Optimal routing of waste collection vehicles

Carel Calitz

26/08/08

Executive Summary

Waste management forms an essential part of any municipality's service towards the public. This includes the collection, transport and disposal of solid waste. This project aims at improving the transportation side of waste management by reducing the distance run by waste collection vehicles. Several models in the literature exist that can be used to improve the collection routes that the waste collection vehicles travel on. The contribution of this project is the creation of a model that is able the handle several extensions:

1. Turn penalties e.g. right turns at a busy intersection.
2. Forbidden turns e.g. U-turns.
3. Several dumping sites.
4. Windy edges, whose cost depends on the direction of travel e.g. a sloping street.

A generic Tabu Search Algorithm is proposed for the design of this model. The results obtained from testing the model on several benchmark problems as well as a set of problems that were created specifically for this model indicates the models capability to improve waste collection routes. This document can be viewed as a step toward solving more general real live waste collection problems that include the extensions mentioned above, as is often the case in practice.

Contents

List of Figures

List of Tables

Acronyms

ARP	Arc Routing Problem
CARP	Capacitated Arc Routing Problem
CARPIF	Capacitated Arc Routing Problem with Intermediate Facilities
CARPTP	Capacitated Arc Routing Problem with Turn Penalties
CPP	Chinese Postman Problem
GAs	Genetic Algorithms
IFs	Intermediate Facilities
MA	Memetic Algorithm
PCARP	Periodic Capacitated Arc Routing Problem
RPP	Rural Postman Problem
SA	Simulated Annealing
SCARP	Stochastic Capacitated Arc Routing Problem
TS	Tabu Search
WCP	Waste Collection Problem

Chapter 1

Introduction

Waste management forms an essential part of any municipality's service towards the public. This includes the collection, transport and disposal of solid waste. In recent years waste management has become an area of concern for municipalities world wide due to population growth (especially in urban areas), environmental concerns and the progressive increase in waste management cost. These factors force municipalities to asses their solid waste management to identify improvement opportunities like the optimal location of dumping sites, location of waste bins, the number of vehicles required and the optimal routing of waste collection vehicles.

Toth and Vigo (2002) classify waste into three major categories: commercial waste; residential waste and roll-on-roll-off waste. These three categories bring about three different waste collection strategies.

Commercial waste is found at small businesses, restaurants and apartment flats that place their waste in large containers that are scattered throughout the city's geographic area. Waste collection involves point-to-point collection as service is only required at selected points.

Residential waste is found at houses along street networks in small containers or garbage bags. Waste collection is required at nearly all locations along the given street network. Residential waste forms the biggest percentage of waste collection area that need to be collected by municipalities.

Roll-on-Roll-off waste collection involves the pickup, transportation, unloading and drop-off of large containers typically found at construction sites, industrial areas and other high volume locations. The roll-on-roll-off waste collection vehicle can usually only carry one container and follow one of two methods, round trip or exchange trip. For a round trip the vehicle picks up a full container at a site, transport it to the landfill for emptying and then returns the empty container to the site. Whereas, for exchange trip the vehicle picks up an empty container at the landfill, transport it to a site to exchange it with a full container and then returns the full container to the landfill.

The context of this project is to focus on residential waste collection which can be described as follows: Residential waste is located in containers along the streets of a defined urban area. All the containers must be collected by a fleet of vehicles whose capacity cannot be exceeded. Each vehicle starts the day from a depot and can serve several streets before its capacity is reached. Once the vehicle's capacity is reached it travels to a landfill located outside the city to unload. The vehicle then returns to the city to start its second trip. This process is repeated until all of the streets and containers assigned to the vehicle have been serviced. The vehicle completes its final trip by proceeding to the landfill to unload for the last time, and then returns to the depot. The vehicle fleet and the service demand are considered a given and the problem will only address the improvement of current waste collection routes and will from here on be referred to as the Waste Collection Problem (WCP).

1.1 Research Question

Waste collection can be considered as a logistical activity where municipalities must collect waste at residents in the most cost effective way as possible. This is usually done by subdividing the municipality's service area into collection quarters and then assigning a vehicle to each quarter. However, the astonishing rate at which the countries urban areas are expanding and the ever increasing fuel price may lead to insufficient solutions in terms of cost and time constraints. This has forced municipalities to critically re-evaluate their waste management systems and identify improvement opportunities that can reduce their operational cost. This gave birth to the following research question:

> *How should waste collection vehicles be routed to ensure quality efficient service at a minimum cost by reducing the total distance travelled by the collection vehicles within each day?*

1.2 Research design

The foremost goal of this project is to develop a model capable of computing optimal collection routes for a municipalitys service area. The model should be able to accurately represent this service area by including turn penalties and Intermediate Facilities (IFs). Comprehensive research is performed with the intent to discover possible techniques to solve the routing problem. The developed model addresses the need to improve the routing of waste collection vehicles to reduce the cost associated with waste collection. Several benchmark problems created by various authors are available in the literature to be used as experimental data to verify and evaluate the quality of the results obtained by the designed model.

1.3 Research methodology

Research was done with the aim of developing a generic model that can be used to design or redesign any municipalities waste collection network. In order to achieve this, the Operation Research process adapted from Rardin (1998) will be followed:

Problem The first and most important phase of the research process is to ensure correct understanding of the problem. The problem should be scoped in such a matter that the solution will represent and address the core problem of waste collection.

Model The creation of the model will be based upon the problem definition and relative literature study to present an accurate representation of the waste collection environment. The model consists out of decision variables that will guide the solution strategy developed, constraints that limit the problem according to certain decisions, assumptions and facts and an objective function that indicates the output choices.

Solution This phase entails the identification of appropriate solution techniques, either exact or approximate, to design optimal or near optimal waste collection routes.

Decision This phase entails interpreting the numerical results found in the solution phase that should be used as a decision support tool to solve the problem. A detailed analysis should be done to assess whether or not the impact of the decisions will address the core objective of the problem. The implemented changes may either address the core objective of the problem or bring about new problems that should undergo the operational research process of modelling, solving and decision making. This will ensure that the best solution to the WCP is found.

1.4 Document structure

Chapter 2 provides insight into various literature about the different solutions strategies, both exact and approximate, that are available to solve the WCP. A detailed description of the development of the metaheuristic used to optimise the routes for the WCP is given in Chapter 3. The metaheuristic is tested on a set of benchmark problems, followed by the computational results for the implementation on a real network. Final conclusions are made in Chapter 5, indicating possible improvements and opportunities for future research.

Chapter 2

Literature review

Various studies on the WCP have been documented in literature including several ideas as well as modelling approaches that can be followed. The WCP is generally modelled as either a node routing problem or an arc routing problem. The approach used depends on whether service is required at selected points throughout the geographical area, as in the case of commercial waste, or if service is required at almost all locations in a street segment, such as residential waste.

This project will focus on the latter, since it requires the optimal routing of waste collection vehicles within an urban area that must collect all of the waste, along the given street network. As a result, the WCP is defined as an Arc Routing Problem (ARP)

2.1 Arc Routing Problem

ARPs are constrained to traverse and service certain arcs within a graph, instead of just visiting certain nodes. The arcs typically represent streets that require service along most or the entire street segment.

Eiselt and Laporte (1995) formulate the ARP as follows: let $\mathbf{G} = (\mathbf{V},\mathbf{A})$ be a connected graph without loops where $\mathbf{V} = \{v_1,\ldots,v_n\}$ is the vertex set (or node set), and $\mathbf{A} = \{(v_i,v_j) : v_i,v_j \in \mathbf{V} \text{ and } i \neq j\}$ is the arc set. With every arc (v_i,v_j) is associated a nonnegative cost, distance or length c_{ij}, assume that $c_{ij} = \infty$ if (v_i,v_j) is not defined. The matrix C = (c_{ij}) is symmetric if and only if $c_{ij} = c_{ji}$ for all i, j. When C is symmetric, it is common to associate an edge (or undirected arc) with every vertex pair. Hence, depending whether $\mathbf{A}$ is a set of arcs, edges or a combination of both, the associated graph will be termed directed, undirected or mixed (Figure 2.1). The aim of an ARP is to determine a least cost traversal of a specified arc subset of a graph, with or without constraints.

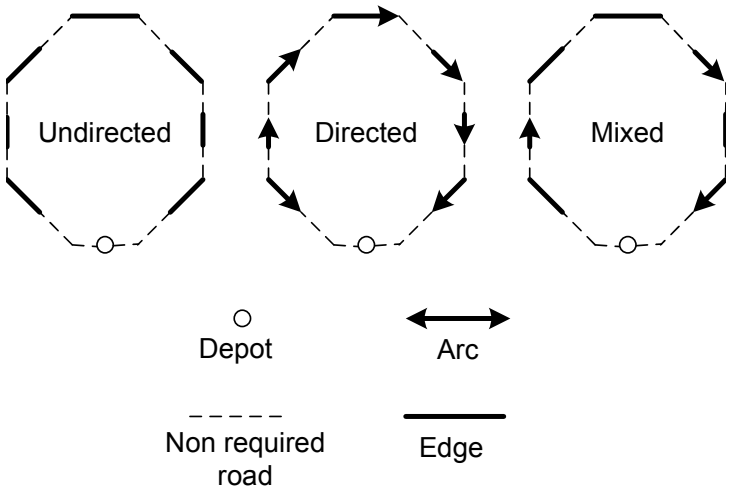

Figure 2.1: Different graph representations

More generally for the WCP, a vertex set represents street crossings or dead-ends. An edge represents small two way streets whose sides can be collected in parallel and in any direction. Arcs represent one way streets or larger two way streets whose sides is collected independently and in the direction of the arc (Figure 2.2). The aim is to find the least cost traversal of all the given streets within the network

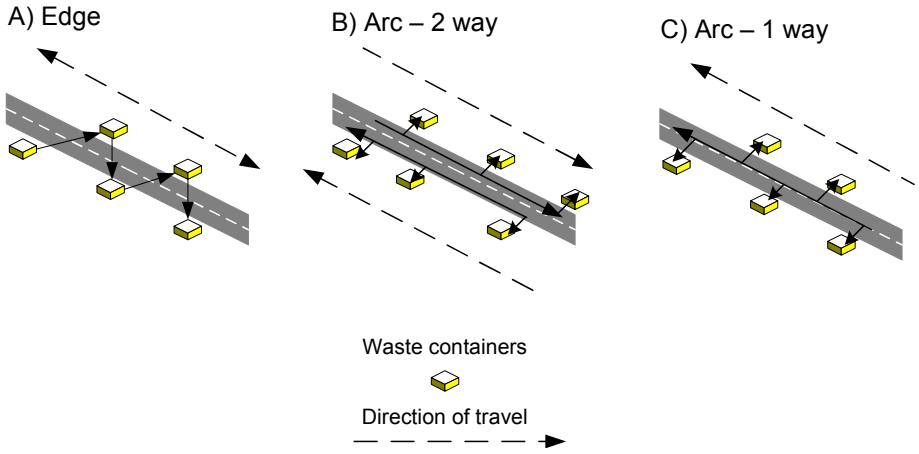

Figure 2.2: Difference between edge and arc

Two important ARPs can be derived from general routing problems, the Chinese Postman Problem (CPP) and the Rural Postman Problem (RPP). The CPP is commonly associated with mail delivery in urban settings that seeks the least cost traversal of all the streets. In turn, the RPP is associated with rural settings and can be described as follows: There are a number of villages whose set $\mathbf{R}$ of streets has to be serviced by a postman, and a set $\mathbf{A} \setminus \mathbf{R}$ of links between the villages that do not have to be served, but may be used for travelling between villages. It seeks the least cost traversal of the subset $\mathbf{R} \subseteq \mathbf{A}$ of required streets (Eiselt and Laporte, 1995). Hence, the RPP can more accurately model

real life arc routing applications such as the WCP. The WCP consist of various streets that may not have to be serviced as they have no waste and will just be used to travel to streets that require services. However, the WCP involve additional characteristics such as capacity constraints and multiple vehicles and for this reasons can not be modelled as a pure RPP but rather be modelled as a Capacitated Arc Routing Problem (CARP).

2.2 Capacitated Arc Routing Problem

The CARP is closely related to the RPP and was first introduced by Golden and Wong (1981). It is an important problem in the area of arc routing since it considers the capacity restrictions of the vehicles involved, which make it more applicable for real life applications. The routing of waste collection vehicles, snow removal vehicles and street sweepers are good examples of CARPs.

Golden and Wong (1981) describes the CARP as follows: consider an undirected graph $\mathbf{G} = (\mathbf{V},\mathbf{E})$ with a vertex set $\mathbf{V}$, a edge set $\mathbf{E}$ and a set of required edges $\mathbf{R} \subseteq \mathbf{E}$. A fleet of k homogeneous vehicles of capacity W are based at a designated depot vertex v_0. Each edge of the graph (v_i,v_j) incurs a cost c_{ij} and has a demand q_{ij} associated with it. The subset $\mathbf{R}$ of edges must be serviced by a vehicle and the remaining edges of $\mathbf{E}$ may be traversed any number of times. The CARP consists of determining a set of vehicle routes of minimum total cost, such that each trip starts and ends at the depot, each required edge is serviced by a single trip and during one traversal and the total demand for each trip does not exceed vehicle capacity W. The graph or network on which the CARP is based may be undirected, directed or mixed depending on the road network topology and operating policies involved.

Eiselt and Laporte (1995) proposed an integer linear programming formulation for the undirected CARP by replacing eah edge with two arcs creating a directed formulation of the CARP that is presented as follows:

$$x_{ijk} \triangleq \begin{cases} 1 & \text{if edge } (v_i, v_j) \in \mathbf{A} \text{ is traversed from } v_i \text{ to } v_j \text{ by vehicle } k, \\ & \text{where } k = \{1,\ldots,m\}, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ijk} \triangleq \begin{cases} 1 & \text{if edge } (v_i, v_j) \in \mathbf{A} \text{ is serviced by vehicle } k \text{ while traveling} \\ & \text{from } v_i \text{ to } v_j, \text{ where } k = \{1,\ldots,m\}, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

$c_{ij} \triangleq$ The cost or distance of edge (v_i, v_j)

$q_{ij} \triangleq$ The demand of edge (v_i, v_j)

$W \triangleq$ Capacity of the vehicles

$S \triangleq$ A given vertex set.

$$\min \sum_{k=1}^{m} \sum_{(v_i,v_j)\in \mathbf{A}} c_{ij} x_{ijk} \tag{2.1}$$

s.t.

$$\sum_{(v_i,v_j)\in \mathbf{A}} x_{jik} - \sum_{(v_i,v_j)\in \mathbf{A}} x_{ijk} = 0 \qquad \forall\ v_i \in \mathbf{V}, k = 1,\ldots,m \tag{2.2}$$

$$\sum_{k=1}^{m} (y_{ijk} + y_{jik}) = \begin{cases} 0 & \text{if } q_{ij} = 0 \\ 1 & \text{if } q_{ij} > 0 \end{cases} \qquad \forall\ (v_i, v_j) \in \boldsymbol{A} \tag{2.3}$$

$$x_{ijk} \geq y_{ijk} \qquad \forall\ (v_i, v_j) \in \mathbf{A}, k = 1,\ldots,m \tag{2.4}$$

$$\sum_{(v_i,v_j)\in \mathbf{A}} q_{ij} y_{ijk} \leq W \qquad \forall\ k = 1,\ldots,m \tag{2.5}$$

$$\sum_{v_i,v_j\in \mathbf{S}} x_{ijk} \leq |S| - 1 + n^2 u_k^S \qquad \forall\ S \subseteq V\backslash\{v_1\}; S \neq \varnothing; k = 1,\ldots,m \tag{2.6}$$

$$\sum_{v_i\in \mathbf{S}} \sum_{v_j\notin \mathbf{S}} x_{ijk} \geq 1 - w_k^s \qquad \forall\ S \subseteq V\backslash\{v_1\}; S \neq \varnothing; k = 1,\ldots,m \tag{2.7}$$

$$u_k^s + w_k^s \leq 1 \qquad \forall\ S \subseteq V\backslash\{v_1\}; S \neq \varnothing; k = 1,\ldots,m \tag{2.8}$$

$$u_k^s, w_k^s \in \{0,1\} \qquad \forall\ S \subseteq V\backslash\{v_1\}; S \neq \varnothing; k = 1,\ldots,m \tag{2.9}$$

$$x_{ijk}, y_{ijk} \in \{0,1\} \qquad \forall\ (v_i, v_j) \in \mathbf{A}; k = 1,\ldots,m \tag{2.10}$$

In this formulation, the objective function (2.1) minimises the total cost induced by the k vehicles. Constraints (2.2) are flow conservation equations for each vehicle. Constraints (2.3) ensure that service arcs correspond to those with a positive demand. Constraints (2.4) state that an arc is serviced by a vehicle only if it is traversed by the same vehicle. Constraints (2.5) enforce capacity restrictions of vehicle k. Constraints (2.6) to (2.9) ensure that the solution does not contain any illegal subtours.

The basic CARP can sometimes be too simplistic to model accurate representations of real world instances, leading to various variants of the CARP.

Capacitated Arc Routing Problem with Intermediate Facilities(CARPIF)

The CARPIF, first introduced by Ghiani et al. (2001), requires that a vehicle need to unload or replenish at Intermediate Facilities (IFs). For the WCP, a vehicle starts in the morning at an assigned depot. Waste is collected along the streets until the vehicle's capacity is reached, after which the vehicle needs to dump the waste at the nearest dumping site (IF), which may or may not include the depot.

Ghiani et al. (2001) and Polacek et al. (2007) solve this problem by adding a subset **I** of intermediate facilities to the CARP. Constraints are added such that the waste collected

between the depot and first IF, or between two IFs may never exceed the vehicle's capacity W.

Capacitated Arc Routing Problem with turn penalties (CARPTP)

The basic CARP assumes that all turns are allowed and are not time consuming, but this may not by the case when routes have to be operated within a city. Some turns can be considered forbidden, others more time consuming or dangerous, especially for large waste collection vehicles. U-turns may be impossible to make due to narrow streets or forbid by traffic rules. Even right turns at robots or busy intersections may be more time consuming and should therefore be penalized.

Belenguer et al. (2006) provides a method for including turn penalties in the CARP based on the work of Benavent and Solver (1999). This method allows the solving of the problem by adding a penalty cost associated with each turn to the objective function.

Stochastic Capacitated Arc Routing Problem (SCARP)

The SCARP can be defined as a problem having some element of uncertainty. The uncertainty can for instance be the demand of the customer or the travel times of the vehicles. In the waste collection problem the stochastic component refers to the demand of the customers and is treated as a decision variable in the problem (Chu et al., 2006).

Periodic Capacitated Arc Routing Problem (PCARP)

In waste collection problems it is not uncommon for municipalities to schedule service only on certain days for instance only twice a week. This period can vary for area to area depending on the population and demand of the given areas. Daily removal may be too expensive and trips will rather be planned over a multi-period time frame (Chu et al., 2006).

In order to make our model for the WCP more realistic, the problem formulation will be extended to include IFs and turn penalties.

2.3 Solution Approaches

According to Winston and Venkataramanan (2004) problems that can be solved in polynomial time are typically solved to optimality using efficient algorithms and exact methods.

Golden and Wong (1981) demonstrated that even finding a near optimum (within 50%) solution to a approximate, restricted version of the CARP of a cost less than 1.5 times the optimal is NP-hard. Given that the CARP is NP-hard it becomes necessary to use heuristics or metaheuristics to solve the CARP. However literature indicates several exact solution approaches for the CARP. Hirabayashi and Nishida (1992) proposed a Branch and Bound algorithm and Belenguer and Benavent (2003) proposed a Cutting Plane algorithm. Belenguer et al. (2006) extended the cutting plane algorithm for the mixed CARP. However,

exact method are still limited to small instances of the CARP and can seldom cope with additional characteristics of real world waste collection problems. This again emphasises the importance of using heuristics or metaheuristics to solve larger instances of the CARP.

2.3.1 Heuristics

Heuristics are approximate techniques used to determine good feasible solutions for problems that are difficult or impossible to solve to optimallity. Winston and Venkataramanan (2004) state that heuristic are characterized by using a greedy approach to obtain good solution in efficient time.heuristics make incremental improvements to an exiting solution by neighbourhood changes or local searches. Heuristics only allow movements that will improve the objective function (increase for a maximize, decrease for a minimize). As a result they tend to get trapped in a local optima and fail to find a global. optimum[1]

Figure 2.3 indicates how a problem can have many different local optima's which may or may not be the global optimum. Depending on where the heuristic starts different solutions can be obtained. Starting at a it can only improve until local optimum 1 is reached. Whereas, starting at b will result in improvements up to the global optimum, indicating that heuristics can not guarantee certainty about how close the solutions is to the global optimum.

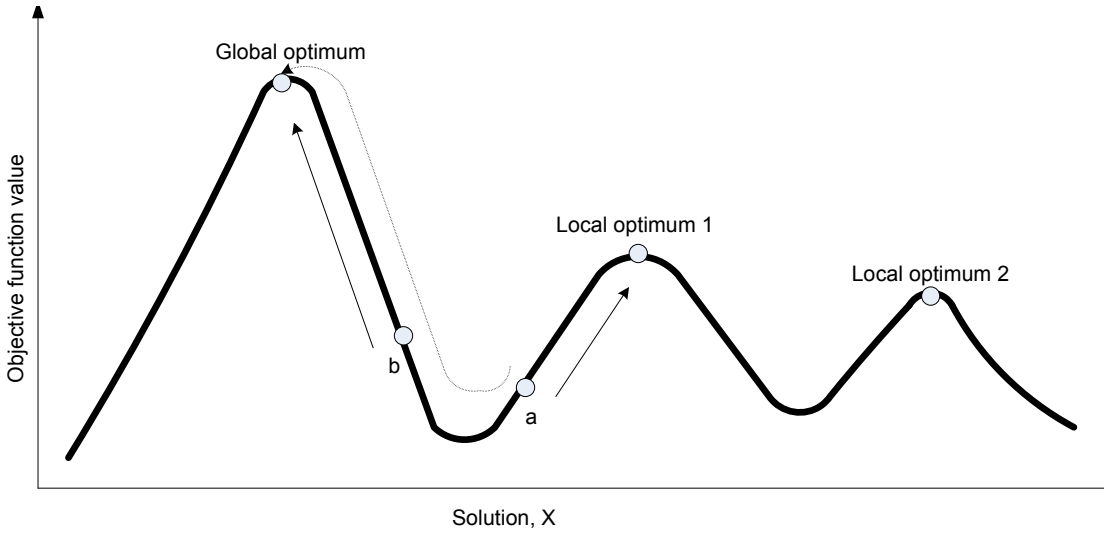

Figure 2.3: Local vs Global optimum

2.3.2 Metaheuristics

Metaheuristics are a subset of heuristics that are based on intelligent search techniques which can over-come the problem of being trapped in a local optimum. This is achieved by

[1]A global optimum is a feasible solution such that no other solution has a superior objective function value.

accepting solutions that may not be an improvement or by considering several solutions at a time (Winston and Venkataramanan, 2004).

Metaheuristics search through the entire solution space, not simply excepting the first local optimum. They keep on searching for an improved solution, increasing the probability of finding a global optimum. Referring back to Figure 2.3 starting at point a it can reach the global optimum by excepting worse solutions.

The four metaheuristics that will be investigated further are Simulated Annealing (SA), Genetic Algorithms (GAs), Tabu Search (TS) and Variable Neighborhood Search (VNS).

Simulated Annealing (SA)

First introduces as a technique to solve complex non-linear optimisations problems, SA emulates the physical process of aggregating particles in a system as it is cooled. By slowly lowering temperature, the energy exchange allows true equilibrium in each stage until the global minimum energy level is reached (Winston and Venkataramanan, 2004).

SA randomly generates feasible moves and calculates the net objective function improvement that these changes will bring about. Since the WCP is a minimization problem a decrease in the objective function will result in the acceptance of the move. An increase in the objective function will result in the acceptance of the move according to a certain probability function, whice may result in a worse solution. This ensures that the algorithm does not get stuck in local optima but searches through the entire solution space in the pursuit of a better solution.

Genetic Algorithms (GAs)

GAs utilizes ideas from biology such as a population of chromosomes, natural selection for mating, offspring production using crossover and mutation for diversity. GAs begin by randomly generating an initial population of strings of chromosomes. These strings represent possible solutions to the given problem. Each solution is evaluated by measurable criteria that result in a fitness (usually the objective function value) being associated with each string. The selection of parents is probabilistically chosen from the current solution by the principle of survival of the fittest: the most fit has the greatest chance of being chosen. Reproduction of parents occurs such that the offspring consist out of a recognisable portion of both parents. The offspring undergoes mutation to randomly alter its genetic makeup to avoid being trapped in a local optima. The offspring becomes part of the new generation of solutions (Winston and Venkataramanan, 2004).

Lacomme et al. (2006) use a Memetic Algorithm (MA) to solve the CARP. This Memetic Algorithm is a genetic algorithm hybridised with a local search. Moreover, the MA ddresses several extensions of the CARP like mixed networks, parallel arcs and turn penalties. It provide excellent performance results on three sets of bench mark problems. Belenguer et al. (2006) further extended this MA to also address problems with several dumping sites. Prins et al. (2003) use a GA with population management to solve the CARP. Result

obtained by the authors suggests that the GA is highly successful for solving the CARP and its extensions.

Tabu Search (TS)

The TS emulates heuristic rules people use in day-to-day decision making by making use of short term and long term memory. The short term memory prevents cycling around a local neighbourhood in the solutions space. Long term memory allows searches to be conducted in the most promising neighbourhoods. It moves away from a local optima by temporarily classifying some moves tabu or forbidden (Winston and Venkataramanan, 2004). The TS algorithm keeps track of the best solution found far and when the search stops it is reported as an approximate optima for the problem.

Hertz et al. (2000) use a TS algorithm called CARPET to solve the CARP; Greistorfer (2003) uses a Tabu Scatter Search metaheuristic; and Brandao and Eglese (2008) a deterministic TS algorithm. Ghiani et al. (2001) use the TS algorithm to solve the CARPIF. All of the authors were able to provide high quality solution for benchmark problems.

Variable Neighborhood Search (VNS)

Hansen and Mladenovic (2001) describes the VNS as a metaheuristic that proceeds to a systematic change of neighborhoods within a possibly randomised local search algorithm. A local search proceeds from an initial solution by a sequence of local changes.Each local change improves the value of the objective function until a local optima is found. The VNS avoids being trapped in a local optima through the systematic changes of neighborhoods.

VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current incumbent (best feasible solution so far), and jumps from this solution to a new one, if and only if, an improvement has been made. In this way often favourable characteristic of the incumbent solution, e.g. the many variables that are already at their optimum value, will be kept and used to obtain promising neighbouring solutions.

Hertz et al. (2001) describe an adaptation of a variant of the VNS, called Variable Neighborhood Descent (VND) algorithm for the CARP. The VND algorithm proved highly successful for solving three sets of benchmark problems. On large instances the VND tend to perform better than the TS algorithm CARPET, proposed by Hertz et al. (2000), in terms of computational time and solution quality. Polacek et al. (2007) use the VNS to solve the CARP and extended it to solve the CARPIF. Excellent results were obtained on four sets of benchmark problems of which two included the extension of IFs. Again the VNS showed slightly better performance results for the CARPIF compared to the results that Ghiani et al. (2001) obtained through the use of a TS algorithm. Another major advantage of VNS is that high quality final solutions can be achieved independently from the quality of the initial solution. As a result of the highly successful results obtained on benchmark problems for the CARP, a VNS solution strategy will be used to solve the WCP.

2.4 Conclusion

The WCP was identified as being an ARP but due to additional characteristics such as capacity constraints and multiple vehicles, rather be classified as a CARP. Various variant on the CARP were investigated to be more able to model a real world representation of the WCP that adhere to traffic rules and allows dumping at multiple IFs. Multiple solution strategies were identified, where the TS proved to be the most appropriate. The TS algorithm in conjunction with various heuristic algorithms will be used in the following chapter to solve the WCP.

Chapter 3

Model formulation

In the previous chapter we have taken an in depth look at the Waste Collection Problem (WCP) and the various solution approaches that are available to solve the problem at hand. Drawing on the knowledge gained, this chapter focuses on the development of the Tabu Search (TS) algorithm, which will be designed to solve the WCP with the following extensions:

A) Mixed multigraph that presents edges, arcs and/or parallel edges and arcs.

B) Turn penalties and forbidden turns.

C) Several dumping sites(Intermediate facilities).

D) Windy edges, whose service cost, $\boldsymbol{c}$, depends on the direction of travel.

To give a quick recap the WCP consists of a depot (which may or may not be a dumping site); several dumping sites; and a fix fleet of k homogeneous vehicles that must collect waste along the required streets of a defined network. The aim of the WCP is to determine a set of vehicle routes of minimum total cost, such that each route starts at the depot, all the required streets are serviced and the routes never exceed the vehicles' capacity.

For the sake of clarity the following section present the WCP and the extended modelling notations and techniques used to solve the problem. The high level structure of the TS algorithm is presented in Section 3.3, with the lower level imbedded algorithms presented in Section 3.4 and 3.5.

3.1 Graph Transformation and Notations

All graph transformations and notations within this chapter were adapted from the work of Belenguer et al. (2006) and Lacomme et al. (2006).

To ease algorithmic design, the mixed multigraph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A}, \boldsymbol{R})$ is transformed into a fully directed multigraph $\boldsymbol{G^*} = (\boldsymbol{V}, \boldsymbol{A^*}, \boldsymbol{R^*})$, by replacing each edge by two opposite arcs and by adding one dummy loop for the depot and one for each of the dumping sites. We define $\boldsymbol{V}$ as the set of n nodes which includes the depot node V_0, as well as all the

dumping nodes. The transformed $\boldsymbol{A^*}$ is defined as a set of m arcs, identified by indexes from 1 to m, instead of pairs of nodes to avoid ambiguities for parallel arcs. Each arc $u \in \boldsymbol{A^*}$ begins at node $\boldsymbol{b}(u)$, ends at node $\boldsymbol{e}(u)$ and has a service cost $\boldsymbol{c}(u)$. The $\boldsymbol{R}$ required tasks in $\boldsymbol{G}$, comprising of $\|\boldsymbol{E}\|$ required edges and $\|\boldsymbol{A}\|$ required arcs, given by $\boldsymbol{R^*} \subseteq \boldsymbol{A^*}$ with $\boldsymbol{R^*} = 2\|\boldsymbol{E}\| + \|\boldsymbol{A}\|$ tasks. Only one of the two arcs that represent an edge has to be collected in any feasible solution. To ensure this, both arcs u and v are linked by a pointer variable, $\boldsymbol{inv}(u)$ and $\boldsymbol{inv}(v)$, which ensures that when the algorithm selects one direction, both arcs can be marked "collected". This is done by coding an arc task in $\boldsymbol{G}$ as one arc u with $\boldsymbol{inv}(u) = 0$ in $\boldsymbol{G^*}$, while each edge task in $\boldsymbol{G}$ gives two opposite arcs, u and v, such that $\boldsymbol{inv}(u) = v$, $\boldsymbol{inv}(v) = u$, $\boldsymbol{q}(u) = \boldsymbol{q}(v)$ and $\boldsymbol{c}(u) = \boldsymbol{c}(v)$, possibly with distinct cost if the edge is windy ($\boldsymbol{c}(u) \neq \boldsymbol{c}(v)$). Each arc $u \in \boldsymbol{R^*}$ has a demand $\boldsymbol{q}(u)$ and a pointer $\boldsymbol{inv}(v)$. From now on all algorithms will work according to the directed encoding of the mixed multigraph.

3.1.1 Forbidden Turns, Turn Penalties and Distance Matrix

Forbidden turns and turn penalties are made transparent by including a set of permitted turns $\boldsymbol{turn}(u, v)$, with associated turn penalties, $\boldsymbol{pen}(u, v)$, into $\boldsymbol{G^*}$. A set of allowed successor arcs for arc u, $\boldsymbol{suc}(u)$, is created, such that arc $v \in \boldsymbol{suc}(u)$ if $\boldsymbol{e}(u) = \boldsymbol{b}(v)$ and $\boldsymbol{turn}(u, v)$ is allowed. Following the work of Lacomme et al. (2006) for the Capacitated Arc Routing Problem (CARP) with turn penalties, a feasible path from arc u to arc v is defined as a sequence of arcs $\mu = (\mu = u_1, u_2 \dots u_k = v)$, such that $u_{i+1} \in \boldsymbol{suc}(u_i)$ for $i = 1, \dots, k-1$. The cost of μ is defined by equation (3.1). Note that the cost of u and v is not included.

$$\boldsymbol{c}(\mu) = \boldsymbol{pen}(u_1, u_2) + \sum_{i=2}^{k-1} \big(\boldsymbol{c}(u_i) + \boldsymbol{pen}(u_i, u_{i+1})\big) \tag{3.1}$$

Using an adaptation of Dijksta's shortest path algorithm, forbidden turns and turn penalties are included by pre-computing two m-by-m arc-to-arc matrices $\boldsymbol{D}$ and $\boldsymbol{P}$. $\boldsymbol{D}(u, v)$ is the cost/distance of the shortest path found from arc u to arc v and $\boldsymbol{P}(u, v)$ is the predecessor of v on this shortest path. The full structure of the adapted Dijkstra's shortest path algorithm is presented in Algorithm 1.

The adapted Dijkstra's algorithm computes arc u of $\boldsymbol{D}$ and $\boldsymbol{P}$. It is called m times with u = 1,2,...,m to calculate the arc-to-arc matrix $\boldsymbol{D}$ and predecessor matrix $\boldsymbol{P}$. The algorithm starts of by setting all arcs' fixed value to false and distance value to infinite. After each iteration the algorithm determines the next arc v that is closest to arc u. Once a shortest path between arcs u and v is obtained, arc v is fixed by setting $\boldsymbol{Fix}(v)$ = true. This ensures that arc v cannot be selected again and determines each succesessor-arc z of arc v to see if the current shortest distance from arc u to arc z can be improved. The output of the algorithm is matrices $\boldsymbol{D}$ and $\boldsymbol{P}$, which includes turn penalties and forbidden turns. Matrices $\boldsymbol{D}$ and $\boldsymbol{P}$ are used as input for the TS algorithm to solve the WCP.

Algorithm 1: Adapted Dijkstra's Shortest Path algorithm

Input : Begin nodes $\boldsymbol{b}(u)$
End nodes $\boldsymbol{e}(u)$
Distance/Service Cost $\boldsymbol{c}(u)$
$\boldsymbol{turn}(u, v)$ and $\boldsymbol{pen}(u, v)$

Output : Shortest Path Matrix $\boldsymbol{D}(u, v)$
Predecessor Matrix $\boldsymbol{P}(u, v)$

Generate $\boldsymbol{suc}(u)$, *the set of allowed successor-arcs for arc; u*

for $i \leftarrow 1$ **to** m **do**
  $\boldsymbol{D}(u, v) = \infty$;
  $\boldsymbol{Fix}(v)$ = False;
end

Find all successor arcs of arc u;

for $i \leftarrow 1$ **to** m **do**
  Set for each v in $\boldsymbol{suc}(u)$:
  $\boldsymbol{D}(u, v) = \boldsymbol{pen}(u, v)$ and
  $\boldsymbol{P}(u, v) = u$;
end

Update matrices $\boldsymbol{D}$ *and* $\boldsymbol{P}$ *with shortest path*;

for $i \leftarrow 1$ **to** m **and** $\boldsymbol{Fix}(v)$ = False **do**
  Chose min v and set $\boldsymbol{Fix}(v)$ = True;
  for *each* z *in* $\boldsymbol{suc}(v)$ *with* $\boldsymbol{D}(u, v) + \boldsymbol{c}(v) + \boldsymbol{pen}(v, z) \leq \boldsymbol{D}(u, z)$ **do**
    $\boldsymbol{D}(u, z) \leftarrow \boldsymbol{D}(u, v) + \boldsymbol{c}(v) + \boldsymbol{pen}(v, z)$;
    $\boldsymbol{P}(u, z) \leftarrow v$;
  end
end

3.1.2 Multiple Dumping Sites

In order to effectively handle multiple dumping sites a set $\boldsymbol{I}$ of all allowable dumping nodes is created. The distances between arc $u \in \boldsymbol{R^*}$ and all other dumping nodes from set $\boldsymbol{I}$, is tackled by including in $\boldsymbol{A^*}$ one fictitious loop $\sigma = (s, s)$, where s is an element of $\boldsymbol{I}$, for each dumping node. This method is also followed to include the depot, V_0 in $\boldsymbol{A^*}$. Thus $\boldsymbol{R^*} = 2\|\boldsymbol{E}\| + \|\boldsymbol{A}\| + \|\boldsymbol{I}\| + \|\boldsymbol{V_0}\|$. After each arc u has been serviced the shortest distance to the closest dumping loop can be pre-computed for every arc-task $\boldsymbol{A^*} \in \boldsymbol{R^*}$ by using equation (3.2).

$$\boldsymbol{dump}(u) = \min_{s \in \boldsymbol{I}} \{\boldsymbol{D}(u, s) \ \forall \ u\} \tag{3.2}$$

3.2 WCP Algorithm Terminology

The basic terminology introduced in Section 3.1 and summarized in Table 3.1, will be used by the WCP algorithm, with the addition of certain algorithm specific terms. The input to the WCP include: a fully directed mixed multigraph $\boldsymbol{G^*} = (V, \boldsymbol{A^*}, \boldsymbol{R^*})$; a distance matrix $\boldsymbol{D}$; a set of dumping nodes $\boldsymbol{I}$ which includes the depot node V_0; the number of waste collection vehicles k; and the capacity of the waste collection vehicles W.

A feasible solution consists out of a set $\boldsymbol{T}$ of k routes, with $\boldsymbol{T} = \{\boldsymbol{R_1}, \ldots, \boldsymbol{R_k}\}$ such that each route $\boldsymbol{R_i}$ with $i = 1,\ldots,k$ consist of a list of sub routes, $\boldsymbol{R_i} = \{\boldsymbol{SR_1}, \ldots, \boldsymbol{SR_x}\}$ and each sub route consist out of a sequence of required arc-task $\boldsymbol{A^*} \in \boldsymbol{R^*}$. The purpose of the sub routes is to indicate the capacity feasible trips that exist within route $\boldsymbol{R_i}$ such that each sub routes is either a trip between the depot and the first IF; between two successive IFs and\or the last IF and the depot. Meaning only streets that require service are included within a feasible solution $\boldsymbol{T}$. Implicitly, $\boldsymbol{R_i}$ starts and ends at the depot, allows dumping at IFs and shortest feasible paths are assumed between two arc-tasks and between one arc-task and the closest dumping loop s, where $s \in \boldsymbol{I}$. Each required arc-task appears once in $\boldsymbol{T}$ and each required edge-task occurs as one of its two opposite arcs in $\boldsymbol{T}$. The objective function value of the entire solution $\boldsymbol{T}$ or route $\boldsymbol{R_i}$ is calculated through $Obj(X)$.

3.3 Tabu Search Algorithm

In order to solve the WCP, two TS algorithms were designed. Algorithm 2 indicates the generic high level representation of the TS algorithm used to create the TS algorithms for the WCP. Algorithm 2 consists out seven basic steps, starting at Step 0 where the initial feasible solution is generated and the stopping criteria are set. At each iteration the best possible non-tabu feasible move from a set $\boldsymbol{M}$ of allowable neighbourhood moves is chosen (Step 2), and applied to the current solution (Step 3). To prevent the algorithm form cycling back to recently visited solutions, Step 5 records the collection of moves made in Step 3 within a tabu list. This forbids moves from being made for a certain number of iterations, after which it is removed from the list and made available for selection. Since a move may either improve or degrade the objective function value, Step 4 will keep track of the best feasible solution (incumbent solution) found so far. The algorithm stops when either no more feasible moves are possible or when the user-specified stopping criteria such as iteration limit, t_{max}, is reached (Step 1). Upon termination the incumbent solution $\hat{\boldsymbol{T}}$ from Step 4 will be reported as the approximate optimum.

Table 3.1: Glossary of mathematical symbols.

Term	Description
$\boldsymbol{G}$	Connected mix graph with $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{A}, \boldsymbol{R})$.
$\boldsymbol{G^*}$	Transformed fully directed graph with $\boldsymbol{G^*} = (\boldsymbol{V}, \boldsymbol{A^*}, \boldsymbol{R^*})$.
$\boldsymbol{V}$	Vertex set representing a streets intersection or dead ends
$\boldsymbol{A^*}$	Set of m arcs.
$\boldsymbol{R^*}$	Set of required arcs.
n	number of nodes in $\boldsymbol{V}$, calculated as $n = \|\boldsymbol{V}\|$
m	number of arcs in $\boldsymbol{A^*}$, calculated as $m = \boldsymbol{A^*}$
$\boldsymbol{b}(u)$	begin node of arc u.
$\boldsymbol{e}(u)$	end node of arc u.
$\boldsymbol{q}(u)$	demand of arc u.
$\boldsymbol{c}(u)$	service cost of arc u.
$\boldsymbol{inv}(u)$	pointer to indicate whether arc u is an edge or arc task in $\boldsymbol{G}$.
$\boldsymbol{suc}(u)$	set of possible successor arcs for arc u.
V_0	depot node.
$\boldsymbol{I}$	set of allowable dumping nodes.
k	number of vehicles.
W	vehicle's capacity.
$\boldsymbol{D}$	m-by-m arc-to-arc distance matrix.
$\boldsymbol{P}$	m-by-m arc-to-arc predecessor matrix.
$\boldsymbol{pen}(u, v)$	penalty for $\boldsymbol{turn}(u, v)$.
$\boldsymbol{T}$	a given solution of the WCP.
$\hat{\boldsymbol{T}}$	incumbent solution.
$\tilde{\boldsymbol{T}}$	solution returned by lower level algorithms.

Algorithm 2: Generic Tabu Search algorithm

Step 0: Initialisation. Choose a high quality starting feasible solution $\boldsymbol{T}^{(0)}$ and an iteration limit t_{max}. Set incumbent solution $\hat{\boldsymbol{T}} \leftarrow \boldsymbol{T}^{(0)}$ and solution index $t \leftarrow 0$. No moves are tabu.

Step 1: Stopping. If no non-tabu move $\Delta \boldsymbol{T}$ in move set M leads to a feasible neighbor of current solution $\boldsymbol{T}^{(t)}$, or if $t = t_{max}$, then stop. Incumbent solution $\hat{\boldsymbol{T}}$ is an approximate optimum.

Step 2: Move. Choose some none-tabu feasible move $\Delta \boldsymbol{T} \in M$ as $\Delta \boldsymbol{T}^{(t=1)}$.

Step 3: Step. Update $\boldsymbol{T}^{t+1} \leftarrow \boldsymbol{T}^{(t)} + \Delta \boldsymbol{T}^{t+1}$.

Step 4: Incumbent Solution. If the objective function value of $\boldsymbol{T}^{t+1}$ is superior to that of incumbent solution $\hat{\boldsymbol{T}}$, replace $\hat{\boldsymbol{T}} \leftarrow \boldsymbol{T}^{t+1}$.

Step 5: Tabu List. Remove from the list of tabu or forbidden moves any move that has been on the list for a suffiecient number of iterations, and add a collection of moves that includes any returning immediately from $\boldsymbol{T}^{t+1}$ to $\boldsymbol{T}^{t}$.

Step 6: Increment. Increment $t \leftarrow t + 1$, and retun to step 1.

Algorithm 2 is used as the basis to create both WCP TS algorithms which is presented in Algorithm 3 & 4, as well as its specific lower level algorithms. All the steps within Algorithm 3 & 4 are exactly the same except for Step 2 & 4. The differences within these two steps will be made apparent later on in the chapter. The remainder of this chapter will provide a detailed explanation of how the main steps of the general TS algorithm were used, implemented and formulated within the two WCP TS algorithms.

3.4 Initialisation

The first step of the WCP TS algorithm is to generate any feasible solution from where the search can begin. The quality of the initial solution has been known to affect the final solution and creating a high quality initial solution can significantly improve the performance of the TS algorithm. This is achieved through the algorithm Construct-Initial-Solution with an imbedded algorithm Improve-Solution that calls the Reduce-Route-Length algorithm for every route $\boldsymbol{R_a}$ in solution $\boldsymbol{T}$.

3.4.1 CONSTRUCT-INITIAL-SOLUTION

The Construct-Initial-Solution algorithm is based on the path-scanning Heuristic of Golden and Wong (1981). The algorithm is extended to handle turn penalties as well as multiple dumping sites. The algorithm builds k routes for each vehicle, one at a time. In constructing each route, the sequence of arc-tasks is extended by joining the tasks looking most promising. For a route ending at task u, the algorithm determines a set $\boldsymbol{L}$ of tasks

Algorithm 3: Complete TS Algorithm 1

Input : Mix Multigraph $\boldsymbol{G^*} = (\boldsymbol{V}, \boldsymbol{A^*}, \boldsymbol{R^*})$
Distance Matrix $\boldsymbol{D}$
Number of vehicles k
Capacity of vehicles W
Iteration limit t_{max}

Output : Incumbent solution $\hat{\boldsymbol{T}} = \{\boldsymbol{R_1}, \ldots, \boldsymbol{R_k}\}$

CONSTRUCT-INITIAL-SOLUTION(k;$\boldsymbol{T}^{(0)}$, $Obj(\tilde{\boldsymbol{T}}^{(0)})$);
$\hat{\boldsymbol{T}} \leftarrow \tilde{\boldsymbol{T}}^{(0)}$;
$Obj_{incumbent} \leftarrow Obj(\boldsymbol{T}^{(0)})$;
$t \leftarrow 0$;
$SelectionNumber \leftarrow 1$;
$NonImprovementCounter \leftarrow 0$;
while $t \leq t_{max}$ **do**
  Call one of the two move algorithms according to $SelectionNumber$:
  (1) ARC-EXCHANGE($\boldsymbol{T}^{(t)}$;$\tilde{\boldsymbol{T}}^{(t+1)}$, $Obj_{current}$);
  (2) RANDOM-TWO-ARC-EXCHANGE($\boldsymbol{T^{(t)}}$;$\tilde{\boldsymbol{T}}^{\boldsymbol{(t+1)}}$, $Obj_{current}$);
  Check if solution objective value is superior;
  if $Obj_{current} < Obj_{incumbent}$ **then**
    $Obj_{incumbent} \leftarrow Obj_{current}$;
    $\hat{\boldsymbol{T}} \leftarrow \tilde{\boldsymbol{T}}^{(t+1)}$;
    $NonImprovementCounter \leftarrow 0$;
    $SelectionNumber \leftarrow 1$;
  else
    $NonImprovementCounter \leftarrow NonImprovementCounter + 1$;
  end
  if $NonImprovementCounter \geq limit$ **then**
    $SelectionNumber \leftarrow 2$;
    $NonImprovementCounter \leftarrow 0$;
  else
    $SelectionNumber \leftarrow 1$;
  end
  Update Tabu List;
  $t \leftarrow t + 1$;
end

Algorithm 4: Complete TS Algorithm 2

Input : Mix Multigraph $\boldsymbol{G^*} = (\boldsymbol{V}, \boldsymbol{A^*}, \boldsymbol{R^*})$

Distance Matrix $\boldsymbol{D}$

Number of vehicles k

Capacity of vehicles W

Iteration limit t_{max}

Output : Incumbent solution $\hat{\boldsymbol{T}} = \{\boldsymbol{R_1}, \ldots, \boldsymbol{R_k}\}$

CONSTRUCT-INITIAL-SOLUTION(k;$\boldsymbol{T^{(0)}}$, $Obj(\tilde{\boldsymbol{T}}^{(0)})$);

$\hat{\boldsymbol{T}} \leftarrow \tilde{\boldsymbol{T}}^{(0)}$;

$Obj_{incumbent} \leftarrow Obj(\boldsymbol{T^{(0)}})$;

$t \leftarrow 0$;

Convert sub routes of $\boldsymbol{T}$ *into one long route boldsymbol*$\tilde{T}$;

while $t \leq t_{max}$ **do**

Call IMPROVE-SOLUTION and return best Objective value found:

IMPROVE-SOLUTION($\boldsymbol{T^{(t)}}$;$\tilde{\boldsymbol{T}}^{(t+1)}$, $Obj_{current}$);

if $Obj_{current} < Obj_{incumbent}$ **then**

$Obj_{incumbent} \leftarrow Obj_{current}$;

$\hat{\boldsymbol{T}} \leftarrow \tilde{\boldsymbol{T}}^{(t+1)}$;

end

Update Tabu List;

$t \leftarrow t + 1$;

end

closest to u, not yet collected and feasible for vehicle capacity W. It uses one of the following five rules to select the next task v in $\boldsymbol{L}$:

1. Maximize the distance $\boldsymbol{D}(v, \boldsymbol{dump}(v))$ to the closest dumping node.
2. Minimize the distance $\boldsymbol{D}(v, \boldsymbol{dump}(v))$ to the closest dumping node.
3. Maximize the yield $\boldsymbol{q}(u)/\boldsymbol{c}(u)$.
4. Minimize the yield $\boldsymbol{q}(u)/\boldsymbol{c}(u)$.
5. Use rule 1 if the vehicle is less than half-full, else use rule 2.

By selecting task v it is flagged as collected and if v belongs to an edge-task, $\boldsymbol{inv}(v)$ is also flagged to ensure it is not reselected in subsequent iterations. Once vehicle capacity W is reached, it returns to the closest dumping node and then continues to construct the route until the maximum number of allowable dumps, θ, by a vehicle is reached. The maximum number of allowable dumps θ is calculated by dividing the total demand of all required arc-tasks in $\boldsymbol{T}$, by vehicle capacity W, rounded to the nearest integer. Once this number is reached, the algorithm will proceed to construct the next route until all k routes have been constructed. The algorithm builds one solution for each selection rule and returns the most promising one.

3.5 Move & Step

The critical element of a TS algorithm is its move set $\boldsymbol{M}$. If it were possible, we should consider constructing every possible neighbourhood solution for the current solution $\boldsymbol{T}$. Then the search would yield promising solutions with a high possibility of finding the global optima because all possible neighbourhoods would have been visited and the search will stop with no solution superior and feasible in objective value to the current solution. However, in practice this is not possible due to the complexity and large number of neigbourhood solutions that exist. If the move set is too large it may take a very long time to find good solutions and on the other hand, if it is too restrictive, very few solutions will be considered at each iteration, resulting in poor quality solutions. This necessitates the importance of considerating a move set that is compact enough to check at each iteration for promising and feasible neighbourhoods. The WCP TS algorithm 1 is confined to a move set where only two routes are modified through one of two algorithms: Arc-Exchange and Random-Two-Arc-Exchange. The first algorithm generates neighbourhood solutions that attempt to reduce the total distance of solution $\boldsymbol{T}$, while the second algorithm diversifies the search space. The WCP TS algorithm 2 only use the Reduce-Route-Length algorithm to make a move to the best objective function value found for each iteration t.

3.5.1 ARC-EXCHANGE

The objective of the WCP is to design waste collection routes that service all required arcs in the shortest possible distance, while taking turn penalties and IFs into consideration. This is achieved through the algorithm ARC-EXCHANGE. The algorithm attempts to reduce the distance of solution $\boldsymbol{T}$, by finding the critical arc u, for each route $\boldsymbol{R_a}$ that is the furthest away from its preceding and succeeding required arcs. For the first and last arc within a sub route, only 1.5 times the distance to the preceding or succeeding arc will be considered, and not the distance to or from the nearest dumping node. This arc is then removed from its current route $\boldsymbol{R_a}$ by calling the REMOVE-ARC algorithm. Once the arc u has been removed it is added to any of the other vehicle routes $\boldsymbol{R_s}$ that will result in the least increase of route length. This is achieved by calling the ADD-ARC algorithm. The new sequence of arcs within solution $\boldsymbol{T}$ is recorded as $\boldsymbol{T_{(l)}^{temp}}$, where $l \in \{1, \ldots, k\}$. The arc exchange from all routes that leads to the best decrease, in total solution distance, is made permanent by the ARC-EXCHANGE algorithm. The complete structure of this algorithm is presented in Algorithm 5.

3.5.2 RANDOM-TWO-ARC-EXCHANGE

As stated before the purpose of this algorithm is to diversify the search space by forcing the search into previously unexplored areas that differ from those visited before. This is accomplished through the RANDOM-TWO-ARC-EXCHANGE algorithm, which is called after β number of non consecutive improvement moves have been made. The algorithm takes any random arc u from any of the vehicle routes $\boldsymbol{R_a}$ and exchange this arc u with a random arc v from any other vehicle route $\boldsymbol{R_s}$. The complete structure of the RANDOM-TWO-ARC-EXCHNGE algorithm is presented in Algorithm 6.

3.5.3 Adding & Removing Arcs

Two simple algorithms, ADD-ARC and REMOVE-ARC are used by the two previous mentioned algorithms. Algorithm REMOVE-ARC removes arc u from route $\boldsymbol{R_i}$, and then calls the algorithm, REDUCE-ROUTE-LENGTH to ensure that the sequence in which the arcs are serviced without arc u minimizes the distance travelled. Algorithm ADD-ARC works in the opposite way by adding arc u to route $\boldsymbol{R_j}$ and also calls algorithm REDUCE-ROUTE-LENGTH for the same purpose.

Algorithm 5: ARC-EXCHANGE

Input : Solution $\boldsymbol{T}^{(t)}$

Output : Neigborhood Solution $\tilde{\boldsymbol{T}}^{(t+1)}$

$k \leftarrow$ number of Routes $\boldsymbol{R}$

for $l \leftarrow 1$ to k **do**

  $\rho \leftarrow$ number of task in $\boldsymbol{R}_l$

  for $t \leftarrow 2$ to $k-1$ **do**

    if arc $t \in \boldsymbol{I}$

      $L(t) = 0$

    else if arc $t-1 \in \boldsymbol{I}$

      $L(t) = 1.5 \times \boldsymbol{D}(t, t+1)$

    else if arc $t+1 \in \boldsymbol{I}$

      $L(t) = 1.5 \times \boldsymbol{D}(t-1, t)$

    else

      $L(t) = \boldsymbol{D}(t-1, t) + \boldsymbol{D}(t, t+1)$

    end

  end

  Find arc t^{temp} where arc $t \leftarrow \max(L(t))$

  Set arc $t \leftarrow$ arc t^{temp}

  REMOVE-ARC($\boldsymbol{R}_a$,t; $\tilde{\boldsymbol{R}}_a$)

  Choose $\boldsymbol{R}_s$ such that adding arc t will result in minimal tour length increase

  ADD-ARC($\boldsymbol{R}_s$,t; $\tilde{\boldsymbol{R}}_s$)

  Update $\tilde{\boldsymbol{T}}^{\boldsymbol{temp}}(l)$ with $\tilde{\boldsymbol{R}}_a$ and $\tilde{\boldsymbol{R}}_s$

end

Find min $\tilde{\boldsymbol{T}}^{temp}(l)$

$\tilde{\boldsymbol{T}}^{(t+1)} \leftarrow \tilde{\boldsymbol{T}}^{temp}(l)$

Algorithm 6: RANDOM-TWO-ARC-EXCHANGE

Input : Current Solution $\boldsymbol{T}^{(t)}$

Output : Neigborhood Solution $\tilde{\boldsymbol{T}}^{(t+1)}$

Randomly choose two routes $\boldsymbol{R}_i and \boldsymbol{R}_j$ such that $i \neq j$

Randomly select an arc u from $\boldsymbol{R}_i$ and

Randomly select an arc v from $\boldsymbol{R}_j$ by calling:

REMOVE-ARC($\boldsymbol{R}_i$,arc u; $\tilde{\boldsymbol{R}}_i^{(1)}$)

REMOVE-ARC($\boldsymbol{R}_j$,arc u; $\tilde{\boldsymbol{R}}_j^{(1)}$)

then

Add arc u to $\boldsymbol{R}_j$ and

Add arc v to $\boldsymbol{R}_i$:

ADD-ARC($\tilde{\boldsymbol{R}}_i^{(1)}$,$u$; $\tilde{\boldsymbol{R}}_i^{(2)}$)

ADD-ARC($\tilde{\boldsymbol{R}}_j^{(1)}$,$u$; $\tilde{\boldsymbol{R}}_j^{(2)}$)

Update $\tilde{\boldsymbol{T}}^{(t+1)}$

3.5.4 REDUCE ROUTE LENGTH

Before any of the moves are made the REDUCE-ROUTE-LENGTH algorithm first creates a giant infeasible route $\boldsymbol{G_a}$ by removing all the intermediate dumping nodes from route $\boldsymbol{R_a}$.It is a local search improvement algorithm that performs successive phases on all arc-task within the giant infeasible route $\boldsymbol{G_a}$. Its main purpose is to reduce the length of a vehicle route $\boldsymbol{R_a}$ by trying the following moves from each of the three phases.

P1: Invert arc-task u, if it is an edge task i.e. replace u by $inv(u)$.

P2: Move arc-task u after arc-task v, or before v if v is the first arc-task of the route.

P3: Perform a pairwise exchange between task u and v.

Once a move is made the algorithm calls the SPLIT algorithm to create a feasible route $\tilde{\boldsymbol{R}}_{\boldsymbol{a}}$, and computes the new objective function value of the proposed move. If the new objective function is better than that of the best one found so far, the algorithm moves to this solution. Each phase ends by performing the first improving move detected.When no more moves can be made the algorithm proceeds to the next phase. The algorithm stops the search process when all phases report no improvements. However, when moving an edge-task in P2 and P3, its service direction may be inverted or not. Meaning a move in P3 consist of four swapping cases: u and v may be replaced by v and u, $inv(v)$ and u, v and $inv(u)$, $inv(v)$ and $inv(u)$. The structure of the REDUCE-ROUTE-LENGTH algorithm is presented in Algorithms 7.

Algorithm 7: REDUCE-ROUTE-LENGTH

Input : Route $\boldsymbol{R}_a$

Output : Improved Route $\hat{\boldsymbol{R}}_a$

$\hat{\boldsymbol{R}}_a \leftarrow \boldsymbol{R}_a$

$Obj_{best} \leftarrow Obj(\boldsymbol{R}_a)$

Create giant infeasible route $\boldsymbol{G}_a$ *by removing intermediate dumping nodes*

while *total length of* $\boldsymbol{R}_a$ *can be reduced* **do**

Perform **phase** move

Create feasible route

SPLIT($\boldsymbol{G}_a$;$\tilde{\boldsymbol{R}}_a^{temp}$)

if $Obj(\tilde{\boldsymbol{R}}_a^{temp}) < Obj_{best}$ **then**

$\hat{\boldsymbol{R}}_a \leftarrow \tilde{\boldsymbol{R}}_a^{temp}$

$Obj_{best} \leftarrow Obj(\tilde{\boldsymbol{R}}_a^{temp})$

end

end

3.5.5 SPLIT

The main purpose of the SPLIT algorithm is to receive a giant infeasible route $\boldsymbol{G_a}$ as input, and perform the split procedure that will produce capacity feasible sub routes. The sub routes are then merged together to form a feasible route $\tilde{\boldsymbol{R}}_{\boldsymbol{a}}$. The SPLIT procedure determines the shortest path to service all tasks in the sequence $(S_i, \ldots, S_j)$ received, while taking capacity constraints into account by enforcing dumping at the nearest dumping node when vehicle capacity is reached. The complete structure of the SPLIT procedure is presented in Algorithm 8.

3.6 Incumbent Solution

During each iteration the objective value, resulting from the move just made, is calculated and compared to that of the incumbent solution. If the current solution improves on the incumbent solution it becomes the new incumbent solution until a better solution is found or no more feasible moves are possible. Once the stopping criteria is met, the incumbent solution is reported as the final solution.

3.7 Tabu List

The WCP TS algorithm 1 Tabu list consists of multiple Tabu lists, one for each vehicle route. Each list records all the arcs recently added to the vehicle routes and forbid these

Algorithm 8: SPLIT

Input : Giant infeasible route $\boldsymbol{G}_a$

Output : Improved Route $\tilde{\boldsymbol{R}}_a^{temp}$

$\tau \leftarrow$ number of tasks in $\boldsymbol{G}_a$

Set $\boldsymbol{V}(1), \boldsymbol{N}(1) = 0$

for $i \leftarrow 2$ to τ **do**

  $\boldsymbol{V}(i) \leftarrow \infty$

end

for $i \leftarrow 2$ to τ **do**

  Reset load and cost variables

  load, cost $\leftarrow 0$

  for $j \leftarrow i$ to τ **do**

    Check if vehicle has sufficient capacity load $> W$

    if load $> W$

      break

    end

    Update load of vehicle

    load $\leftarrow$ load + $\boldsymbol{q}(\boldsymbol{S}(j))$

    if $i = j$ **then**

      cost $\leftarrow$ cost - $\boldsymbol{D}(\boldsymbol{dump}(\boldsymbol{S}(i-1)), \boldsymbol{S}(i)) + \boldsymbol{c}(i) + \boldsymbol{D}(\boldsymbol{S}(i), \boldsymbol{dump}(\boldsymbol{S}(i)))$

    else

      cost $\leftarrow$ cost - $\boldsymbol{D}(\boldsymbol{S}(j-1), \boldsymbol{dump}(\boldsymbol{S}(j-1)))$

    end

    if load $\leq W$ **then**

      $\boldsymbol{V_{new}} = \boldsymbol{V_{(i-1)}}+$ cost

      if $\boldsymbol{V_{new}} < \boldsymbol{V_j}$

        $\boldsymbol{V_j} \leftarrow \boldsymbol{V_{new}}$

        $\boldsymbol{N_j} \leftarrow \boldsymbol{N_{(i-1)}} + 1$

      end

    end

  end

end

$\tilde{\boldsymbol{R}}_a^{temp} \leftarrow V_\tau$

arcs from being removed for x iterations. For the WCP TS algorithm 2 only one Tabu list is kept since there is only one giant route within solution $\boldsymbol{T}$. The Tabu list fix either the arc-task u that is inverted or the first of each two arcs that is interchanged through the pairwise exchange move. Only the move that results in the best objective function value will be recorded with the appropriate TB list. It is important that a robust tabu tenure is selected as the incorrect tenure may negatively influence the search process. If the tenure is too long, the TS may not find good solutions, whereas if it is to short, cycling between previous solutions can occur.

3.8 Conclusion

In this chapter the complete TS algorithms developed to solve the WCP with turn penalties and IFs have been presented. The next chapter will focus on testing the algorithms on several benchmark problems, comparing results with those found in literature, and finally, concluding on their ability to solve real world waste collection problems.

Chapter 4

Computational Evaluation

In this chapter the Tabu Search (TS) Algorithms designed to solve the Waste Collection Problem (WCP) will be analyzed. The performance of the algorithms on the basic Capacitated Arc Routing Problem (CARP) as well as its extensions will be compared to that of the best found results in literature. For the computational evaluation various problems from four different sets of benchmark problems[1] were selected. However, these benchmark problems can only be used to evaluate the TS algorithms effectiveness on the basic CARP and does not include the extensions to evaluate the CARP with forbidden turns, turn penalties and Intermediate Facilities (IFs). For this reason a set of test problems that can be used to evaluate these extensions were created. The TS algorithms were coded in Matlab and run on a 2.66 GHz dual core Pentium-4 processor under Windows XP.

4.1 Test Instances

Several benchmark problems were used, each varying the vehicle capacity W; the number of vehicles available; and the number and percentage of required edges and arcs within the road network. The benchmark problems were divided into four categories, each evaluating a different extension of the CARP. The categories are as follows:

The Basic CARP: In this category all the problems contain only required edges, no arcs and one dumping site. This category consist of 10 of the 34 *val* files designed by Belenguer and Benavent (2003) to evaluate a cutting plane algorithm for the CARP. The problems have 24–50 nodes and 34–97 edges of which all have to be serviced.

The Mixed CARP: The 10 problems within this category were selected from the *mval* files designed by Belenguer et al. (2006). The authors converted the *val* files of Belenguer and Benavent (2003) into 34 *mval* files by keeping each edge with probability 0.4, replacing an edge with two opposite arcs with probability 0.4 or with one arc with probability 0.2. As a result the problems contain 24–50 nodes and 43–138 tasks (edges & arcs) of which all have to be serviced.

[1]The benchmark problems used in this project can be downloaded from http://www.uv.es/belenque/carp.html and http://www.uv.es/belengue/mcarp/index.html.

Required/Non Required CARP: With the above two categories all tasks require service. However, within the real world some streets may not require service and are only used to travel between streets that require service. Therefore to model and evaluate a network with required as well as non required streets, five problems from the following benchmark sets were selected for evaluation. The first four problems were selected from the 24 *egl* files designed by Belenguer and Benavent (2003) and the fifth problem from the *lpr* files designed by Belenguer et al. (2006). The first four problems contain 77–144 nodes, 98–190 edges and no arcs. The fifth problem consists of 366 nodes; 388 required edges; 416 required arcs; and 38 non required arcs.

The CARP with turn penalties and Ifs: As mentioned before no test problems with in the literature provide best solution for problems that contain both turn penalties and IFs. However, Chaini et al created a set of test problems that can be used to evaluate the Capacitated Arc Routing Problem with Intermediate Facilities (CARPIF). These test problems can be used to evaluate the CARP with turn penalties and IFs by setting all turn penalties to zero and thus reducing the CARP with turn penalties and IFs to a CARPIF. However, it was decided to rather create six problems that can be used to evaluate the effect of both turn penalties and Ifs, than to use the benchmark problems for the CARPIF. This was done by selecting six of the *val* files in category 1 and then transforming them into *cval* files by applying the following procedure. To include IFs, two IFs were added at nodes $\lfloor |V|/2 \rfloor$ and $2\lfloor |V|/2 \rfloor$. The demand Q of each edge was doubled to ensure that each vehicle must al least visited two IFs within its designed route. The depot node V_0 was also selected to be a dumping site. All vehicles return to the depot once they have serviced all the required streets within their designed route. To address turn penalties, al U-turns were forbidden by setting the penalty to infinity, meaning no vehicle can return on the same route it just travelled on, which were allowed in all of the above categories. The *cval* problems consist of 24–34 nodes;34–65 edges of which all have to be service and three dumping sites.

4.2 Computational Results

All the benchmark problems were tested on both TS algorithms 1 & 2. TS algorithm 1 was run for 10000 iterations with a Tabu list length of 7 and TS algorithm 2 was run for 1000 iterations with a Tabu list length of 18.

Table 4.2 to 4.5 summarises the results for all four categories. The various column headings are presented in Table 4.1.

Table 4.1: Column headings for results.

Term	Description
File	the instance number as in the benchmark problems.
n	number of nodes in $\boldsymbol{V}$, calculated as $n = \|\boldsymbol{V}\|$
m	number of arcs in $\boldsymbol{A^*}$, calculated as $m = \|\boldsymbol{A^*}\|$
$\boldsymbol{E}$	number of required edges.
$\boldsymbol{A}$	number of required arcs.
W	vehicle' capacity.
Best found	best solution values within the literature.
PS	initial solution value produced with the path-scanning heuristic.
TS Algorithm 1	Results from TS algorithm 1.
TS Algorithm 2	Results from TS algorithm 2.
% Deviation	percentage deviation of the best solution from either TS algorithm 1 & 2 against the best found solution value in the literature.
% Improvement	percentage with which the best solution value from either TS algorithm 1 & 2 have improved from the PS solution value.

The computational results presented in Table 4.2 to 4.5 indicate that a high quality initial solution is obtain through the path scanning heuristic, as the five selection criteria used to calculate the initial solution, are never simultaneously bad. TS algorithm 1 outperforms TS algorithm 2 in most of the test problems but it should be noted that it was run for more iterations. The average deviation from the best found objective function value for category 1 to 3 is 8.8%, with a worst deviation of 15.9%. Good solutions were obtained on most of the problems. However, no best found results were match with either TS algorithms, indicating the need for future improvement and refinement of the algorithms.

The results in Table 4.5 for the six *cval* instances in category 4, indicate an average improvement of 16.8% over the initial solution, with a best improvement of 23.5%. Both algorithms were specifically designed to handle this class of WCPs that include turn penalties and Ifs. With the results obtained it can be seen that the algorithms have the ability to improve on the high quality initial solutions and thus have the potential to improve this type of problems in the real world.

Table 4.2: Computational results for Category 1.

File	n	m	$\boldsymbol{E}$	$\boldsymbol{A}$	W	Number of Vehicles	Best Found	PS	TS Algorithm 1	TS Algorithm 2	% Deviation
val1a	24	79	39	0	200	2	173	186	185	186	6.9
val2a	24	69	34	0	180	2	227	259	244	249	7.5
val3a	24	71	35	0	80	2	81	88	86	87	6.2
val4a	41	139	69	0	225	3	400	451	451	410	2.5
val5a	34	131	65	0	220	3	423	476	468	440	4.0
val6a	31	101	50	0	170	3	223	271	245	243	9.0
val7a	40	133	66	0	200	3	279	326	307	311	10.0
val8a	30	127	63	0	200	3	386	433	418	432	8.3
val9a	50	184	92	0	235	3	323	358	358	344	6.5
val10a	50	195	97	0	250	3	428	453	453	439	2.6
Average deviation											

Table 4.3: Computational results for Category 2.

File	n	m	$\boldsymbol{E}$	$\boldsymbol{A}$	W	Number of Vehicles	Best Found	PS	TS Algorithm 1	TS Algorithm 2	% Deviation
mval1a	24	76	20	35	200	2	230	257	238	245	3.5
mval2a	24	61	16	28	180	2	324	430	345	346	6.5
mval3a	24	64	15	33	80	2	115	143	124	136	7.8
mval4a	41	122	26	69	225	3	580	702	655	630	8.6
mval5a	34	119	22	74	220	3	597	375	346	349	12.4
mval6a	31	92	22	47	170	3	326	707	671	688	6.1
mval7a	40	123	36	50	200	3	364	592	398	381	4.7
mval8a	30	117	20	76	200	3	581	677	637	645	9.6
mval9a	50	165	32	100	235	3	458	523	507	495	8.1
mval10a	50	165	32	106	250	3	634	739	706	706	11.4

Table 4.4: Computational results for Category 3.

File	n	m	$\boldsymbol{E}$	Non Required Edges	$\boldsymbol{A}$	Non Required Arcs	W	# of Vehicles	Best Found	PS	TS 1	TS 2	% Deviation
egle1A	77	197	51	47	0	0	305	5	3515	4115	3943	3889	9.6
egle2A	77	197	72	26	0	0	280	7	4994	6458	5815	6327	15.9
egle3A	77	197	87	11	0	0	280	8	5869	7454	6837	6975	15.9
egls1A	140	381	75	115	0	0	210	7	4992	6382	5799	5919	15.6
lprc05	369	1229	387	0	416	38	10000	23	257890	268012	266763	267232	3.4

Table 4.5: Computational results for Category 4.

File	n	m	$\boldsymbol{E}$	W	Number of Vehicles	PS	TS Algorithm 1	TS Algorithm 2	% Improvement
cval1a	24	81	39	200	2	210	170	187	23.5
cval2a	24	71	34	180	2	273	238	254	14.7
cval3a	24	73	35	80	2	91	79	84	15.2
cval4a	41	141	69	225	3	529	445	439	12.8
cval5a	34	133	65	220	3	529	449	447	17.9
cval6a	31	103	50	170	3	311	267	268	16.5

4.3 Conclusion

In Chapter 4 the WCP TS algorithms created in this project were tested against various benchmark problems. The results obtained were used to verify the quality of WCP TS algorithms solutions against the best found solutions within the literature. It was concluded that even though the WCP TS algorithms improve on the quality of the PS heuristic initial solution, room for further improvement exist.

A set of six additional problems were derived from the *val* files, designed by Belenguer and Benavent (2003) to test the algorithms ability to solve the extended WCP that include turn penalties and IFs. Good results were obtained in terms of the algorithms capability to improve on the initial solution, indicating its readiness to be tested on a real world problem.

The next Chapter will look at possible areas for improvement and further research that can be undertaken to further improve the quality of solutions obtained with the WCP TS algorithms.

Chapter 5

Opportunities for Future Research

This project was tested on benchmark problems to evaluate its effectiveness of being able to solve a WCP that includes the following:

1. Mixed multigraph that presents:
    a. Two way streets that requires service in only one of the two directions of travel (zigzag collection of both sides of the road).
    b. Two way streets that requires service in both directions of travel (Each side of the road collected separately).
    c. One way streets (zigzag collection of both sides of the road).
    d. Large one way streets (Each side of the road collected separately).
2. Turn penalties e.g. right turns at a busy intersection.
3. Forbidden turns e.g. U-turns.
4. Several dumping sites.
5. Windy edges, whose cost depends on the direction of travel e.g. a sloping street.

All result from benchmark problems were verified and validated against the best found result in the literature, but these benchmark problems only represent small, relative simple WCPs that do not include turn penalties and IFs. To test these extensions, a set of six problem instances were created to evaluate the models' capability to solve these extensions efficiently. Since no best result were available, the answers could not be verified but the assumption were made if the model is verifiable for the basic CARP instances by setting the turn penalties to zero and the number of IF to zero it should be sufficient to verify the models' capability to solve the WCP with turn penalties and IFs as well.

During the development phases of the models, it became evident that one can improve on the models' accuracy with which it represents reality. By making simplifying assumptions, ones model can compute good results but will in fact be impractical in real life. Therefore to address and effectively solve a real problem that will obtain good result, it is very import to create a model that represents reality.

Future research should be done to implement this project on a real WCP for a given municipality that could be used to support decisions regarding the problems they experience in reality with optimal routing of waste collection vehicles.

This Chapter present several opportunities for improvement that can be pursued in future research before implementing this project on a real WCP.

5.1 Geographic Information System (GIS)

This project only included test problems for which all the input date were a given. If this project would be implemented on a real problem, input data regarding the distances between nodes need to be calculated. One way of gathering the distances would be to obtain accurate distance through physical measurement of roads but this will be very time consuming for large networks. An improved method would be to investigate the incorporation of a GIS system with the current model. The advantage of this method is that GIS can create an accurate distance matrix, efficiently in a short period of time. Additional benefits of GIS is that it provides the ability to attach distinctive information to each road that can be used to calculate an improved cost matrix rather than basing the cost entirely on the distance between nodes. Such information may include the slope of the road, speed limits as well as traffic flow which will all influence the optimal route selected.

5.2 Multiple Vehicle Types with Different Capacities

The model created for this project only allowed the use of a homogeneous fleet of vehicles. There may however exist situation were a heterogeneous fleet are required such as when a municipality use trucks that differ in their capacity. Further research should be aimed at the possibility of including a heterogeneous fleet in the model to more accurately reflect reality.

5.3 Using Distribution for Uncertain Input Data

There exist the opportunity to be able to more accurately represents uncertain input data with distributions. For example waste quantities may very from week to week and this will affect how quickly the vehicle's capacity is reached. Meaning a vehicle may need to return to the closest dumping site before the designed route is finished as a direct result of higher waste quantities experienced. By using distributions the results obtained from the model will be able to incorporate the effect that uncertain demand will have on the creation of optimal collection routes.

5.4 Further Improving the Search

Out of the result obtained improvements on the initial solution were found but the percentage deviation from the best found result in literature is still relative large indicating that the algorithm have the potential for future improvement to be even more capable of providing a improved answer to real world problems.

This document can be viewed as a step toward solving more general real live WCPs that include turn penalties an IF, as is often the case in practice.

Bibliography

Belenguer, J. and Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705–728.

Belenguer, J., Benavent, E., Lacomme, P., and Prins, C. (2006). Lower and upper bounds for the mixed capacitated arc routing problem. *Computers & Operations Research*, 33(12):3363–3383.

Benavent, E. and Solver, D. (1999). The directed rural postman problem with turn penalties. *Transportation Science*, 33(4):408–418.

Brandao, J. and Eglese, R. (2008). A deterministic tubu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4):1112–1126.

Chu, F., Labadi, N., and Prins, C. (2006). A scatter searh for the periodic capacitated arc routing problem. *European Journal of Operational Research*, 169(2):586–605.

Eiselt, H. and Laporte, G. (1995). Arc routing problems, part ii the rural postman problem. *Operations Research*, 43(3)(3):339–414.

Ghiani, G., Improta, G., and Laporte, G. (2001). The capacitated arc routing problem with intermediate facilities. *Networks*, 37(3):134–143.

Golden, B. and Wong, R. (1981). Capacitated arc routing problems. *Networks*, 11(2):305–315.

Greistorfer, P. (2003). A tabu scatter searh metaheuristic for the arc routing problem. *Computers & Operations Research*, 44(2):249–266.

Hansen, P. and Mladenovic, N. (2001). Variable neigborhood search: principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Hertz, A., Laporte, G., and Mittaz, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1):129–135.

Hertz, A., Laporte, G., and Mittaz, M. (2001). A variable neigborhood descent algorithm for the undirected capacitated arc routing problem. *Transport Science*, 35(4):425–434.

Hirabayashi, R andsaruwatari, Y. and Nishida, N. (1992). Tour construction algorithm for the capacitated arc routing problem. *Asia pacific journal of oprational research*, 9(2):155–175.

Lacomme, P., Prins, C., and W, R.-C. (2006). Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(4):156–185.

Polacek, M., Doerner, K., Hartl, R., and Maniezzo, V. (2007). A variable neigborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of heuristics*, NA:NA.

Rardin, R. (1998). *Optimization in Operation Research*. Prentice Hall Upper Saddle River New Jersey.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*. Philadelphia, PA: SIAM.

Winston, W. and Venkataramanan, M. (2004). *Introduction to Mathematical Programming*. Curt Hinrichs.