.0	-	621.0	-	621.0	-	621.0	-	621.0	-		
FSA	621.0	-	621.0	-	621.2	0.4	621.0	-	621.0	-		
FSS	621.0	-	621.0	-	621.8	1.6	621.0	-	621.0	-		

Table C.10: ALNS test for instance set V2.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	463.4	5.8	470.0	1.7	470.0	—	468.6	1.3	464.4	5.8		
MLA	463.0	3.3	469.8	4.9	470.0	—	469.2	1.1	465.0	3.3		
MLS	466.6	4.5	469.8	3.1	470.0	—	468.0	1.8	463.6	4.5		
MAL	507.2	3.1	513.2	3.4	518.2	1.7	512.0	0.6	510.6	3.1		
MAA	507.8	2.7	512.0	1.9	518.4	1.0	509.8	2.7	509.0	2.7		
MAS	509.0	3.6	513.2	3.3	518.2	1.1	511.0	3.9	508.0	3.6		
MSL	530.6	1.1	531.2	2.2	535.8	1.1	530.6	1.0	530.8	1.1		
MSA	530.6	1.7	533.0	0.8	535.8	1.1	531.6	1.7	531.4	1.7		
MSS	530.4	0.7	531.2	1.5	536.8	0.4	532.6	1.3	529.2	0.7		
ALL	433.4	1.2	451.6	3.1	461.0	3.3	451.2	5.6	438.0	1.2		
ALA	443.2	3.1	448.4	6.7	461.6	3.7	442.0	7.3	440.2	3.1		
ALS	436.2	5.2	446.8	5.1	464.0	5.1	450.8	2.9	442.0	5.2		
AAL	492.4	—	493.6	2.7	502.4	2.3	494.8	2.4	496.0	—		
AAA	491.4	—	492.2	0.8	502.4	2.5	493.8	1.4	496.0	—		
AAS	492.4	—	496.4	1.0	500.0	1.1	494.6	2.3	496.0	—		
ASL	523.0	—	523.0	—	523.8	0.7	523.6	1.2	525.0	—		
ASA	523.0	—	523.0	—	524.6	0.8	523.0	—	525.0	—		
ASS	523.0	0.8	523.0	—	524.4	0.8	523.0	—	525.4	0.8		
FLL	387.8	5.5	395.2	5.6	414.2	5.5	384.0	1.5	385.8	5.5		
FLA	392.0	0.4	389.8	11.8	410.4	9.1	386.2	4.9	383.2	0.4		
FLS	383.0	8.5	388.2	—	406.6	4.4	387.8	5.1	389.2	8.5		
FAL	456.0	3.1	454.2	3.5	457.8	2.2	457.8	3.3	463.6	3.1		
FAA	454.8	0.4	458.4	1.0	460.6	3.8	455.0	2.0	462.2	0.4		
FAS	454.4	—	455.8	0.8	461.0	1.5	457.2	3.8	462.0	—		
FSL	489.6	—	489.2	1.2	493.4	1.6	490.4	1.7	498.0	—		
FSA	490.2	—	493.2	1.6	494.2	2.4	489.0	—	498.0	—		
FSS	489.6	—	489.6	1.2	493.2	0.7	494.0	3.7	498.0	—		

Table C.11: ALNS test for instance set V3.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	464.6	3.4	469.8	0.4	470.0	-	467.8	3.1	462.4	1.6		
MLA	463.4	1.3	467.6	2.5	470.0	-	467.4	2.5	462.4	1.9		
MLS	460.6	2.7	469.4	1.2	470.0	-	468.0	1.9	462.2	2.4		
MAL	507.0	2.8	509.6	2.3	518.6	1.0	510.0	1.1	503.8	1.0		
MAA	506.8	1.3	507.4	1.5	519.4	0.5	509.0	1.1	504.0	0.6		
MAS	506.8	2.9	509.4	1.2	519.2	0.7	507.2	3.0	506.6	2.2		
MSL	530.4	1.6	529.4	1.3	534.4	0.8	531.4	0.5	530.2	1.6		
MSA	530.8	0.7	531.0	1.2	534.8	1.1	530.2	1.3	529.6	1.3		
MSS	529.4	1.0	530.4	0.8	535.8	1.4	532.0	1.9	528.6	1.5		
ALL	440.0	2.7	445.4	3.9	461.0	4.1	443.8	3.0	440.2	1.6		
ALA	434.2	7.1	447.8	2.4	462.0	3.7	445.6	5.1	440.4	1.7		
ALS	439.4	3.6	448.8	1.6	462.4	1.2	443.2	4.0	439.6	0.8		
AAL	487.4	2.2	492.8	2.2	502.6	1.5	492.6	1.8	497.0	-		
AAA	488.0	4.2	493.8	2.8	502.4	1.0	492.8	3.2	497.0	-		
AAS	486.6	1.8	492.4	3.1	501.6	1.8	494.2	3.7	497.0	-		
ASL	517.2	1.4	520.4	0.8	523.0	1.1	519.6	1.0	526.0	-		
ASA	517.4	0.8	518.0	0.9	525.6	1.5	519.2	1.1	526.0	-		
ASS	517.2	1.6	519.6	1.2	525.4	2.3	519.4	2.1	526.0	-		
FLL	375.0	8.4	398.2	6.8	414.8	5.0	393.4	6.0	412.0	-		
FLA	385.2	12.1	399.2	6.1	415.6	7.4	393.2	4.2	412.6	1.2		
FLS	384.6	3.5	388.6	5.7	416.8	3.4	397.4	4.7	412.0	-		
FAL	451.8	6.3	453.4	2.0	462.0	3.0	454.4	8.0	464.8	1.6		
FAA	446.6	4.0	453.4	4.0	461.6	1.7	447.0	1.9	464.0	-		
FAS	453.4	5.2	452.2	2.2	461.8	3.9	457.8	4.9	464.0	-		
FSL	488.6	5.7	490.0	3.5	493.2	3.4	489.0	3.3	497.0	-		
FSA	487.8	3.1	489.0	3.2	492.6	1.3	487.8	1.6	497.0	-		
FSS	490.0	2.6	490.4	4.1	493.2	2.0	490.6	2.0	497.2	0.4		

Table C.12: ALNS test for instance set V4.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	464.8	1.6	468.8	1.4	470.0	-	467.8	2.4	462.4	-	-	-
MLA	464.2	0.4	468.2	1.7	470.0	-	467.4	2.0	462.4	-	1.6	1.6
MLS	465.6	2.3	470.0	-	470.0	-	468.0	2.4	462.2	-	0.8	0.8
MAL	509.4	0.5	509.2	0.4	518.6	1.4	510.0	1.2	503.8	-	-	-
MAA	509.6	1.2	509.4	0.5	519.4	1.0	509.0	1.1	504.0	-	-	-
MAS	509.2	0.4	509.4	0.8	519.2	1.4	507.2	1.3	506.6	-	1.4	1.4
MSL	532.0	-	532.0	-	534.4	1.1	531.4	0.4	530.2	-	-	-
MSA	532.0	-	532.0	-	534.8	0.7	530.2	-	529.6	-	0.4	0.4
MSS	532.0	-	532.2	0.4	535.8	1.1	532.0	-	528.6	-	-	-
ALL	439.8	4.9	446.8	3.0	461.0	0.7	443.8	4.8	440.2	-	3.9	3.9
ALA	439.6	3.8	447.8	3.8	462.0	2.6	445.6	5.5	440.4	-	-	-
ALS	437.2	4.8	449.0	4.9	462.4	1.4	443.2	4.6	439.6	-	5.3	5.3
AAL	490.2	4.0	493.8	1.6	502.6	1.6	492.6	1.9	497.0	-	-	-
AAA	488.4	1.9	492.8	2.1	502.4	2.0	492.8	1.3	497.0	-	-	-
AAS	489.0	3.2	492.2	2.2	501.6	1.1	494.2	4.4	497.0	-	-	-
ASL	518.0	1.6	518.0	0.6	523.0	0.7	519.6	3.1	526.0	-	-	-
ASA	518.6	2.0	519.8	1.0	525.6	1.1	519.2	2.0	526.0	-	-	-
ASS	519.8	1.7	520.0	1.2	525.4	2.0	519.4	1.3	526.0	-	-	-
FLL	390.0	14.2	392.8	5.6	414.8	2.9	393.4	13.5	412.0	-	0.4	0.4
FLA	382.2	6.1	392.4	1.9	415.6	4.8	393.2	7.0	412.6	-	-	-
FLS	376.8	5.7	391.6	1.7	416.8	2.5	397.4	5.1	412.0	-	3.9	3.9
FAL	459.4	8.8	454.8	6.6	462.0	2.1	454.4	10.0	464.8	-	-	-
FAA	454.4	6.5	452.8	3.0	461.6	1.7	447.0	6.4	464.0	-	-	-
FAS	453.8	10.7	454.6	2.1	461.8	2.7	457.8	5.2	464.0	-	-	-
FSL	493.2	1.9	495.6	2.9	493.2	2.4	489.0	3.5	497.0	-	-	-
FSA	493.6	3.1	492.0	-	492.6	3.4	487.8	4.3	497.0	-	-	-
FSS	494.4	2.3	494.6	3.1	493.2	1.3	490.6	2.9	497.2	-	-	-

Table C.13: ALNS test for instance set V5.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	480.2	0.7	480.6	0.8	481.0	-	480.2	1.1	478.4	1.5		
MLA	478.6	0.8	481.0	-	481.0	-	480.8	0.4	479.4	1.3		
MLS	479.2	1.7	481.0	-	481.0	-	480.4	0.8	479.4	1.5		
MAL	524.6	0.8	525.4	0.8	527.8	0.4	525.0	1.4	524.0	1.1		
MAA	524.6	0.5	525.8	0.4	529.4	0.8	525.8	0.7	524.4	0.5		
MAS	524.0	-	524.8	0.4	528.0	0.6	524.4	0.5	525.8	0.7		
MSL	547.0	-	547.2	0.4	548.8	0.4	547.2	0.4	547.0	-		
MSA	547.0	-	547.0	-	548.4	0.5	547.2	0.4	547.2	0.4		
MSS	547.0	-	547.0	-	549.0	1.1	547.6	0.5	547.0	-		
ALL	468.0	-	470.2	1.1	476.2	0.7	471.2	3.2	468.6	1.2		
ALA	469.0	1.2	470.2	1.1	476.4	0.5	469.6	2.0	468.0	-		
ALS	469.0	1.2	471.0	2.9	476.0	2.5	470.8	1.9	469.6	2.0		
AAL	519.0	-	519.0	-	522.2	1.4	519.0	-	519.0	-		
AAA	519.0	-	519.2	0.4	521.0	1.9	519.0	-	519.0	-		
AAS	519.0	-	519.0	-	519.2	0.4	519.0	-	519.0	-		
ASL	545.0	-	545.0	-	545.0	-	545.0	-	545.0	-		
ASA	545.0	-	545.0	-	545.0	-	545.0	-	545.0	-		
ASS	545.0	-	545.0	-	545.0	-	545.0	-	545.0	-		
FLL	441.4	8.4	445.4	1.6	456.4	3.4	443.2	3.0	457.0	-		
FLA	436.8	10.3	450.6	5.9	459.6	3.3	451.2	5.6	457.0	-		
FLS	435.6	6.4	448.0	7.7	458.6	2.8	448.6	6.0	457.0	-		
FAL	495.8	5.3	500.6	3.8	503.0	0.6	497.4	4.7	498.8	1.6		
FAA	490.6	4.5	495.6	4.4	501.6	2.0	490.4	3.3	499.0	2.0		
FAS	493.8	4.4	494.0	3.5	502.0	2.7	493.6	6.0	499.2	1.9		
FSL	523.0	-	524.2	1.6	524.8	0.7	524.8	1.6	523.8	1.6		
FSA	523.0	-	524.6	1.9	524.4	1.2	524.2	1.6	524.2	1.4		
FSS	524.8	1.4	523.8	1.1	524.4	1.3	523.8	1.6	523.2	0.4		

Table C.14: ALNS test for instance set V11.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	463.0	-	463.0	-	463.0	-	463.0	-	463.0	-	478.4	-
MLA	463.0	-	463.0	-	463.0	-	463.0	-	463.0	-	479.4	-
MLS	463.0	-	463.0	-	463.0	-	463.0	-	463.0	-	479.4	-
MAL	518.0	2.1	518.4	0.8	521.0	0.8	518.8	1.1	518.8	1.1	524.0	1.8
MAA	517.4	0.8	518.2	0.4	521.0	0.4	518.4	1.3	518.4	1.3	524.4	0.8
MAS	516.8	1.0	518.2	0.4	521.0	0.4	518.4	1.3	518.4	1.3	525.8	0.4
MSL	543.2	0.4	543.2	0.4	548.4	0.4	544.4	0.8	544.4	0.8	547.0	-
MSA	544.0	1.5	545.4	1.2	548.4	1.2	544.0	0.8	544.0	0.8	547.2	0.8
MSS	543.6	0.5	544.0	-	547.4	-	543.6	0.8	543.6	0.8	547.0	0.5
ALL	446.4	0.8	452.6	0.8	463.0	0.8	454.2	4.9	454.2	4.9	468.6	0.4
ALA	447.4	2.7	455.4	1.9	463.0	1.9	454.8	2.7	454.8	2.7	468.0	-
ALS	448.0	2.7	453.2	3.0	463.0	3.0	450.4	5.4	450.4	5.4	469.6	-
AAL	508.0	-	510.8	2.9	513.0	2.9	509.2	1.4	509.2	1.4	519.0	-
AAA	508.0	-	508.0	-	515.4	-	508.0	2.8	508.0	2.8	519.0	1.2
AAS	508.0	-	509.8	1.4	514.8	1.4	508.0	3.1	508.0	3.1	519.0	-
ASL	539.0	-	539.0	-	539.6	-	539.0	0.5	539.0	0.5	545.0	-
ASA	539.0	-	539.0	-	539.0	-	539.0	-	539.0	-	545.0	-
ASS	539.0	-	539.0	-	539.2	-	539.0	0.4	539.0	0.4	545.0	-
FLL	403.6	11.9	416.6	8.9	440.8	8.9	425.4	2.7	425.4	2.7	457.0	-
FLA	387.2	7.5	425.2	5.1	437.4	5.1	411.6	4.6	411.6	4.6	457.0	-
FLS	392.0	12.8	431.6	6.3	439.8	6.3	417.8	4.4	417.8	4.4	457.0	-
FAL	463.4	4.4	477.8	2.4	488.6	2.4	476.8	4.2	476.8	4.2	498.8	-
FAA	468.6	6.6	475.8	3.6	490.8	3.6	480.2	3.1	480.2	3.1	499.0	-
FAS	468.8	5.4	485.4	5.9	487.2	5.9	475.8	5.0	475.8	5.0	499.2	-
FSL	510.4	2.7	512.0	5.0	515.4	5.0	511.0	1.6	511.0	1.6	523.8	-
FSA	513.6	5.5	511.8	1.3	514.4	1.3	510.4	1.2	510.4	1.2	524.2	-
FSS	511.0	2.2	515.4	3.8	515.0	3.8	510.4	3.9	510.4	3.9	523.2	-

Table C.15: ALNS test for instance set V12.

Instance Variant	500		1000		2000		500		500		500	
	Iterations	Half-width	Iterations	Half-width	Iterations	Half-width	Iterations (Faster Cooling)	Half-width	Iterations (Faster Cooling)	Half-width	Iterations with Heuristic	Half-width
MLL	1267.8	4.6	1275.0	6.4	1291.8	7.0	1273.0	5.1	1273.0	5.1	1272.2	1.0
MLA	1267.8	3.4	1272.4	2.7	1291.4	6.8	1273.4	4.7	1273.4	4.7	1271.4	0.8
MLS	1268.8	3.1	1273.2	5.3	1291.4	3.4	1269.8	2.3	1269.8	2.3	1271.0	-
MAL	1329.6	2.1	1327.8	0.4	1333.8	1.1	1330.4	2.0	1330.4	2.0	1333.6	0.8
MAA	1327.8	1.0	1330.8	1.9	1335.6	1.3	1329.6	1.7	1329.6	1.7	1333.4	0.5
MAS	1329.6	2.4	1331.8	3.0	1334.0	2.4	1329.4	2.5	1329.4	2.5	1333.2	0.4
MSL	1361.8	1.1	1361.0	-	1362.4	1.0	1361.6	1.2	1361.6	1.2	1364.0	-
MSA	1361.2	0.4	1362.6	1.9	1363.2	0.7	1361.0	-	1361.0	-	1364.0	-
MSS	1361.2	0.4	1361.6	0.5	1363.4	0.8	1361.4	0.5	1361.4	0.5	1364.0	-
ALL	1235.6	4.9	1250.2	5.3	1270.4	3.9	1241.4	3.9	1241.4	3.9	1245.0	5.0
ALA	1256.0	-	1260.4	2.2	1271.2	2.3	1258.0	3.0	1258.0	3.0	1256.0	-
ALS	1256.0	-	1256.4	0.8	1269.6	2.3	1256.4	0.8	1256.4	0.8	1256.2	0.4
AAL	1318.0	-	1318.0	-	1321.6	2.4	1318.0	-	1318.0	-	1318.2	0.4
AAA	1318.0	-	1318.4	0.5	1321.0	1.1	1318.2	0.4	1318.2	0.4	1318.0	-
AAS	1318.0	-	1318.0	-	1320.2	1.4	1318.0	-	1318.0	-	1318.0	-
ASL	1349.0	-	1349.0	-	1350.0	0.6	1349.2	0.4	1349.2	0.4	1349.0	-
ASA	1349.0	-	1349.0	-	1349.8	1.0	1349.0	-	1349.0	-	1349.0	-
ASS	1349.0	-	1349.0	-	1350.4	1.0	1349.2	0.4	1349.2	0.4	1349.0	-
FLL	1057.0	35.7	1099.4	18.8	1148.8	13.7	1063.4	31.8	1063.4	31.8	1049.6	30.8
FLA	1172.0	5.1	1169.2	1.4	1177.8	7.9	1169.4	2.7	1169.4	2.7	1180.2	0.4
FLS	1171.8	5.7	1170.6	3.1	1180.4	7.1	1169.2	2.4	1169.2	2.4	1180.4	0.8
FAL	1244.6	1.2	1249.6	4.8	1249.6	4.7	1246.2	4.3	1246.2	4.3	1244.0	-
FAA	1245.4	2.7	1244.4	0.8	1251.4	4.4	1244.4	0.8	1244.4	0.8	1244.6	1.2
FAS	1245.6	3.1	1244.0	-	1244.6	0.8	1245.0	1.2	1245.0	1.2	1248.2	3.6
FSL	1282.8	1.6	1282.4	0.5	1283.2	1.0	1282.2	0.4	1282.2	0.4	1286.4	0.8
FSA	1283.4	1.9	1282.0	-	1283.8	1.3	1284.8	3.8	1284.8	3.8	1287.0	2.0
FSS	1285.0	2.6	1283.2	-	1283.8	1.3	1282.0	-	1282.0	-	1286.0	-

Table C.16: ALNS test for instance set M4.

Bibliography

- [1] Aaby, K., Abbey, R.L., Herrmann, J.W., Treadwell, M., Jordan, C.S., and Wood, K. (2006) Embracing computer modeling to address pandemic influenza in the 21st century. *Journal of Public Health Management and Practice*, 365-372.
- [2] Aaby, K., Herrmann, J.W., Jordan, C.S., Treadwell, M., and Wood, K. (2006) Montgomery County's public health service uses operations research to plan emergency mass dispensing and vaccination clinics. *Interfaces*, **36.6**, 569-579.
- [3] Baita, F., Ukovich, W., Pesenti, R., and Favaretto, D. (1998) Dynamic routing-and-inventory problems: A review. *Transportation Research*, **32**, 585-598.
- [4] Ball, M.O., and Lin, F.L. (1993) A reliability model applied to emergency service vehicle location. *Operations Research*, **41**, 18-36.
- [5] Beasley, J.E. OR-Library, (2009) Brunel University, 5 Aug. 2009, <<http://people.brunel.ac.uk/~mastjjb/jeb/info.html>>.
- [6] Bramel, J., and Simchi-Levi, D. (1997) *The Logic of Logistics*, Springer, New York, NY.
- [7] Brandeau, M.L., McCoy, J.H., Hupert, N., Holty, J.E., and Bravata, D.M. (2009) Recommendations for modeling disaster responses in public health and medicine: a position paper of the society for medical decision making. *Medical Decision Making*, 1-23.
- [8] Campbell, A., Clarke, L., Kleywegt, A.J., and Savelsbergh, M.W.P. (1998) The inventory routing problem in *Fleet Management and Logistics*, Crainic, T.G., and Laporte, G. (eds), Kluwer Academic Publishers, Dordrecht, The Netherlands, Chapter 4.
- [9] Campbell, A.M., Clarke, L.W., and Savelsbergh, M.W.P. (2001) *The Vehicle Routing Problem*, Society for Industrial and Applied Mathematics, Philadelphia, PA.
- [10] Ceyhun, A., Selim, H., and Ozkarahan, I. (2007) A fuzzy multi-objective covering-based vehicle location model for emergency services. *Computers and Operations Research*, **34**, 705-726.
- [11] Chen, Z.-L. (2008) Integrated production and outbound distribution scheduling: Review and extensions. To appear in *Operations Research*.
- [12] Chen, Z.L., and Powell, W.B. (1999) Solving Parallel Machine Scheduling Problems by Column Generation. *INFORMS Journal on Computing*, **11**, 78-94.
- [13] Chen, Z.L., and Xu, H. (2006) Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows. *Transportation Science*, **40**, 74-88.

- [14] Daskin, M.S., and Stern, E.H. (1981) Hierarchical objective set covering model for emergency medical service vehicle deployment. *Transportation Science*, **15**, 137-152.
- [15] Dror, M., Ball, M., and Golden, B. (1985) Computational comparisons of algorithms for the inventory routing problem. *Annals of Operations Research*, **4**, 3-23.
- [16] Eglese, R.W. (1990) Simulated Annealing: A tool for operational research. *European Journal of Operational Research*, **46**, 271-281.
- [17] Gillett, B.E., and Miller, L.R. (1974) A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, **22.2**, 340-349.
- [18] Haghani, A., Tian, Q., and Hu, H. (2004) Simulation model for real-time emergency vehicle dispatching and routing. *Transportation Research Record*, no. 1882, 176-183.
- [19] Herrmann, J.W., Lu, S., and Schalliol, K. (2009) A routing and scheduling approach for planning medication distribution. *Proceedings of the 2009 Industrial Engineering Research Conference*.
- [20] Herrmann, J.W., Lu, S., and Schalliol, K. (2009) Delivery volume improvement for planning medication distribution. *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*.
- [21] Hupert, N., Wattson, D., Cuomo, J., Hollingsworth, E., and Neukermans, K. (2009) Predicting hospital surge after a large-scale anthrax attack: a model-based analysis of CDC's cities readiness initiative prophylaxis recommendations. *Medical Decision Making*, **29**, 424-437.
- [22] Lee, E.K., Chen, C.H., Pietz, F., and Benecke, B. (2009) Modeling and optimizing the public health infrastructure for emergency response. *Interfaces*, **3.5**, 476-490.
- [23] Mertens, S. Improving Solutions. (1999) Otto-von-Guericke University, 21 July 2009, <<http://www-e.uni-magdeburg.de/mertens/TSP/node3.html>>.
- [24] Moin, N.H., and Salhi, S. (2007) Inventory routing problems: A logistical overview. *Journal of the Operational Research Society*, **58**, 1185-1194.
- [25] Sivanandan, R., Hobeika, A.G., Ardekani, S.A., and Lockwood, P.B. (1988) Heuristic shortest-path method for emergency vehicle assignment - a study on the Mexico City network. *Transportation Research Record*, no. 1168.
- [26] Shaw, P., (1997) A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems. Technical Report, Department of Computer Science, University of Strathclyde.

- [27] Toth, P., and Vigo D. (2001), An overview of vehicle routing problems in *The Vehicle Routing Problem*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1-26.

- [28] Ropke, P., and Pisinger, D., (2005) An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, **40.4**, 455-472.

- [29] Weintraub, A., Aboud, J., Fernandez, C., Laporte, G., and Ramirez, E. (1999) An emergency vehicle dispatching system for an electric utility in Chile. *Journal of the Operational Research Society*, **50**, 690-696.