The Bees Algorithms for the Vehicle Routing Problem

Michael J. Dinneen Aisha J.L. Fenton

Department of Computer Science
University of Auckland
Auckland, New Zealand

Abstract

We examine the Pickup and Delivery problem for Full truck load vehicles - an important problem in moving bulk freight. We provide background on the related problems VRP and PDP as well as providing a brief survey of the more common heuristics for these problems. We provide an algorithm based on adapting classic heuristics and provide a set of test instances to benchmark the problem against. We discuss implementation challenges and solutions and present a real-world case study.

Contents

1 Introduction **4**

1.1 Motivation 4

1.1.1 Road transport industry 4

1.1.2 Vehicle routing 5

2 Background **6**

2.1 TSP introduction and history 6

2.2 VRP introduction and history 8

2.2.1 Exact Methods 8

2.2.2 Classic Heuristics 9

2.2.3 Meta-Heuristics 11

2.3 Classic VRP heuristics in depth 13

2.3.1 Nearest neighbour 13

2.3.2 Nearest insertion 13

2.3.3 Clark and Wright algorithm 14

2.3.4 2-opt 15

2.3.5 Or-opt 15

2.3.6 Osman-opt 15

2.3.7 Tabu search 15
2.3.8 Large neighbourhood search 15
2.4 An introduction to Swarm Intelligence 15

3 Problem Definition **16**

3.1 Industry 16
3.1.1 Operation 17
3.1.2 Objectives 17
3.1.3 Constraints 17
3.2 Formal definitions 17
3.2.1 Classic VRP 17
3.2.2 PDP 18
3.2.3 PDP-FTL 20
3.2.4 VRPTW 21
3.2.5 PDPTW 21

4 Algorithm **22**

4.1 Goals 22
4.2 Algorithm 23
4.2.1 Classic Bees algorithm 23
4.2.2 Representation 23
4.2.3 Changes from classic Bees Algorithm 23
4.2.4 Operations 23

5 Results **24**

5.1 Test instances 24

5.2 Results 24

5.3 Comments 24

6 Conclusion **25**

6.1 Future Directions 25

7 Appendix A - Implementation **26**

7.1 Parallelization 26

7.2 Source Code 26

*

Chapter 1

Introduction

1.1 Motivation

1.1.1 Road transport industry

Road transport is a vital part of New Zealand. It is important for providing basic needs: virtually every product grown, made or used in New Zealand will be carried on a truck at least once during its lifetime[17]. And it is important for the New Zealand economy. The success of New Zealand's export industries are inextricably linked to the reliability and cost effectiveness of road transport.

New Zealand is inline with other countries in having 80% of all freight transported by road (by comparison Europe also carries 80% of all freight road). Road transport is particularly important to regional New Zealand and the export industries which drive these local economies. Trucks carry[17]:

- 95% of export fruit
- 86% of export wool
- 85% of export dairy products
- 65% of export logs
- 35% of export meat

Road transport is a also very closely tied to the New Zealand economy. A study by [**?**] showed that a 1% growth in national output requires a 1.5% increase in transport services.

In addition to this Road transport is itself a major contributor to the New Zealand economy. The road transport industry has a total turnover of around $4 billion a year. This equates to

around 1.4% of economic activity nationally and goes as high as 2.5% of economic activity in regional areas such as Southland[17].

Most road transport business runs on tight margins [**?**]. Therefore technology innovations are seen as key area for investment and as having a great potential for enabling the industry to grow. Technology research is focused on providing solutions in the following areas:

- Improving fuel efficiency (e.g. improved engines, improved aerodynamics)
- Reducing environmental impact of Road transport
- Improving utilization of the fleet
- Reducing running costs of a fleet
- Strain on transport network. Limiting factor for economic growth.

Improvements to vehicle routing has benefits in each of these areas. An optimal schedule will reduce the amount of distance travelled to perform the same amount of work, therefore it: reduces fuel costs, reduces environment impact as less work is required to move the same amount of freight, allows more good to be carried for the same cost, allows for more efficient use of the existing road network.

SOMETHING ABOUT LOGGING TRUCKS

1.1.2 Vehicle routing

TODO

Chapter 2

Background

This section reviews the background of Vehicle Routing Problem (VRP) and Pickup Delivery Problem (PDP). Classic and more modern heuristics for solving the VRP and related problems are reviewed as well providing a review of real-world implementations and results.

2.1 TSP introduction and history

The traveling salesman problem can informally be defined as given n points on a map provide a route through each of the n points such that each point is only used once and the total distance travelled is minimized. The problem's name, traveling salesman, comes from the classic real-world example of the problem: a salesman is sent on a trip to visit n cities. What is order should she visit the cities in, such that she covers the least distance? See figure 2.1

TSP is related to a classic graph theory problem, the Hamiltonian path. Hamiltonian circuits have been studied since 1856 by both Hamilton [10] and Kirkman [13]. The traveling salesman problem has been informally discussed probably for many years [22] but didn't become actively studied until after 1928 where Menger, Whitney, Flood and Robinson produced much of the early results in the field. Robinson's RAND report [21] might have been the first article to call the problem by the name it has since become known, the Traveling Salesman Problem.

> The purpose of this note is to give a method for solving a problem related to the traveling salesman problem. One formulation is to find the shortest route for a salesman starting from Washington, visiting all the state capitals and then returning to Washington. More generally, to find the shortest closed curve containing n given points in the plane.

An early result was provided by Dantzig, Fulkerson, and Johnson [5]. Their paper gave an

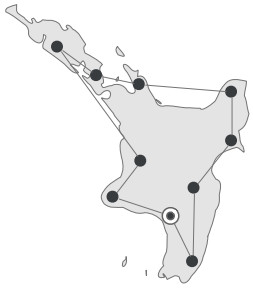

Figure 2.1: Traveling Salesman Problem

exact method for solving a 49 city problem, a large number of cities for the time. Their algorithm used the cutting plane method to provide an exact solution. This approach has been the inspiration for many subsequent approaches, and is still the bedrock of algorithms that attempt to provide an exact solution.

Karp's famous paper, Reducibility Among Combinatorial Problems, in 1972 showed that the Hamiltonian Circuit problem is NP-complete. This implied the NP-hardness of TSP, and thus supplied the mathematical explanation for the apparent difficulty of finding optimal tours.

A generalization of TSPis MTSP, where multiple routes are allowed to be constructed (i.e. multiple salesman can be used to visit the cities). The pure MTSP can trivially be turned into a TSPby constructing a graph G with $n-1$ additional copies of the starting node to the graph and forbidding travel directly between each n starting nodes. Note however that the pure formulation of MTSP places no additional constraints on a route. In real life applications MTSP is typically employed with additional constraints, such as each route should be of equal size (e.g. work should be balanced among the salesman), or no route should exceed a total distance or time (e.g. a anyone salesman shouldn't be asked to work more than 8 hours a day).

MTSPleads us naturally into the a class of problems derived from the Vehicle Routing Problem (VRP). VRP– and it's family of related problems – can be understood as being a combination of MTSP along with other constraints, which are in themselves often combinatorially hard problems.

2.2 VRP introduction and history

The Vehicle Routing Problem (VRP) seeks to solve the problem of constructing routes for a fleet of vehicles. Each route takes the vehicle through a set of customers to deliver a good (or service) at different locations. The routes must be constructed such that all constraints are meet, classically, the vehicle's capacity isn't exceeded by the number of customers being serviced. The goal is to minimize the cost of distributing the goods, which typically has meant minimizing the distance travelled across all routes.

VRP was first formally defined by Dantzig and Ramser, in [6] and has remained an important problem in logistics and transport (the original name given to the problem presented by Dantzig and Ramser was The Truck Scheduling Problem). The Vehicle Routing Problem is closely related to two problems, the Multiple Traveling Salesman Problem MTSP and the Bin Packing Problem BPP.

2.2.1 Exact Methods

The first efforts were concerned with exact solutions, and proceeded using many of the same techniques brought to bear on TSP. We follow Laporte and Nobert's survey [14] and classify exact algorithms for the VRP into three families: Direct tree search methods, Dynamic programming, and Integer linear programming.

The first classic Direct tree search results are due to Christolds and Ellison. Their 1969 paper in Operations Research Quarterly provided the first branch and bound algorithm for exactly solving the VRP[4]. Unfortunately it's time and memory requirements where such that it was only able to solve problems of up to 13 customers. This result was later improved upon by Christolds in 1976 by using a different branch model. This improvement allowed him to solve for up to 31 customers.

Christofides, Mingozzi, and Toth, [3] provide a lower bound method that is sufficiently quick (in terms of runtime performance) to be used to as a lower bound for excluding nodes from the search tree. They used this method to provide solutions as for a number of problems containing between 15 to 25 customers. Laporte, Mercure and Nobert [**?**] used MTSP as a relaxation of VRP within a branch and bound framework to provide solutions for 'realisticly' sized problems containing up to 250 customers.

The Dynamic Programming approach was first proposed for VRP by Eilon, Watson-Gandy and Christofides (1971). Their approach allowed them to solve exactly for problems of 10 to 25 customers. Since then, Christofides has made improvements to this algorithm to solve exactly for problems up to 50 vertices large.

A Set Partitioning algorithm was given by Balinski, and Quandt in 1964 to produce exact VRP solutions [1]. The problem sets they used where very small however, having only between 5 to 15 customers. And even despite this they weren't able to produce an solution for some of the problems. However, taking their approach as a starting point many authors

have been able to produce more powerful methods. Rao and Zionts (1968), Foster and Ryan (1976), Orloff (1976), Desrosiers, Soumis and Desrochers (1984), Agarwal, Mathur and Salkin (1989), and Desrochers, Desrosiers and Solomon (1990), have all extended the basic set partitioning algorithm, using the Column Generation method, to produce more practically useful results.

Notwithstanding the above results, exact methods have been of more use in advancing theoretical understanding of VRP than to providing solutions to real life problems. This can mostly be attributed to the fact that real-life VRP instances often involve hundreds of customers, and involve richer constraints than are modeled in a simple VRP.

2.2.2 Classic Heuristics

Classic VRP heuristics can be classified into three families. Constructive heuristics; Two-phase heuristics, which again can be classified into two sub-families: cluster first and then route, and route first and then cluster; and Improvement methods.

Constructive heuristics

We start by looking at Constructive heuristics. Constructive heuristics build a solutions from the ground up. They typically provide a recipe for building each route, such that the total cost of all routes is minimized. An early and influential result was given by Clarke and Wright in their 1964 paper, Scheduling of vehicles from a central depot to a number of delivery points [8]. In this paper they present a heuristic method for solving VRP problems that improved upon Dantzig and Ramser's work – it is commonly known as the Clarke-Wright heuristic. The heuristic is based on the simple process of combining routes (starting with each route containing a single customer) such that the combination reduces the overall cost (typically distance) while still producing feasible solutions.

The heuristic has been used to solve problems of up to 1000 customers with results often within 10% of optimal using only a 180 seconds of runtime [24]. This classic algorithm has been extended by Gaskell (1967), Yellow (1970) and Paessens (1988), who have suggested alternatives to the savings formulas used by Clarke and Wright. These approaches typically introduce additional parameters to guide the algorithm towards selecting routes with geometric properties that are likely to produce better combinations.

Alkinkemer and Gavish provide an interesting variation on the basic savings heuristic [12]. They use a matching algorithm to combine multiple routes in a step. To do this they construct a graph where each vertex represents a route, each edge represents a feasible saving, and the edge's weight represents the saving that can be realized by the merge of the two routes. The algorithm proceed by solving a maximum cost weighted matching of the graph.

A group of construction heuristics, know as Sequential Insertion Heuristics, had their first

results due to Mole and Jameson [20].

Two-phase heuristics

We next look at two-phase heuristics. We start by looking at the cluster-first, route second sub-family. One of the foundational algorithms for this method is due to Gillett and Miller who provided a new approach called the Sweep Algorithm in their 1974 paper[2]. This popularized the two-phase approach, although this method was suggested earlier by Wren in his 1971 book, and subsequently in Wren and Holliday's 1972 paper for Operations Research Quarterly. In this approach, a initial clustering phase is used to cluster the customers into a base set of routes. From here the routes are treated as separate TSP and optimized accordingly. The two-phase approach typically doesn't prescribe a method for how the TSP is solved and assumes that already developed TSP methods can be used. The classic sweep algorithm uses a simple geometric method to cluster the customers, and has been shown to solve several benchmark VRP problems to within 2% to 9% of the best known solutions [24].

Fisher and Jaikumas's 1981 paper builds upon the two-phase approach by providing a more sophisticated clustering method. They solve a General Assignment Problem to form the clusters instead. A limitation of their method is that the amount of vehicle routes must be fixed up front. Their method often produces results that are 1% to 2% better than similar results produced by the classic sweep algorithm [24].

Christofides, Mingozzi, and Toth expanded upon this approach in [**?**] and proposed a method that uses a truncated branch and bound technique (similar to Christofiedes Exact method). At each step it builds a collection of feasible routes containing a customer i for evaluation. It then evaluates each route and selects the route that the TSPwith the shortest distance can be produced from.

The Petal algorithm is a natural extension to the Sweep Algorithm. It was first proposed by Balinski and Quandt [**?**] and then extended by Foster and Ryan [**?**]. The basic process is to produce a collection of overlapping candidate routes (called petals) and then to solve a set partition problem to produce a feasible solution. As with other two-phase approaches it's assumed that the order of the customers within each routes is solved using any pick of existing TSPheuristics. It has produced some competitive results for small solutions, but quickly becomes impractical where the set of candidate routes that must be considered is large.

Lastly, there are route first, cluster second methods. The basic premise of these techniques are to first construct a "grand" TSP tour such that all customers are visited. The second phase is then concerned with splitting this tour into feasible routes. Route first, cluster second methods are generally thought to be less competitive than other methods [9], although interestingly Haimovich and Rinnooy Kan have shown that if all customers have unit demand then a simple shortest path algorithm (which can be solved in polynomial time) can be used to produce a solution from a TSP tour that is asymptotically optimal [16].

Iterative Improvement Heuristics

Iterative Improvement methods follow a basic hilling climbing approach. They start with a solution, which may have been randomly generated, or produced by another heuristic technique, and then iteratively apply an operation to that solution to improve it. This continues until a local minimum is reached.

There have been a number of operations suggested. Christofides and Eilon gave one of the earliest iterative improvement methods in their paper [4]. In this they made a simple change to the classic TSP operation, 2-opt, increasing the amount of edges removed to three - the operation fittingly being called 3-opt. They found that their heuristics produced better results than the a 2-opt procedure could by itself.

In general operations, such as 3-opt, that remove edges and then search for a more optimal recombination of components take $O(n^y)$ where y is the number of edges removed. A rich strain of research has been on producing operations that reduce the amount of recombinations that must be searched. Or presents an operation that has since come to be known as Or-opt [18]. Or-opt is a restricted 3-opt. It searches for a relocation of all sets of 3 consecutive vertices (i.e. chains), such that an improvement is made. If an improvement can't be made then it tries again with chains of 2 consecutive vertices, and so on. Or-Opt has been shown to produce similar results to that of 3-opt, but with a running time of $O(n^2)$. More recently Renaud, Boctor, and Laptorte have presented a restricted version of 4-opt (in a similar vain to Or-Opt) that has a running time of $O(wn^2)$ [11].

Iterative improvement heuristics are often used in combination with the other techniques. In this case they are run on the candidate solution after the initial heuristics has completed. However, for this purpose there is often a fine balance between producing an operation that improves a solution, and one that is sufficiently destructive enough to escape a local minimum. Recently interest in iterative improvement heuristics has grown because of their use as part of meta-heuristic methods.

We present a more detailed description of some classic improvement operations in section 2.3.

2.2.3 Meta-Heuristics

Meta-Heuristics are a broad collection of methods that make few or no assumptions about the type problem being solved. They provide a framework that allows for individual problems to be modeled and 'plugged in' to the meta-heuristic. Typically meta-heuristics take an approach where a candidate solution (or solutions) is initially produced and then is iteratively refined towards the optimal solution. Intuitively meta-heuristics can be thought of searching a problem's problem space. Each iteration searches the neighbourhood of the current candidate solution(s) looking for new candidate solutions that move closer to the goal.

INSERT PIC OF SEARCH SPACE

A limitation of meta-heuristics is that they aren't guaranteed to find an optimal solution (or even a good candidate). Indeed the theoretical underpinnings of what makes a meta-heuristic more effective than another is still poorly understood. Instead meta-heuristics in the literature tend to be tuned for specific problems and then validated empirically.

There have been a number of meta-heuristics produced for the VRP in recent years. Many of the most competitive results produced in the last ten years have been from meta-heuristic approaches. We next review some of the more well known meta-heuristic results for VRP.

Simulated Annealing

Simulated Annealing is inspired by the annealing process used in metallurgy. The algorithm starts with a candidate solution (which can be randomly selected) and then move to nearby solutions with a probability dependent on the quality of the solution and a global parameter T, that is reduced over the algorithm. By analogy to the metallurgy process T represents the current temperature of the solution. Initially T is high. This lets the algorithm free itself from any local minimum that it may be caught in. It is then cooled over time forcing the algorithm to converge to a new solution.

One of the first results was given by Robuste, Daganzo and Souleyrette [7]. They define the search neighbourhood all solutions that can be obtained from the current solution by applying one of three operations: relocating part of a route to another position within the same route, and exchanging customers between two routes. They tested their solution on some large real-world instances of up to 500 customers. They self-reported some success with their method but as their test cases are unique no direct comparison is possible.

Osman has given probably the best know Simulated Annealing results for VRP[19]. His algorithm expands upon many areas of the basic Simulated Annealing approach. The method start by using the Clark and Wright algorithm to produce a starting position. It defines its neighbourhood as candidate solutions that can be searched by applying a $\lambda - interchange$ operation. $\lambda - interchange$ works by selecting two sequences (i.e. chains) of customers S_p, S_q from two routes, p and q, such that $|S_p|, |S_q| < \lambda$ (note the chains aren't necessary of the same length). The customers within each set are then exchanged until an exchange produces an infeasible solution. As the neighbourhood produced by $\lambda - interchange$ is typically large Osman restricts λ to ≤ 2 and take the first move that provides an improvement. Osman also uses a sophisticated cooling schedule; his main change being that the temperature is cooled continuously while improvements are found, or if no improvement he resets the temperature (using $T_i = max(\frac{T_r}{2}, T_b)$, where T_r is the reset temperature, and T_b is temperature of the best solution found so far).

Although Simulated Annealing has produced some good results, and in many cases outperforms the classic heuristics (compare [9] with [15]), it is not competitive with the tabu search implementation.

Tabu Search

Tabu Search follows the general approach shared by many meta-heuristics; it iteratively improving a candidate solution by searching for improvements within the current solution's neighborhood. Tabu search starts with a candidate solution, that may be generated randomly or by using another heuristic. Unlike Simulated Annealing, the best improvement within the current neighbourhood is always taken as the next move. This introduces the problem of cycling between candidate solutions. To overcome this Tabu Search introduces a list of solutions that have already been investigated and are forbidden as next moves (hence its name of Tabu List).

The first instance of Tabu Search being used for VRP is by Willard [25].

Biology inspired methods

2.3 Classic VRP heuristics in depth

This section reviews some of the more common heuristics algorithms. BECAUSE THEY CAN BE USED AS IMPROVEMENT TECHNIQUES WHEN DESIGNING META-HEURISTICS

2.3.1 Nearest neighbour

The most simple algorithm, this codifies the intuitive principle of constructing a solution by picking the job to the depot, then the closest job to this, and so on until a route is filled.

Formally the algorithm creates the sequence $R = [v_0]$ and picks $v \in V$ such that $d_{0,i}$ is minimized. It repeat this until the route represented by R is exceeds it capacity. Then a new route R is constructed and the process is repeated with the remaining jobs. This continues until all jobs are allocated.

Although the Nearest Neighbour algorithm is simple to understand and implement it situations can occur where the performance is bad

2.3.2 Nearest insertion

The algorithm was first proposed by ?

This algorithm slightly improves upon the previous algorithm. The algorithm proceeds much the same as the Nearest Neighbour algorithm but instead of considering only the previous node it considers the entire sequence of jobs route so far in R. It finds an insertion

point along R such that for job v_k insertion between vertices v_i, v_j the function $c_{ik} + c_{kj} - c_{ij}$ is minimized.

2.3.3 Clark and Wright algorithm

Clark-Wright's widely known construction algorithm first appeared in [8]. The algorithm uses a concept of *savings*. The algorithm starts with a separate route for each customer: from depot, to customer, and back to the depot. A saving value is calculated for every pair of customers. The saving value is the distance saved by one customer being serviced directly after the other without returning to the depot in between – in other words the savings gained by combining two partial routes. Routes are then merged based on the largest savings.

The algorithm comes in two flavours: sequential and parallel. The sequential version sequentially adds customers to a route until a route has reached capacity (or based on some other constraint such as maximum route length) and then produces the next. The parallel version builds each route in parallel. The parallel version out performs the sequential version in most cases[9] and is the version considered in our approach.

The algorithm is surprising adaptable and has been extended to deal with more specialized vehicle routing problems were additional objectives and constraints must be considered. The algorithm's flexibility derives from its algebraic treatment of the problem rather than exploiting the problem's underlying spatial properties, as many of the two-phase algorithms do **??**. The algorithms savings formula can be adjusted to accommodate other objectives. An example of this is Solomon's equally ubiquitous algorithm [23] which extends the Clark and Wright algorithm to cater for time constraints.

The algorithm works as follows:

- Prime the solution with routes $R = \{x_0, x_i, x_0\}$ for all jobs $x_i \in X$.
- For all i, j calculate $s_{ij} = c_{i0} + c_{j0} - c_{ij}$.
- Process each S_{ij} in descending order, find routes $r_1 \in R : r_1 = \{0, ..., x_i, x_0\}$ and $r_2 \in R : r_2 = \{0, x_j, ..., 0\}$. Merge these together such that $R' = \{0, ..., x_i, x + j, ..., 0\}$. Stop once capacity constraints or route length constraint is exceeded.

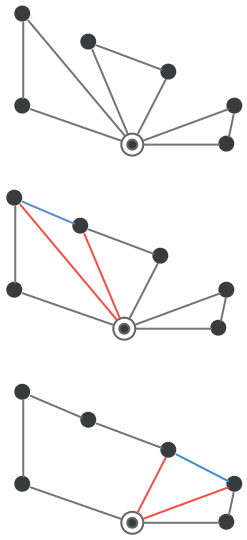

Figure 2.2: Clark Wright Savings Algorithm

2.3.4 2-opt

2.3.5 Or-opt

2.3.6 Osman-opt

2.3.7 Tabu search

2.3.8 Large neighbourhood search

2.4 An introduction to Swarm Intelligence

Chapter 3

Problem Definition

- Full-truck-load pickup and deliveries
- Multiple trucks
- Start at depot. Return to depot
- Time window constraints

3.1 Industry

Full truck load (FTL) movement of freight is a common. Most bulk materials are moved as full truck loads. The classic VRP problem examines the case where a truck is Less than Truck Load (LTL). In this case a truck is loaded with many goods that are to be transported to many different customers. The truck is contained by its capacity and sometime also by the length of the trip. The VRP maps well to the real-world freight business of distribution where once goods have been moved between major centres then they are distributed out to retailers or end customers.

Although in the literature VRP has been extensively studied a comparatively small amount has been published on PDP. As far as we know no papers have been published on the PDP for full truck loads.

In transporting bulk materials, such as Logging, Petroleum, Cement, ? the more common model is where the truck is filled to capacity at the primary manufacturing plant (e.g. cement plant, forestry skid site) and then distributed as a FTL (or many FTL) to a resellers such as a concrete plant or saw mill. In bulk transport the transport planner will try to minimize the amount of empty running by maximizing the amount of back-loading possible.

Empty-running is where a vehicle is traveling between jobs without any goods. For the bulk transport planner this total distance traveled isn't important as their rates are structured

so that this travel is reflected in the cost of the job. A bulk transport company on the other hand isn't paid for time spent traveling between jobs.

A back-load is simply were a series of pickups and deliveries can be chained together such that empty-running time is minimized.

3.1.1 Operation

3.1.2 Objectives

- Minimize between job travel distance

3.1.3 Constraints

3.2 Formal definitions

3.2.1 Classic VRP

We formulate the VRP here as an integer programming problem. Although it is possible to solve the VRP as an integer programming problem this approach is only feasible for small problem sizes. Rather the problem is presented here in this fashion so that it can be stated precisely.

The VRP is defined on a graph (V, E). The vertices of the graph V represent customers with v_0 and v_{n+1} representing the depot where the vehicle starts and ends each route. Customers are denoted by $C = 1, 2, ..., n$. The set of edges E corresponds to possible connections between customers. For our use here all connections are possible – that is it is possible for a vehicle to drive between any two customers – except where the depot is concerned. No edge terminates at v_0 and no edge originates at v_{n+1} and there is no edge $0, n + 1$. Each edge $i, j \in E$ has a corresponding cost c_{ij} which typically represents the travel distance between customers.

The set of routes is denoted by R and each route has a maximum capacity q. Each customer has a demand $d_i, i \in C$.

The model contains the decision variable X_{ij}^r which is defined $\forall i, j \in E, \forall r \in R$. X_{ij}^r is equal to 1 if route r contains a trip from customer c_i to customer c_j and 0 otherwise.

(3.1)

Minimize:

$$\sum_{r \in R} \sum_{ij \in E} c_{ij} X_{ij}^r \tag{1}$$

Subject to:

$$\sum_{r \in R} \sum_{j \in V} X_{ij}^r = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{i \in C} d_i \sum_{j \in C} X_{ij}^r \leq q \qquad \forall r \in R \tag{3}$$

$$\sum_{j \in V} X_{0j}^r = 1 \qquad \forall r \in R \tag{4}$$

$$\sum_{j \in V} X_{j0}^r = 1 \qquad \forall r \in R \tag{5}$$

$$\sum_{i \in V} X_{ik}^r - \sum_{j \in V} X_{kj}^r = 0 \qquad \forall k \in C \text{ and } \forall r \in R \tag{6}$$

The objective function (1) aims to minimize costs c_{ij}. Constraint (2) states that each customer can only be serviced by a single route. Constraint (4) enforces the capacity constraint: each route can not exceed the maximum vehicle capacity. Constraint (5) and (6) ensure that each route starts at exactly 1 depot vertex and finishes at exactly one depot vertex. Lastly, constraint (7) is a flow constraint that ensures that the number of vehicles entering a customer is equal to the number of vehicles leaving.

3.2.2 PDP

We now similarly define the PDP.The PDP generalizes the VRP. Goods can now be picked up from a customer or delivered to a customer along a single route. In contrast in the VRP all jobs are either picking up goods or delivery goods to a customer. The PDP is defined on a graph (V, E). The vertices of the graph V represent all pickup and delivery locations along with v_0 and v_{2n+1} representing the depot where the vehicle starts and ends each route. Pickup jobs are denoted by $P = \{p_1, p_2, ..., p_n\}$ and delivery jobs by $D = \{d_1, d_2, ..., d_n\}$. The set of edges E corresponds to the possible connections between jobs. For our use here all connections are possible – that is it is possible for a vehicle to drive between any two jobs – except where the depot is concerned. No edge terminates at v_0 and no edge originates at v_{2n+1} and there is no edge $0, 2n + 1$. Each edge $i, j \in E$ has a corresponding cost c_{ij} which typically represents the travel distance between customers.

The set of routes is denoted by R and each route has a maximum capacity q. Each location has a demand $d_i \in V$. Pickup jobs have a positive demand whereas deliveries are assumed to have a negative demand. A route containing a pickup $p_i \in P$ must also contain its corresponding delivery $d_i \in D$.

The model contains the two decision variables: X_{ij}^r which is defined $\forall i, j \in E, \forall r \in R$. X_{ij}^r is equal to 1 if route r contains a trip from location v_i to location v_j and 0 otherwise.

Let y_i denote a an intermediate variable that stores the total load of a vehicle traveling route r and visiting location v_i.

(3.2)

Minimize:

$$\sum_{r \in R} \sum_{ij \in E} c_{ij} X_{ij}^r \tag{1}$$

Subject to:

$$\sum_{r \in R} \sum_{j \in V} X_{ij}^r = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{j \in V} X_{0j}^r = 1 \qquad \forall r \in R \tag{4}$$

$$\sum_{j \in V} X_{j0}^r = 1 \qquad \forall r \in R \tag{5}$$

$$\sum_{i \in V} X_{ik}^r - \sum_{j \in V} X_{kj}^r = 0 \qquad \forall k \in C \text{ and } \forall r \in R \tag{6}$$

$$p_i \in r \Rightarrow d_i \in r \forall r \in R \tag{7}$$

$$p_i \text{ appears before } d_i \in r \forall r \in R \tag{8}$$

$$X_{ij}^r = 1 \Rightarrow y_i + d_i = y_j \forall r \in R \text{ and } \forall i, j \in V \tag{9}$$

$$y_0 = 0 \tag{10}$$

$$\sum_{r \in R} \sum_{j \in V} X_{ij}^r y_i \leq q \qquad \forall i \in V \tag{11}$$

The objective function (1) aims to minimize costs c_{ij}. Constraint (2) states that each customer can only be serviced by a single route. Constraint (4) and (5) ensure that each route starts at exactly 1 depot vertex and finishes at exactly one depot vertex. Constraint (6) is a flow constraint that ensures that the number of vehicles entering a customer is equal to the number of vehicles leaving.

Constraint (7) states that jobs pickup and delivery jobs are serviced by the same route. Constraint (8) enforces that goods are picked up before being delivered. And constraints (9) through (11) ensure that each vehicle's capacity isn't exceeded.

3.2.3 PDP-FTL

The mathematical model can be represented by PDP above. As the demand on each job fills the truck we can simplify the model so that each pickup and delivery constraint is paired. We also add two additional constraints: vehicles fleet size is fixed and each route has a maximum distance travelled (this is used to limited the work undertaken to a shift). These aren't required for the general case but are important in most real-world applications of the problem.

We start by using the same framework given in the PDP formulation. We replace the PDP concept of pickup and delivery locations, with the simpler notation of jobs $J = \{1, 2, ..., n\}$. Each job $j\ inJ$ is represented by the arc j_p, j_d which represents the pickup and delivery locations of the job. Each job can only be serviced by a single route $rinR$.

The number of routes is set to the fixed number of vehicles we have available $|R| = k$.

(3.3)

Minimize:

$$\sum_{r \in R} \sum_{ij \in E} c_{ij} X_{ij}^r \tag{1}$$

Subject to:

$$\sum_{r \in R} \sum_{j \in V} X_{ij}^r = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{j \in V} X_{0j}^r = 1 \qquad \forall r \in R \tag{4}$$

$$\sum_{j \in V} X_{j0}^r = 1 \qquad \forall r \in R \tag{5}$$

$$\sum_{i \in V} X_{ik}^r - \sum_{j \in V} X_{kj}^r = 0 \qquad \forall k \in C \text{ and } \forall r \in R \tag{6}$$

$$p_i \in r \Rightarrow d_i \in r \forall r \in R \tag{7}$$

$$p_i \text{ appears before } d_i \in r \forall r \in R \tag{8}$$

$$X_{ij}^r = 1 \Rightarrow y_i + d_i = y_j \forall r \in R \text{ and } \forall i, j \in V \tag{9}$$

$$y_0 = 0 \tag{10}$$

$$\sum_{r \in R} \sum_{j \in V} X_{ij}^r y_i \leq q \qquad \forall i \in V \tag{11}$$

The objective function (1) aims to minimize costs c_{ij}. Constraint (2) states that each job can only be serviced by a single route. Constraint (4) enforces the capacity constraint: each

route can not exceed the maximum vehicle capacity. Constraint (5) and (6) ensure that each route starts at exactly 1 depot vertex and finishes at exactly one depot vertex. Constraint (7) is a flow constraint that ensures that the number of vehicles entering a customer is equal to the number of vehicles leaving.

Constraint (8) states that jobs pickup and delivery jobs are serviced by the same route. Constraint (9) enforces that goods are picked up before being delivered. And constraints (10) through (**??**) ensure that each vehicle's capacity isn't exceeded.

Objectives

- Minimize between job travel distance

Constraints

3.2.4 VRPTW

3.2.5 PDPTW

IN REAL WORLD CASE OFTEN VEHICLE NUMBER IS SET. IN THIS CASE WE WANT TO MAKE MAXIMUM USE OF THE VEHICLES THAT ARE AVAILABLE SINCE WAGES ARE PAID REGARDLESS.

Chapter 4

Algorithm

4.1 Goals

Design goals:

- Fast
- Minimal
- Take advantage of modern parallel hardware

4.2 Algorithm

4.2.1 Classic Bees algorithm

4.2.2 Representation

4.2.3 Changes from classic Bees Algorithm

Deeper driller

Tabu List

Aged Sites

Different Operations

4.2.4 Operations

2-Opt

3-Opt

Vertex-Swap

Inverse

Large Neighbourhood Search

Chapter 5

Results

5.1 Test instances

5.2 Results

5.3 Comments

Chapter 6

Conclusion

6.1 Future Directions

Chapter 7

Appendix A - Implementation

This chapter details how the optimization system is implemented. The system's architecture, technologies employed in the system and optimisations done to the system are detailed here.

7.1 Parallelization

Actor Model

7.2 Source Code

Get it here ...

Bibliography

[1] M.L. Balinski and R.E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

[2] L.R. Miller B.E. Gillett. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22:340–349, 1974.

[3] Mingozzi A. Christofides, N. and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981a.

[4] N. Christofides and S. Eilon. An algorithm for the vehicle dispatching problem. *ORQ*, 20:309–318, 1969.

[5] Fulkerson Dantzig and Johnson. Solution of a large-scale traveling salesman problem. *Operations Research*, (2):393410, 1954.

[6] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, Oct, 1959.

[7] C.F. Daganzo F. Robuste and R. Souleyrette II. Implementing vehicle routing models. *Transportation Research*, 24B:263–286, 1990.

[8] J.W. Wright G. Clark. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[9] Frederic Semet Gilbert Laporte. Classical heuristics for the vehicle routing problem. Technical report, Les Cahiers du Gerad, 1999.

[10] W.R. Hamilton. Memorandum respecting a new system of roots of unity (the icosian calculus). *Philosophical Magazine*, 12, 1856.

[11] G Laporte J Renaud, F F Boctor. A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing*, 8:134–143, 1996.

[12] B. Gavish K. Altinkemer. Parallel savings based heuristic for the delivery problem. *Operations Research*, 39:456–469, 1991.

[13] T.P. Kirkman. On the representation of polyhedra. *Philosophical Transactions of the Royal Society of London Series A*, 146:413–418, 1856.

[14] G. Laporte and Y Nobert. *Surveys in Combinatorial Optimization*, chapter Exact algorithms for the vehicle routing problem. 1987.

[15] J Potvin M Gendreau, G Laporte. Metaheuristics for the vehicle routing problem. Technical report, Les Cahiers du Gerad, 1998, revised 1999.

[16] A H G R Kan M Haimovich. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10:527–542, 1985.

[17] Road Transport Forum NZ. Road transport forum nz - transport facts. Website.

[18] I Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Evanston, IL: Northwestern University, 1976.

[19] I. H. Osman. Metastrategy simulated annealing and tabu search algorithm for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

[20] S.R. Jameson R.H. Mole. A sequential route-building algorithm employing a generalized saving criteria. *Operations Research*, 27:503–511, 1976.

[21] J. Robinson. On the hamiltonian game (a traveling salesman problem). *Research Memorandum RM-303*, 1949.

[22] Alexander Schrijver. On the history of combinatorial optimization (till 1960). *?*, ?:?, ?

[23] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.*, 35(2):254–265, 1987.

[24] Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[25] J. A. G. Willard. Vehicle routing using r-optimal tabu search., 1989.