Bees Algorithms, and the Logging 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

Chapter 1

Introduction

1.1 Motivation

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[**?**]. 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[**?**]:

- 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[**?**].

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

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 A short history of the Vehicle Routing Problem

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 [**?**] and Kirkman [**?**]. The traveling salesman problem has been informally discussed probably for many years [**?**] 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 [**?**] 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.

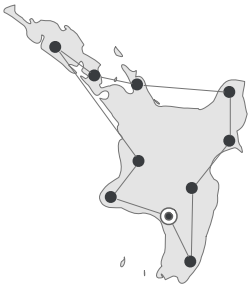

Figure 2.1: Traveling Salesman Problem

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

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 [**?**] 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.

Exact Methods

The first efforts were concerned with exact solutions, and proceeded using many of the same techniques brought to bear on TSP. Following the Laporte and Nobert's survey [**?**], exact algorithms for the VRP can be classified into three categories: Direct tree search methods, Dynamic programming, and Integer linear programming.

Balinski, and Quandt used a set partitioning algorithm to produce exact solutions [**?**]. 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. 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.

A 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.

Christolds and Ellison gave the original branch and bound algorithm for exactly solving the VRP[**?**]. Unfortunately it's time and memory requirements where such that it was only able to solve problems of up to 13 customers. This was later improved upon again 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, [**?**] 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.

Notwithstanding the above results, exact methods have been of more use in advancing theoretical understanding of VRP than to providing solutions to real life cases. 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.

Approximate Methods

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 [**?**]. In this paper they present an approxmiate approach to solving

VRP is one of the most the studied combinator optimization problems. Hundreds of papers been written on it

Out of VRP comes a family of related problems. These attempt to model other constraints that are encountered in real-world applications of the VRP. Classic problems have included: VRP with Time Windows that introduces a time window constraints against each customer, that the vehicle must arrive within. VRP multiple

depot, where the vehicles are displayed from multiple start points.
VRP with Pick

VRPTW

PDPTW

2.2 An introduction to classic VRP heuristics

This section reviews some of the more common heuristics algorithms.

Classic VRPheuristics can be broadly classified into three categories. Constructive heuristics gradually build a feasible solution while keeping an eye on teh solution cost, but do not contain an improvement phase per se. In two-phase heuristics, the problem is decomposed into its two natural components: clustering of vertices into feasible routes and actual route construction, with possible feedback loops between the two stages. Two-phase heuristics will be divided into two classes: cluster-first, route second, methods adn route-first, cluster-second methods. In the first case, vertices are first organized into feasible clusters and a vehicle route is constructed for each of them. In the second case, a tour is first built on all vertices and is then segmented into feasible vehicle routes. Finally, improvement methods attempt to upgrade any feasible solution by performing a sequence of edge or vertex exchanges within or between routes.

Construction algorithms

These following algorithms are heuristics for constructing an initial solution

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

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.

Clark and Wright algorithm

Clark-Wright's widely known construction algorithm first appeared in [**?**]. 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[**?**] 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 2.2. The algorithms savings formula can be adjusted to accommodate other objectives. An example of this is Solomon's equally ubiquitous algorithm [**?**]

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.

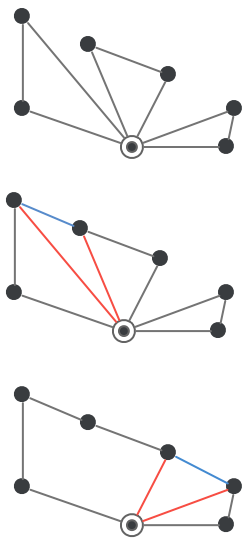

Figure 2.2: Clark Wright Savings Algorithm

2-phase construction algorithms

Cluster first

We use kmeans clustering. The kmeans algorithm isn't guaranteed to produce same result each run. This is conventionally viewed

as limitation of the algorithm. However for our purpose this is an advantage as we can use the randomness to better seed the population with different solutions.

Sweep

Improvement heuristics

2-opt

Or-opt

Or-opt improvements

GENI

λ-interchanges

Ejection chains

I want to know more about ejection chains

Meta-heuristics

Tabu search

Genetic algorithms

Memetic algorithms

Large neighbourhood search

2.3 Real-world implementations

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.

Operation

Objectives

- Minimize between job travel distance

Constraints

3.2 Formal definitions

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.

PDP

We now similarly define the PDPThe 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.

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 $j\ in J$ 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 $r in R$.

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

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 Objective

Design goals:

- Make things fast
- Interactive with user
- Soft variables
- Take advantage of modern parallel hardware

- Full-truck-load pickup and deliveries
- Multiple trucks
- Start at depot. Return to depot
- Time window constraints
- Number of vehicles set already.
- If jobs are charged on travel time then we'd want to minimize the amount of distance travelled between jobs. In which case travel distance (between jobs) would be the thing to optimize.
- Note: We don't care about AVG number of jobs completed, because each job is charged for time taken.

Algorithm

Sketch

Memetic representation

Direct representation. Using 2-dim array of routes vs. job assignments.

Search neighbourhood

Tabu list. Probability plan.

Construction phase

Something on modifications to classic methods to add in random element

Nearest Insertion Heuristic for PDP-FTL

Convex whole improvement doesn't work

Lemma 1. *Moreover we can also show that for there doesn't exist an algorithm $A \in P$ that produces a subset $S \subset V$ such that S is part of an optimal tour.*

PROOF REQUIRED

Clark-Wright Savings Heuristic for PDP-FTL

Selection

Local improvement

Two-opt

Or-opt

Re-combination

Permutation cross

Successful move

Optimizations

CHEAP SPATIAL INDEXING USING GEOHASHES

In addition to these basic implementation alternatives, Johnson and McGeoch (2002) describe four speed up rules: 1) avoiding search space redundancies: when implementing r-opt, time can be saved by avoiding some exchanges that cannot improve the solution; 2) bounded neighbour lists: only consider the p closest neighbours of a vertex when per- forming reconnections (this rule

was implemented by Zweig, 1995); 3) dont look bits: avoid considering certain moves if such moves have proved fruitless in the past; 4) tree-based tour representation: use a tree-based representation of the tour to accelerate computations. [**?**]

Chapter 5

Problem Dataset

Chapter 6

Results

6.1 Test instance generation

6.2 Test instance results

- Some comments on results showing that 80% improvements are made in first 20% of time.

Solomon instances [**?**].

The geographical data are randomly generated in problem sets R1 and R2, clustered in problem sets C1 and C2, and a mix of random and clustered structures in problem sets by RC1 and RC2. Problem sets R1, C1 and RC1 have a short scheduling horizon and allow only a few customers per route (approximately 5 to 10). In contrast, the sets R2, C2 and RC2 have a long scheduling horizon permitting many customers (more than 30) to be serviced by the same vehicle. The customer coordinates are identical for all problems within one type (i.e., R, C and RC). The problems differ with respect to the width of the time windows. Some have very tight time windows, while others have time windows which are hardly constraining. In terms of time window density, that is, the percentage of customers with time windows, I created problems with 25, 50, 75 and 100 % time windows. The larger problems are 100 customer euclidean problems where travel times equal the corresponding distances. For each such problem, smaller problems have been created by considering only the first 25 or 50 customers.

6.3 Case Study - McCarthys Transport Limited

Chapter 7

Conclusion

Chapter 8

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.

8.1 Implementation goals

SECTION DESCRIPTION

Take advantage of parallelization possible in memetic algorithm.

8.2 Architecture

Logical architecture

Technology stack

Heuristic algorithms

The Strategy Design pattern was used extensively for allowing different heuristics to be swapped out. This made it easy to increase the search neighbourhood size and it also made it easy to extend the system.

Process distribution

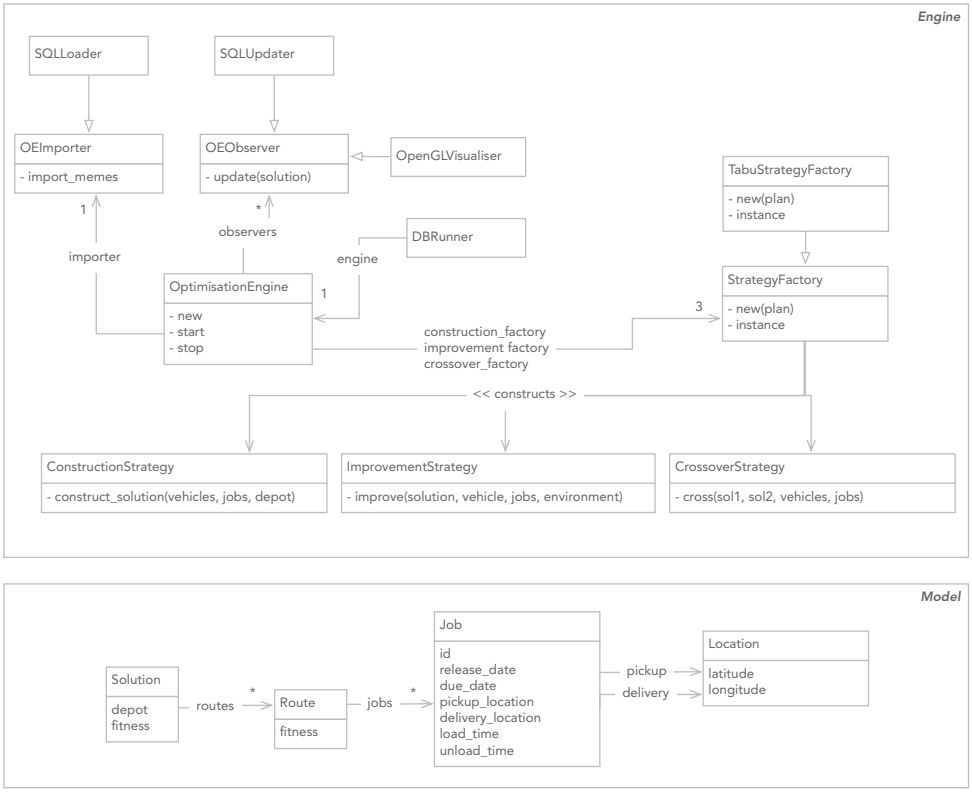

Figure 8.1: Logical Architecture