ABSTRACT

Title of Document:	SOLVING CONTINUOUS REPLENISHMENT INVENTORY ROUTING PROBLEMS
	Samuel Fomunyam Fomundam, Master of Science, 2008
Directed By:	Dr. Jeffrey Herrmann, Department of Mechanical Engineering and Institute of Systems Research

This research investigates the problem of resupplying points of dispensing (PODs), which will dispense medications to millions of people in case of a bioterrorist attack such as anthrax. After receiving an initial but limited supply of medication, the PODs will operate continuously. Vehicles will resupply the PODs continuously from a central depot that has a stockpile of medication. Each vehicle will repeatedly follow the same route and will deliver at each POD enough medication to replace what was consumed since the last visit. Because the number of drivers and trucks may be limited during an emergency, we wish to minimize the number of vehicles used to resupply the PODs. This thesis presents heuristics and a branch-and-bound approach for solving this NP-hard problem and evaluates their performance. We also analyze a special case in which all of the PODs have the same demand.

SOLVING CONTINUOUS REPLENISHMENT INVENTORY ROUTING PROBLEMS

By

Samuel Fomunyam Fomundam

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
2008

Acknowledgements

My deep appreciation goes to Dr. Jeffrey Herrmann for his guidance, patience and the meticulous nature with which he aided me throughout this research. He went beyond the call of duty as my advisor to help make this work successful.

I also deeply appreciate my wife and daughter who have been very supportive and patient with me during the long hours I spent on this work.

Table of Contents

List of Tables

List of Figures

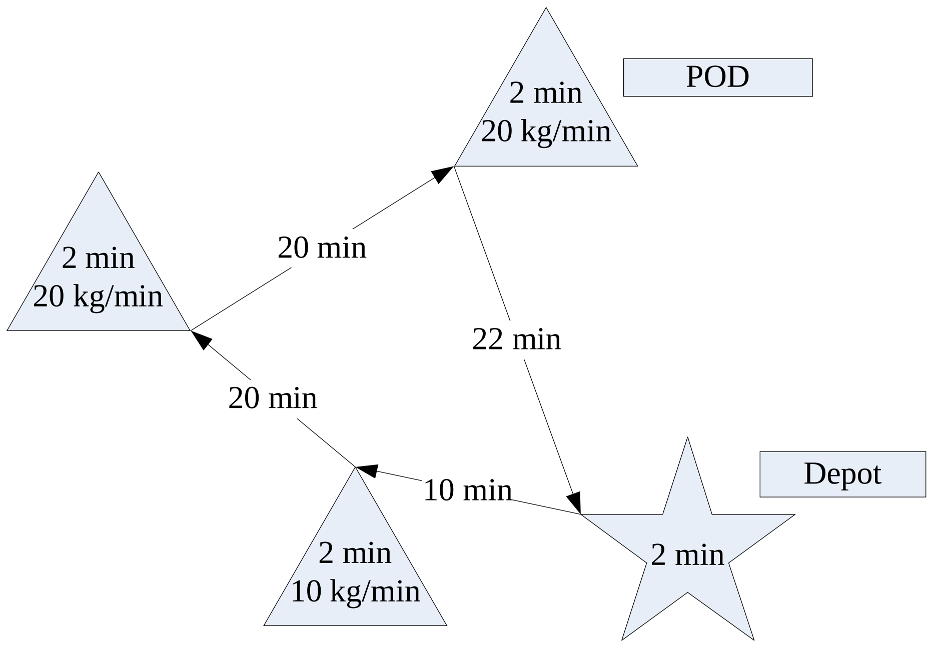
Chapter 1: INTRODUCTION

This research is motivated by work with public health officials who must plan the logistics for resupplying points of dispensing (PODs), which will dispense medications to the public in case of a health crisis or emergency that requires immediately deploying medication or other medical supplies to the population. An example of such an incident is a bioterrorist anthrax attack or a dangerous strain of the flu which requires that the entire populations of cities travel to and receive emergency vaccination at certain predetermined points.

Once the decision has been made to supply various PODs from the depot, the first problem is getting an initial quantity of medical supplies to the PODs. After this happens and the PODs are up and running, public health officials must determine the best way to continuously resupply these PODs so that their supplies do not run out. This thesis addresses the second part of this problem: the PODs are operating and require a steady stream of supplies. Suppose that during this second phase, supplies run out at PODs. This would result in POD staff and their supporting equipment being idle and thus the POD would be operating below its capacity. In an emergency, this would be an unwise use of scarce resources. Also, because queues would continue to build up while people are not served, this could also result in anxiety in the population and a lack of confidence in the competence and preparedness of the public health system in handling emergencies.

Vehicles will resupply the PODs continuously from a central depot that has a stockpile of medication. This is in contrast with existing applications which typically supply customers once a day and return to resupply the next day or after a few days. Given the vast quantities of medical supplies needed by an entire population in the case of an emergency, one truck load per day may just not be sufficient. Instead, each vehicle repeatedly follows the same route, starting out as soon as it can after returning to the depot. The capacity of the vehicles is given. At each site, the vehicle delivers enough medication to replace what was consumed since the last visit and which must last until the next visit. Figure 1 shows an example of the quantities that a vehicle must deliver at each POD on its route. The inventory capacity at each POD is not constrained. The amount that is delivered to any site is thus limited only by the capacity of the vehicle, which must supply other PODs as well. The inventory capacity needed by each POD may thus be obtained after solving the problem.

Figure 1: Meeting a route's demand rate



Demand per POD= 10, 20 & 20 kg/min
Route duration = 80 min
Required delivery quantities: 80 x [10,20,20]
=800, 1600 & 1600 kg
Required truck capacity
=sum of delivery quantities
or route demand(50) x duration(80)
=4000 kg

1.1 Parameters

We assume that the following information is given:

- Truck capacity
- Travel times: time it takes to travel between each pair of PODs or any POD and the depot. This would depend on the vehicle speed and the length of routes between PODs and/or the depot.

- Demand rates: the rate at which supplies are consumed at each POD. This is subject to the nature of the medical emergency and the capacity of dispensing operations (for example, available staffing for dispensing operations). It is prescribed by health planners.
- Load times: the time required to load the truck at the depot and unload it at each POD.
- Maximum route duration: the maximum allowed duration for a vehicle to complete its route.

1.2 Variables

These variables represent problem parameters which we are free to manipulate to obtain solutions to the problem:

- Assignment of PODs to vehicles: determination of which PODs each vehicle would supply.
- Routing: determination of the path the vehicle would take to load at the depot and unload at the PODs.

1.3 Constraints

These are the problem constraints:

- Each POD is supplied by only one vehicle.
- The route duration must not exceed the prescribed maximum.
- Vehicle supply rate at PODs (quantity to route duration ratio) may not be less than the prescribed POD demand rates.

- The quantity loaded onto the vehicle at the depot may not exceed the vehicle capacity.

1.4 <u>Objective Function</u>

The number of vehicles: Because the number of drivers and trucks may be limited during an emergency, the principal objective in this study is to minimize the number of vehicles. The cost of supplying the PODs also depends on the number of vehicles and drivers.

1.5 <u>Problem Scope</u>

This thesis investigates a single-depot, single-product, deterministic, symmetric, steady-state problem. The quantity of medical supplies is treated as a continuous variable. Symmetry means that the time required to travel between a pair of PODs or a POD and depot is independent of which of the pair is the start point.

This problem, which we call the continuous-replenishment inventory routing problem (CRIRP), is a new type of inventory routing problem, which is known to be NP hard. This study develops and investigates the CRIRP problem and presents the results of heuristic solutions and exact solutions that use branch and bound and pruning techniques.

Chapter 2: LITERATURE SURVEY

Previous research has added a great deal to our knowledge about vehicle routing problems in general and inventory routing problems in particular. A complete review of this literature is beyond the scope of this thesis. However, there are several problems that have been studied which may be comparable to or useful in solving the CRIRP.

2.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) seeks to minimize the total distance of the route that is followed by a salesman who must visit a number of cities exactly once, before returning home (Lenstra, 1977). The intercity distances, c_{ij} between each pair of cities, from city i to city j are given. If $c_{ij} = c_{ji}$, it is a symmetric TSP; otherwise, it is asymmetric. Because the total distance depends on the order in which the cities are visited, the number of possible routes amongst which to choose the shortest is $n!$, where n is the number of cities.

Because the TSP is NP-hard, various heuristic approaches have been developed. Golden *et al.* (1980) present the nearest neighbor algorithm of the TSP, which can have any node as its start point. The closest node to the last one previously added to the route is appended. This procedure continues until all nodes are in the route at which time the last and first nodes are connected to complete the route.

2.2 Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) is a specific instance of the Vehicle Routing Problem (VRP). The VRP seeks the optimal routing for a fleet of vehicles (which may be homogenous or heterogeneous) to supply a set of customers from one or more depots (Toth and Vigo, 1998). A wide array of constraints (for example, vehicle capacity limits, route length limit, supplying customers within certain time windows, and customer supply precedence relationships) may limit route construction.

In the CVRP, a fleet of identical vehicles supplies customers from a central depot, each customer having a given quantity requirement. Vehicles leave the depot, supply a certain number of customers and return to the depot. The quantity delivered by each vehicle on a route is limited by the vehicle's capacity. The objective of the CVRP is to minimize costs, which could be based on route length or travel times between customers or customers and the depot. These costs could be symmetric or asymmetric. In the asymmetric case, the cost would depend on the direction taken between a pair of points while in the symmetric case, cost is independent of direction. The largest arbitrary CVRP problem instances that can be solved to obtain exact solutions have about 50 sites. Larger problems can be solved practically only with heuristics (Toth and Vigo, 1998).

One of the heuristics for obtaining near-optimal solutions to the CVRP is the Clarke-Wright savings algorithm. Clarke and Wright (1962) assume a fleet of trucks with different capacities that need to supply a large number of customers (each having a quantity requirement) from a central depot. The goal is to minimize the total distance covered by all the trucks. The number of trucks available is not predetermined, but as many trucks as needed are assumed available.

The algorithm starts with an initial solution in which every customer is assigned a truck. Then the distance savings of including a pair of customers on the same route is computed for all possible pairs of customers. Now, these pairs are ordered in decreasing amount of savings which are taken advantage of one at a time using the following procedure as applicable:

- If both customers have not been included in a multi-customer route, include them in a new route.
- If one customer has not been included in a multi-customer route and the other is a non-interior node in a multi-customer route, add the former to the latter's route, between the latter and the depot. An interior customer is one that is both preceded and followed by other customer nodes (as opposed to the depot).
- If both customers are in separate multi-customer routes and are both non-interior nodes, join the two routes by breaking the link between each of these customers and the depot and linking the said customers.

Adding customers to multi-customer routes as described is subject to the capacities of the trucks left.

2.3 <u>Period Vehicle Routing Problem</u>

In the Period Vehicle Routing Problem, there is a planning horizon of *p* days and *i* customers, each of which must be serviced during *k* of the *p* days (Christofides and Beasley, 1984). There are only a certain number of *k*-day allowable combinations. For example, for a problem with a planning horizon of Monday through Friday, a customer that requires servicing twice a week could have {Monday and Thursday} or {Tuesday and Friday} as the only options available. The goal is to assign a set of k_i days to each customer in a manner that minimizes total costs over the planning horizon. To solve this problem, one must first decide which set of days to assign to each customer. Then, on each particular day of the planning horizon, one must determine how to route a fleet of vehicles to service the customers assigned to that day (this is a vehicle routing problem).

2.4 Inventory Routing Problem

The classic Inventory Routing Problem (IRP) combines routing and inventory as follows (Golden et al., 1984; Bard et al., 1998; Jaillet et al., 2002): based on their expected daily demand, customers must be assigned to one or more days, and then a vehicle routing problem (VRP) must be solved for each day to assign vehicles to customers and determine routes for the vehicles, with a goal of minimizing the total delivery cost.

In the IRP, there is a single product, which each customer consumes at a certain rate (Campbell et al. 1998). Each customer also has a predetermined inventory capacity. A customer's existing inventory must not run out before a vehicle resupply. Typically the IRP is solved over a planning horizon (for example one week). There is a fleet of homogenous vehicles of a given capacity and the objective is minimizing the cost of supplying the customers.

Here are the questions to answer in solving the IRP:

- Based on inventory capacity and demand rate, how many and what days of the week should each customer be supplied?
- Based on inventory level, what quantity should be supplied to each customer?
- How does one best route the fleet of vehicles to supply the determined quantities to the customers assigned to a particular day?

In more recent work Campbell and Savelberg (2004a) take a two-phase approach to solving the IRP. The first phase uses integer programming to determine what customers to serve over the next several days and the quantities to be delivered. The results of the first phase are used as inputs for the second phase. This phase uses the VRP and scheduling techniques to plan delivery routes and schedules. Constraints encountered in the second phase may lead to a modification of the results obtained in the first phase. In another recent work, Campbell and Savelsbergh (2004b), present Vendor Managed Inventory Replenishment. In this version of the IRP, a vendor monitors customers' inventories and conducts replenishment of their inventories by coordinating inventory levels and vehicle deliveries to minimize long term costs.

2.5 <u>Strategic Inventory Routing Problem</u>

While the fleet size is given in the classic IRP, the Strategic Inventory Routing Problem (SIRP) seeks to minimize the fleet size needed over an extended period of time. Webb and Larson (1995) point out the importance of anticipating the needed vehicle resources over a period of years needed to then solve classic IRP problems over a short period of say one week. It may be required to purchase or lease such vehicles months before the first anticipated use. The probability distribution of the demand rate and the minimum acceptable probability of stock-out for each customer are known. The SIRP seeks the fleet size that minimizes costs while meeting the needs of the problem subject to constraints. While the SIRP seeks to minimize vehicle fleet size, the underlying problem that the SIRP fleet solves in the short term is still a classic IRP as discussed above.

2.6 Bin Packing Problem

The bin packing problem (BPP) is a classic NP-hard combinatorial optimization problem. Given a finite set U of n items, $\{ u_1, u_2, \ldots, u_n\}$, each of maximum size 1, the Bin Packing Problem (BPP) seeks a partition of U into disjoint subsets (bins): $U_1, U_2, \ldots, U_k$ such that the sum of the sizes of each subset (bin) is at most 1 and for which the number of partitions k (the bins) is minimized (Garey and Johnson, 1979).

Garey and Johnson (1979) also present the First Fit Decreasing algorithm for the BPP problem. First sort the n items of the set U in decreasing order of size. An infinite number of bins, $U_1, U_2, \ldots$ are then made available. The n items are added from the largest-sized to the smallest-sized, one at a time into the first bin into which they can fit without exceeding the capacity of the bins (the bin capacity equals 1). When all of the items have been placed in bins, the number of bins that contain items of the set U is the minimum number of bins obtained by this heuristic algorithm.

2.7 Minimum Spanning Tree

The Minimum Spanning Tree (MST) problem is important in network design problems. A spanning tree is a subgraph of a graph that connects all its vertices but has no circuits. There is thus exactly one path for each pair of vertices. The minimum spanning tree problem seeks the spanning tree that has the minimum sum of paths between vertices.

Kruskal (1956) shows that the minimum spanning tree of a graph is unique and provides a procedure for obtaining the unique solution. This procedure repeatedly performs the following step until a spanning tree results: amongst the edges of the graph not yet chosen, choose the shortest edge that does not form a circuit with those previously chosen.

2.8 Branch and Bound

The problems considered in this thesis usually have a large number of possible solutions. Because they are NP-hard, there are no algorithms that allow the determination of an optimal solution in polynomial time. Lawler and Wood (1966) describe the branch and bound technique as an intelligent, systematic way to search a solution space of feasible solutions for the optimum. The solution space is repeatedly partitioned (via branching) to obtain smaller spaces. For each subspace, this technique computes a lower bound (assuming that minimization of some cost function is the objective). If the subspace's lower bound is greater than the smallest known cost, the subspace is discarded. Partitioning of the remaining sub-spaces continues until a solution is found whose cost is not greater than the smallest known bound.

2.9 Summary

The CRIRP is a distinct problem that is closely related to classical vehicle routing and inventory routing problems. This chapter has reviewed some of the most relevant work and heuristics that may be useful for solving the CRIRP. After presenting the formulation of the CRIRP, Chapter 3 will describe the differences between the CRIRP and other problems previously studied.

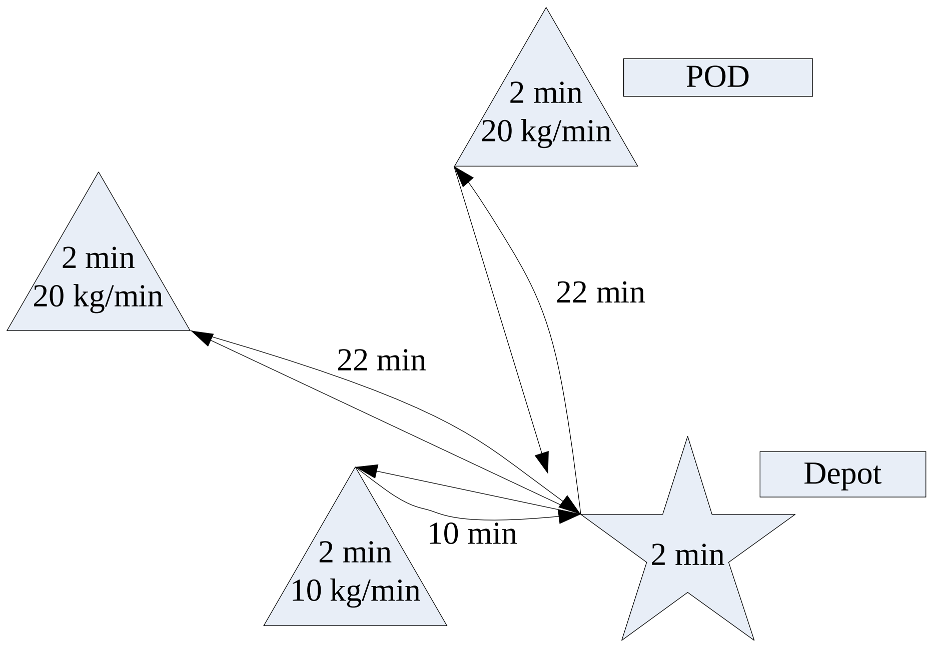
Chapter 3: PROBLEM FORMULATION

In the general CRIRP, there are n sites (customers). Each site (i = 1, …, n) has a demand rate of L_i items per time unit. This is the rate at which the site consumes material. There is a depot (i = 0) that has an unlimited amount of material. The time spent at site i (to refill a vehicle or deliver material) is p_i for i = 0, …, n. The time to go from site i to site j is c_{ij}. The vehicles are identical, each with capacity of C items of material.

The problem is to find a feasible solution with the smallest number of vehicles. A feasible solution specifies a route for each vehicle, and each site is assigned to one route. The delivery amount at a site is the route duration multiplied by the site's demand rate.

A vehicle may visit the depot multiple times during a route to refill. A partial route that starts at the depot and ends at the depot is a "subroute." A vehicle may have multiple subroutes but visits each site just once on its route. Figure 2 modifies the example in Figure 1 to illustrate the concept of subroutes. The depot and PODs are the same but this time, the vehicle visits the PODs one at a time, refilling at depot after each POD is supplied. Each POD constitutes a subroute and the vehicle capacity only needs to accommodate the subroute with the maximum demand rate.

Figure 2: Subroutes belonging to a route



Demand per POD= 10, 20 & 20 kg/min
Route duration = 2+10+2+10+
2+22+2+22+
2+22+2+22=120
Required delivery quantities: 120 x [10,20,20]
= 1200, 2400 & 2400 kg
Required truck capacity = 2400 kg

Given a solution, we evaluate its feasibility as follows. Let vehicle *v* have r_v subroutes. Let the sequence $s_{vj} = \{0, i_1, ..., i_k\}$ be subroute *j* for vehicle *v*, where *k* is the number of sites on the subroute. The total demand for the subroute is $D(s_{vj}) = L_{i_1} + \cdots + L_{i_k}$. The total time to complete the subroute is $T(s_{vj}) = p_0 + c_{0i_1} + p_{i_1} + c_{i_1 i_2} + \cdots + p_{i_k} + c_{i_k 0}$. The total time for vehicle *v* to complete all of its subroutes is $T_v = T(s_{v1}) + \cdots + T(s_{vr_v})$.

When the vehicle visits site *i*, it will need to deliver $L_i T_v$ units of material in order to keep the site supplied until the vehicle's next visit. When vehicle *v* starts subroute s_{vj}, it should take $D(s_{vj})T_v$ items in order to satisfy the demand of all the sites on that subroute; this quantity is the *load* of that subroute. Let $D_v^* = \max\{D(s_{v1}),\dots,D(s_{vr_v})\}$. The maximum load for vehicle *v* is $M_v = D_v^* T_v$. The solution is feasible if each site is assigned to exactly one vehicle and each vehicle's maximum load is not greater than the vehicle capacity. That is, $M_v \le C$ for all vehicles $v = 1,\dots,K$.

In order to demonstrate the existence of feasible solutions, consider the trivial subroutes $z_i = \{0,i\}$, for $i = 1,\dots,n$. Then, $T(z_i) = p_0 + c_{0i} + p_i + c_{i0}$ and $D(z_i) = L_i$. It is easy to see that there are feasible solutions to CRIRP if and only if $D(z_i)T(z_i) \le C$ for all $i = 1,\dots,n$.

The objective is to find a feasible solution with the minimal number of vehicles. CRIRP, like virtually all vehicle routing problems, is NP-hard (Lenstra and Rinnooy Kan, 1981).

3.1 Unique Characteristics of the CRIRP

Although similar in some ways to other routing problems, the CRIRP is unique in certain respects. In the PVRP, IRP and SIRP, customer demand is expressed as a rate (quantity per unit time) as opposed to just a quantity. Based on the customer demand rates, vehicle routing decisions are made periodically (e.g. daily). The routes start and end in the same day; they don't go into the next day. All of the vehicles are available at the beginning of the next day. There is a “jump” from one day to the next where no vehicles are operating. In the CRIRP, the customer demand is also a rate, but customers are supplied continuously, around the clock. Instead of waiting for the next day to determine routing, when vehicles return to the depot, they immediately reload and resupply their customers.

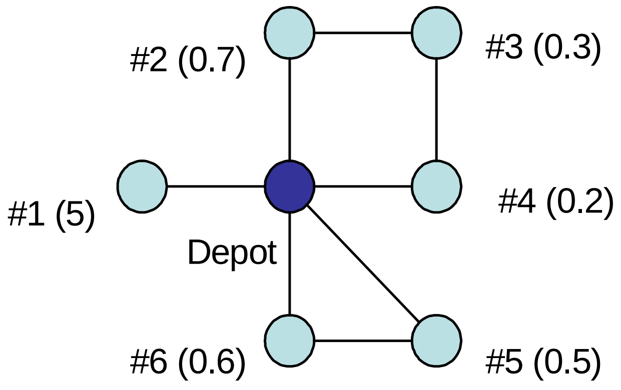
In IRP problems, the inventory capacity at customer locations is predetermined. In the CRIRP, the customers are also expected to have inventory, but their required inventory capacity is determined only after the CRIRP has been solved.

Among the problems previously studied, the SIRP best resembles the CRIRP. Both express customer needs as demand rates and minimize the required vehicle fleet size. However, the classic IRP is the problem that underlies the SIRP which means that ultimately the SIRP anticipates supplying customers in a non-continuous manner.

3.2 Example

Consider a six-site problem instance. The depot and site locations are shown in Figure 3. The travel time between sites is proportional to the distance. In this instance, the travel time equals one time unit between the depot and sites 1, 2, 4, and 6 as well as between sites 2 and 3, between sites 3 and 4, between sites 5 and 6. The travel time between the depot and site 5 equals 1.4 time units. The demand rate L_i at each site is shown in parentheses. The service time $p_i = 1$ time unit at the depot and all sites.

Figure 3: Feasible solution to a CRIRP instance



If the vehicle capacity $C = 20$ units, then the solution in Figure 1 is feasible with two vehicles as follows. The first vehicle follows only one subroute $s_{11} = \{0,1\}$. The demand $D_1^* = D(s_{11}) = 5$ items per time unit, and the route duration $T_1 = T(s_{11}) = 4$ time units, so the load $M_1 = 20$ items. The second vehicle has two subroutes: $s_{21} = \{0,2,3,4\}$ and $s_{22} = \{0,5,6\}$. The first subroute demand $D(s_{21}) = 1.2$ items per time unit, and the subroute duration $T(s_{21}) = 8$ time units. The second subroute demand $D(s_{22}) = 1.1$ items per time unit, and the subroute duration $T(s_{22}) = 6.4$ time units. Therefore, the total route duration $T_2 = T(s_{21}) + T(s_{22}) = 14.4$. $D_2^* = \max\{D(s_{21}), D(s_{21})\} = 1.2$ items, so $M_2 = D_2^* T_2 = 17.28$ items. The load for the first subroute equals 17.28 items, and the load for the second subroute equals 15.84 items.

3.3 *The Special Case of Identical Demand*

Consider the special case in which all $L_i = L$. (This special case is a useful model for the POD resupply problem if the jurisdiction's mass dispensing plans call for a set of identical PODs.) In this case, as we show below, the non-trivial subroutes of a feasible solution can be split into the trivial subroutes without increasing the maximum load of any vehicle. Thus, there is an optimal solution in which every vehicle's route is the concatenation of trivial subroutes.

Consider a feasible solution in which a vehicle *v* visits *n* sites using *r* subroutes. Suppose $r < n$, so at least one subroute has more than one site. Let $m_0 = 0$. Renumber the sites and define m_k ($k = 1, \dots, r$) so that the first subroute visits sites $1, \dots, m_1$, the second subroute visits sites $m_1 + 1, \dots, m_2$, and so forth, with $m_r = n$.

Let $h = \max_{1 \le k \le r} \{ m_k - m_{k-1} \}$. Note that $h \ge 2$ and $hr \ge n$. Let TT_k be the travel time of subroute *k*. Note that $TT_k \ge 2c_{0i}$ for any $i \in \{ m_{k-1} + 1, \dots, m_k \}$.

Now consider the duration of each subroute *k*, and let T_0 be the duration of the current route:

$$T(s_{vk}) = p_0 + \sum_{i=m_{k-1}+1}^{m_k} p_i + TT_k$$

$$\begin{aligned} T_0 &= \sum_{k=1}^{r} T(s_{vk}) \\ &= rp_0 + \sum_{i=1}^{n} p_i + \sum_{k=1}^{r} TT_k \end{aligned}$$

On subroute *k* the demand $D(s_{vk}) = (m_k - m_{k-1})L$. The maximum subroute demand is therefore *hL*, and the maximum load is hLT_0. Because the solution is feasible, $hLT_0 \le C$.

Now, modify this solution to construct a new solution in which this vehicle visits all of the same sites using trivial subroutes. Let $t_i = p_0 + 2c_{0i} + p_i$ for all $i = 1, \ldots, n$. Let T_1 be the duration of the new route:

$$\begin{aligned} T_1 &= \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} \left(p_0 + c_{0i} + p_i + c_{0i} \right) \\ &= np_0 + \sum_{i=1}^{n} p_i + 2\sum_{i=1}^{n} c_{0i} \end{aligned}$$

In this solution, the maximum subroute demand is *L*, and the maximum load is LT_1. Now, we will show that $LT_1 < hLT_0$ by proving that $hT_0 - T_1$ is positive.

$$hT_0 - T_1 = p_0 \left(hr - n \right) + \left(h - 1 \right) \sum_{i=1}^{n} p_i + \left(\sum_{k=1}^{r} hTT_k - 2\sum_{i=1}^{n} c_{0i} \right)$$

Because $hr \ge n$, the first term is non-negative. Because $h \ge 2$, the second term is positive. To analyze the third term, we regroup the terms in the last summation by the subroutes to get the following:

$$\begin{aligned} \sum_{k=1}^{r} hTT_k - 2\sum_{i=1}^{n} c_{0i} &= \sum_{k=1}^{r} \left(hTT_k - 2 \sum_{i=m_{k-1}+1}^{m_k} c_{0i} \right) \\ &\ge \sum_{k=1}^{r} \sum_{i=m_{k-1}+1}^{m_k} \left(TT_k - 2c_{0i} \right) \end{aligned}$$

Each term of this double summation is non-negative. Therefore, $hT_0 - T_1$ is positive, and $LT_1 < hLT_0 \le C$. This shows that using the trivial subroutes is also feasible because they reduce the load of the vehicle. Therefore, there is an optimal solution with all trivial subroutes.

Which vehicle should do which subroutes? Let $t_i = p_0 + 2c_{0i} + p_i$ for all $i = 1, \ldots, n$. Suppose vehicle *k* completes a set S_k of trivial subroutes. The route is feasible if and only if $M_k = L \sum_{i \in S_k} t_i \leq C$, which is equivalent to $\sum_{i \in S_k} t_i \leq C / L$. Thus, the problem becomes a bin packing problem in which the item size is t_i and the bin size is *C*/*L*. The packing of items into bins corresponds to the assignment of sites to vehicles. Interestingly, the routing is trivial, because the load does not depend upon the sequence, so any sequence for a vehicle's route is sufficient.

3.4 Combining Subroutes with Unequal Demand

The result above reflects the fact that it is desirable to create subroutes that have equal demand. In the special case, this means using only trivial subroutes. In the general case, we can see that combining subroutes with low demand is desirable.

Consider a feasible solution in which a vehicle visits *n* sites using *r* subroutes. Renumber the sites as done in Section 3.3.

For each subroute *k*, let TT_k be the travel time of subroute *k*, let $D(s_k)$ be the subroute demand, and let $T(s_k)$ be the duration of the subroute:

$$D(s_k) = \sum_{i=m_{k-1}+1}^{m_k} L_i$$

$$T(s_k) = p_0 + \sum_{i=m_{k-1}+1}^{m_k} p_i + TT_k$$

Let $D_v^* = \max_{1 \le k \le r} D(s_k)$ be the largest subroute demand. Let $T_v = \sum_{k=1}^{r} T(s_k)$ be the total route duration. Then the maximum load $M_v = T_v D_v^*$.

Consider two subroutes *p* and *q* such that $D(s_p) + D(s_q) \le D_v^*$. Combining these two subroutes into one subroute eliminates one stop at the depot, which decreases the total route duration T_v (by p_0 and any distance savings). Because the total demand on the new subroute is not greater than D_v^*, the maximum load M_v also decreases. Thus, this new route is also feasible. This implies that low-demand subroutes should be combined when possible.

3.5 Summary

In this chapter we have developed the equations that govern the feasibility of a potential solution to a CRIRP problem – that is the relationship among the required vehicle capacity, route duration and subroute demands. We also determined the conditions required for a CRIRP to have a solution. For a special case of the CRIRP in which all PODs have an identical demand rate, we showed that there is an optimal solution which reduces to a bin packing problem. This finding then led us to the conclusion that combining relatively low-demand subroutes is beneficial. In the next chapter, we develop heuristics for solving the CRIRP. Some of these heuristics are based on insight obtained from this chapter.

Chapter 4: HEURISTICS

Because the general CRIRP problem is NP-hard, we will consider the use of heuristics to find good solutions in a reasonable amount of time. In this chapter three heuristics are presented along with the results they produce.

4.1 <u>H1 Heuristic: Subroute Demand Packing</u>

The first heuristic (H1) is a three-stage bin-packing approach that has a parameter *W*, representing subroute demand. *W* is varied in the range

$$W \in \left[\max\left(L_i\right), \sum_i L_i \right] \tag{4.1}$$

that is, from the maximum POD demand to the sum of all POD demands.

4.1.1 Rationale for the Heuristic

Before presenting the heuristic, we will describe the idea behind it. Each subroute is visited once during a vehicle route. For two subroutes on the same vehicle's route, a lower demand subroute requires a smaller delivery quantity than a higher demand subroute. While the higher capacity subroute may fully utilize vehicle capacity, the lower demand subroute would be wasting the same. It is thus desirable that all the subroutes of a route have about equal demand. The parameter, *W* in this heuristic represents subroute demand. Site demands are packed into subroutes of demand capacity, *W*. On the one hand, to accommodate the largest demand POD, *W* must be at least as large as $\max(L_i)$. On the other hand the largest possible subroute would contain all sites and have a demand of $\sum_i L_i$.

Consider the vehicle capacity constraint for each route:

$$D_v^* T_v \le C$$

Since *W* is now the maximum subroute demand,

$$WT_v \le C \quad \text{or } T_v \le C/W \qquad (4.2)$$

From (4.2) subroute durations can be packed into routes of duration C/W.

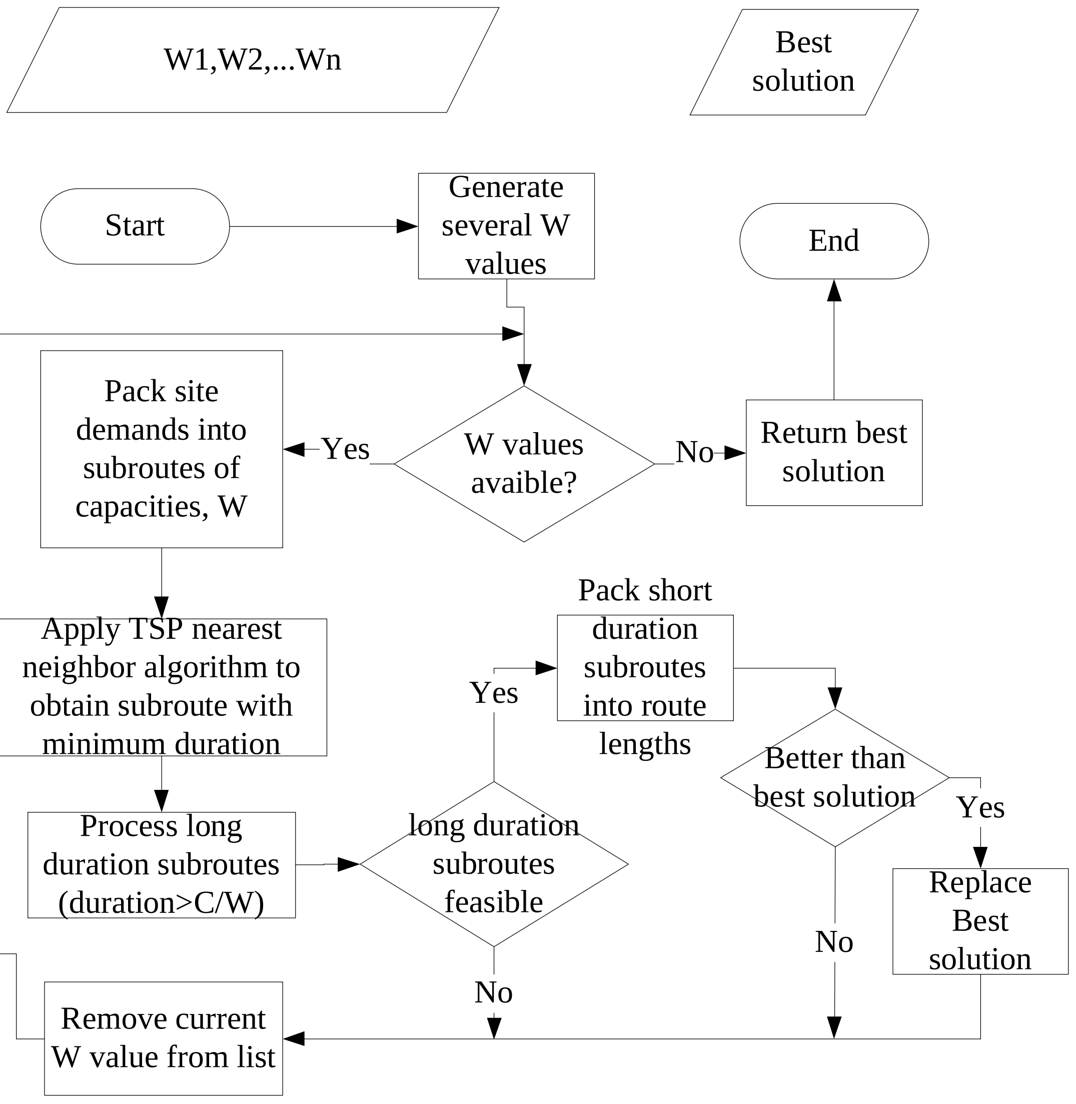
It is possible that a subroute's duration may be greater than C/W, in which case the heuristic can't pack it into a route. Moreover, any solution with that subroute may be infeasible unless the subroute demand is much less than *W*. That would mean this heuristic would not be able to return a solution. If $T\left(s_{vj}\right)$, the duration of subroute *j*, is greater than C/W, but $T\left(s_{vj}\right) \le C/D\left(s_{vj}\right)$, then the subroute by itself can constitute a route even though it cannot be packed into the specified route duration.

To illustrate this point, suppose that *W* = 400 lbs/hr and *C* = 1000 lbs. We require the total route length $T_v \le C/W$ = 2.5 hrs. Now, suppose that, after packing the site demands, subroute *j* has demand $D\left(s_{vj}\right)$ = 250 lbs/hr and a subroute duration $T\left(s_{vj}\right)$=3 hrs. Subroute *j* cannot be packed into a route, but the subroute by itself is feasible because 3 hrs = $T\left(s_{vj}\right) \leq C / D\left(s_{vj}\right)$ = 4 hrs. Equivalently, $D\left(s_{vj}\right)T\left(s_{vj}\right) \leq C$ as 250(3) < 1000 lbs.

4.1.2 Algorithm and Implementation

The H1 or subroute demand packing heuristic is depicted in Figure 4. In the first step, the heuristic uses the first fit decreasing algorithm to find a solution to the bin-packing problem in which each site i is an item, the item size is the demand rate L_i, and the bin capacity is W. This assigns sites to subroutes. The order in which sites in a subroute are visited is determined by applying the nearest neighbor algorithm starting at the depot.

Figure 4: The H1 heuristic


W1,W2,...Wn
Best solution
Start
Generate several W values
End
Pack site demands into subroutes of capacities, W
Yes
W values avaible?
No
Return best solution
Pack short duration subroutes into route lengths
Apply TSP nearest neighbor algorithm to obtain subroute with minimum duration
Yes
Better than best solution
Yes
Process long duration subroutes (duration>C/W)
long duration subroutes feasible
Replace Best solution
No
No
Remove current W value from list

The next step packs subroutes into routes. First, if there are any subroutes whose duration does not permit them being packed in the prescribed route duration, such subroutes require separate processing (Figure 5). For the subroutes whose durations can be packed into route durations, the first fit decreasing algorithm is once again applied. The routes formed from the long subroutes and the packed routes constitute the total number of routes. If this is thus far the best solution, it replaces the previous best solution (if one existed). This process continues for each value of *W*, and the solution with the lowest number of vehicles is returned. Our implementation of this heuristic used 6 equally spaced values of *W*, with the lowest and highest values obtained from the bounds in Equation 4.1.

Figure 5: Processing long subroutes for the H1 heuristic

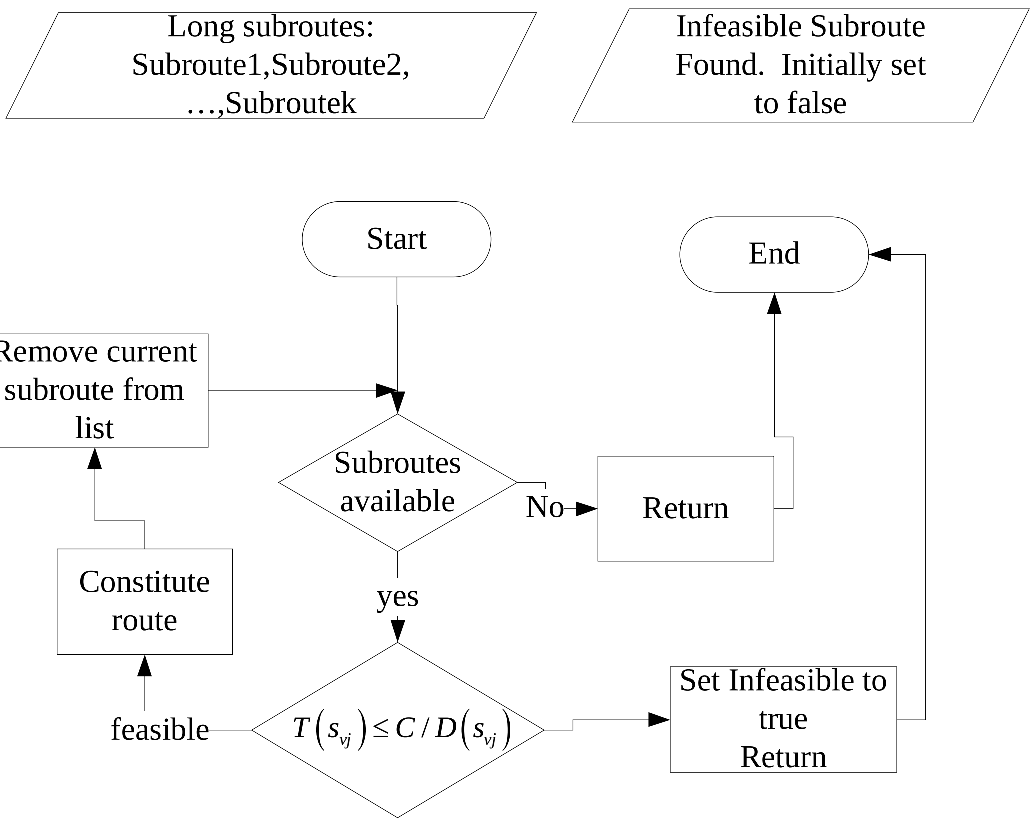
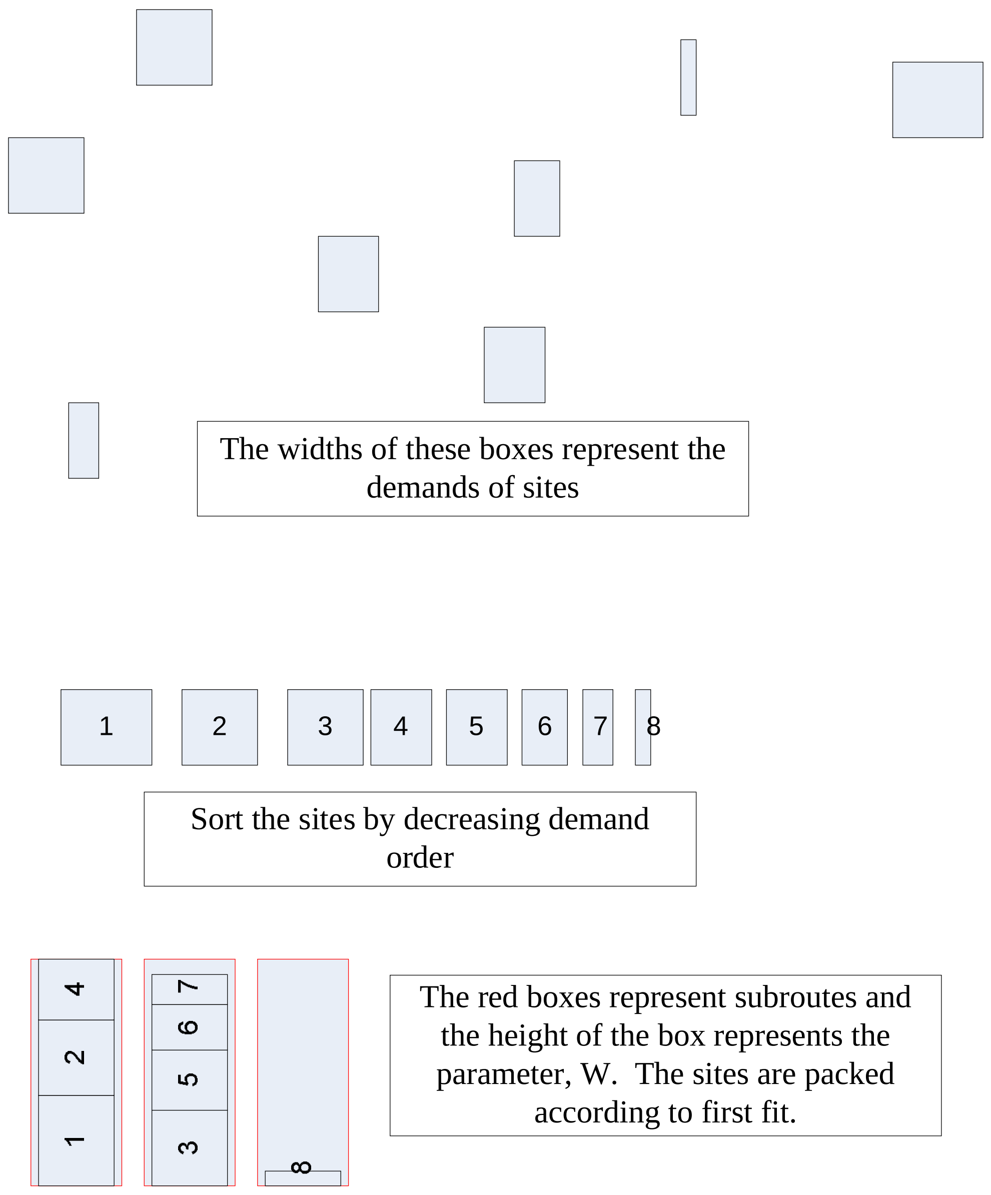

Long subroutes: Subroute1,Subroute2, …,Subroutek
Infeasible Subroute Found. Initially set to false
Start
End
Remove current subroute from list
Subroutes available
No
Return
yes
Constitute route
feasible
$T(s_{vj}) \le C / D(s_{vj})$
Set Infeasible to true Return

Figure 5 depicts how the above-mentioned long subroutes are processed. If any of the long subroutes cannot by itself constitute a route (because $D(s_{vj})T(s_{vj}) > C$), the value of *W* during which the failure occurs is abandoned. If, for all the values of *W*, no feasible solution is found, then this heuristic fails to return a feasible solution.

Figure 6 depicts the first fit decreasing algorithm applied when packing site demands into W, the subroute demand capacity.

Figure 6: The first fit decreasing algorithm applied to site demands


The widths of these boxes represent the demands of sites
1
2
3
4
5
6
7
8
Sort the sites by decreasing demand order
4
2
1
7
6
5
3
8
The red boxes represent subroutes and the height of the box represents the parameter, W. The sites are packed according to first fit.

4.1.3 Heuristic Improvements

Some work could be done to improve the solutions computed by this algorithm. If the long duration subroutes fail to be packed into routes and fail to constitute routes by themselves for all values of *W* tested, this heuristic will fail to return a solution. A simple way to attempt to counter this situation is to break up long subroutes into individual sites and try to reconstitute them to obtain feasible routes. The most trivial attempt could simply make each of the individual sites into a route.

This implementation uses six evenly spaced values of *W*. There are an infinite number of *W*'s and intuitively, the more values that are processed, the better the chance of getting close to the optimum. However, processing more values of *W* is time intensive and provides no guarantee of obtaining a better solution. It would be interesting to investigate if there is a more systematic way to select values from the range of possible *W* values so as to obtain better solutions.

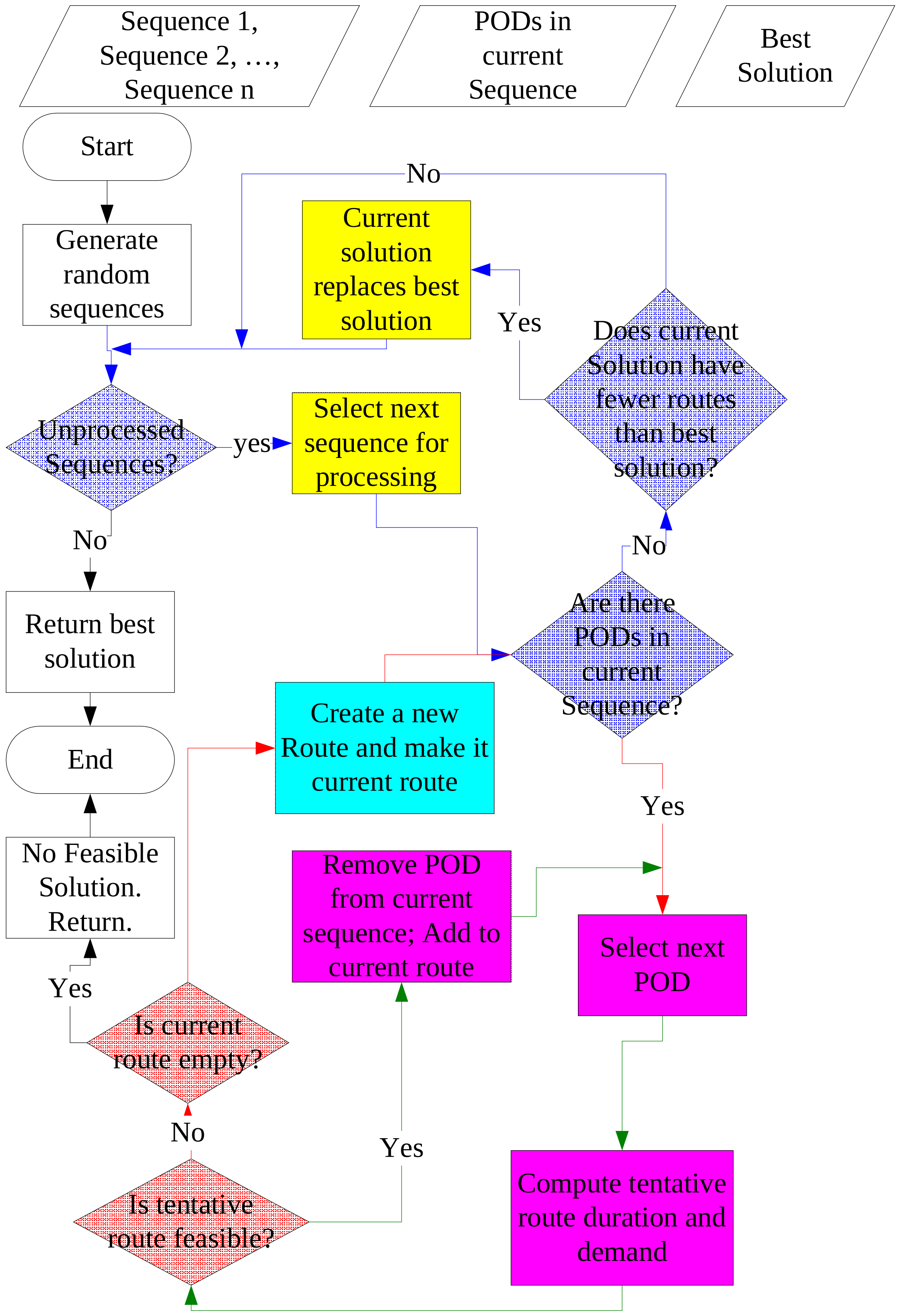
4.2 H2 Heuristic: Random POD Sequences

This heuristic generates several random sequences of PODs. For each such sequence, a new route is initially created and PODs are added one at a time to the current route up to the point where adding a POD makes a route infeasible. At this point, the heuristic starts a new route and adds the POD to it. This procedure continues until all PODs are in routes. We compare the number of routes generated by each of the POD sequences and select the minimum. The routes generated by this heuristic consist of one subroute only. This is a very simple heuristic based on random POD sequences. Its main purpose is to serve as a benchmark for all heuristics. Any heuristic that does not significantly out-perform this one, is probably not a good heuristic.

4.2.1 Algorithm and Implementation

The H2 heuristic is depicted in Figure 7. Our implementation generates 10 random sequences of the PODs. To determine route feasibility, the heuristic needs the route duration T_v and the route demand (in this case the route demand is the same as the maximum subroute demand D_v^*). Before adding the first POD to the route, the load time at the depot is added to the route duration. When adding subsequent PODs to a route, the current route duration is augmented by the travel time to the POD and the POD unload time; meanwhile the route demand is augmented by the POD demand.

Figure 7: The H2 heuristic


Sequence 1, Sequence 2, …, Sequence n
PODs in current Sequence
Best Solution
Start
Generate random sequences
No
Current solution replaces best solution
Yes
Does current Solution have fewer routes than best solution?
Unprocessed Sequences?
yes
Select next sequence for processing
No
No
Return best solution
Are there PODs in current Sequence?
Create a new Route and make it current route
End
Yes
No Feasible Solution. Return.
Remove POD from current sequence; Add to current route
Select next POD
Yes
Is current route empty?
No
Yes
Is tentative route feasible?
Compute tentative route duration and demand

4.2.2 Determining whether to add a POD to a Route

When adding a POD to a route, it is unknown if this would be the last POD added. The heuristic should therefore not add the travel time back to the depot to the route duration. However, to determine whether the route would be feasible with the addition of this POD, the travel time from the POD back to the depot is tentatively added to the route duration in order to determine feasibility.

If this new POD cannot feasibly be added to the route, the current POD is removed and used to start a new route: the current route is now complete. Because the feasibility of a route is determined before a POD is added to it, we simply permanently augment the route duration with the travel time back to the depot.

4.2.3 Improvements

This heuristic is straightforward and easy to implement, but one envisions possible improvements. Before rejecting a new POD and closing a route, a TSP heuristic could be applied to the route to reduce the duration and thus increase the demand that the route can handle. Also, when a POD fails to be added to a route, the heuristic could look further down into the sequence of remaining PODs to determine if there are any that could be feasibly added to this route. We leave these improvements for future study.

4.3 H3 Heuristic: Utilizing the Special Case

The H3 heuristic attempts to group PODs having similar demand on the same route. The CVRP is then applied to each of these sets of PODs. The vehicle capacity is the CVRP constraint. For a set of PODs, the CVRP would return routes. These CVRP routes are used as subroutes of a CRIRP route. The resulting route duration and the aggregate maximum subroute demand are used to determine the feasibility of the CVRP solution. To create a CRIRP route, this heuristic progressively adds PODs to the CVRP problem until the feasibility constraints are violated.

4.3.1 Rationale for the Heuristic

This heuristic is inspired by two observations made in Chapter 3. First, in the special case, all PODs have the same demand, and there is an optimal solution for which routes are constituted from trivial subroutes. Secondly, in the general CRIRP problem, it was shown in Chapter 3 that, if the sum of the demand of two subroutes of a route is smaller than the demand of any single subroute, the solution cannot be worse and indeed should be better if those subroutes are combined into one subroute. Thus, if PODs have about the same demand, we want them to each constitute a trivial subroute. We would combine subroutes only if some PODs on the route have a much smaller demand than others. Solving the CVRP provides a mechanism for implementing these ideas. To apply CVRP techniques to solve the CRIRP, we convert POD demands to POD delivery quantities that are proportional the demand rates. By varying the ratio of delivery quantity to demand, we permit the vehicle capacity constraint of the CVRP to control the number of PODs per subroute while ensuring that low demand PODs are combined on the same subroute. In the next sections we discuss in more detail how this heuristic achieves these aims.

4.3.2 Converting Demand Rates to Absolute Quantities

In order to use solution techniques for the CVRP, the demand at each site must be a quantity, not a rate. The solution of the CVRP is then executed periodically for each route to get the effect of a rate. The absolute quantities assigned to PODs must thus be proportional to the POD demands. Consider a set of PODs that we wish to constitute into a route. At one extreme the quantities assigned to the PODs could be such that the vehicle can barely deliver medical supplies to the maximum demand POD. At the other extreme, the quantities could be such that the vehicle could deliver to all the PODs with one vehicle load. For example, suppose we want to constitute a route with four PODs that have the following demands: 75, 100, 125, and 200 *kg/hr*. Let the truck capacity be 1000 *kg*. In converting demand rates into quantities, at one extreme, we would multiply the demands by a ratio of 1000/200 = 5 (using the maximum demand rate) to obtain delivery quantities of 375, 500, 625, and 1000 *kg*. If these were an instance of the CVRP, the POD with 1000 *kg* would become one subroute, and the other three PODs would be combined in a manner that seeks to minimize the route duration. At the other extreme, if we multiply the demands by 1000/500 = 2 (using the sum of the demands), solving the CVRP would yield one giant subroute of our route. Of course, we could use ratios anywhere between these extremes. Whatever ratio we use, the advantage of using CVRP is that it lumps POD demands together to vehicle capacity while also attempting to minimize route duration (which is the goal of CVRP).

4.3.3 Feasibility of the Adapted CVRP Solution

CVRP requires that the quantities delivered to each POD on a route do not exceed vehicle capacity. CVRP feasibility is a prerequisite to applying the CVRP solution to the H3 heuristic. To apply this solution, the heuristic first converts the routes of the CVRP solution to subroutes of a single route in CRIRP. We still need to ensure that the PODs' demand rates are not outstripping supply rates. Consider POD j with demand D_j on the

CRIRP route just obtained from CVRP. Let *r* be the *quantity - demand ratio* applied to convert the demand rates to quantities for the CVRP. Then the quantity delivered to the POD during the route is rD_j. If the demand rate should not outstrip supply, then

$$\frac{quantity\_delivered}{route\_duration} \geq demand$$
$$\frac{rD_j}{routeDuration} \geq D_j$$
$$r/_{routeDuration} \geq 1$$
$$r \geq routeDuration$$

The sum of the CVRP route durations obtained must be less than or equal to the *quantity-demand ratio* by which the demand rates were multiplied.

4.3.4 Selecting PODs for a Route

Since this heuristic seeks to operate as close to the special case as possible, first, all PODs are sorted in order of increasing demand. This is the order in which PODs must be added to routes. Whenever PODs are successfully added to a route, they are removed from the list of sorted PODs. We seek lower and upper bounds to the number of low demand PODs that will fit in the next route. The initial upper bound is the number of PODs left. To obtain a lower bound, the heuristic adds PODs to the route as trivial subroutes up to the point where the problem constraints are violated. The use of trivial subroutes (as in the CRIRP special case) as opposed to one all-inclusive subroute, is justified by the proximity of the sorted POD demands. When applying CVRP to the PODs, the heuristic then selects a certain number of the next available PODs, where the number selected is greater than the lower limit and less than or equal to the upper limit. The CVRP solution technique is then applied to these selected PODs. Depending on the feasibility of results of the CVRP solution technique, the heuristic revises the lower and upper bounds on the number of PODs in the route. This revision continuously reduces the difference between the lower and upper bounds until they converge on the same number. When this happens, the indicated PODs are the best that this heuristic can put in a route.

4.3.5 Algorithm and Implementation

Figure 8 depicts the "bounds routine": the process that this heuristic uses to select lower and upper bounds. When this routine is first called for a new route, the lower bound is set to zero PODs. The number of PODs in the last CVRP attempt, the route length and maximum subroute duration are also all set to zero. For subsequent calls of the routine, the heuristic must have tried to run a CVRP solution technique on a certain number (non-zero) of PODs. Since the lower bound is the least number of the next available PODs that is guaranteed to fit in a route, the CVRP solution technique that is called should try to fit more than the lower bound. The bounds routine then receives the feasibility of the CVRP-attempted route. If feasible, the routine also needs the route duration and maximum subroute demand. It is this information that the bounds routine uses to determine new lower and upper bounds for the number of PODs that can fit in a route. The routine either reduces the upper bound or increases the lower bound.

Figure 8: Bounds routine for the H3 heuristic

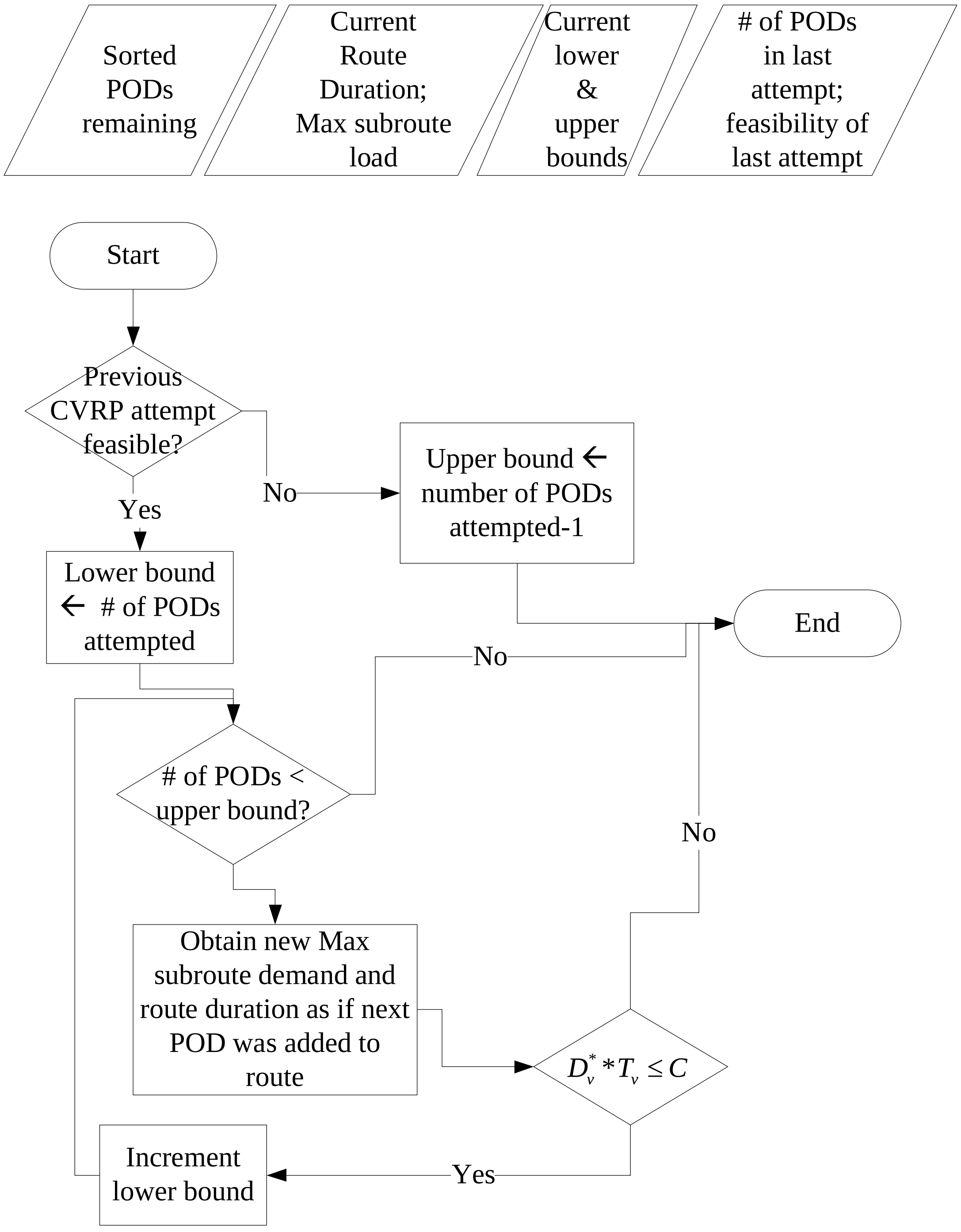
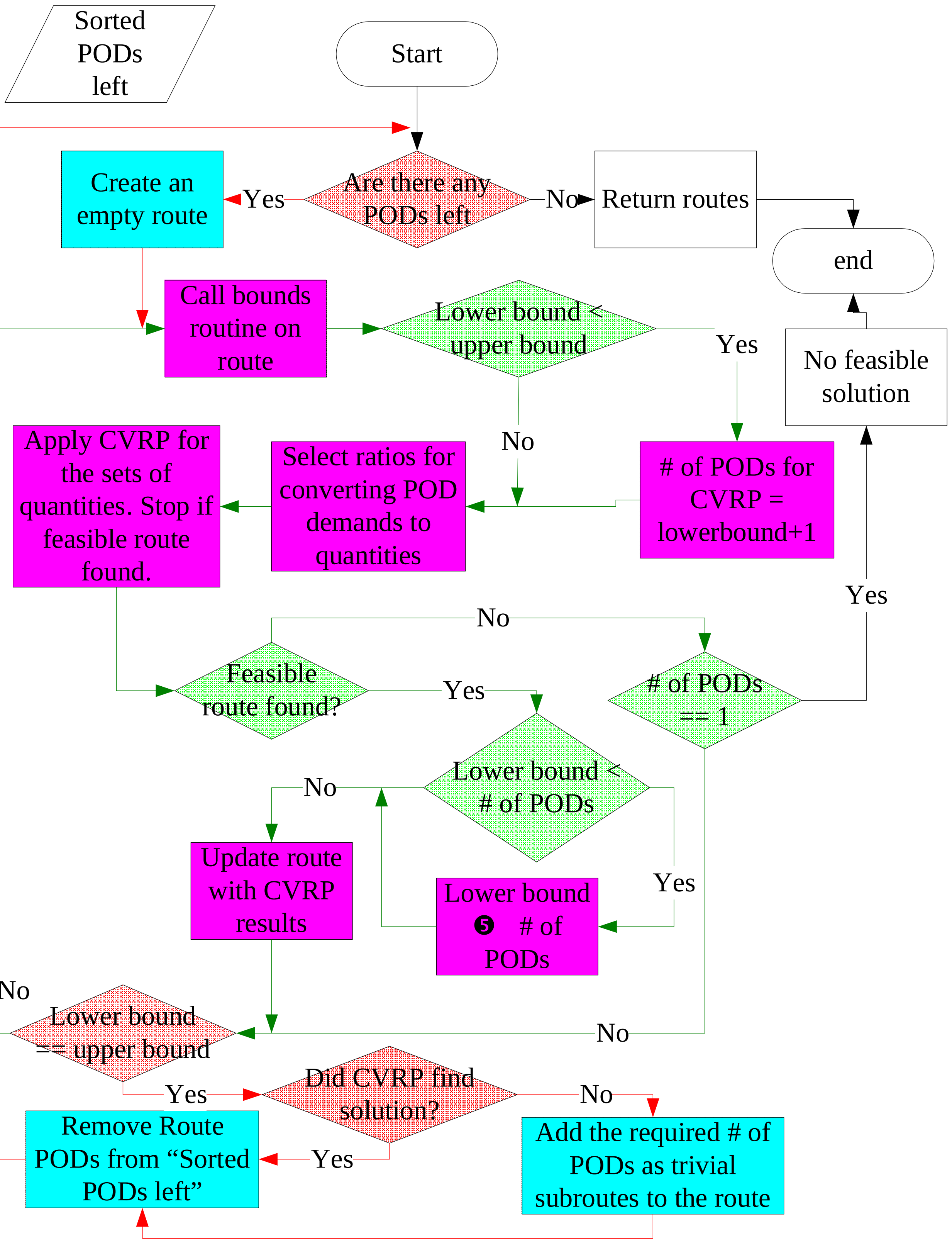

Sorted PODs remaining
Current Route Duration; Max subroute load
Current lower & upper bounds
# of PODs in last attempt; feasibility of last attempt
Start
Previous CVRP attempt feasible?
No
Upper bound ← number of PODs attempted-1
Yes
Lower bound ← # of PODs attempted
End
No
# of PODs < upper bound?
No
Obtain new Max subroute demand and route duration as if next POD was added to route
$D_v^* * T_v \leq C$
Yes
Increment lower bound

Figure 9 depicts the H3 heuristic. The heuristic starts with all the PODs sorted in increasing order of demand and creates a new empty route. The bounds routine (Figure

8) is called to obtain bounds on the number of PODs that can fit in the route. When a new route is created, the upper bound is the total number of PODs left. If the lower bound is less than the upper bound, in this implementation, the heuristic tries to solve the CVRP with one POD more than the lower bound. Otherwise we simply use the lower bound as the number of PODs.

To convert the POD demands to quantities, our implementation multiplies the POD demand rates by six equally spaced *quantity – demand rate ratios* to obtain six route *quantity sets*. The smallest and largest ratios are obtained from the extremes described in Section 4.3.2 (Converting Demand Rates to Absolute Quantities). The Clarke-Wright CVRP algorithm is then applied to each *quantity set*. We are interested only in finding one feasible route, so our implementation stops cycling through the *quantity sets* as soon as a set is found that results in a feasible route. Feasibility is determined as described in Section 4.3.3 (Feasibility of the Adapted CVRP Solution). If a feasible route is found with this number of PODs, the lower bound is updated to indicate the number of PODs in our best proven solution. The current route is also updated.

Figure 9: H3 heuristic



Sorted PODs left
Start
Are there any PODs left
Yes
No
Create an empty route
Return routes
end
Call bounds routine on route
Lower bound < upper bound
Yes
No feasible solution
No
Apply CVRP for the sets of quantities. Stop if feasible route found.
Select ratios for converting POD demands to quantities
# of PODs for CVRP = lowerbound+1
Yes
No
Feasible route found?
Yes
# of PODs == 1
Lower bound < # of PODs
No
Update route with CVRP results
Lower bound ❺ # of PODs
Yes
No
Lower bound == upper bound
No
Yes
Did CVRP find solution?
No
Remove Route PODs from "Sorted PODs left"
Yes
Add the required # of PODs as trivial subroutes to the route

If the CVRP solution technique fails to find a feasible route and it was considering only one POD, the heuristic exits: there is no solution because it is impossible to fit one POD in a route. If it fails with multiple PODs, processing continues.

On the one hand, if the lower bound is equal to the upper bound, this heuristic knows the number of PODs in the best makeup of the a route. It is possible however that CVRP solution technique failed to find a feasible route (which in fact happened while running this heuristic) despite having a lower bound and an upper bound that are equal. If this happens, the method that this heuristic uses to put all the desired PODs in the route is the same method that was used to obtain the lower bound: starting with the current route, the heuristic simply adds PODs to the route as trivial subroutes until the number of PODs in the route is the lower limit. This is guaranteed to work because that is how the heuristic obtained the lower bound in the first place. Otherwise the route found by a feasible CVRP solution is sufficient. This route is now complete: the heuristic removes the PODs in this route from the sorted PODs left and proceeds to create a new empty route. On the other hand, if the lower bound is less than the upper bound, the heuristic proceeds to call the bounds routine in order to improve the current route or confirm that a better route is not possible. The heuristic continues with this procedure until there are no sorted PODs left.

4.4 Testing the Heuristics

4.4.1 CRIRP Test Problems

To test the heuristics developed, we use four sets of location data obtained from the TSPLIB (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html July 08, 2008).

TSPLIB is a library of sample instances that provide either location data or the costs associated with the paths of a graph. They serve as test data for TSP solvers. We selected the following 4 sets of data:

- berlin52 (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/berlin52.tsp): 52 locations in Berlin, Germany.
- bier127 (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/bier127.tsp): 127 beer gardens in Augsburg Germany
- burma14 (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/burma14.tsp): 14 cities in Burma
- ulyssess22 (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/ulysses22.tsp): 22 locations from the Odyssey of Ulysses

For each of these problems, the locations are sequentially indexed using positive integers. Each location also has Cartesian coordinates. While these are sufficient for testing the intended TSP method, more data is needed for the CRIRP.

We made the first location the depot. The other locations are then designated as PODs and numbered from 1. The Euclidean distance between each pair of points was computed to obtain a complete graph whose path weights are the travel times. For each problem we generated 4 sets of load times. For each set, the load times at all PODs and the depot are equal. First, we calculated the average travel time A of a problem's graph. Then the following sets of load times were generated $\left\{\frac{A}{50}, \frac{A}{5}, A, 3A\right\}$. We then arbitrarily chose an average demand rate of 200 per POD. The depot demand was set to zero. We assigned the POD demand rates using a random number generator generating values about a mean of 200 and each of the following standard deviations: $\left\{\frac{average\_demand}{3}, \frac{average\_demand}{6}, \frac{average\_demand}{9}\right\}$

To designate the problem instances' vehicle capacities, we had to avoid a situation wherein the vehicle capacities are so low that the problems have no solution. Prior to determining an acceptable range for vehicle capacity, we supposed that each POD would be served exclusively by one vehicle and obtained the required vehicle capacities for each of the PODs. We used the set of demand rates that have standard deviation $\frac{average\_demand}{3}$ and the set of load times that equal the average travel time A to determine these vehicle capacities. We set the highest of the required vehicle capacities as our "low" capacity, and the sum of the required vehicle capacities as our "high" capacity. To further ensure that the "low" capacity is reasonable, we ran the H1 heuristic with using this low capacity. If this failed to give a solution, we gently crept up the "low" capacity until this set of data gave us a feasible solution. We then selected three possible capacities: the "low" capacity, the "high" capacity and the average of the two as variations of the problem data. This is done for each of the 4 original TSP instances.

For a problem instance, it may be required to place a limit on the route duration. For each problem we assigned the following route duration limits: 100, 1000, 10000, 100000 and one case with no limits.

We thus ended up with 4 sets of load/unload times, 3 sets of POD demand rates, 3 possible vehicle capacities and 5 sets of duration limits (that is, 4 duration limits and one case with no limit) for a single TSP problem. By taking all possible combinations of unload times, POD demand rates, vehicle capacities and route durations, from one TSP instance we have 4 x 3 x 3 x 5= 180 CRIRP problems. The 4 TSP problems generated 720 CRIRP problem instances. These problem instances are contained in a MATLAB code file posted (with the rest of the implementation software) at the project website: http://www.isr.umd.edu/Labs/CIM/projects/clinic/.

4.4.2 CRIRP Heuristic Solutions

We implemented all three heuristic algorithms in the MATLAB programming language.

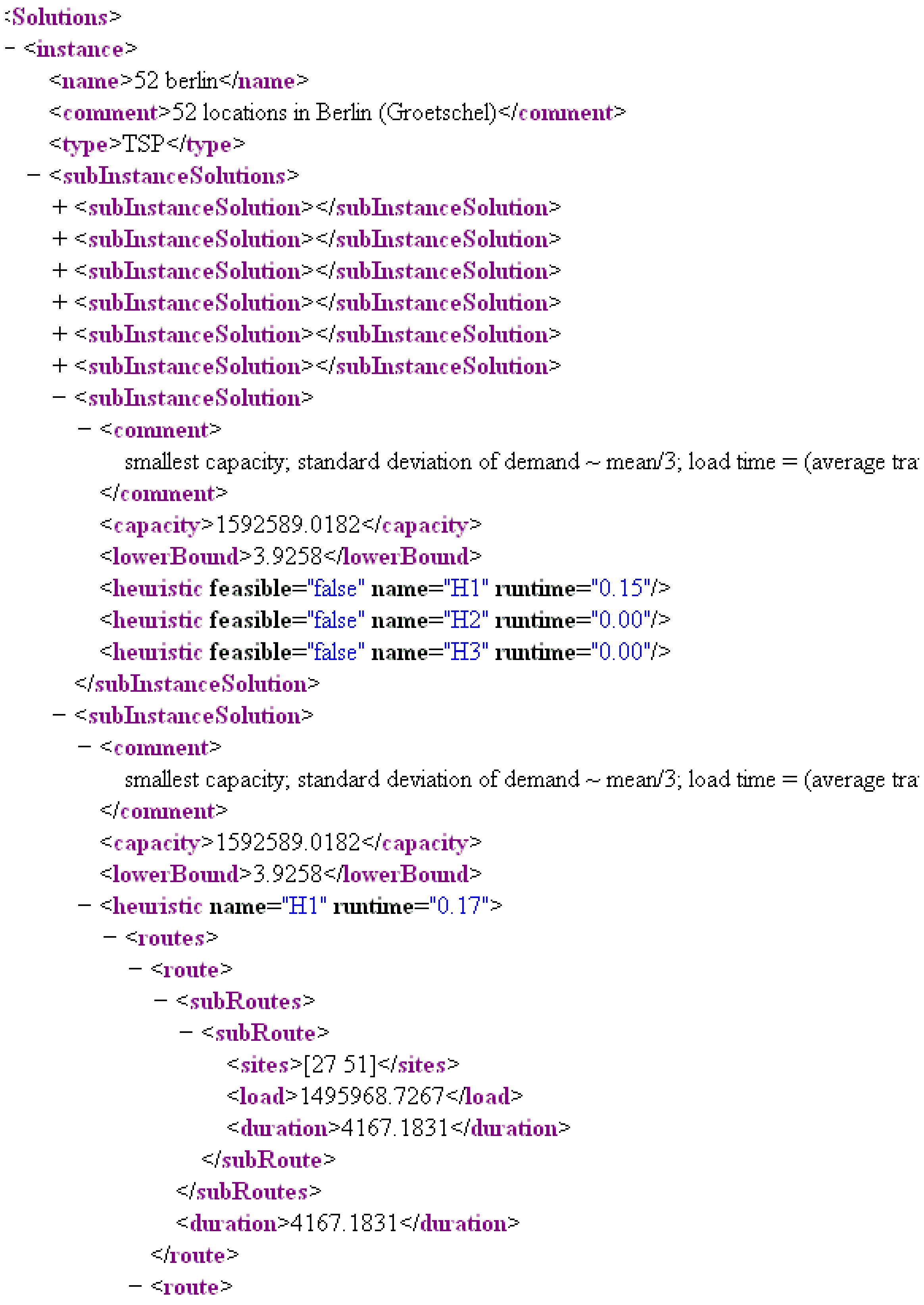
We applied each of the three heuristics to all 720 CRIRP problem instances. Figure 10 is an image of the beginning of the file that holds the test results. The instance node represents one of the original 4 TSP problem instances. Under the instance node, there are the following nodes:

- The “name” node which holds the name of the TSP instance (52 berlin);
- The “comment” node (under instance) with a description of the origin of the problem;
- The “type” node holds the problem for which the TSPLIB intended for the data;

- The “subInstanceSolutions” node made up of “subInstanceSolution” nodes each of which holds one of the 180 CRIRP combinations of data we generated for each TSP problem instance.
    - The “comment” node (under “subInstance”) which informs us of what combination of generated data this sub-instance holds
    - The capacity input
    - A theoretical lower bound to the number of vehicles needed
    - “heuristic” nodes that hold the detailed solution generated by each heuristic

The full results for all problem instances and heuristics are posted at the project website.

Figure 10: Sample view of heuristic test results



```
<Solutions>
- <instance>
    <name>52 berlin</name>
    <comment>52 locations in Berlin (Groetschel)</comment>
    <type>TSP</type>
  - <subInstanceSolutions>
    + <subInstanceSolution></subInstanceSolution>
    + <subInstanceSolution></subInstanceSolution>
    + <subInstanceSolution></subInstanceSolution>
    + <subInstanceSolution></subInstanceSolution>
    + <subInstanceSolution></subInstanceSolution>
    + <subInstanceSolution></subInstanceSolution>
    - <subInstanceSolution>
      - <comment>
          smallest capacity; standard deviation of demand ~ mean/3; load time = (average tra
        </comment>
        <capacity>1592589.0182</capacity>
        <lowerBound>3.9258</lowerBound>
        <heuristic feasible="false" name="H1" runtime="0.15"/>
        <heuristic feasible="false" name="H2" runtime="0.00"/>
        <heuristic feasible="false" name="H3" runtime="0.00"/>
      </subInstanceSolution>
    - <subInstanceSolution>
      - <comment>
          smallest capacity; standard deviation of demand ~ mean/3; load time = (average tra
        </comment>
        <capacity>1592589.0182</capacity>
        <lowerBound>3.9258</lowerBound>
      - <heuristic name="H1" runtime="0.17">
        - <routes>
          - <route>
            - <subRoutes>
              - <subRoute>
                  <sites>[27 51]</sites>
                  <load>1495968.7267</load>
                  <duration>4167.1831</duration>
                </subRoute>
              </subRoutes>
              <duration>4167.1831</duration>
            </route>
          - <route>
```

In addition to holding a solution, a heuristic node is annotated with other information such as the name of the heuristic used (*H1*, *H2* or *H3*), "*runtime*" which represents the amount of time in seconds, it took for the heuristic to generate the solution. If the heuristic failed to generate a solution, the heuristic is annotated with 'feasible="false"' as can be observed in the first sub-instance in the figure. When a heuristic finds a solution, it would have a routes node which contains vehicle routes. Under the routes node one can find the subroutes that make up a route as well as the following subroute information: subroute sites (ordered by precedence), required vehicle load and subroute duration.

The "XML" format in which this data is stored is a hierarchical format for storing text data. This allows the data to be retrieved and manipulated from any computer platform. One could easily transform such data into tables of vehicle routes. Such tables would list the subroutes for each vehicle and could be printed and handed to drivers.

Table 1 is a summary of the results, aggregated over different problem parameters. The categories in the table analyze how changes in the category value affect the solution. The numbers in parentheses associated with the categories represent the number of problem instances in the sub-category that have feasible solutions for all three heuristics. For example, the numbers "158/240" associated with small vehicle capacities signify that of 240 problem instances with small vehicle capacities, all three heuristics provide solutions in only 158 of them. The H3 heuristic gives the best heuristic solutions for most of the problem instances. The H2 heuristic performs worst, as it finds the best heuristic solution for only a few of the problem instances.

Table 1: Heuristic performance measures by category

	Category	Average Number of Vehicles			Number of Best Solutions			Time		
		H1	H2	H3	H1	H2	H3	H1	H2	H3
Vehicle capacity	small (158/240)	12.55	18.45	10.56	60	5	151	0.32	0.03	0.30
	medium (177/240)	5.13	7.06	3.98	103	9	173	0.32	0.03	1.36
	high (177/240)	3.94	5.58	3.04	132	7	175	0.32	0.03	1.21
Standard deviation	small (177/240)	7.46	10.85	6.19	110	10	170	0.34	0.03	0.91
	medium (172/240)	7.66	10.67	6.19	103	6	168	0.32	0.03	1.00
	high (163/240)	5.83	8.56	4.60	82	5	161	0.30	0.03	0.96
Load time	very small (135/180)	3.30	6.60	2.72	86	0	135	0.32	0.03	1.47
	small (135/180)	3.96	7.40	3.33	98	0	133	0.32	0.03	1.06
	medium (135/180)	7.42	10.57	5.99	78	5	129	0.32	0.03	0.81
	high (107/180)	15.00	17.15	12.02	33	16	102	0.32	0.03	0.48
Problem	14 burma (170/180)	3.53	5.09	3.31	133	5	165	0.02	0.01	0.10
	22 ulysses (167/180)	4.23	6.37	3.59	102	5	164	0.04	0.02	0.23
	52 berlin (105/180)	9.98	13.72	7.90	40	5	102	0.19	0.04	0.85
	127 bier (70/180)	17.63	25.46	13.11	20	6	68	1.03	0.06	2.63
Maximum route duration	10000000 (139/144)	6.69	12.17	5.63	89	1	138	0.33	0.05	1.63
	100000 (139/144)	10.04	13.17	7.77	69	7	136	0.33	0.05	2.28
	10000 (104/144)	7.22	8.72	5.68	49	6	100	0.32	0.02	0.65
	1000 (69/144)	3.84	5.87	3.41	49	1	68	0.31	0.01	0.07
	100 (61/144)	4.03	5.20	3.66	39	6	57	0.32	0.01	0.14

Table 2 evaluates the relative performance of the heuristics by computing the difference between the number of vehicles in the solutions of each possible pair of heuristics. For example, a column with “H2-H1” signifies the difference between the number of vehicles needed by the solutions of the H2 and H1 heuristics. We compute the median, average and relative standard deviation of these paired differences aggregated over different values of the problem parameters. These results confirm the results of Table 1. The H3 and H1 heuristics consistently result in fewer vehicles than the H2 heuristic. H3 also results in fewer vehicles than H1 and does this with a low relative standard deviation.

Table 2: Heuristic relative performance by category

	Category	Median			Average			σ/Average		
		H2-H1	H2-H3	H1-H3	H2-H1	H2-H3	H1-H3	H2-H1	H2-H3	H1-H3
Vehicle capacity	small (158/240)	3	4	1	5.90	7.89	1.99	0.92	4.40	0.06
	medium (177/240)	2	2	0	1.93	3.08	1.15	1.27	2.27	0.17
	high (177/240)	2	2	0	1.64	2.54	.90	1.37	1.98	0.21
Standard deviation	small (177/240)	2	3	0	3.40	4.66	1.27	1.38	4.92	0.07
	medium (172/240)	2	2	0	3.01	4.48	1.47	1.16	4.04	0.12
	high (163/240)	2	2	0	2.74	3.96	1.23	1.07	3.88	0.11
Load time	very small (135/180)	2	2	0	3.30	3.88	.59	1.05	7.60	0.06
	small (135/180)	2	2	0	3.44	4.07	.64	1.09	7.60	0.07
	medium (135/180)	2	2	0	3.15	4.58	1.43	1.10	3.95	0.10
	high (107/180)	2	3	1	2.15	5.13	2.98	1.52	2.59	0.08
Problem	14 burma (170/180)	2	2	0	1.56	1.78	.22	0.49	3.25	0.72
	22 ulysses (167/180)	2	3	0	2.14	2.77	.63	0.42	1.95	0.92
	52 berlin (105/180)	4	5	1	3.74	5.82	2.08	0.75	1.69	0.14
	127 bier (70/180)	7.5	7.5	3	7.83	12.34	4.51	0.99	2.49	0.03
Maximum route duration	10000000 (139/144)	3	4	0	5.48	6.55	1.06	0.88	6.90	0.08
	100000 (139/144)	2	3	1	3.13	5.40	2.27	1.37	3.05	0.07
	10000 (104/144)	2	2	1	1.50	3.04	1.54	1.11	1.78	0.21
	1000 (69/144)	2	2	0	2.03	2.46	.43	0.35	2.49	1.37
	100 (61/144)	1	1	0	1.16	1.54	.38	0.87	2.41	0.41

We also see the following trends:

- Not surprisingly, as vehicle capacity increases, the number of vehicles needed reduces.
- The performance of the H1 heuristic approaches that of H3 as vehicle capacity increases or as problem size decreases.
- The number of vehicles required for the solution increases with both the load times and the problem size (number of PODs).

Appendices A and B respectively show the detailed problem data and heuristic solutions for three problem instances. These are 3 instances of the Burma 14 problem having all parameters the same except for the vehicle capacity. Appendix B shows the results of the three instances by each of the three heuristics. The solutions generated by the H2 heuristic have routes consisting of one subroute each. For these instances, the H1 and H3 perform equivalently to each other and better than H2. Both the H1 and H3 are constituting routes with trivial subroutes. PODs are combined into non-trivial subroutes only if their demands are smaller than that of the trivial subroutes.

4.4.3 Application to the State of Maryland

To provide a more realistic test of the heuristics, this section presents a solution for an instance of CRIRP generated using data obtained from the state of Maryland about the dispensing plans of 3 counties with a total of 189 PODs in the Washington, D.C., suburban area. The data gave the travel times between POD locations, the anticipated demand at each POD, and other information about the trucks available to deliver medication. For convenience, some of this data is shown in this section.

Table 3 shows the planning timeline; from it, we discern that the actual delivery of supplies to the PODs should last 24 hours.

Table 3: State of Maryland scenario: timeline

Scenario Timeline		
Hour	**Event**	**Regimens Delivered to Depot**
-4	Attack detected	
0	Federal and state decision to dispense	
12	Push pack trucks arrive	324,000
14	Managed inventory--first shipments arrive	324,800
16	next shipment	313,600
18	next shipment	313,600
20	next shipment	313,600
22	next shipment	324,800
24	Managed inventory--last shipments arrive	313,600
24	PODs begin dispensing operations	
48	PODs scheduled to complete dispensing operations	

Table 4 shows how PODs are distributed in through the counties and the number of regimens required per POD. By dividing the required number of regimens by the time allotted for distribution (24 hours) we obtain demand rates of 4.65, 5.56 and 29.44 regimens per minute for the PODs in counties A, B and C respectively.

Table 4: State of Maryland Scenario: Regimen requirements per POD

County	Population	PODs	Regimens per POD
A	248,000	37	6,700
B	1,040,000	130	8,000
C	932,800	22	42,400
Total	**2,220,800**	**189**	

We had to modify the data to create CRIRP instances. Since our implementation of the CRIRP assumes symmetric travel times, we adjusted the asymmetric travel times provided by using the higher travel time in both directions. Although the state has a heterogeneous fleet of trucks, our model assumes a homogenous fleet. Furthermore, in the given scenario, county B has a local depot. In conformity with the CRIRP model developed in this thesis, we ignore the local depot and supply all PODs from the central depot. We solved two instances of the problem using each of the vehicle sizes in the state's fleet (53 and 20 foot trucks).

Table 5 presents a summary of the results obtained for all the heuristics. The H3 heuristic gives the best results for both the 53-foot trucks and 20-foot trucks. Note that a 53-foot truck can hold 268,800 regimens, and a 20-foot truck can hold 112,000 regimens. By applying the CRIRP to the state's scenario, we thus generate routes and the associated number of vehicles which are within the limits of the state of Maryland's fleet of forty-nine 53-foot trucks and twenty-two 20 foot trucks (subject to choosing the appropriate maximum route duration). It is clear from the results that by allowing a longer route duration, we can reduce the number of vehicles required. At low route durations, the 53 foot truck has excess capacity.

Table 5: State of Maryland scenario: solution summary

Max Route Duration (hours)	**Number of Vehicles**					
	20 foot truck			**53 foot truck**		
	H1	**H2**	**H3**	**H1**	**H2**	**H3**
4	-	-	-	-	-	-
6	-	48	32	-	48	32
8	41	29	18	41	29	18
12	23	16	11	23	16	10
24	11	13	9	11	8	6

4.5 *<u>Summary</u>*

In this chapter we developed three heuristics for solving the CRIRP and tested them on a variety of test problems. The results of our tests were generally as expected. The H3 heuristic which takes advantage of the CRIRP special case and utilizes CVRP techniques performed best, while the H2 heuristic (using random sequences of PODs) performed worst. Runtime data showed us that the H3 heuristic takes longer to run with the time increasing rapidly with problem size. While more runtime studies may be needed for much larger problems, for the application at hand, heuristic runtimes of a few seconds are not an issue.

Finally, we applied the heuristics to a scenario from the State of Maryland and found feasible solutions that can supply the designated counties with the available fleet. The results also serve as a tool to help health planners decide how to balance between short route durations and the number of required vehicles. Our results also demonstrate that in some cases, low-capacity vehicles are sufficient and high capacity vehicles do not need to be committed.

Chapter 5: EXACT APPROACHES

In Chapter 4 several heuristics were developed and applied to solve the CRIRP. These heuristics use our understanding of the problem and its properties to obtain solutions that we hope are close to the optimum. Each vehicle must be assigned one set of PODs among all the possible sets that may be assigned. Given a vehicle and a set of PODs, a solution chooses one of several possible routes. Even for small problems, there are many ways to assign PODs to vehicles and sequence the PODs assigned to a vehicle to form a route. The CRIRP is a combinatorial problem. In this chapter, we develop an exact approach for determining the optimal solution by applying branch and bound techniques as we span the entire potential solution space for all possible solution combinations and find an optimum.

5.1 *An Exact Approach for the CRIRP*

The goal of the CRIRP is to minimize the number of vehicles. Since the CRIRP does not allow a POD to be supplied by more than one vehicle, clearly, the maximum number of vehicles is equal to the number of PODs. Suppose that we have a problem with n PODs. Our initial solution would have n vehicles. By progressively decreasing the size of the vehicle fleet and trying to supply the PODs with the currently reduced fleet of vehicles, eventually we reach a point at which it is impossible to supply the PODs with the current fleet size, say k ($< n$). If the PODs cannot be supplied by k vehicles, it is also infeasible to use less than k vehicles. Therefore $k + 1$ is the minimum number of vehicles. Instead of using n vehicles as the starting point of this procedure, we can quickly get a head-start by using some of the heuristic approaches discussed in Chapter 4 to start with a smaller fleet.

To show that a certain number of vehicles *r*, is sufficient to supply the PODs it is sufficient to find one feasible solution that uses *r* vehicles. However, to show that *r* vehicles are insufficient to supply the PODs it is necessary to explore and rule out all possible assignments of PODs to vehicles. To rule out the feasibility of a set of PODs being associated with one vehicle, it is necessary to explore all routing possibilities amongst the depot and the PODs and show that they are all infeasible. For each fleet size under consideration a branching mechanism is needed to explore all of the possible assignments of PODs to vehicles.

5.2 <u>Assigning PODs to Vehicles</u>

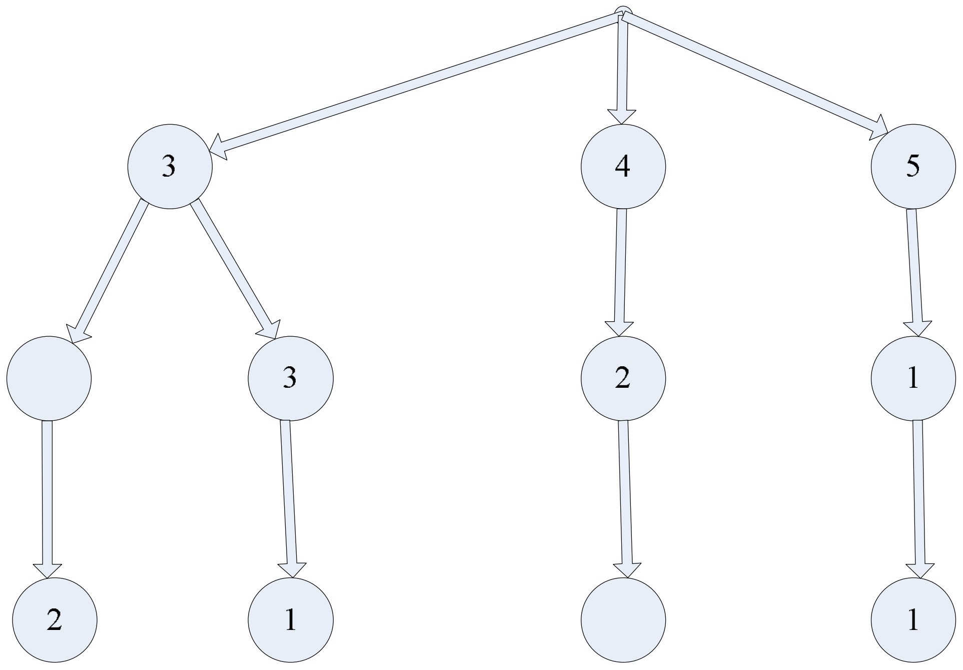
In this section we develop a method for generating all the possible ways of assigning PODs to vehicles.

5.2.1 Size Sequence Branching

We know that vehicles are indistinguishable and vehicle routes are independent. In particular, there are no ordering or precedence relationships among routes. However, for the purposes of this method, let the vehicles be listed in decreasing order of the route size (number of PODs they supply). Because every vehicle is assigned at least one POD, the first vehicle can be assigned a maximum of $n-(r-1)$ PODs where n is the number of PODs and r is the number of vehicles. (The case in which a vehicle is assigned zero PODs is same as having one less vehicle, which is the direction this method would take if we succeed to find a feasible solution with the current number of vehicles. We therefore need not be concerned about vehicles with zero PODs.) Due to the ordering we impose, the first vehicle is always assigned the greatest number of PODs and each vehicle has at least the same number of PODs as the subsequent PODs. This implies that the first vehicle has the minimum possible number of PODs when all routes have the same number of PODs, which is n/r. Because the number of PODs assigned to a vehicle is an integer and no vehicle can have more PODs than the first, if n/r is not an integer, it is rounded up to the next higher integer. Therefore the possible number of PODs in the first vehicle of a solution must be an element of the set: $\left\{ceiling\left(n/r\right), ceiling\left(n/r\right)+1, \ldots, n-r, n-r+1\right\}$. Having assigned the number of PODs to the first vehicle, we turn to subsequent PODs. Suppose that the vehicle that precedes that under consideration has been assigned *m* PODs and that there are n' PODs left to be assigned to r' vehicles. The maximum number of PODs that can be assigned to the next vehicle is the smaller of *m* and $n'-(r'-1)$. Similar to the first vehicle, the minimum number of PODs that can be assigned to the next POD is $ceiling\left(n'/r'\right)$.

As an example, consider a CRIRP with 7 PODs that we want to fit in 3 routes. The possible number of PODs per vehicle is shown in Figure 11. The first nodes from the root in the figure represent the possible number of PODs for the first vehicle. The number of leaf nodes represent the total number of possible route size sequences which are {3,2,2}, {3,3,1}, {4,2,1} and {5,1,1}.

Figure 11: Size sequence branching



5.2.2 Method of Combinations

The method described in Section 5.2.1 generates all the possible route sizes for a given vehicle fleet size. For each of these size sequences, there are several ways to assign PODs to vehicles. Each of the paths in Figure 11 is thus a root for the assignment of PODs to vehicles. We assign PODs to vehicles, one vehicle at a time.

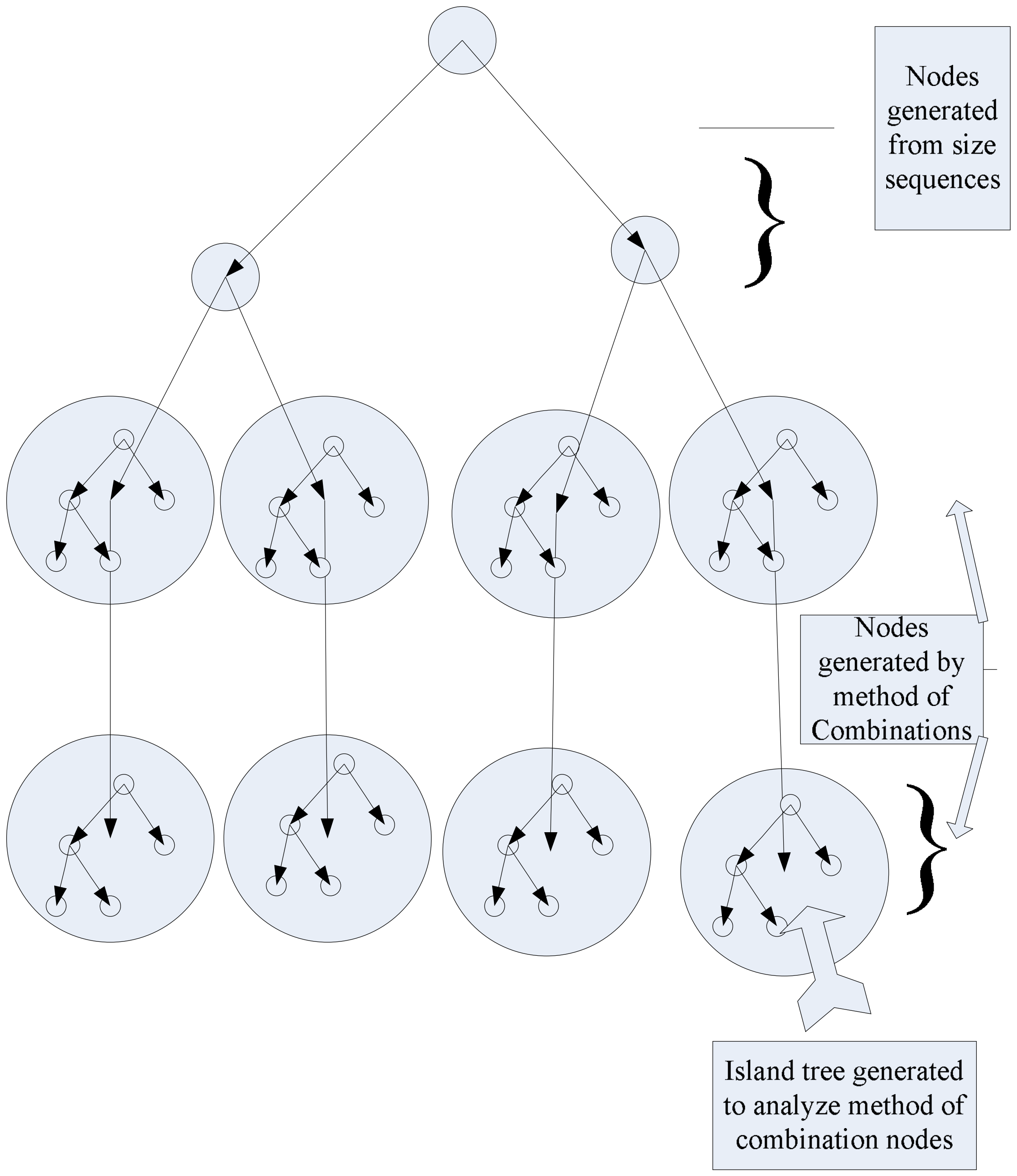
Suppose we have n PODs to assign to r vehicles according to the size sequence $\{s_1, s_2, ..., s_r\}$ $\left(\sum_{i=1}^{r} s_i = n\right)$. If $s_i \neq s_j$ for every $i, j,\ i \neq j$, that is, the sizes in the sequence are distinct, then there are $C_{s_1}^{n}$ distinct ways of assigning PODs to the first vehicle. $C_{s_1}^{n}$, which is associated with the first vehicle, is also the number of nodes that descend from the leaf node of the size sequence tree. For each of these nodes, there are $C_{s_2}^{n-s_1}$ ways of assigning PODs to the second vehicle. Similarly, $C_{s_2}^{n-s_1}$ represents the number of nodes that descend from each of the nodes associated with the first vehicle. Proceeding with this train of thought, there are $C_{s_k}^{n-\sum_{i=1}^{k-1} s_i}$ ways of selecting vehicles for the kth vehicle. We call this scheme for generating nodes of a vehicle as children nodes of the preceding vehicle the "*method of combinations*".

Let us consider the case where the sizes are not distinct. For example, let us attempt to fit seven PODs in three vehicle routes with size sequence {3, 3, 1}. Using the method of combinations just described we would obtain the following POD allocations {[1,2,3],[4,5,6],[7]}, {[4,5,6],[1,2,3],[7]} among several others. In the first allocation, PODs 1, 2, and 3 are assigned to vehicle 1 and PODs 4, 5 and 6 are assigned to vehicle 2. In the second allocation, PODs 4, 5 and 6 are assigned to vehicle 1 and PODs 1, 2 and 3 are assigned to vehicle 2. Because all of the vehicles are identical, these two allocations are identical and would lead to extra, unnecessary branches in a tree containing identical solutions. To prevent this from happening, we combine sizes in the size sequence that have the same value and form a *size-group*. Size sequences are thus transformed to *size-group sequences*. As an example, the {3, 3, 1} size sequence becomes a {6, 1} size-group sequence. We can now use the *method of combinations* to generate nodes from PODs. However, when using size-group sequences, a node represents one or more vehicle routes of the same size.

5.2.3 Analyzing the Nodes Generated by the Method of Combinations

Each of the nodes added to the solution tree by the method of combinations represents one or more vehicle routes (having the same size) and the PODs to be assigned to the vehicles. If there are several vehicles in the node, we still need to explore all the possible allocations of PODs to vehicles. To assign PODs to vehicles, we follow a procedure similar to the method of combinations. The difference here is that the vehicle routes have the same size. Consider fitting *ps* PODs in *p* routes each of size *s*. For the first route, because the sites are similar, we can arbitrarily select any POD as the first POD in the route. There are thus C_{s-1}^{ps-1} ways of selecting PODs for the first vehicle route. For each of these selections, there are C_{s-1}^{ps-s-1} ways of assigning PODs to the second vehicle. Similarly, there are $C_{s-1}^{ps-(k-1)s-1}$ ways of selecting vehicles for the k^{th} vehicle. We call this the "*modified method of combinations*". This results in a tree structure that resides in the nodes generated by the method of combinations. Take note that these trees are not branches in the overall solution tree but are island trees that reside within its nodes. The nodes of these island trees each represent a single vehicle route and the PODs that would be supplied on that route. This level of branching of the solution tree structure (which we call the route tree) ends at POD assignments to individual routes and is shown in Figure 12.

Figure 12: Analysis of the nodes generated by the method of combinations results is island nodes



Nodes generated from size sequences
Nodes generated by method of Combinations
Island tree generated to analyze method of combination nodes

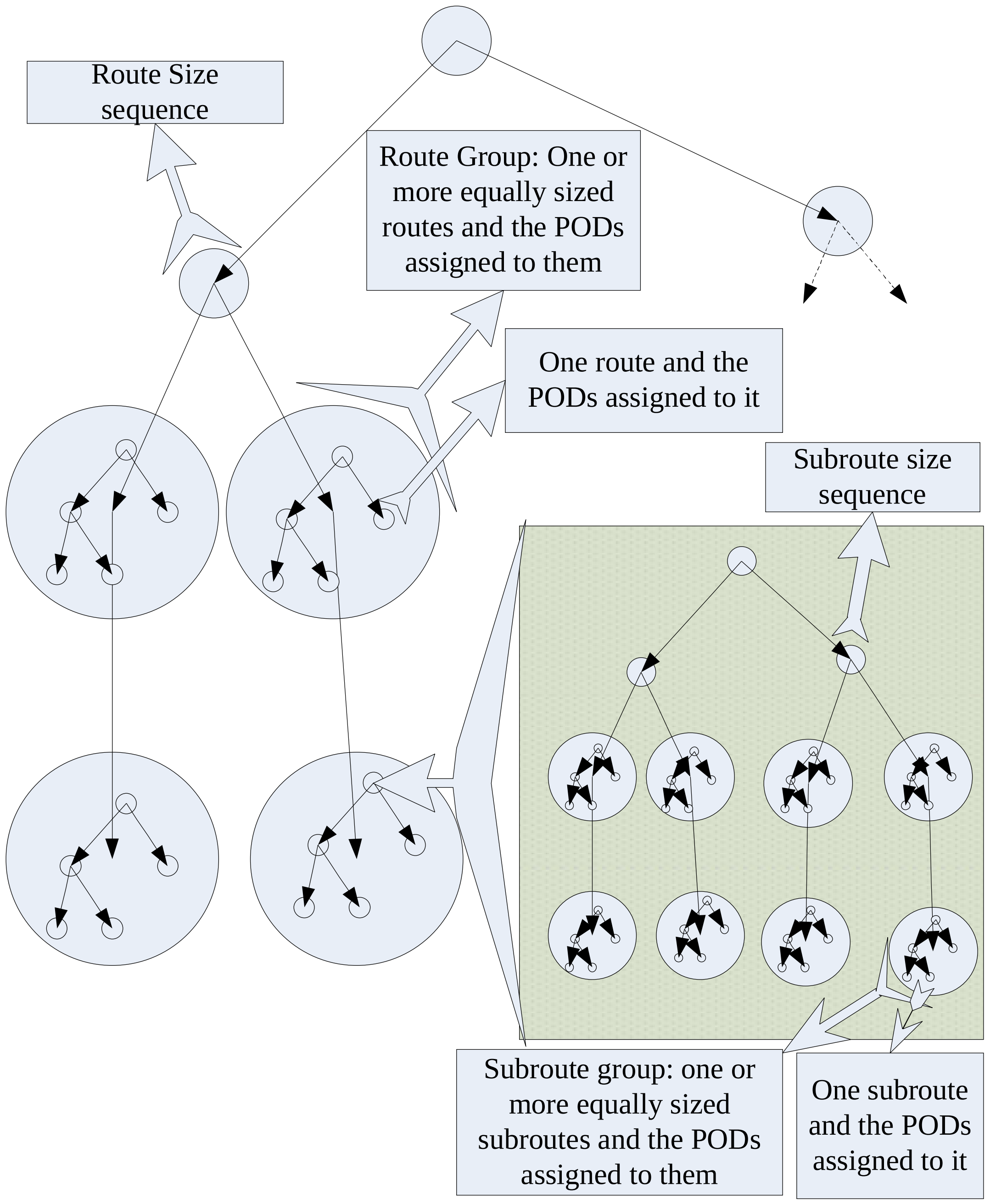
5.3 Routing a vehicle to a set of PODs

From the method described in Section 5.2, we have all of the possible route-PODs combinations i.e. all the possible ways that PODs can be assigned to routes. These are represented by the nodes of the island trees shown in Figure 12. The PODs on a vehicle route can belong to one more subroutes. This section addresses methods for determining all the possible ways of assigning PODs on a route to subroutes.

5.3.1 Assigning PODs to Subroutes

Essentially, we will follow the same procedure we did in assigning vehicles to routes. The number of subroutes on a route ranges from a minimum of one to a maximum equal to the number of PODs intended for the route. The minimum corresponds to having all PODs on one subroute, while at the maximum each subroute would only supply one POD. Like the routes, the order in which the subroutes are supplied is irrelevant. However, for the purposes of this method we order the subroutes in decreasing order of the number of PODs supplied. Just like we did for routes, for each of the possible number of subroutes, we generate all the size sequences. We convert size sequences into size-group sequences and apply the method of combinations. We apply the modified method of combinations to the nodes that result from the method of combinations. This results in a structure exactly the same as that depicted in Figure 12. The difference here is that the nodes of the island trees represent subroutes and these are the inner-most nodes in our solution tree. Also, the entire tree structure we have generated in this section (which we call the subroute tree) resides inside the each of the nodes of the island trees of the previous section. The resulting tree is shown in Figure 13.

Figure 13: The solution space tree


Route Size sequence
Route Group: One or more equally sized routes and the PODs assigned to them
One route and the PODs assigned to it
Subroute size sequence
Subroute group: one or more equally sized subroutes and the PODs assigned to them
One subroute and the PODs assigned to it

There is another major difference between the route tree and the subroute tree. Within the route group nodes of the route tree, each path of the enclosed island trees represents a possible assignment of PODs to a number of equally-sized vehicle routes. If we find one such feasible path, the enclosing route group node is feasible independent of parent and children route group nodes. We have no need to explore more island tree paths within the route group node. The converse is true in the case of the subroute tree. A subroute is feasible only in the context of the route to which it belongs. We cannot therefore say that a subroute group node by itself is feasible. We must compare all the combinations of the paths of the enclosed island tree of one subroute group to the paths of the island trees of other associated subroute group nodes.

To illustrate this point, suppose we want to fit PODs 1 to 6 in two routes each of size 3 (the first route group) and PODs 7 to 10 in two routes each of size 2 (the second route group). Furthermore, suppose that in the first route group the PODs are feasibly assigned to routes as follows: {1, 2, 3} and {4, 5, 6}. Independent of the second route group, we do not need to explore other ways of assigning the PODs to the routes in the first. Consider a similar scenario for the subroute tree. We want to constitute a route by assigning PODs 1 to 6 to two subroutes each of size 3 (the first subroute group) and PODs 7 to 10 to two subroutes each of size 2. Recall that the feasibility of a route depends on the maximum subroute demand and the total route length (that is, the sum of all the subroute lengths). We can therefore make no conclusions about one subroute group without taking associated subroute groups into consideration. In fact, to conclude that this route of 10 PODs is infeasible, we must consider and rule out all the possible combinations of subroute assignments in the first subroute group with those in the second.

5.3.2 Obtaining the Subroute Path

From Section 5.3.1, all possible PODs assignments to subroutes are known. We must now determine the best path that a vehicle should take as it leaves the depot, visits each of the PODs on the subroute before returning to the depot. The best subroute path is that which is shortest. Choosing the best subroute path thus reduces to solving the well-known TSP.

5.4 Bounding and Pruning

Thus far in this chapter we have used branching to subdivide the solution space down to the lowest level. This allows us to explore all the possible ways of assigning PODs to subroutes (in the subroute tree) and subroutes to routes (in the route tree) for any solution size (number of vehicle routes). In an effort to reduce the computational burden, this section seeks methods for eliminating tree branches from the solution space.

5.4.1 Lower Bound for the Number of Routes

By applying heuristics, an upper bound of the optimal fleet size (which can be improved during branching) is obtained. When we apply the method of combinations or the modified method of combinations, we attempt to fit a given set of PODs into routes for a given number of vehicles (based on the fleet size upper bound). For a particular branch in the tree, if we can show that the given set of PODs cannot fit in the given number of vehicle routes, we can abandon that branch because it does not have a solution. In this section we determine a lower bound for the number of vehicles required to supply a given set of PODs.

Suppose that a vehicle has the ability to supply a set of PODs with known demand rates. If we reduce the demand rates of some PODs, the vehicle should still be able to feasibly supply those PODs. To obtain a lower bound for the number of vehicles required, we set all the POD demand rates to the minimum demand rate in the set. This reduces to an instance of the special case (Section 3.3) and is a BPP problem. To further simplify the BPP, we fill the bins as if quantities are continuous. For example, let the minimum POD demand for a set of PODs be 200 *kg/min*; let the trivial subroute durations be 20, 35, 35 and 45 *minutes* and the vehicle capacity be 10000 *kg*. Then the maximum allowable route length that allows a truck to meet demand is $\frac{10000}{200} = 50\,min$. The minimum number of vehicles is the number of bins of capacity 50 *min* in which we can pack the subroute durations. To find the lower bound we simply take $\frac{20+35+35+45}{50} = 2.7$, which is rounded up to yield a lower bound of 3 vehicles. If this lower bound is more than the number of vehicles for which the set of PODs is destined, this branch is infeasible.

5.4.2 Pruning Routes

During bounding, we eliminate branches that cannot yield a solution with less than optimal fleet size upper bound. Sometimes, it is also possible to show that there is another solution combination that is always better that the current branch. If *A* always performs worse than *B*, why waste time on *A* when either *B* had failed to do the job or we are destined to encounter *B*? In the case of fitting PODs in routes, consider two possible solutions. The first *k* routes of both are identically filled. The *k*+1th route of solution *A* has PODs 1, 2 and 3 with 10 PODs left to fit in 3 routes. The *k*+1th route of solution *B* has PODs 1, 2, 3 and 4 with 9 PODs left to fit in 3 routes. Solution B in all cases would do better than solution A. If we encountered solution A, we may therefore abandon it because when we get solution B, we are guaranteed better results.

To determine if a better solution exists, for each route that we have filled with a set of PODs, we look at the PODs left (destined for subsequent routes) and determine if any of them can be feasibly added to this route. If so, we abandon this branch. To reduce the computational expense of determining the addition of a POD, heuristics could be used.

5.4.3 Capacity Lower Bound

In the subroute tree, we attempt to fit a given set of PODs in one vehicle route. In this section we determine a lower bound to the capacity required by a set of PODs on a route. If we can show that this lower bound exceeds the vehicle capacity, then the route is infeasible. Recall the vehicle capacity constraint $D_v^* T_v \leq C$ from Chapter 3. D_v^* is the maximum subroute demand of vehicle *v*, T_v is the route duration and C is the vehicle capacity. $D_v^* T_v$ represents the required route capacity which must not exceed C. To determine a lower bound for the required vehicle capacity, we use the multiple of the lower bounds of D_v^* and T_v.

The maximum single POD demand in the set is a lower bound for D_v^*. Since a vehicle must visit all the PODs on its route, we can utilize the cost of the minimum spanning tree of all the PODs and the depot as a lower bound. The travel times are used as the path costs in the spanning tree. In a spanning tree, each vertex is connected by only one path. In our scenario, a vehicle route must include both a departure from and a return to the depot. Therefore the depot has at least two paths. To obtain a better lower bound, we add the cost of the lowest cost depot path to the cost of the minimum spanning tree.

5.4.4 Subroute Demand and Duration Lower Bound

Consider a CRIRP problem in which the vehicle capacity, C = 1000 kg. The route assigned to vehicle v has duration T_v = 30 minutes and maximum subroute demand D_v^* = 20 kg/minute. The route is feasible because $D_v^* T_v \le C$. Now suppose we want to add a subroute group of 2 subroutes, each of size 2 constituted of PODs 1, 2, 3 and 4. Let the possibilities for the subroute group be as follows:

- {1,2},{3,4} with total duration = 30 minutes and the greater subroute demand = 35 kg/minute
- {1,3},{2,4} with total duration = 35 minutes and the greater subroute demand = 32 kg/minute
- {1,4},{2,3} with total duration = 33 minutes and the greater subroute demand = 25 kg/minute

To obtain a lower bound for this subroute group, simply select the smallest duration and the smallest maximum subroute demand, that is 30 minutes and 25 kg/minute respectively. Before checking each of the listed possibilities for addition to the route, we use the lower bound. To constitute the augmented route with the lower bound values, the new value for T_v=30+25=55 minutes and for D_v^*=max (25, 20) =25 kg/minute.

$D_v^* T_v = (25)(55) = 1375 > C$. Since the lower bound is not feasible, we need not test each of the listed possibilities.

5.4.5 Pruning Subroutes

In the subroute tree, we can also prune branches if it can be shown that a better solution exists. From Section 3.4 (Combining Subroutes with Unequal Demand), if there are two subroutes whose combined demand is less than the maximum subroute demand, we obtain a better solution by combining them into one subroute. When this situation is encountered in the subroute tree, the applicable branch may be abandoned.

5.5 Other Computation Saving Techniques

5.5.1 Storing Partial Solutions

Consider the two CRIRP solutions that follow: {[1,2,3,4], [5,6],[7,8]} and {[1,2,3,4], [5,7],[6,8]} where the contents of the square brackets are the PODs in a route. Both solutions assign the same set of PODs to the first route. In general several solutions could have a lot of common routes and subroutes, which once computed are stored and available for retrieval.

5.5.2 Storing Failed Attempts

At a branch in the solution tree, suppose we want to fill 3 vehicle routes with PODs 1 – 10 according to the size sequence {5, 3, 2}. That is, 5 PODs in the first route, 3 second and 2 in the third. There are C_5^{10} ways (which represent children nodes at the branch) of selecting PODs for the first route. On the one hand, all of these trials could fail and the following information stored: “It is impossible to constitute a route of 5 PODs using PODs 1 – 10.” On the other hand we may successfully create several possible routes of 5 PODs. Each of these would have C_3^5 children nodes, which represents the number of ways of constituting the second route given a feasible first route. If all of the nodes that successfully created the first route of 5 PODs failed to create a second route of 3 PODs, the following information is stored: “It is impossible to constitute 2 routes of sizes 5 and 3 using PODs 1 – 10.”

The results of such failed attempts are useful in eliminating other branches. Let us revisit the last example: it is impossible to constitute from PODs 1 – 10, routes of sizes 5 and 3. PODs 1 – 10 would equally fail to constitute routes of the following sizes: {5,4,1}, {6,3,1}, {5,3,1,1}. In general, suppose it is impossible to fill routes of sizes $\{a_1, a_2, ..., a_m\}$ from a set of PODs. For the same set of PODs, we may rule out routes of sizes $\{b_1, b_2, ..., b_n\}$ if the following conditions are met:

- $n \geq m$
- $b_i \geq a_i$ for all $i \leq m$

5.5.3 Multilevel Search

To show that a certain number of vehicles *r* is sufficient to supply the PODs it is sufficient to find one feasible solution that uses *r* vehicles. A lot of computational resources may be expended on one branch of the solution tree while a computationally inexpensive solution exists on the next branch. The premise of this section is that applying heuristic methods on tree branches would result in locating the solutions that are easy to find more quickly. The multilevel search has three levels of increasing complexity with the full solution tree of Figure 13 being the most complex (level 3). At level 1, when we get to the point at which we need to assign a set of PODs to one vehicle (island nodes of the route tree), we use heuristics instead of using a subroute tree. At level 2, we use the subroute tree. However, when we get to its island nodes (how to connect PODs into a subroute) we use a TSP heuristic instead of solving the TSP optimally. In order to show that r vehicles are insufficient to supply the PODs, it is not sufficient for the solution tree to fail at levels 1 or 2. It must fail at level 3.

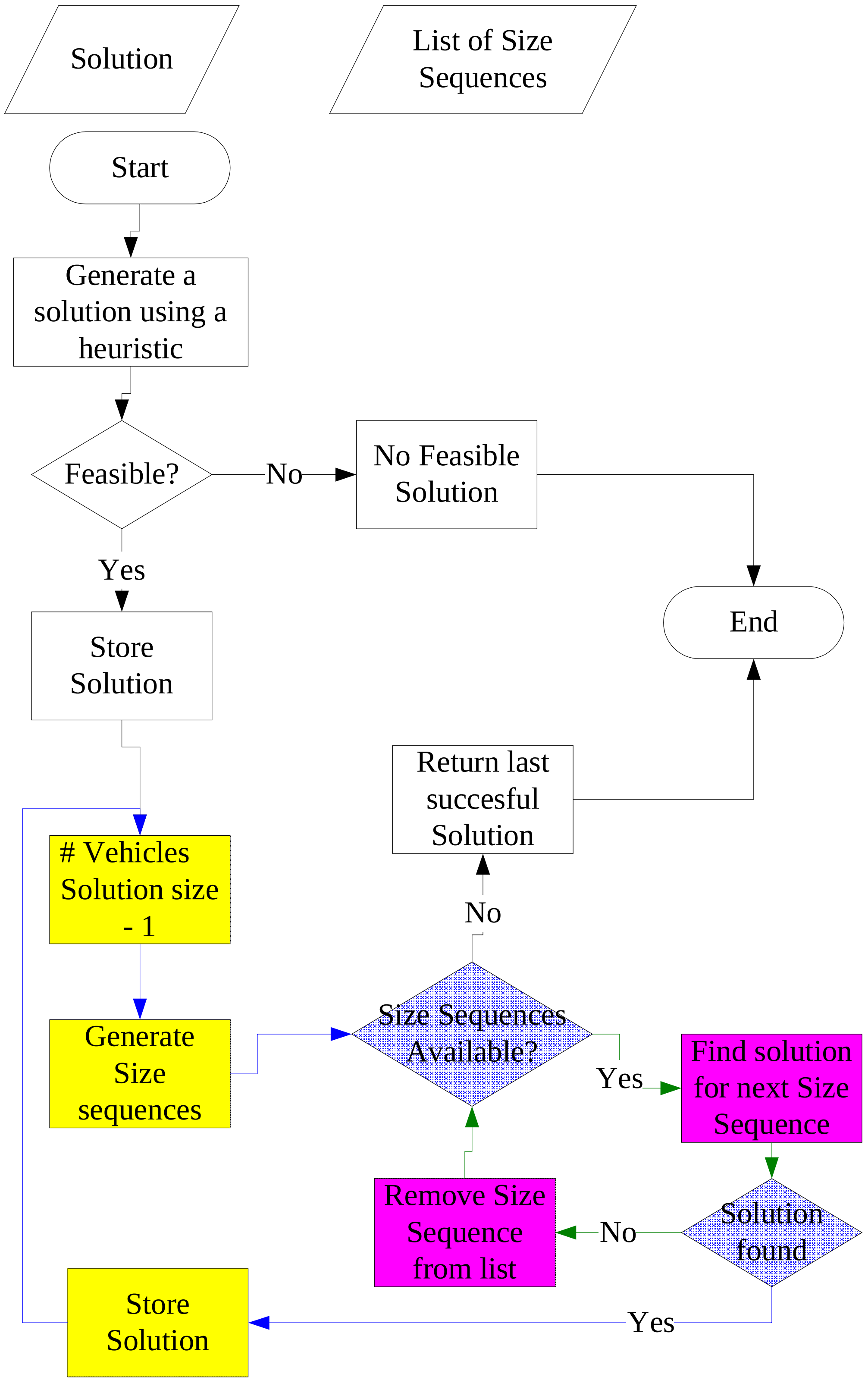
5.6 <u>Algorithms and Implementation</u>

Like the heuristics, our implementation of the exact approach was coded mainly in MATLAB programming language. MATLAB's built-in java interface was also used to create and utilize custom java objects for storing computed routes, subroutes and infeasible size sequences for sets of PODs.

For its first step, the branch and bound algorithm (Figure 14) calls a heuristic to find an upper bound to the optimal number of vehicles. The heuristic called must not return an infeasible solution if a feasible solution exists. The H2 and H3 heuristics are guaranteed to return a feasible solution if one exists. These algorithms can do this because ultimately, they only fail if one POD cannot be fitted into a route. The H1 heuristic does not meet this criterion because it first packs PODs into subroutes and when one such subroute fails to be packed into a route, the heuristic fails. It may however be possible to break up such a subroute to constitute several routes. This implementation of the branch and bound uses the H3 heuristic to obtain an initial upper bound for the optimum number of vehicles.

The initial heuristic solution is stored and the number of vehicles is set to one less than the heuristic solution. Using the new solution size, vehicle size sequences are then generated as described in Section 5.2.1. For each size sequence, the algorithm in Figure 15 is called to generate a solution. If any such call returns a feasible solution, the solution is stored and the solution size is again reduced to seek a better solution. If all the sequences fail, the last feasible solution found is an optimal solution and is returned.

Figure 14: Top level branch and bound Algorithm



Solution
List of Size Sequences
Start
Generate a solution using a heuristic
Feasible?
No
No Feasible Solution
Yes
End
Store Solution
Return last succesful Solution
# Vehicles Solution size - 1
No
Generate Size sequences
Size Sequences Available?
Yes
Find solution for next Size Sequence
Remove Size Sequence from list
No
Solution found
Store Solution
Yes

Figure 15: Algorithm that given a sequence of vehicle route sizes (size sequence) generates a feasible solution

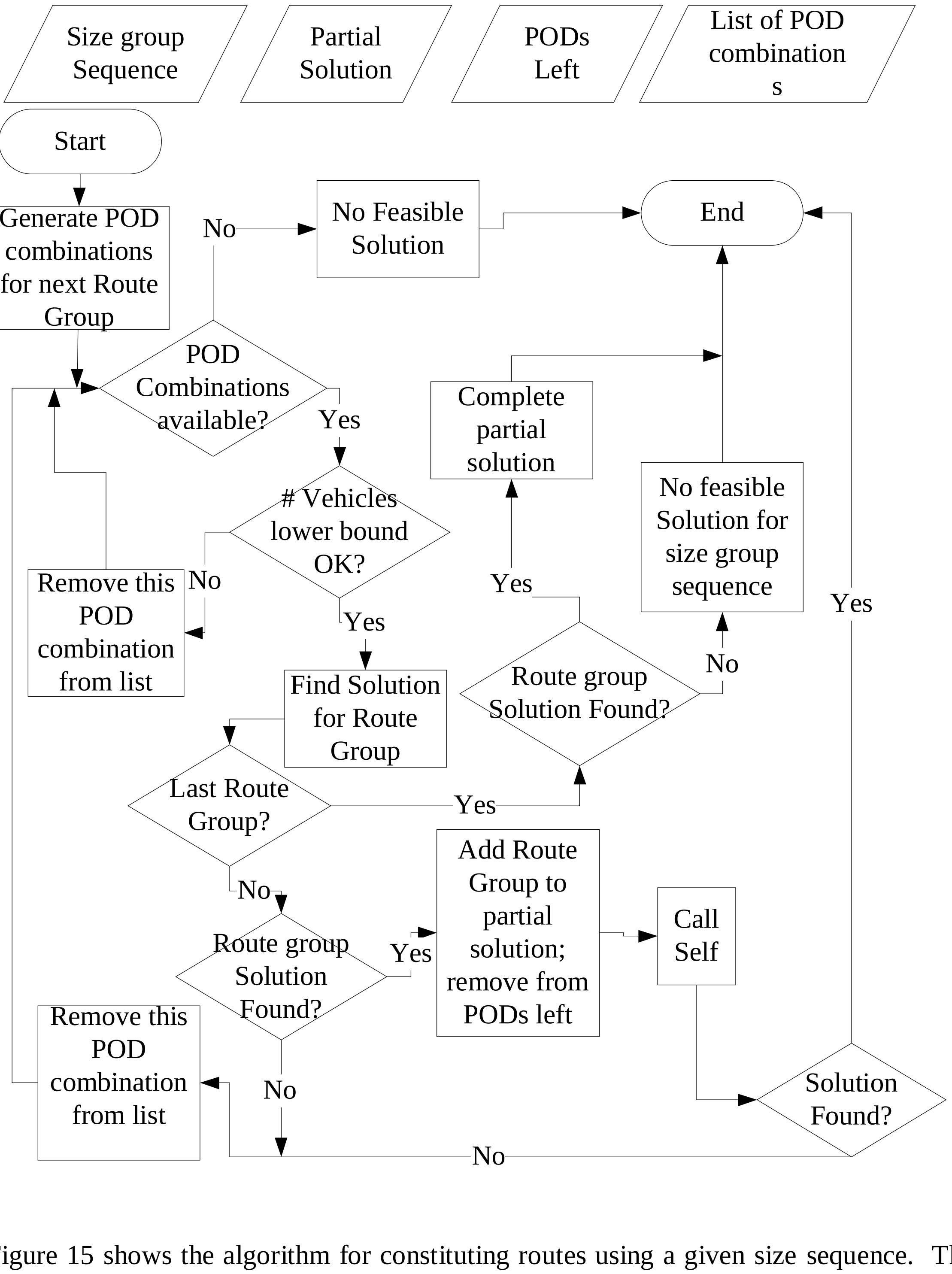
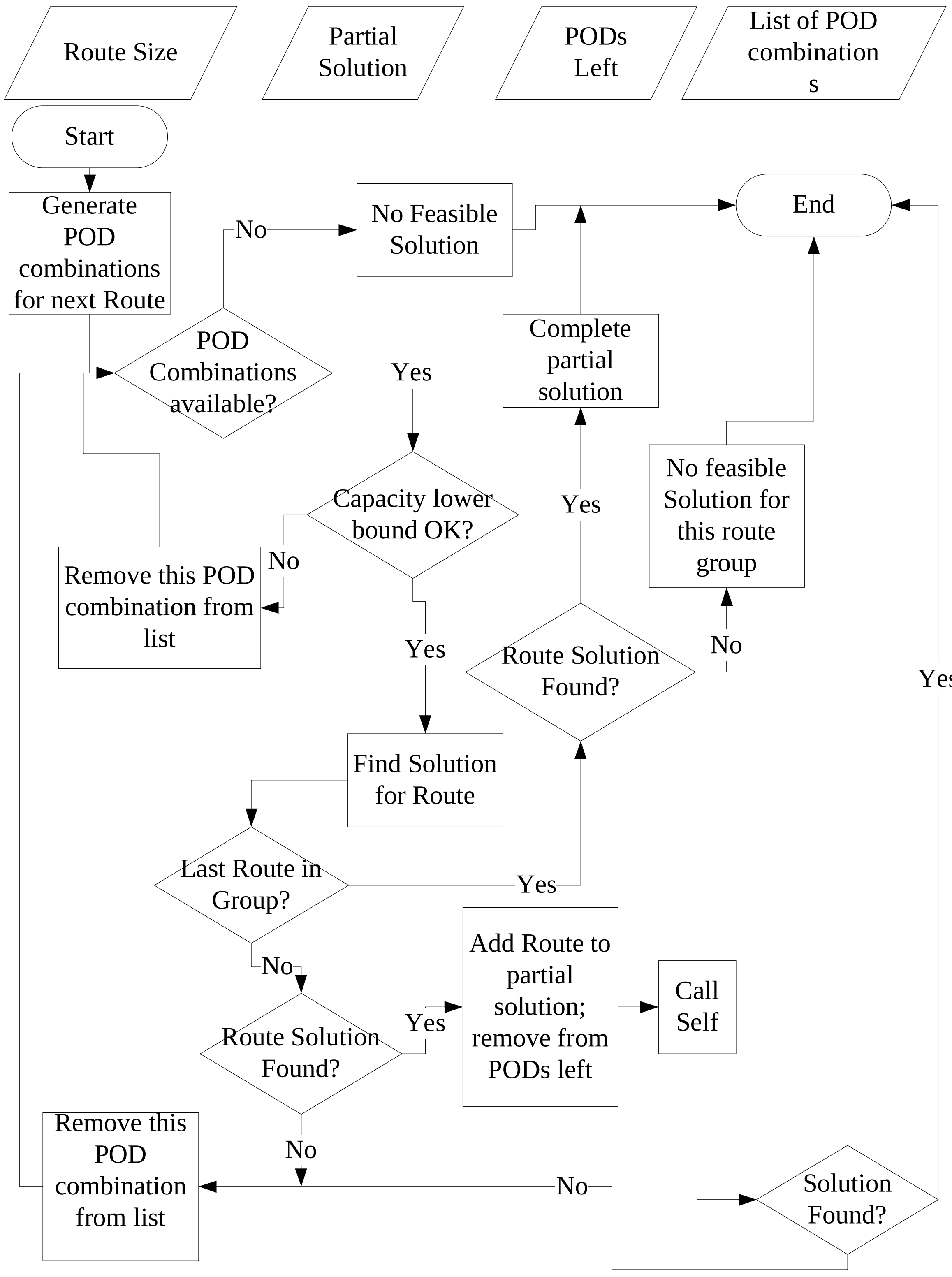

Size group Sequence
Partial Solution
PODs Left
List of POD combination s
Start
Generate POD combinations for next Route Group
No
No Feasible Solution
End
POD Combinations available?
Yes
Complete partial solution
No feasible Solution for size group sequence
# Vehicles lower bound OK?
No
Remove this POD combination from list
Yes
Yes
Yes
Route group Solution Found?
No
Find Solution for Route Group
Last Route Group?
Yes
No
Add Route Group to partial solution; remove from PODs left
Call Self
Route group Solution Found?
Yes
Remove this POD combination from list
No
Solution Found?
No

Figure 15 shows the algorithm for constituting routes using a given size sequence. The size sequence is first converted to a size-group sequence as described in Section 5.2.2. In general, this algorithm works by filling the first route group, reducing the number of route groups left to be filled and recursively calling itself. Recursion ends when there is one route group left. The method of combinations is used to assign PODs to a size group. Our implementation uses a combination generator that loops from one possible combination to the next until all combinations have been exhausted. If the *number of vehicles lower bound* (Section 5.4.1) for this combination exceeds the size of the first route group, the algorithm proceeds to the next combination. Otherwise, the algorithm in Figure 16 is called to generate routes for the first route-group.

If the first route group is also the last, the algorithm returns with the solution found and recursion ends. Otherwise, if there is a failure to fill either the first route group or the subsequent ones through recursion, the algorithm proceeds to the next combination of PODs for the first route group. If at any combination, the algorithm finds a solution the algorithm returns the solution and exits. Otherwise, it runs through all combinations and returns the size sequence as having no feasible solution.

Figure 16: Algorithm that uses a set of PODs and a given number of vehicles generates routes having an equal number of PODs.


Route Size
Partial Solution
PODs Left
List of POD combination s
Start
Generate POD combinations for next Route
No
No Feasible Solution
End
POD Combinations available?
Yes
Complete partial solution
No feasible Solution for this route group
Capacity lower bound OK?
No
Yes
Remove this POD combination from list
Yes
Route Solution Found?
No
Yes
Find Solution for Route
Last Route in Group?
Yes
No
Add Route to partial solution; remove from PODs left
Call Self
Route Solution Found?
Yes
No
Remove this POD combination from list
No
Solution Found?

The algorithm in Figure 16 assigns a set of PODs to the routes in a route group. It is recursive and works similarly to the algorithm in Figure 15. However, this algorithm has routes instead of route groups; it uses the modified method of combinations for reasons explained in Section 5.2.2; it applies the capacity lower bound to PODs intended for the first route; and recursion works by the algorithm constituting one route and calling itself to constitute the subsequent routes. If the set of PODs intended for the first route passes the lower bound test, the algorithm in Figure 17 is called to constitute the route.

In order to constitute a route from a given set of PODs, the algorithm in Figure 17 starts with an initial number of subroutes equal to the number of PODs. For this initial case, constituting the route is trivial. In general however, subroute size sequences are generated for a given number of subroutes and the algorithm in Figure 18 is called to constitute a route according to the size sequence. If a size sequence generates a feasible route, the algorithm returns the result. Otherwise, after looping through all size sequences, it reduces the number of subroutes and tries again. Notice that if the number of subroutes is 1, no further reduction is possible and the algorithm returns no feasible route for the set of PODs.

Figure 17: Algorithm that given a set of PODs generates one route

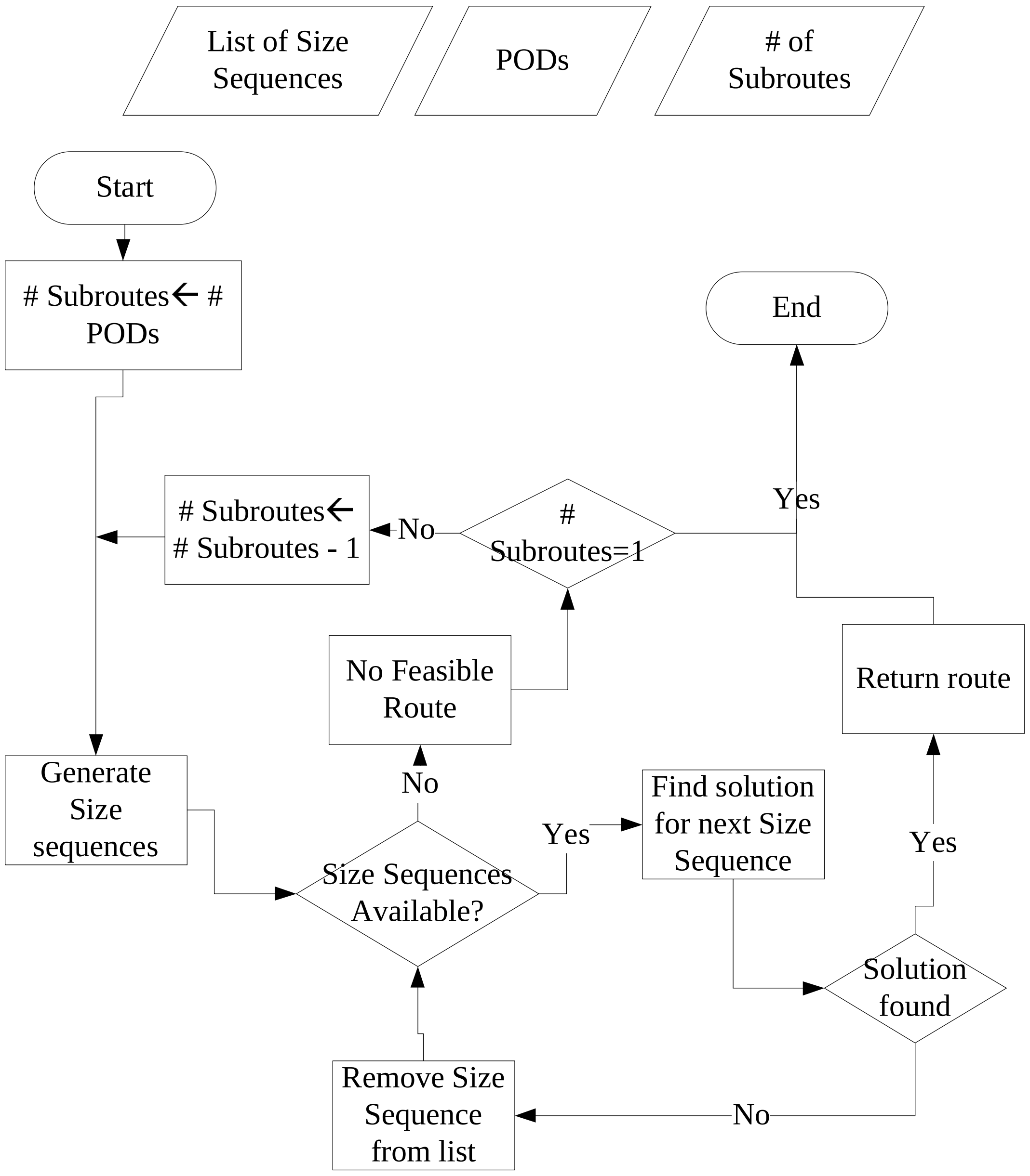
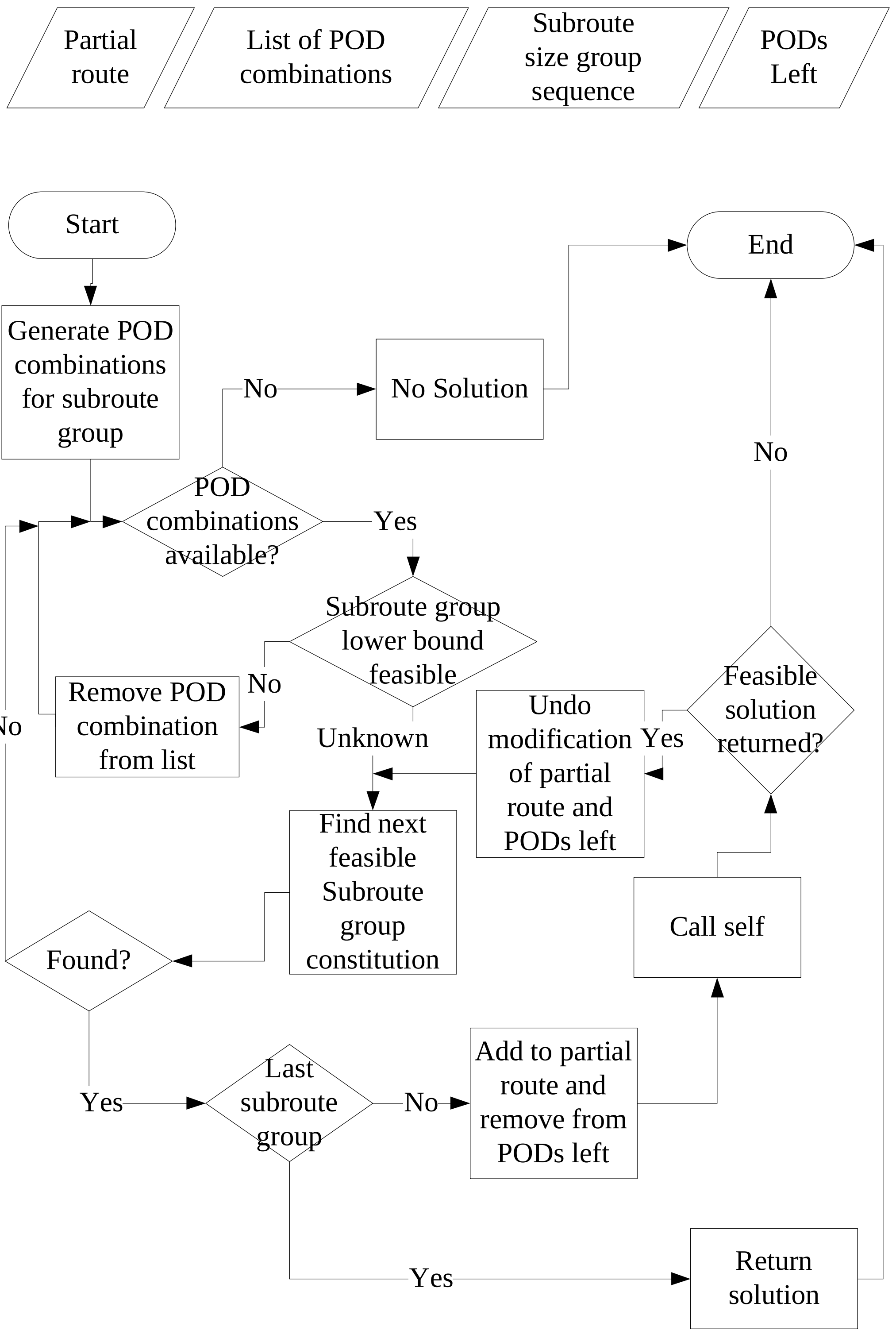


Figure 18: Algorithm that given a sequence of subroute sizes, completes a partially constituted route with a given set of PODs


Partial route
List of POD combinations
Subroute size group sequence
PODs Left
Start
End
Generate POD combinations for subroute group
No
No Solution
No
POD combinations available?
Yes
Subroute group lower bound feasible
No
Remove POD combination from list
No
Unknown
Undo modification of partial route and PODs left
Yes
Feasible solution returned?
Find next feasible Subroute group constitution
Call self
Found?
Yes
Last subroute group
No
Add to partial route and remove from PODs left
Yes
Return solution

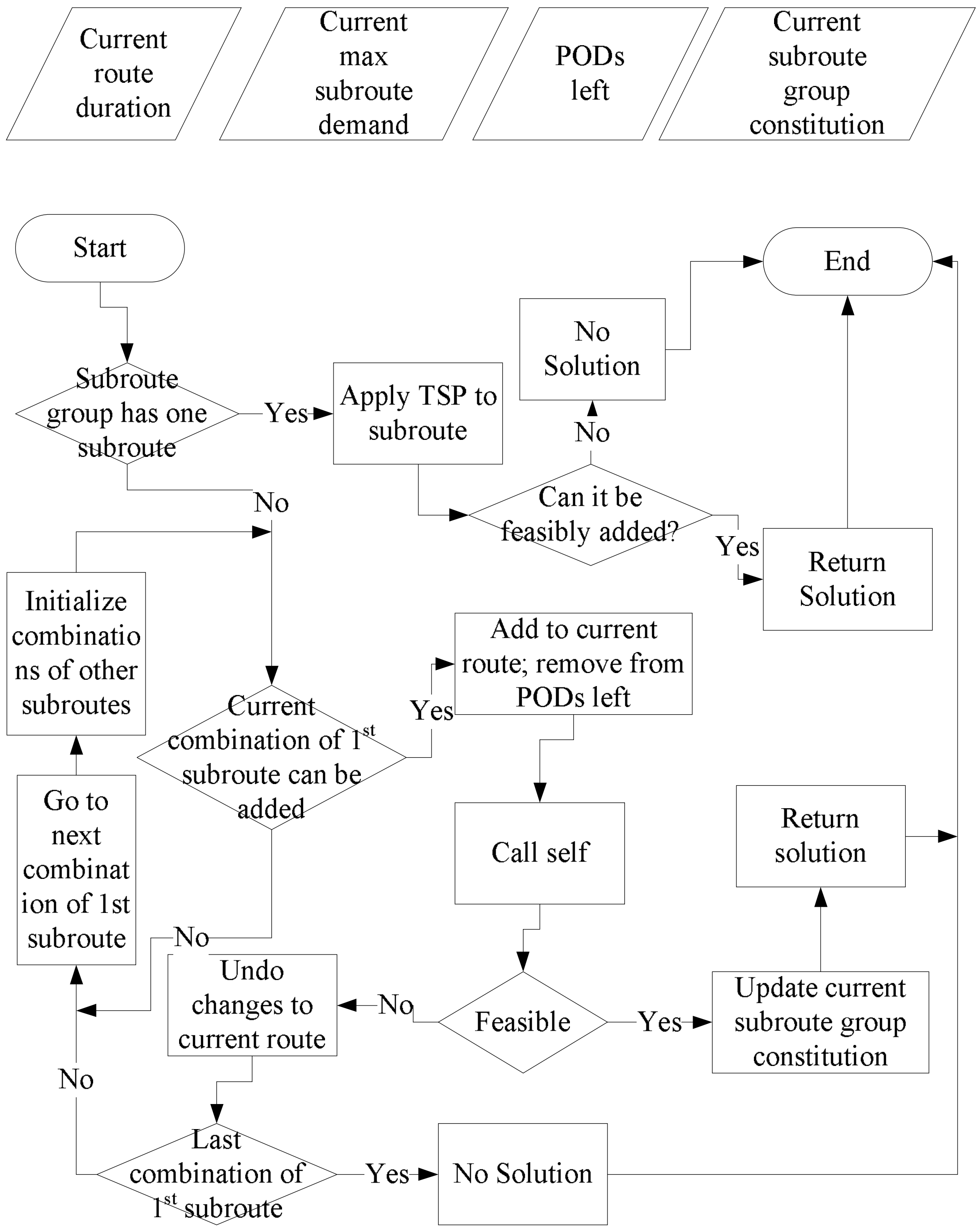
The algorithm in Figure 18 fills a route with a set of PODs according to a given subroute size sequence and is similar to Figure 15 which fills a route size sequence. As described in section 5.3.1 (Assigning PODs to Subroutes), the main difference is that unlike route groups, this algorithm cannot simply constitute a subroute group and conclude on its feasibility. Here, even after the first subroute group is constituted and feasibly added to a route, if there is a failure to successfully add the other subroute groups (recursive call), the algorithm finds the next feasible configuration of the first subroute group and tries again. It is only when all possible configurations of the first subroute group fail to be combined with the recursive call is failure final. The algorithm that finds the next feasible configuration of a subroute group is in Figure 19.

To find the next feasible configuration of a subroute group, the algorithm uses recursion and progresses from the current configuration. The configuration of a subroute is the current combination of each of its subroutes. To illustrate this point first consider how the combination generator we implemented works. Suppose we want combinations for three of nine PODs numbered 1 to 9. The combination generator would generate the following: {1,2,3}, {1,2,4}, …, {1,2,9}, {1,3,4}, {1,3,5}, …, {1,3,9}, {1,4,5}, …, {7,8,9}. Now suppose that the 9 PODs make up 3 subroutes of size 3 in a subroute group. An example of a subroute configuration is {[1,5,6], [1,3,5], [1,2,3]} where the first pair of square brackets represent the combination for 3 of 9 PODs for the first subroute and the second pair of square brackets represent the combination for 3 of remaining 6 PODs for the second subroute. Notice that the third angular bracket must be

[1, 2, 3] because there would be only three PODs left. For this example, the next subroute group configuration is {[1,5,6], [1,3,6], [1,2,3]}.

Returning to the algorithm, if the current subroute is the last in the subroute group that needs to be filled, then there is no “next” configuration and there is an attempt to simply add the subroute to the route. Otherwise, an attempt is made to add the first available subroute. If this fails, the algorithm goes to the next combination of this subroute for another trial. If the first subroute is successfully added, the algorithm recursively calls itself to find the next feasible combination of the subsequent subroutes. If the last combination is reached and is not feasible, then no configuration of the subroute group can feasibly be added to the route. Notice that for a given subroute group configuration, the algorithm calls a TSP routine to determine the best order in which to connect the PODs into a subroute.

Figure 19: Given a set of PODs that constitute a subroute group, this algorithm finds a combination of subroutes of the group that can feasibly be added to an existing route.



Current route duration
Current max subroute demand
PODs left
Current subroute group constitution
Start
End
Subroute group has one subroute
Yes
Apply TSP to subroute
No Solution
No
Can it be feasibly added?
Yes
Return Solution
No
Initialize combinations of other subroutes
Current combination of 1st subroute can be added
Yes
Add to current route; remove from PODs left
Call self
Return solution
Go to next combination of 1st subroute
No
Undo changes to current route
No
Feasible
Yes
Update current subroute group constitution
No
Last combination of 1st subroute
Yes
No Solution

5.7 *Testing the Exact Method*

After developing and implementing the branch and bound method earlier in this chapter, in this section we test the implementation and interpret the results. A major concern for exact solutions of NP hard problems is the time required to obtain a solution. In Sections 5.4 (Bounding and Pruning) and 5.5 (Other Computation Saving Techniques) we discussed techniques for reducing the time required to obtain an exact solution to the CRIRP. In this section, we first identify the techniques that work best for our problem. We then solve modified versions of the CRIRP based on the dispensing plans of the State of Maryland (Section 4.4.3), using different problem sizes to study the effects of problem size on time required to obtain a solution.

5.7.1 Evaluating Computation Saving Techniques

We used one of the variants of the Burma 14 problem adapted from TSPLIB as described in Section 4.4.1. This variant has a low capacity, demand rates with a standard deviation of about one-sixth of the average and a load time of one-fiftieth of the average travel time. There is no limit on the maximum subroute duration. We seek the effects of the computation saving techniques on computer runtimes. We look particularly at the effects of turning on or off the following:

- Pruning of routes
- Pruning of subroutes
- Storing feasible routes
- Multilevel search

For the default test, routes and subroutes are pruned, feasible routes computed are stored to memory and retrieved when needed and the multi-level search is not applied. Data is collected on the number of nodes in the route and subroute trees of the solution tree, the memory used and the program runtime. The computation-saving techniques listed above are each varied with respect to the default and the results in Table 6 are obtained. All tests are conducted on a Microsoft Windows XP Professional (version 2002, service pack 3) platform running on a DELL OPTIPLEX GX620 system having a 3.20 GHz Pentium processor and 2 gigabytes (GB) of random access memory (RAM). MATLAB version 7.5.0.342 of August 15, 2007 is used.

Table 6: Evaluating computation saving techniques on one problem instance

Comment	**Number of Nodes**				**Memory (bytes)**	**Runtime (minutes)**
	Route size sequences	**Route combinations for route sequences**	**Subroute size sequences**	**Subroute combinations for subroute sequences**		
Default	8	3032	118625	3033	52525	36.04
Route pruning OFF	8	3032	118828	3033	53006	34.15
Subroute pruning OFF	8	3032	118625	3033	52524	36.14
Feasible Route storage OFF	8	3032	118625	3033	26110	227.83
Multilevel search ON	24	9096	236392	9097	53180	70.67

As expected, these results confirm that when computed routes are not stored in memory, it requires much more time to obtain the solution and much less memory. With the multilevel search, the process also runs slower than the default case. The intent of the multilevel search is to improve existing solutions as quickly as possible by quickly finding solutions through a combination of exhaustive search and heuristics. However, the exact method only stops when it fails to fit the PODs into a certain number of vehicles. For this to happen the solution search goes through all three levels. This is what most likely accounts for the multilevel search having a high runtime. However, the multilevel search may still be useful in finding a solution more quickly if the goal is to improve the heuristic solution as opposed to finding the optimal solution.

The results show that pruning routes and subroutes have little effect on the runtime. Normally, one would expect that the pruning of branches in the route and subroute trees, would lead a solution in a shorter amount of time. A possible explanation is that pruning is also a computationally expensive operation. When the implementation is in route-pruning mode, for each route that is computed, heuristics are called several times for various combinations of PODs. The effects of pruning on runtime seem inconclusive. In order to further investigate the effects of pruning, we ran some more tests.

Table 7 shows the results of testing various variants of the Burma 14 problem. The first 3 columns specify the variant, the fifth column informs us if the exact approach was able to find a solution better than that obtained by the H3 heuristic and the last 4 columns show the effect of pruning routes and subroutes on the runtime. In most cases pruning has little effect; however, in the last two cases, not pruning routes results in much shorter runtimes. There is a possible reason for this significant difference. Consider fitting 10 PODs into a route using the size sequence {4, 3, 2, 1}. Furthermore, suppose that there is no combination of 4 and 3 PODs that can constitute the first two routes. Then the branches with the following size sequences would be eliminated: {5,3,2}, {4,3,3}, {6,3,1} among others. Now, suppose that route-pruning is in effect and we prune a branch because of a particular combination of the first route of 4 PODs. Because we did not go far enough to discover that there is no possible combination of the second route of 3 PODs, the branches mentioned above would not be eliminated. Therefore when using pruning, we may save computation time on one little branch within a size sequence node, but end up not eliminating several other size sequences that would otherwise have been eliminated.

Even though all the problem instances in Table 7 are of the same size, the time required to obtain solutions vary widely. This leads us to conclude that the runtime of a problem strongly depends on the particular problem and not just the size of the problem. In the first three cases, the runtime is zero because the heuristic solution size is 1 and no improvement is possible. For the rest where improvement is possible, the exact approach found a better solution in 2 out of 9 cases.

Table 7: Evaluating computation saving techniques on several instances

Vehicle Capacity	Demand Rate Standard Deviation	Load / unload Time	Size of Solution	Branch and Bound Improves Heuristic Solution	Time (Minutes)			
					Prune Neither	Prune Subroutes Only	Prune Routes Only	Prune Both
high	small	very small	1	FALSE	0	0	0	0
high	medium	very small	1	FALSE	0	0	0	0
high	high	very small	1	FALSE	0	0	0	0
high	small	medium	2	FALSE	37.78	37.84	37.76	37.88
high	medium	medium	2	FALSE	40.32	42.93	40.29	43.02
medium	high	very small	2	FALSE	571.64	569.73	566.93	564.6
small	high	very small	3	FALSE	55.76	60.61	59.05	60.04
small	small	very small	3	FALSE	13.97	13.95	14.92	15.88
medium	medium	medium	3	FALSE	50.74	51.62	54.26	53.63
small	high	medium	6	FALSE	5.36	5.36	74.16	74.8
small	medium	medium	6	TRUE	1.24	1.34	14.11	14.78
small	small	medium	6	TRUE	0.51	0.53	10.32	9.99

5.7.2 Application to the State of Maryland

In this section, we solve modified versions of State of Maryland scenario. We start by electing to use only PODs in county C, which has a total of 22 PODs, using 12 foot trucks and setting the maximum route duration to 8 hours. From the observations made in the previous section, we used neither route nor subroute pruning while solving for this scenario. To prevent the problem from reducing to the CRIRP special case, the PODs are assigned 3 different demand rates: ten are assigned a demand of 37.42, five are assigned a demand of 29.44 and seven a demand of 21.91 all in regimens per minute. The problem sizes considered are 7, 12, 17 and 22. For the first three sizes, we randomly selected the PODs from the 22 PODs in county C. The results are shown in Table 8.

Table 8: State of Maryland scenario: exact solution

Problem size	Runtime (min.)	Heuristic solution	Exact solution
7	0.23	2	2
12	69.83	3	2
17	6540.81	4	3
22	2400.00 (still running)	5	4 (tentative)

The exact solution consistently improves on the heuristic solution but as the problem size increases, we pay a high computation price for these improvements. It is clear from these results that the runtime increases much faster than the problem size.

5.8 Summary

In this chapter, we have developed a branch and bound technique for finding an optimal solution to the CRIRP. This involves a branching method that allows us to explore all possible solutions and a bounding method using various lower bounds to systematically eliminate branches from the solution tree. We also used additional computation-saving techniques such as storing intermediate results in memory, pruning and multilevel search. We found that the pruning methods used failed to reduce computation time, but tended to increase them instead. The multilevel search also performed poorly, but it might have the potential to quickly find solutions. Storing computed routes proved quite successful in reducing runtime. The best combination of techniques includes using storage of computed routes and excluding the use of the multilevel search and pruning.

Finally, the solution runtime clearly increases much faster than the problem size.

Chapter 6: SUMMARY

In this thesis, we have studied the Continuous Replenishment Inventory Routing Problem, in which a homogenous fleet of vehicles continuously delivers a single product to a set of sites that consume the material at a constant rate. The sites do not have predetermined inventory limits, and it is required that sites do not run out of material before the next visit by a vehicle. The number of vehicles available is limited, so we seek a solution that minimizes the number required.

6.1 Insights

Significant observations made during this research on the CRIRP include:

- Both the H1 and H3 heuristics perform significantly better than the benchmark H2 heuristic suggesting that they may be good heuristics. The H3 heuristic which exploits problem properties and uses CVRP techniques almost always performs better than the other heuristics.
- The performance of the H1 heuristic approaches that of H3 as the vehicle capacity increases or as the problem size decreases.
- For the exact approach, the computation resources required increase very rapidly with the problem size. A problem instance with 17 PODs required 4 days of computation time on a 3.20 GHz Pentium processor with 2 gigabytes of random access memory (RAM). However, size is only one of the factors that determine computation time. For problems of the same size, the computation resources required may vary by a couple of orders of magnitude. The required resources thus also depend on the particular problem.

- The exact method is practical only in problems with few PODs.
- The enormous amount of computational resources required for the exact approach make it challenging to extensively test the exact approach. Such tests are important for providing insight into the factors that make the method successful (by reducing computation time) in solving CRIRP problems.

6.2 Contributions

These are the major contributions of this work:

- The development of a new type of Inventory Routing Problem to address the need for continuous delivery of medical supplies which is motivated by emergency preparedness planning.
- The identification of a special case of the CRIRP that is equivalent to the bin-packing problem.
- The development of heuristics that provide near-optimal solutions (at least for the problem sizes for which we were able to obtain optimal solutions). These heuristics use only a tiny fraction of the computational resources required by the exact solution.
- The development and implementation of tests that demonstrate the relative strengths of these solution approaches.
- The development of an exact solution approach for the CRIRP. Various bounding and pruning techniques were tested on small problems to determine their effectiveness is speeding up the exact approach.

6.3 Future work

Future work should focus on the following:

- Improving the solution lower bounds and pruning techniques used in the exact solution approach. The use of better lower bounds would eliminate more solution branches from consideration, thus speeding up the arrival at an optimum solution.
- Using high performance computer systems to test more features of the exact approach. This would also permit us to compare how close to the optimum our heuristic solutions are in the case of problems of large size.
- A comparison of the time it takes to arrive at the optimal solution as opposed to the time required for eliminating the first infeasible solution. To rule out an infeasible number of vehicles, all possible route combinations must be ruled out. Therefore, it is quite possible that after arriving at the optimum, a considerable fraction of computational resources is used in eliminating the first infeasible number of vehicles.
- Developing approaches for problems with a heterogeneous fleet of vehicles (that is, the vehicles come in different sizes), which will require modifications to the current heuristics.
- Developing approaches that account for variability in the travel times and demand at each site. Adding slack to the solutions will be necessary. The maximum load of a subroute must be kept less than the vehicle capacity. The difficult problem is to determine the minimal amount of slack that still provides a high service level.
- Developing approaches for multi-depot problems.

- An attempt to use a modified CRIRP to solve the classic IRP is also worth some investigation. The classic IRP plans for one route per day for each vehicle. While customers may have delivery time windows, there is no reason for which supplying customers in the IRP must be done discretely. Applying the CRIRP to the IRP, once a vehicle returns to the depot, subject to customers' delivery time windows, it immediately reloads and returns to its route. . In order to be adapted to the IRP, the CRIRP would need some modification to take time windows and customer inventory into account.

APPENDIX A: PROBLEM DATA FOR SAMPLE PROBLEM INSTANCES

Smallest capacity= 8822.3971

Medium capacity= 22980.7794

Large capacity= 37139.1618

Load time= 0.8816

Maximum route duration= 100

Table 9: Travel Times

	Depot	POD #1	POD #2	POD #3	POD #4	POD #5	POD #6	POD #7	POD #8	POD #9	POD #10	POD #11	POD #12	POD #13
Depot	0	1.66	5.08	6.52	8.83	5.53	4.1	0.75	1.29	3.15	1.28	5.08	3.12	3.94
POD # 1	1.66	0	4.09	6.02	9.2	5.76	4.76	1.99	2.94	4.4	2.94	5.18	3.98	3.62
POD # 2	5.08	4.09	0	2.45	6.96	4	4.5	4.73	6.15	8.22	6.01	3.37	4.64	2.01
POD # 3	6.52	6.02	2.45	0	4.8	2.71	4.12	5.96	7.29	9.6	7.1	2.38	4.8	2.58
POD # 4	8.83	9.2	6.96	4.8	0	3.44	4.77	8.09	8.93	11.21	8.7	4.06	5.82	5.8
POD # 5	5.53	5.76	4	2.71	3.44	0	1.81	4.81	5.85	8.22	5.63	0.66	2.81	2.43
POD # 6	4.1	4.76	4.5	4.12	4.77	1.81	0	3.35	4.19	6.51	3.96	1.77	1.07	2.5
POD # 7	0.75	1.99	4.73	5.96	8.09	4.81	3.35	0	1.41	3.64	1.28	4.38	2.36	3.37
POD # 8	1.29	2.94	6.15	7.29	8.93	5.85	4.19	1.41	0	2.37	0.23	5.52	3.12	4.73
POD # 9	3.15	4.4	8.22	9.6	11.21	8.22	6.51	3.64	2.37	0	2.59	7.89	5.45	7.02
POD # 10	1.28	2.94	6.01	7.1	8.7	5.63	3.96	1.28	0.23	2.59	0	5.3	2.89	4.55
POD # 11	5.08	5.18	3.37	2.38	4.06	0.66	1.77	4.38	5.52	7.89	5.3	0	2.61	1.77
POD # 12	3.12	3.98	4.64	4.8	5.82	2.81	1.07	2.36	3.12	5.45	2.89	2.61	0	2.67
POD # 13	3.94	3.62	2.01	2.58	5.8	2.43	2.5	3.37	4.73	7.02	4.55	1.77	2.67	0

Table 10: Demands with medium standard deviation

POD #	Depot	1	2	3	4	5	6
Demand	0	162.73	260.44	205.03	190.57	255	222.21

POD #	7	8	9	10	11	12	13
Demand	188.45	191.2	178.52	169.82	223.89	199.76	106.17

APPENDIX B: HEURISTIC SOLUTIONS FOR SAMPLE PROBLEM INSTANCES

B.1. Smallest Capacity

Table 11: H1 heuristic (4 routes)

Route						**Duration**
Subroutes	4	5				32.25
Subroute Durations	19.43	12.82				
Subroute Load	6147	8226				
Subroutes	3	2	1			31.8
Subroute Durations	14.8	11.92	5.08			
Subroute Load	6521	8283	5176			
Subroutes	11	6	13			31.53
Subroute Durations	11.91	9.97	9.64			
Subroute Load	7059	7006	3348			
Subroutes	9	12	8	10	7	28
Subroute Durations	8.07	7.99	4.35	4.33	3.27	
Subroute Load	5000	5595	5355	4756	5278	

Table 12: H2 heuristic (6 routes)

Route		**Duration**
Subroutes	[2 9]	19.1
Subroute Durations	19.1	
Subroute Load	8384	
Subroutes	[10 6]	11.99
Subroute Durations	11.99	
Subroute Load	4700	
Subroutes	[4 8]	21.7
Subroute Durations	21.7	
Subroute Load	8285	
Subroutes	[11 12]	13.45
Subroute Durations	13.45	
Subroute Load	5698	
Subroutes	[13 7 1]	14.49
Subroute Durations	14.49	
Subroute Load	6626	
Subroutes	[5 3]	17.4
Subroute Durations	17.4	
Subroute Load	8006	

Table 13: H3 heuristic (4 routes)

Route						Duration
Subroutes	13	1	10	9	7	
Subroute Durations	9.64	5.08	4.33	8.07	3.27	30.39
Subroute Load	3227	4946	5161	5425	5727	
Subroutes	4	8	12			
Subroute Durations	19.43	4.35	7.99			31.77
Subroute Load	6055	6075	6347			
Subroutes	3	6	11			
Subroute Durations	14.8	9.97	11.91			36.69
Subroute Load	7522	8153	8215			
Subroutes	5	2				
Subroute Durations	12.82	11.92				24.74
Subroute Load	6310	6444				

B.2. <u>Medium Capacity</u>

Table 14: H1 heuristic (2 routes)

Route								Duration
Subroutes	4	3	5	2	11	6	1	
Subroute Durations	19.43	14.8	12.82	11.92	11.91	9.97	5.08	85.94
Subroute Load	16378	17621	21916	22384	19242	19098	13986	
Subroutes	13	9	12	8	10	7		
Subroute Durations	9.64	8.07	7.99	4.35	4.33	3.27		37.64
Subroute Load	3997	6721	7520	7198	6393	7095		

Table 15: H2 heuristic (4 routes)

Route		**Duration**
Subroutes	[11 10 3]	27.52
Subroute Durations	27.52	
Subroute Load	16479	
Subroutes	[12 4 5]	21.44
Subroute Durations	21.44	
Subroute Load	13833	
Subroutes	[9 1 13 2]	22.67
Subroute Durations	22.67	
Subroute Load	16051	
Subroutes	[6 8 7]	13.98
Subroute Durations	13.98	
Subroute Load	8417	

Table 16: H3 heuristic (2 routes)

Route												**Dura tion**
Subrtes	13	1	10	9	7	4	8	12	3	6	11	98.85
Subrte Dura tions	9.64	5.08	4.33	8.07	3.27	19.43	4.35	7.99	14.8	9.97	11.91	
Subrte Load	10495	16086	16786	17647	18628	18837	18900	19746	20266	21965	22131	
Subroutes	5	2										24.74
Subroute Durations	12.82	11.92										
Subroute Load	6310	6444										

B.3. <u>Large Capacity</u>

Table 17: H1 heuristic (2 routes)

Route			**Duration**
Subroutes	[7 13 4 8]	[2 5 12]	43.08
Subroute Durations	24.56	18.52	
Subroute Load	29140	30812	
Subroutes	[6 11 3]	[10 9 1]	31.77
Subroute Durations	18.31	13.46	
Subroute Load	20686	16237	

Table 18: H2 heuristic (3 routes)

Route		**Duration**
Subroutes	[4 3 13 12 1]	29.82
Subroute Durations	29.82	
Subroute Load	25774	
Subroutes	[11 10 8 7 6]	24.76
Subroute Durations	24.76	
Subroute Load	24656	
Subroutes	[2 5 9]	23.97
Subroute Durations	23.97	
Subroute Load	16633	

Table 19: H3 heuristic (2 routes)

Route								**Duration**
Subroutes	[4 3]	[13 11]	[10 8]	[7 12]	[1 6]	9	5	84.61
Subroute Durations	22.8	13.43	5.45	8.88	13.17	8.07	12.82	
Subroute Load	33472	27928	30546	32847	32571	15105	21577	
Subroutes	2							11.92
Subroute Durations	11.92							
Subroute Load	3104							

BIBLIOGRAPHY

Bard, Jonathan F., Liu Huang, Patrick Jaillet, and Moshe Dror, “A decomposition approach to the inventory routing problem with satellite facilities,” *Transportation Science,* Volume 32, Number 2, pages 189 – 203, 1998.

Campbell, A., Clarke, L., Kleywegt, A., and Savelsbergh, M., “The Inventory Routing Problem,” *Fleet Management and Logistics* (eds. Crainic, G. L. and Laporte, G.), Kluwer Academic Publishers, Norwell, Massachusetts, pages 95-113, 1998.

Campbell, Ann Melissa, and Martin W. P. Savelsbergh, “A Decomposition Approach for the Inventory-Routing Problem,” *Transportation Science*, Volume 38, Number 4, pages 488–502, November 2004a.

Campbell, Ann Melissa, and Martin W. P. Savelsbergh, “Delivery Volume Optimization,” *Transportation Science*, Volume 38, Number 2, pages 210-223, May 2004b.

Christofides, N., and Beasley, J.E., “The Period Routing Problem,” *Networks*, Volume 14, pages 237-256, 1984.

Clarke, G., and Wright, J.W. “Scheduling of vehicles from a central depot to a number of delivery points,” *Operations Research*, Volume 12, Number 4, pages 568-581, 1964.

Garey, M.R. and Johnson, D.S., “Coping with NP-Complete Problems”, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Bell Telephone Laboratories, pages 121-151, 1979.

Golden, B., Bodin, L., Doyle, T., and Stewart Jr., W., "Approximate Traveling Salesman Algorithms," *Operations Research*, Volume 28, Number 3, Part 2, pages 694-711, May-Jun 1980.

Golden, Bruce, Arjang Assad, and Roy Dahl, "Analysis of a large scale vehicle routing problem with an inventory component," *Large Scale Systems*, Volume 7, pages 181-190, 1984.

Jaillet, Patrick, Jonathan F. Bard, Liu Huang, and Moshe Dror, "Delivery cost approximations for inventory routing problems in a rolling horizon framework," *Transportation Science*, Volume 36, Number 3, pages 292-300, 2002.

Kruskal, Jr., J.B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", *Proceedings of the American Mathematical Society*, Volume 7, Number 1, pages 48-50, Feb 1956.

Lawler, E.L., and Wood, D.E., "Branch-And-Bound Methods: A Survey," *Operations Research*, Volume 14, Number 4, pages 669-719, Jul – August 1966.

Lenstra, J.K., "Quadratic Assignment Problems," *Sequencing By Enumerative Methods*, Mathematical Centre Tracts, Amsterdam, pages 12-17, 1977.

Lenstra, J.K., and A. Rinnooy Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, Volume 11, pages 221-227, 1981.

Toth, P. and Vigo, D. (1998), "Exact Solution of the Vehicle Routing Problem," *Fleet Management and Logistics* (eds. Crainic, G. L. and Laporte, G.), Kluwer Academic Publishers, Norwell, Massachusetts, 1-31.

TSPLIB (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html July 08, 2008).

TSPLIB: 52 locations in Berlin, Germany (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/berlin52.tsp, July 08, 2008)

TSPLIB: 127 beer gardens in Augsburg Germany (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/bier127.tsp, July 08, 2008):

TSPLIB: 14 cities in Burma (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/burma14.tsp, July 08, 2008):

TSPLIB: 22 locations from the Odyssey of Ulysses (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/ulysses22.tsp, July 08, 2008):

Webb, Ian R., and Richard C. Larson, "Period and phase of customer replenishment: a new approach to the strategic inventory/routing problem," *European Journal of Operational Research*, Volume 85, pages 132-148, 1995.