Tour Generation

Applied combinatorial optimization

A hybrid approach for large Pickup and Delivery Problems

(Public version containing limited non-confidential information)

Master project BA
J. Arentshorst
July 2013

Supervisor VU
J. Gromicho

Supervisors ORTEC
I. Meuffels
F. van der Wal

Tour Generation
Applied combinatorial optimization
A hybrid approach for large Pickup and Delivery Problems

Master Project Business Analytics
Jan-Willem Arentshorst
July 2013

Supervisor: Prof. Dr. Joaquim dos Santos Gromicho
Second reader: Prof. Dr. Ger Koole
Vrije Universiteit Amsterdam
Faculteit der Exacte Wetenschappen
Studierichting Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

Primary supervisor: Ineke Meuffels, MSc.
Secondary supervisor: Frank van der Wal, Drs.
ORTEC
Houtsingel 5
2719 EA Zoetermeer

I. PREFACE

This report is part of the mandatory six month internship of the Master 'Business Analytics' at the VU University Amsterdam. The goal of this internship is to apply the acquired theoretical knowledge of business mathematics and informatics to a practical problem in Operations Research. I've been working in the Courier, Express and Parcel department of ORTEC from February to July 2013. I would like to thank everyone who has made a contribution to my internship. I explicitly want to mention Ineke Meuffels and Frank van der Wal who played a crucial part in my internship. I also would like to thank Joaquim Gromicho for his expertise in operations research, guidance and constructive feedback which guided me in the right direction. I also would like to thank second reader Ger Koole for reading this report.

Jan-Willem Arentshorst
July 2013

II. INTRODUCTION TO ORTEC

ORTEC is one of the largest providers of advanced planning and optimization software solutions and consulting. ORTEC's products and services result in optimized fleet routing and dispatch, vehicle and pallet loading, workforce scheduling, delivery forecasting, logistics network planning and warehouse control. ORTEC offers stand-alone, custom-made and SAP® certified and imbedded solutions, supported by strategic partnerships. ORTEC has 650 employees, offices in Europe, North America, Asia and the Pacific Region and over 1,650 customers worldwide including Shell, Coca-Cola and TNT. In 2012, ORTEC has won the annual Franz Edelman Award, an award honoring great accomplishments in operations research worldwide for their intense collaboration with TNT Express. TNT Express, supported by ORTEC's modeling capabilities and solutions, carried out hundreds of optimization projects on a global scale which resulted in saving 60 million kilometers and 54 million kg of CO2 emissions by optimizing its domestic networks in Germany, France, Spain and Italy.

III. ABSTRACT

The Tour Driver Problem (TDP) is the optimization problem of combining movements into driver schedules called tours. A movement is a transportation task between two locations at a certain time with a given duration. In contrast to regular Pick-Up and Delivery Problems a constraint is added which forces a tour to end near the start location. This problem is common in services like parcel delivery or the taxi industry. The algorithm design is confidential.

The approach is tested on three different datasets consisting of up to 1750 tasks over 6 days between up to 62 locations. The results were compared to another method based on column generation using four selected KPI's; unscheduled movements, waiting time and repositioning in between and at the end. The results, conclusions and recommendations are confidential as well.

Keywords: Tour Generation, Tour Driver Problem, Combinatorial optimization, Pickup and Delivery Problem, Hybrid, Set Partitioning Problem.

IV. SUMMARY

This report describes the Tour Driver Problem (TDP), which was the subject of the mandatory half-year internship at ORTEC for the Master degree 'Business Analytics'. The TDP is the optimization problem of finding the optimal schedule which assigns a given set of movements to tours. A movement is a transportation task between two locations at a certain time with a given duration. A tour can be interpreted as a driver schedule and consists of one or more movements. A key aspect of this problem is the limitation in repositioning, i.e. the distance that a vehicle is allowed to drive without carrying any freight. There are two kinds of limitations on repositioning; at the end of the tour and in between movements. The first kind ensures that a driver ends the tour close to where it started. The second kind ensures that movement B can only follow movement A, if B starts near where A ended. The cost of a tour usually depends on the total amount of repositioning, the waiting time between movements, the number of drivers needed and the number of days. Capacity constraints are disregarded since the business case assumes that either the entire capacity is required for each delivery or that performing a second pick-up before the first delivery leads to infeasibility due to the time-windows of the delivery. The TDP is a common problem in services like the parcel delivery business and the taxi industry.

[CONFIDENTIAL]

The method was tested on three different datasets, each consisting of respectively 1025, 1770 and 765 movements between 49 or 62 locations over a period of six days. The solution is compared to a different method studied by ORTEC based on column generation. The exact objective function of the method could not be reproduced, so KPI's are reported instead. The four KPI's are the amount of repositioning at the end of a tour, repositioning in between tours, waiting time and number of one-ways. One-ways are tours consisting of only one movement and are considered inefficient. These KPI's should give a sufficient indication regarding the quality of the tours.

[CONFIDENTIAL]

TABLE OF CONTENTS

1. PROBLEM DESCRIPTION

This chapter contains a description of the Tour Driver Problem (TDP). In the first segment we will introduce the context of the problem. The second segment will contain the problem description, followed by sections containing explanations of the costs, assumptions and constraints. The final segment will give an overview of the structure of the report.

1.1. Definitions

The Courier, Express and Parcel (CEP) department of ORTEC supports express-, courier- and mail services in optimizing their infrastructure and managing their logistical networks. These businesses offer a service in which a client pays to have its parcel delivered to a specific location within a specific amount of time. ORTEC-CEP develops various tools for these companies; for example giving insight into the routes of the parcels, optimizing loads of their vans or quantifying the average flows between depots. The journey of a package consists of three phases. First, parcels are collected from the customers in local distribution centers called **depots**. Second, the parcels are sent from the origin depot to the destination depot through one or several **hubs**, which are transshipment centers in which incoming parcels are sorted and forwarded to the next hub or depot. The transportation tasks between hubs and depots are called **movements**. A movement is defined as a transportation task with a certain vehicle type from location A to location B with specific start- and end times. Finally, after arriving at the end depot, the packages are distributed to customers. The express-chain can be split into a **Pick-up-and-delivery** part between customers and depots and a **line-haul** part consisting of the movements between hubs and depots. These two parts are separated by **cut-off times**; the time limit when all packages need to be collected at the origin depot and the latest time when the last movement has to arrive at the destination depot. In this thesis we are only interested in the line-haul part of the express chain, an example of which is shown in Figure 1.

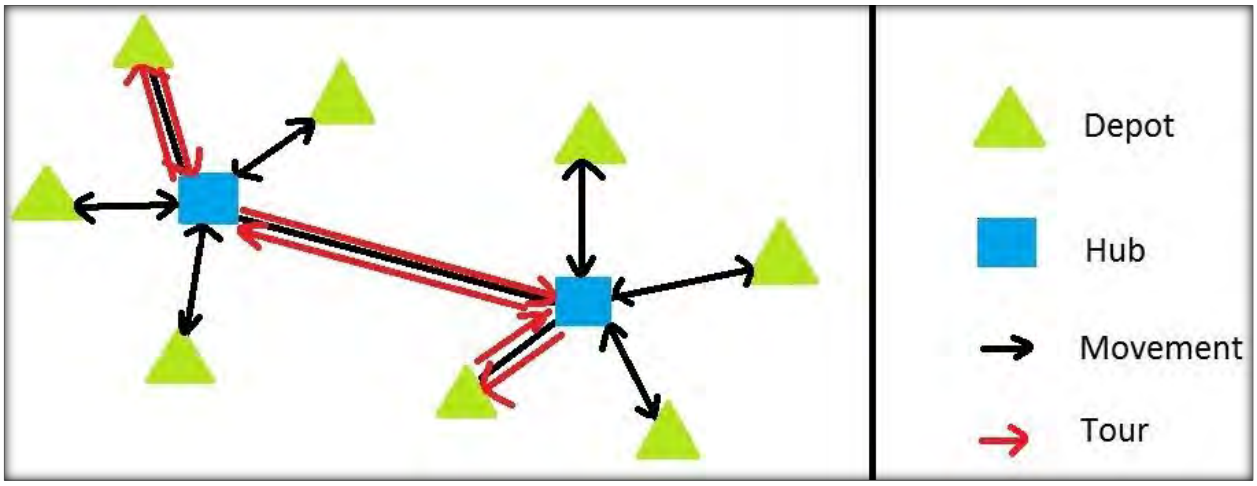

Figure 1: Example of a line-haul network

Having a different driver and vehicle for every single movement is obviously inefficient. A **tour** is a combination of multiple movements in one shift for a specific driver/vehicle-combination. For example, a truck driver starts with a movement at 09:00 from depot A, arriving at hub B at 13:00. He takes a 45 minute break, and then waits fifteen more minutes until the staff at hub B is done (un)loading his truck. He then carries out the movement from B to A arriving at 18:00. The idea behind generating tours is the

creation of practical work schedules for truck drivers given a set of transportation tasks. That is why in many cases a tour is required to end where the first movement started, such that drivers can easily return to their own homes. If a tour does not end where it started, **repositioning** is needed. Repositioning is the act of driving an empty vehicle from one location to another. If the same tour is identically performed on multiple days (for example on Mondays, Tuesdays and Wednesdays) this is called a **tour template.** Some companies find it easier to sell tour templates to subcontractors than to sell the tours individually. In certain cases it might therefore be cheaper to form a few 'great' tour templates and be stuck with a handful of single movements, than to have perfect but different tours on every day of the week. These tour templates are left out of scope in this thesis.

1.2. The Tour Driver Problem (TDP)

OBJECTIVE

This thesis is about finding an algorithm which finds the 'optimal' set of tours for the TDP, such that every movement is assigned to a tour. Like many other practical optimization problems, the optimal set of the TDP means the cheapest set of tours, but since the cost function and constraints might be very different for each business setting, the optimal result will heavily depend on the company's preferences. For example, companies which offer their tours to subcontractors might be less interested in repositioning costs and more in minimizing the total amount of tours.

MATHEMATICAL NOTATION

MODELING

The TDP can be modeled as a Set Partitioning Problem; where every movement has to be carried out exactly once. In our context, carrying out a movement twice will actually increases the service level in the real-world and does not make the solution infeasible. Since deleting a movement from a tour will never result in a more expensive tour, an optimal Set Covering solution will be an optimal Set Partitioning. Optimization with inequality (Set Covering; movements are executed at least once) is much easier than optimization with equalities (Set Partitioning; every movement exactly once) since even the slightest mutation in a Set Partitioning Problem could lead to infeasibility, whereas the same mutation in a Set Covering Problem would only lead to a different target function value. This makes the optimization of Set Covering much more stable and therefore easier than Set Partitioning as explained in (1). Unfortunately, allowing movements to be performed multiple times might lead to tours using small movements in between as loopholes to circumvent the limit on maximum waiting time. Therefore, we will be using Set Partitioning. Set Partitioning optimization is a classical problem in combinatorics and is known to be NP-hard. Exact calculation of the optimal solution of NP-hard problems is a hopeless task for large instances; therefore we will be looking for other means of solving the TDP, for example by using metaheuristics.

NOTATION

Let F be the set of all possible tours. Then:

x_f = tour *f* is used in the solution

a_{if} = movement *i* is part of tour *f*

c_f = costs of tour *f*

$$\text{Min} \quad \sum_{f \in F} c_f x_f$$

$$\text{S.T.} \quad \sum_{f \in F} a_{if} x_f = 1 \qquad \forall i \in I$$

$$x_f \in [0,1] \qquad \forall f \in F$$

Note that the size of F quickly becomes extremely large as the number of movements increases, which makes this problem difficult. Finding the best combination of all of these tours yields even more possibilities, so this should not be done by full enumeration.

1.3. Assumptions

MOVEMENTS

We assume that the movements are immutable, so a movement cannot be replaced by other movements and every movement must be assigned to a tour. We also assume that distances and travel times between depots and hubs are deterministic. They do not have to be symmetrical (i.e. d[A,B] ≠ d[B,A]), but they do have to be constants. Furthermore, since not every movement is on the critical time path, there should be slack available to delay (or bring forward) the departure times of certain non-time-critical movements, even though movements are considered to have a static departure and arrival time. Expanding the algorithm to incorporate time-windows could have a significant added value, but this movability of departure times will not be discusses to reduce the complexity of the problem.

VEHICLE TYPES

Transportation can be carried out by different types of vehicles. In some cases parcels are transported by aircrafts, trains or ferries, but this is usually done by trucks or vans. We assume that one specific vehicle type is assigned to each movement, which means that if the movement from A to B is scheduled to a van, then the van cannot be replaced by a truck even though the truck has more capacity. There are some reasons to justify this assumption, for example that compared to trucks; vans are usually faster and less expensive. Making this assumption divides the problem into several sub-problems (one for every vehicle type), making the problem less complex.

1.4. Constraints

REPOSITIONING

One of the constraints of generating tours is that drivers should be able to easily return to their homes at the end of the tour. In most cases, repositioning at the end of the tour is allowed to some extent. This means that in most cases the ending location only has to be close to the starting location. How much repositioning is allowed depends on the situation. Some companies do not allow any repositioning, while other companies are not interested in this constraint. Some situations also allow repositioning in between movements. Again, whether this is allowed or to which extent fully depends on the preferences of the company.

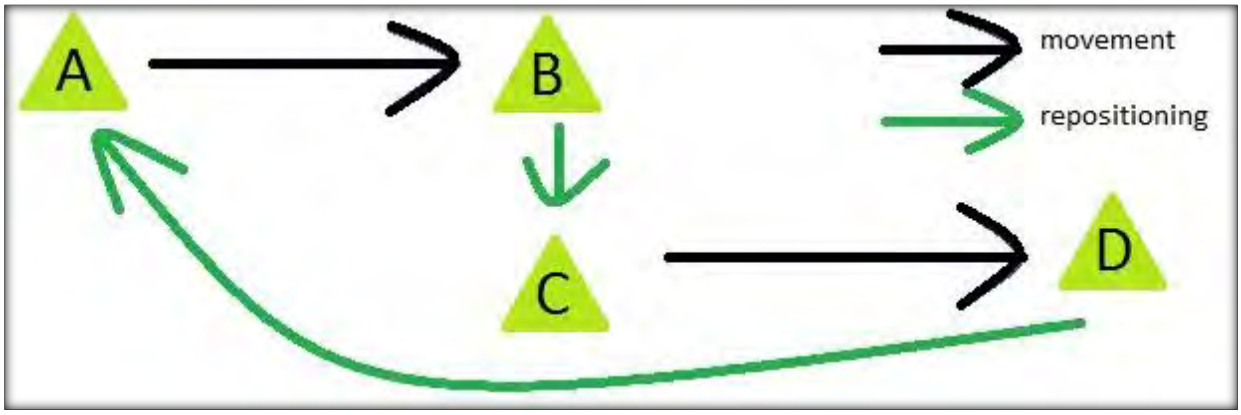

Figure 2: Example of a tour with repositioning

Figure 2 shows an example of a tour. The tour starts with a movement from depot A to depot B. From there, the driver repositions to the nearby depot C, from which he will execute the movement from C tot D. At the end of the day, the driver repositions back to his starting location at depot A.

LABOUR LAW AND DRIVING LEGISLATION

Drivers are required to take a 45 minute break after 4.5 hours of driving, but they are allowed to split this break up into a 30 minute break and a 15 minute break. The maximum amount of driving per day is 10 hours for one driver, and 22 hours for two drivers. The maximum amount of working time for one driver is limited to 16 hours per day. We will not be discussing weekly (or monthly) working time restrictions, since allocating drivers to tours is not in the scope of this thesis.

1.5. Costs

There are many factors influencing the cost of a specific tour. Below is a list of these factors.

REGULAR DRIVING COSTS

We assume different driving costs for different vehicle types, but only one vehicle type is considered at a time. The cost of a tour is usually composed of a fixed amount per day plus a variable amount per kilometer or hour.

REPOSITIONING COSTS

If the ending location of a tour is not equal to the starting location, then the vehicle has to drive back (empty) to the starting location. Users might want to avoid these 'empty' kilometers so they could be given an additional cost on top of the regular costs per kilometer.

MULTIPLE DRIVERS OR DAYS

There might be very long movements which require two drivers instead of one, since two drivers are allowed to driver longer. Obviously, having two drivers instead of one is more expensive. It should also be possible to schedule multiple-day tours.

WAITING COSTS

Not all movements will connect perfectly in timing. In many cases, a tour from A to B and back to A requires the driver to wait at depot B for a certain amount of time. These waiting times could be advantageous or not, depending whether the user prefers additional breaks or higher productivity. Therefore, it should be possible to alter the cost function to increase (decrease) the cost associated with waiting. Depending on the situation, waiting time might be considered working time or not.

TOUR CATEGORIES

Certain tours are considered better than others. Some of ORTEC's clients prefer to outsource the driving to subcontractors. The price these subcontractors ask for a specific tour depends on the characteristics of the tour. For example, most subcontractors seem to ask less for perfect round-tours (where the starting and ending location are equal) or tours with a duration of exactly 8 working hours and ask a higher price for 5-hour one-way 'tours' to the other side of the country. Therefore, tours could be given a certain discount or premium depending on the characteristics of the tour. This is not in the scope of this thesis.

COST FUNCTION

The above mentioned costs lead to the following function:

Costs of tour = Fixed cost (per day)
+ Variable cost per km (hour) * the amount of kilometers (hours)
+ Additional repositioning costs * total repositioning in kilometers
+ Waiting costs * total waiting time
+ Penalty for requiring an additional driver
+ Penalty for a multiple-day tour (due to overnight costs)

1.6. Structure of the report

In the next chapter similar problems in literature will be discussed. Chapter 3 contains an overview of several methods to solve the TDP and chapter 4 contains a review of three methods already researched by ORTEC. Chapter 5 gives an elaboration on the chosen solving method for the TDP and its results will be discussed in chapter 0. Concluding remarks and suggestions towards further research are provided at the end of this thesis in chapter 7.

2. PROBLEMS IN LITERATURE SIMILAR TO THE TOUR DRIVER PROBLEM

This thesis deals with the Tour Driver Problem (TDP); a problem regarding combining movements (transportation tasks) into tours (a practical day schedule). The problem lies in NP because given a set of tours it can be checked in polynomial time if all tasks are covered in these tours and if the total costs are less than an arbitrary value *k*. It can be reduced to a Vehicle Routing Problem (VRP) by adding arcs with length zero from each location to one fictive node, and updating the costs of those arcs for all tours that do not end where they started. Since the TDP is in NP and can be reduced to a VRP which is shown to be NP-hard in (2), the TDP is NP-hard as well. This problem is similar to other problems in combinatorial optimization like the Set Partitioning Problem, the Pick-up and Delivery Problem and the Crew Scheduling problem. These problems are briefly described in the following chapter. Since the size of these problems is usually large, many theses are focused on finding means which use the information regarding the structure of the problem to their advantage in finding better solutions quicker. Each thesis usually has a slightly different problem which makes it difficult to compare the quality of the implementation's results. What these theses can provide is information about methods and implementations, some of which are used in the chosen algorithm which will be elaborated in chapter 5.

2.1. Set Partitioning Problem

The Set Partitioning Problem (SPP) can be seen as a generalization of the TDP, with the elements being the movements between depots and hubs, and the subsets being the tours. The objective is to select subsets to cover all elements at least once while minimizing the total costs associated with selecting subsets. In Figure 3, the elements are the twelve dots and there are six different sets available; three blue and three red sets. The only feasible solution consists of the blue sets.

Set Partitioning Optimization is proven to be NP-hard. Note that the Set Cover Problem, in which all elements must be covered at least once, might have a different optimal solution depending on the costs of each set. SPP is a very well-known combinatorial optimization problem which received extensive attention from literature. The Set Partitioning decision variant is one of Karp's 21 NP-complete problems. Exact solvers for our problem not using the structure of the problem actually optimize an instance of the SPP. Examples of overviews of solution methods are (3), (4), (5) and (6). But since the amount of sets grows exponentially in the number of elements and the amount of solution grows even faster, we should be looking at ways to use the problem and its structure to make educated guesses which combinations to try.

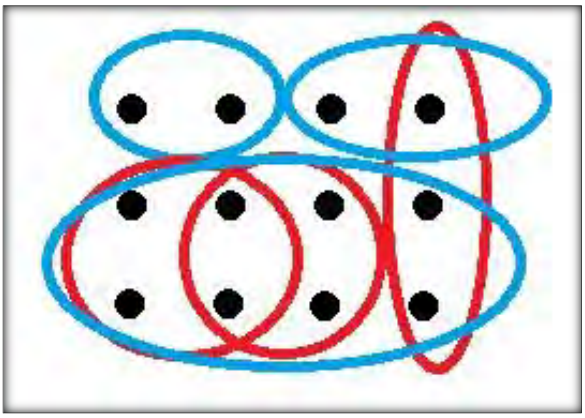

Figure 3: Example of a Set Partitioning Problem

2.2. Pickup and Delivery Problem

The Pickup and Delivery Problem (PDP) consists of transportation tasks with a certain load from origin locations to destination locations with homogenous vehicles departing from and returning to a central depot. It is a generalization of the Vehicle Routing Problem (VRP), which assumes some central depot to be the starting point of every tour. Since in our case vehicles can start from any location, the TDP and PDP can be reformulated as a VRP in which the central depot is a fictive node with arcs to all 'customers' with costs zero. The costs of the arcs to the fictive central location can also be used to represent the repositioning costs if the starting location is not equal to the ending location. Extensions of the PDP incorporate time-windows in which the transportation has to take place (PDP-TW). A survey of solution techniques for PDP and some of its extensions like multiple vehicles, uniform goods, specific capacity constraints and stochastic demand is given in (7), (8) and (9).

2.3. Crew Scheduling Problem

The Crew Scheduling Problem (CSP) deals with assigning staff to a set of tasks. A common example of a CSP is the scheduling of airline crew while respecting labor legislation. The CSP is commonly solved with column generation, since it is an exact solver and long-term schedules are made weeks in advance so computation time is abundant. The CSP is usually broken down into two sequential phases due to the computational complexity; in the first phase staff is optimally grouped into cabin crew and in the second phase crews are assigned to flights. An overview of methods for solving crew scheduling problems is given in (10).

3. COMMON METHODS IN COMBINATORIAL OPTIMIZATION

NP-hard optimization problems like the TDP usually have to deal with intractably many combinations. If tours consist of between one and four movements, then a total of 32 movements will lead into nearly one million possible tours. Combining these million tours into solutions consisting of up to 16 tours lead to over 10^{82} million possible solutions, which is more than the amount of atoms in the universe. Since most datasets consist of much more than 32 movements, an intelligent way to traverse the search space is required to find the global optimal solution. In the following paragraphs several approaches of exact solutions and heuristics are explained and compared to each other. At the end of the chapter a recommendation for a solution method is given.

3.1. Exact solutions

BRANCH-AND-PRICE: COLUMN GENERATION AND BRANCH-AND-BOUND

CONCEPT

Column Generation only considers a small portion of the potential variables at a time instead of solving the problem as a whole. After each iteration, variables (columns) are added (generated) to the problem, and the problem is solved again until there are no useful variables left to add. Which columns to generate is a difficult and problem-specific sub-problem known as the “pricing problem”. Most scheduling and network problems are (Mixed-)Integer Problems, i.e. you cannot produce 8.6 items from a machine, schedule 2.4 nurses to an operating room or assign 0.7 trucks to a delivery. Integer problems are much more difficult to solve than linear problems, so the integer constraints are usually either ‘relaxed’ using Lagrange multipliers or completely relaxed making the problem linear. The reduced cost of an unused variable in linear programming is the value by which the objective function would change if the variable would be added to the solution. We are therefore interested in unused variables with negative reduced costs. Reduced costs of new columns can be calculated using the Lagrangian multipliers or the dual information from the relaxed linear program. Obviously the reduced cost is not calculated for all of the billions of unused columns, but usually the pricing problem is reformulated, for example, as a Knapsack-Problem or a Shortest-Path Problem and then a heuristic is used to find some good columns with negative reduced costs. Note that it is not necessary to find the column with the most negative reduced costs in every iteration, so a heuristic or approximation is sufficient. The problem is solved over and over until no columns with reduced costs can be found. Now, hopefully the solution is completely integer, otherwise the problem is branched fixing a chosen non-integer variable to 0 in one branch and 1 in the other. The lower bound of each branch is given by the LP-relaxation of that branch, since the best integer solution cannot be better than the best linear solution. The global upper bound is the best found integer solution so far. Now, we can discard all branches with an LP lower bound worse than the global integer upper bound. It is crucial to design the branching in such a way that the least amount of branching is needed, so it is important to branch at the best possible node and branch on the best possible variable. A variation on this method called “Branch-and-Price-and-Cut” adds cutting planes to tighten the linear relaxation. (1) (11) (12)

STRENGTHS AND WEAKNESSES

The strength of branching is that it is an *exact* method which guarantees that it will always find the global optimum eventually. Finding the right heuristic for the pricing problem is not trivial. The biggest problem lies in the computation time, which explodes exponentially in the number of locations or movements, even when using efficient branching and intelligent heuristics for the pricing problem. This technique should be preferred for small instances, but might not even give feasible results for large datasets as mentioned in (13) and (14). Since our problem should be able to cope with thousands of movements, we will be looking into metaheuristics instead.

3.2. Metaheuristics

A heuristic is a problem-specific and efficient method to generate a near optimal solution; a local optimum. Metaheuristics are general-purpose strategies to efficiently guide the search for the global optimal solution, making their concept applicable to every type of problem. The main goal is to avoid the disadvantage of iterated improvement of converging to a local optimum, either by allowing worsening moves or by generating new starting solutions for the local search in an 'educated' way. Metaheuristics need to make a trade-off between intensifying the search for the global optimum near the best found solution so far, and diversifying to avoid getting stuck in local optima. Below is an overview of some of the commonest metaheuristics. A basic overview of metaheuristics is given in (15) and (16).

SIMULATED ANNEALING (SA)

CONCEPT

The idea of Simulated Annealing (SA) comes from the annealing of metals in which metals in a solid state are heated to a high temperature, which allows atoms to leave their bonds. The metal is then slowly cooled down to room temperature, increasing the size of the crystals within the material which decreases the amount of defects. SA is based on this principle of a slowly decreasing temperature. In each iteration of SA, the neighborhood of a solution is searched and improvements are always accepted. But unlike other local search algorithms, worse solutions are also accepted with a probability depending on the temperature. As the temperature decreases over iterations, the probability of accepting worse moves decreases as well. Threshold Acceptance is a variation of simulated annealing which allows 'bad' moves if it is below a certain periodically lowered threshold instead of using a random generator and an exponential function. (17) (18) (19)

STRENGTHS AND WEAKNESSES

SA is a hill climbing algorithm which also accepts some downhill movements. It can be great to analyze large neighborhoods and is very easy to code. The downside is that the temperature-parameters need a tuning, since setting the initial temperature too low will result in only part of the search space to be searched and the nature of the cooling schedule is the trade-off between computation time and the chance of being trapped in local optima. The optimal result can be very dependent on the starting location, and several restarts might be required. Simulated Annealing seems like a reasonable meta-heuristic especially in the final stages of the problem, but as an iterated local-search-based algorithm it might lack the diversifying power of a population-based algorithm like GA.

TABU SEARCH (TS)

CONCEPT

Tabu Search (TS) is an iterated local-search-based algorithm like SA. In every iteration of TS (most of) the neighborhood of the current solution is searched. The solution is then moved towards the best non-tabu neighbor, even if the result is worse, and the "move" is added to the tabu list. This way, the tabu list prevents re-visiting recently visited states therefore escaping local optima. Some extensions of TS use dynamic tabu list lengths depending on the frequency of revisits to certain states. Others use "aspiration criteria" which accept tabu solutions, for example when a tabu solution is the best found solution so far. Granular Tabu Search (20) decreases the neighborhood size by excluding some of the costliest mutations. (15) (19) (21) (22) (23)

STRENGTHS AND WEAKNESSES

TS can converge quickly to an optimum, but the tabu list is mainly used to prevent cycling and might not be sufficient to prevent converging to a local optimum. Therefore multiple random restarts might be needed. If the search space is very large then keeping a list of tabu solutions might not be valuable since there are many similar solutions with similar (and therefore less interesting) results. Instead, tabu lists can be more effective if they contain attributes of a solution, which might also lead to the avoiding of some very good solutions. Formulating what information is used by the tabu list is therefore quite complex and needs a lot of tuning. In our case, with thousands of movements and therefore gigantic neighborhood sizes, defining suitable tabu attributes might be difficult to do, making TS a less attractive option.

GENETIC ALGORITHM (GA)

CONCEPT

Genetic Algorithms (GA) uses the principal of natural selection and 'survival of the fittest'. The fittest solutions are stored in the population and in each iteration two parent solutions are chosen for reproduction. The most common policy for parent selection is a probability proportional to (a function of) the fitness function. But since this can be very time-consuming, other policies like *k*-tournament selection (in which two pools of *k* specimens are selected, and the fittest specimens from both pools are selected for breeding) or selection based on ranking is commonly used as well. When two parents are selected, parts of these parents are combined into two children. Usually one or more crossover point(s) are selected and the entire segment between these points is swapped, as shown in Figure 4.

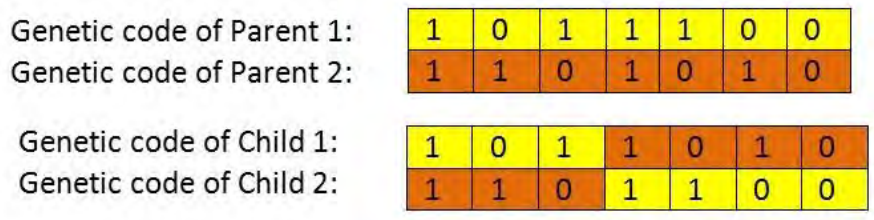

Figure 4: Example of a crossover in a genetic algorithm

Describing the solution in a binary array to reduce the computation time is preferred, but not essential. Any crossover structure is allowed, as long as it is well defined. For example in a multi-depot VRP, in which a set of customers can be serviced from several depots, one could divide the solution in two parts by randomly assigning each depot to either part A or part B. Now, we can combine part A of parent1 with part B of parent2 forming child1, and combine part A of parent2 with part B of parent1 forming child2. In many complex combination problems like the multi-depot VRP, the crossover of two distinct parents will rarely result in a feasible solution by itself, requiring some 'repair' to take place. The key to mutation is keeping the beneficial characteristics from the parents intact while adding enough new potentially interesting "genes" to the child solutions to reduce the chance of getting stuck in a local optimum. In most complex problems, a neighborhood search is done after the mutation to increase the fitness of the solution. At the end of an iteration, part of the population is replaced by the newly generated children depending on the fitness functions. In some situation all but the fittest solutions are replaced by children to increase the diversity, other situations always keep the fittest to increase the intensification. (17) (24) (25) (26) (27) (28) (29)

STRENGTHS AND WEAKNESSES

GA is very robust and can be implemented for any optimization problem. The concept is easy to explain and usually easily implemented, but it might be difficult to find an easy way to represent the solution in a chromosome (array). The great advantage of GA is that it allows the search in multiple neighborhoods at the same time, while gradually applying information gained from a single neighborhood to the entire population of solutions. This means that GA is great in diversification of the solution, covering a great part of the search space. But the nature of the algorithm is very probabilistic, which might result in a different solution every run, which might not be preferable for managers. The calculation time of GA is usually longer than other methods, since keeping a large population of solutions requires a lot of unnecessary computation time especially in the final stages of the algorithm when many children look alike. Therefore, GA can be a very efficient method to search a gigantic solution space, but it should be combined with some kind of local search to increase the convergence speed as showed in (30). GAs can be difficult to manage and tend to either converge too little or too much giving disappointing results. Combinations of evolutionary-based algorithms with local improvement heuristics are called 'memetic algorithms' (MA), but have also received various other names like 'Baldwinian/Lamarckian Evolutionary Algorithms' or 'Genetic local search'.

ANT COLONY OPTIMIZATION (ACO)

CONCEPT

Ants randomly explore the area around their anthill for food. After finding food, they leave a chemical pheromone trail on their way back for others. There are many paths an ant can take to the food source, but due to the pheromone trail only the shortest path remains in use after a few passages. Eventually, the majority of the ants will follow the shortest path and only a few will keep exploring other paths. This concept of reinforcing the shortest path while exploring alternatives was translated to optimization problems. In each iteration of Ant Colony Optimization (ACO), each ant traverses the network constructing a tour on its way. The probability that an ant moves from node X to Y depends on η_{xy} and τ_{xy}, respectively the "attractiveness" of moving from X to Y and the pheromone trail between X and Y. This attractiveness is a heuristically chosen constant, usually depending on the costs of the movement.

One could choose $\eta_{xy} = \frac{1}{d(x,y)}$ such that shorter routes are preferred, but one could also add some value regarding constraint satisfaction, such as a penalty for shorter tours or a penalty if Y is a node with little connections. With a probability *q* the ant chooses the next node Y: Y = arg max { $\eta_{xy} * \tau_{xy}$ } and with probability *1-q* chooses a next node *Y* with probability: P(Y)= $\frac{\eta_{xy} * \tau_{xy}}{\sum_i \eta_{xi} * \tau_{xi}}$. Each ant adds segments to its tour until it has found a feasible solution. The ants' solutions are usually optimized using a local neighborhood search heuristic to speed up convergence. When all ants have found a tour, the pheromone trails are updated. All pheromone trails evaporate with a certain rate α and new pheromone is added to the newly found tours depending on the fitness of the tour. Some extensions of the ACO only add pheromone to the best new tour found in an iteration. Other extensions also reinforce the best global solution each iteration (Elitist Ant Systems) or sort newly found tours by their fitness and add pheromone corresponding to their rank (Rank-based Ant Systems). Other extensions also add local pheromone updates, which make recently travelled paths less desirable for future ants to increase diversification. (15) (19) (31) (32) (33)

STRENGTHS AND WEAKNESSES

Compared to other metaheuristics ACO works well with dynamically changing maps, since ACO can instantaneously update all the attractiveness factors directly in the routing probabilities whereas even Genetic Algorithms would require several generations of very specific mutations before obtaining newly favorable properties in the entire population. This might be useful for situations like bandwidth allocation in computer networks or ambulance planning in which a schedule must be updated every few minutes depending on accidents and traffic jams. ACO is also less dependent on poor initial results and allows efficient (parallel) computing. Furthermore, it retains memory of all iterations through pheromones instead of only a part of the visited states and the pheromone distribution can give additional insight into bottlenecks or opportunities in the network. Downsides to ACO are a difficult theoretical analysis since probabilities are not independent, and the fact that it has to update the entire pheromone distribution every iteration which might take long for large and complex networks. ACO might be interesting to review for on-the-spot calculation of dynamic networks, but might not be as efficient as Simulated Annealing when it comes down to static optimization.

3.3. Recommendation

[CONFIDENTIAL]

4. CURRENTLY USED ALGORITHMS IN ORTEC'S SOFTWARE

[CONFIDENTIAL]

5. ALGORITHM DESIGN

[CONFIDENTIAL]

6. RESULTS

This chapter elaborates on the results from the approach which is specified in the previous chapter. The approach was tested on three different datasets from the year 2009. The reason to use these datasets was that they are also used in another ORTEC thesis as specified in Section **Fout! Verwijzingsbron niet gevonden.**, allowing their usage as some kind of benchmark. In Section 6.1 we will discuss which objective function is used for solving and which key performance indicators will be used to evaluate the qualities of the tour schedule. Section 6.2 will give an overview of used settings and parameters. Section 6.3 shows an overview of the datasets used for testing. Section 6.4 will give an overview of the main results, and Section 6.5 contains a sensitivity analysis regarding parameters and settings. Final conclusions and recommendations will be given in Chapter 7.

[CONFIDENTIAL]

6.1. Objective function and Key Performance Indicators

In most business cases, the most prominent Key Performance Indicator (KPI) is the overall cost. However, cost parameters are not always known or deductible from context and the calculation of some cost variables might not even be implemented. Since using different cost objectives will usually lead to different results, comparison based on costs alone might give a biased perspective. Therefore, other KPI's should be used for comparison if the full cost function cannot be reproduced. The aim of the scheduling problem is to increase productivity and efficiency, so the chosen KPI's should be indicators of either slack or efficiency.

Based on the business case explained in Chapter 1, the following four KPI's were selected:

- Amount of repositioning at the end of the tour to the start location
- Amount of repositioning in between movements
- Total amount of waiting time in between movements
- Number of one-way tours (tours with only one movement)

The first KPI is the amount of repositioning at the end of tours since tours are required to return to the starting point. An obvious objective is to reduce the distance travelled and additional kilometers should be penalized. Repositioning in between movements and waiting time are statistics that directly translate into costs; drivers are paid to wait and repositioning incurs costs. The final KPI is the number of one-way movements. These single-movement 'tours' are usually undesirable because they require repositioning by definition, so if the algorithm ends up with many one-ways it could imply an inability to generate enough quality tours.

6.2. Default parameter settings

[CONFIDENTIAL]

6.3. Dataset characteristics

[CONFIDENTIAL]

6.4. Main results

[CONFIDENTIAL]

6.5. Sensitivity analysis

[CONFIDENTIAL]

7. CONCLUSIONS AND RECOMMENDATIONS

[CONFIDENTIAL]

APPENDIX A: BIBLIOGRAPHY

1. **Barnhart, Johnson, Nemhauser, Savelsbergh and Vance.** Branch-And-Price: Column Generation for Solving Huge Integer Programs. *Operations Research.* 1996, Vol. 46, 3.

2. **Lageweg, Leenstra, Lawler and Rinnooy Kan.** Computer-Aided Complexity: Classification of Combinatorial Problems. *Communications of the ACM.* 1982, Vol. 25, pp. 817-822.

3. **Schilling, Jayaraman and Barkhi.** A Review of Covering Problems in Facility Location. *Location Science.* 1992, Vol. 1, 1, pp. 25-55.

4. **Dutta.** *Survey of Approximation Algorithms for Set Cover Problem.* s.l. : Master Thesis, University of North Texas, 2009.

5. **Paschos.** A Survey of Approximately Optimal Solutions to Some Covering and Packing Problems. *ACM Computing Surveys.* 1997, Vol. 29, 2, pp. 171-209.

6. **Caprara, Toth and Fischetti.** Algorithms for the Set Covering Problem. *Annals of Operations Research.* 2000, Vol. 98, 1-4, pp. 353-371.

7. **Doerner, Parragh and Hartl.** A Survey on pickup and delivery problems. *Journal für Betriebswirtschaft.* 2008, Vol. 58, 2, pp. 81-117.

8. **Manfrin.** *Ant Colony Optimization for the Vehicle Routing Problem.* Masther thesis, Université Libre de Bruxelles : s.n., 2004.

9. **Cordeau, Laporte and Ropke.** Recent Models and Algorithms for One-to-One Pickup and Delivery Problems. [ed.] Golden, Raghavan and Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges.* s.l. : Springer US, 2008, pp. 327-357.

10. **Ernst, Jiang and Krishnamoorthy.** Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research.* 2004, Vol. 153, 1, pp. 3-27.

11. **Danna, and Le Pape.** Branch-and-Price Heuristics: A case study on the Vehicle Routing Problem with Time Windows. [book auth.] Desrosiers, Solomon Desaulniers. *Column Generation.* s.l. : Springer US, 2005, pp. 99-130.

12. **Desrosiers and Lubbecke.** Generic Branch-Cut-and-Price. *Wiley Encyclopedia of Operations Research and Management Science.* 2010.

13. **Gerrits.** *The Resource Assignment Problem.* Master thesis, Erasmus Universiteit Rotterdam : s.n., 2011.

14. **Venkateshan and Mathur.** An efficient column-generation-based algorithm for solving a pickup-and-delivery problem. *Computers & Operations Research.* 2011, Vol. 38, 12, pp. 1647-1655.

15. **Blum and Roli.** Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys.* 2003, Vol. 35, 3, pp. 268-308.

16. **Baghel, Agrawal and Silakari.** Survey of Metaheuristic Algorithms for Combinatorial Optimization. *International Journal of Computer Applications.* 2012, Vol. 58, 19.

17. **Hosny.** *Comparing Genetic Algorithms and Simulated Annealing for Solving the Pickup and Delivery Problem with Time Windows.* College of Computer and Information Sciences, King Saud University : s.n., 2011.

18. **Brandao de Oliviera, and Vasconcelos.** A hybrid search method for the vehicle routing problem with time windows. *Annals of Operations Research.* 2008, Vol. 180, 1.

19. **Coello, Coello, Lamont and van Veldhuizen.** *Evolutionary Algorithms for Solving Multi-Objective Problems.* 2007. ISBN: 978-0-387-33254-3.

20. **Toth and Vigo.** The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing.* 2003, Vol. 15, 4, pp. 333-346.

21. **Lau and Liang.** Pickup and delivery with time windows: algorithms and test case generation. *International Journal on Artificial Intelligence Tools.* 2002, Vol. 11, 3, pp. 455-472.

22. **Braysy and Gendreau.** Tabu Search Heuristics for the Vehicle Routing Problem with Time Windows. *Sociedad de Estadística e Investigación Operativa.* 2002, Vol. 10, 2, pp. 211-237.

23. **Cordeau and Laporte.** A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological.* 2002, Vol. 37, 6, pp. 579–594.

24. **Créput, Koukam, Kozlak and Lukasik.** An Evolutionary Approach to Pickup and Delivery. [ed.] Bubak, et al. *Computational Science - ICCS 2004.* s.l. : Springer Berlin Heidelberg, 2004, Vol. 3038, pp. 1102-1108.

25. **Krasnogor, Aragón and Pacheco.** Chapter12: Memetic Algorithms. [ed.] Alba and Martí. *Metaheuristic Procedures for Training Neutral Networks.* 2006, pp. 225 - 248.

26. **Haupt and Haupt.** *Practical Genetic Algorithms.* s.l. : Wiley-Interscience, 2004. ISBN 0-471-45565-2.

27. **Beasley and Chu.** A Genetic Algorithm for the Set Covering Problem. *European Journal of Operation Research.* 1994, Vol. 94, pp. 392-404.

28. **Luke.** *Essentials of Metaheuristics.* s.l. : Lulu, 2011. 978-0557148592.

29. **Masum, Faruque, Shahjalal and Sarker.** Solving the Vehicle Routing Problem using Genetic Algorithm. *International Journal of Advanced Computer Science & Application.* 2011, Vol. 2, 7, p. 126.

30. **Marti, Laguna and Campos.** Scatter Search vs. Genetic Algorithms. [ed.] Rego and Alidaee. *Metaheuristic Optimization via Memory and Evolution.* 2005, 12, pp. 263-282.

31. **Dorigo and Stutzle.** Ant Colony Optimization: Overview and Recent Advances. [ed.] Gendreau and Potvin. *Handbook of Metaheuristics.* s.l. : Springer US, 2010, Vol. 146, pp. 227-263.

32. **Ren, Feng, Ke and Zhang.** New ideas for applying ant colony optimization to the set covering problem. *Computers and Industrial Engineering.* 2010, Vol. 58, 4, pp. 774-784.

33. **Cordón, Herrera and Stützle.** A Review on the Ant Colony Optimzation Metaheuristic: Basis, Models and New Trends. *Mathware & Soft Computing.* 2002, Vol. 9, pp. 141-175.

34. **Jourdan, Basseur and Talbi.** Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research.* 2008, Vol. 199, 3, pp. 620-629.

35. **Talbi.** A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics.* 2002, Vol. 8, 5, pp. 541-564.

36. **Puchinger and Raidl.** Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. [ed.] Mira and Álvarez. *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach.* 2005, Vol. 3562, 5, pp. 41-53.

37. **van Krieken, Fleuren and Peeters.** *A Lagrangean relaxation based algorithm for solving set partitioning problems.* 2004. 0924-7815.

38. **van Krieken, and Fleuren.** Problem reduction in set partitioning problems. *CentER Discussion Paper.* 2003, Vols. 2003-80, pp. 1-21.

39. **Bossers.** *Tour Driver Scheduling.* Master thesis, Erasmus Universiteit Rotterdam : s.n., 2009.

40. **Gromicho, van Hoorn, Kok and Schutten.** Restricted dynamic programming: a flexible framework for solving realistic VRPs. *Computers & Operations Research.* 2009, Vol. 39, 5, pp. 902-909.

41. **Hosny.** *Investigating Heuristic and Meta-Heuristic Algorithms for Solving Pickup and Delivery Problems.* Ph.D. thesis, Cardiff University : s.n., 2010.

42. **Hosny and Mumford.** *New Solution Construction Heuristics for the Multiple Vehicle Pickup and Delivery Problem with Time Windows.* Cardiff University : s.n., 2009.

APPENDIX B: LIST OF TABLES AND FIGURES

[CONFIDENTIAL]

APPENDIX C: INTERNSHIP MOSCOW-LIST

At the start of the internship an action plan was written in agreement with ORTEC. Next to a planning, this consisted of a MoSCoW-list regarding the functionalities of the proposed algorithm, which is shown below. Behind each item on the list is noted whether or not the functionality was delivered, or if it can easily be implemented.

MUST

- Generate a nearly optimal set of tours given any set of movements, which satisfy working conditions regarding maximum driving time per day. (**Delivered**)
- Generate acceptable tours for "smaller" movement-sets (<100) within a few minutes. The result should be a local optimum and therefore not easily improvable by the user by optical inspection. (**Delivered**)

SHOULD

- Generate acceptable tours for "large" movement-sets (>1000) within an hour. (**Delivered**)
- Able to deal with situations which allow for different restrictions and costs of repositioning (repositioning both in the middle and at the end of a tour) (**Delivered**)

COULD

- Optimize movements not on the critical time paths, where a delay would lead to better tours. (In Greenfield, the difference between departure time and the critical departure time is calculated, so this slack is available.) This feature has significant added value. (**NOT delivered**)

- Generate tours for "large" movement-sets (>1000) within minutes. (**Delivered**)
- Programmed in C# (or AIMMS). (**Delivered**)
- Generate tour templates instead of regular tours using template preference parameters. (**NOT delivered**)
- Try to fit movements "in the middle" of given tours. These 'old' tours are part of the input then. (For example, if there is an old tour with a repositioning from B to C at a certain time and we have that same movement in our input, the algorithm could show that this movement will fit. The same could apply to tours with a lot of waiting time in the middle of their tour.) **(NOT delivered, but can easily be implemented**)

WONT (BUT WOULD LIKE)

- Optimize tours by allowing multiple vehicle types in a tour, or changing vehicle types of movements as long as capacity and time constraints are met. (**NOT delivered, but can be implemented**)
- Allocate drivers to tours satisfying driver preferences, working conditions and legislation regarding the week or longer periods. (**NOT delivered**)
- Work with different sets of working/driving legislation depending on, for example, the starting location of the tour. **(NOT delivered, but can be implemented)**
- Do any optimization regarding movements (instead it will only do optimization regarding tour generation). (**NOT delivered**)