Chapter 2

2 PROBLEM BACKGROUND: VRP WITH MULTIPLE CONSTRAINTS

2.1. The Vehicle Routing Problem

Logistics can be defined as the provision of goods and services from a supply point to various demand points. The transportation of raw materials from the suppliers to the factory, from the factory to the depots, and the distribution to customers can be described as a complete logistic system. With an effective logistic system, cost can be reduced due to less penalties for late delivery, lowered trucking cost, shorter distances and effective use of capacity of the vehicle. One of the most significant measures of a logistic system is effective vehicle routing. Optimising of routes is the basis of vehicle routing problems.

The VRP originated from the Travelling Salesmen Problem (TSP). According to Winston [53] (p. 519) the TSP can be define as a problem where a salesperson must visit each of ten cities once before returning to his home. The cities need to be selected to minimise the total distance the salesmen travels.

According to Barbarosoglu et al. [3] (p. 256) the VRP can be described as the problem of designing optimal delivery or collection of routes from one or several depots to a number of customers subject to side constraints. Thus, the basic VRP can be described as vehicles that depart from the depot, visit one or more customers and return to the depot.

The VRP has a finite number of feasible solutions. The VRP solution space increase exponentially as the number of customers increases. Thus the VRP is known as a non-polynomial hard (NP-hard) problem.

The basic VRP is today no more than a classical problem. The advance of science has prompted the industry to ask for more real life solutions The basic VRP is given by a set of identical vehicles, a depot, a set of customers to be visited and a directed network connecting the depot and customers. Let us assume there are K vehicles, V = {0,1,2,3,...K-1}, and N+1 customers, C = {0,1,2,3,...N}. We denote the depot as customer 0, or C_0. Each arc in the network corresponds to a connection between two nodes. A route is defined as starting from the depot, going through a number of customers and ending at the depot. A cost c_{ij} and a travel time t_{ij} are associated with each arc of the network.

The problem is to find tours for the vehicles in such a way that:

- The objective function is minimized. The objective function can be the total travel distance, the number of vehicles used, or any cost related function.

Several constraints must be applied on the **basic** VRP:

- Only one vehicle handles the deliveries for a given customer. We will not split deliveries across multiple vehicles. A customer can only be visited once a day.

- The number of vehicles is equal to the number of routes, meaning that a vehicle can only complete one route per day.

The VRP has a finite number of feasible solutions. The VRP solution space increase exponentially as the number of customers increases. Thus the VRP is known as a non-polynomial hard (NP-hard) problem.

The basic VRP is today no more than a classical problem. The advance of science has prompted the industry to ask for more real life solutions The basic VRP is given by a set of identical vehicles, a depot, a set of customers to be visited and a directed network connecting the depot and customers. Let us assume there are K vehicles, V = {0,1,2,3,…K-1}, and N+1 customers, C = {0,1,2,3,…N}. We denote the depot as customer 0, or C_0. Each arc in the network corresponds to a connection between two nodes. A route is defined as starting from the depot, going through a number of customers and ending at the depot. A cost c_{ij} and a travel time t_{ij} are associated with each arc of the network.

The problem is to find tours for the vehicles in such a way that:

- The objective function is minimized. The objective function can be the total travel distance, the number of vehicles used, or any cost related function.

Several constraints must be applied on the **basic** VRP:

- Only one vehicle handles the deliveries for a given customer. We will not split deliveries across multiple vehicles. A customer can only be visited once a day.

- The number of vehicles is equal to the number of routes, meaning that a vehicle can only complete one route per day.

- The demand of the customers on every route is known with certainty. The demand of the customers in total on one route cannot exceed the capacity of the specific vehicle that will cover that route.

- The travelling distance between customer *i* and *j* are the same as the travel distance between *j* and *i*.

- The vehicles have the same capacity with the same fixed and variable cost, thus a homogeneous fleet are assumed.

- The vehicles must complete their route within a maximum length of time, usually the time the depot is open.

- The vehicle returns to the depot at the end of the route.

The VRP can be formulated as follows:

- A set of identical vehicles V

- A special node called the depot,

- A set of customers C to be visited

- A directed network connecting the depot and the customers

Let us assume there are K vehicles, $V = \{0, 1, 2, \ldots, K - 1\}$, and $N + 1$ customers, $C = \{0, 1, 2, \ldots, N\}$.

- For simplicity, we denote the depot as customer 0.

- Each arc in the network corresponds to a connection between two nodes.

- A route is defined as starting from the depot, going to any number of customers and ending at the depot.

- The number of routes in the traffic network is equal to the number of vehicles used, K. Therefore, exactly K directed arcs leave the depot and K arcs return to the depot.

- A cost c_{ij} and a travel time t_{ij} are associated with each arc of the network.

- Every customer in the network must be visited only once by one of the vehicles.

- Since each vehicle has a limited capacity q_k, and each customer has a varying demand m_i, q_k must be greater than or equal to the summation of all demands on the route travelled by vehicle k.

- Vehicles are also supposed to complete their individual routes within a total route time, which is essentially the time window of the depot.

There are two types of decision variables in a VRP.

- The decision variable $x_{ijk,}(i,j=0,1,2..N;k=0,1,2..K;i\neq j)$ is 1 if vehicle k travels from node i to node j, and 0 otherwise.

- The decision variable t_i denotes the time a vehicle starts service at node i. The triangular inequality, i.e. $c_{ij}<c_{ih}+c_{hj}$ and $t_{ij}\leq t_{ih}+t_{hj}\forall h,i,j\in N$ need not apply.

The objective is to design a set of cost-minimizing routes that service all the customers while all the constraints stated above are satisfied. The model can be mathematically stated as follows:

Notation:

K = total number of vehicles.

N = total number of customers.

c_i = customer i, where i = 1, 2, … ,N.

c_0 = the depot.

c_{ij} = cost incurred on arc from node i to j.

t_{ij} = travel time between node i and j.

m_i = demand at node i.

q_k = capacity of vehicle k.

e_i = open time at node i.

l_i = close time at node i

t_i = arrival time at node i.

f_i = service time at node i.

r_k = maximum route time allowed for vehicle k.

p_i = polar coordinate angle of customer i, i = 1, 2, ..., N.

R_k = vehicle route k, k = 1, 2, ..., K.

O_k = total overload for vehicle k, k = 1, 2, ..., K.

T_k = total tardiness for vehicle *k*, *k* = 1, 2, ..., K.

D_k = total travel distance for vehicle *k*, *k* = 1, 2, ..., K.

W_k = total travel time for vehicle *k*, *k* = 1, 2, ..., K.

$C(R_k)$ = cost of the route R_k based on a cost function.

$C(S)$ = sum total cost of individual routes $C(R_k)$.

α = weight factor for the total distance travelled by a vehicle.

β = weight factor for the latest arrival time of a customer.

γ = weight factor for the difference in polar coordinate angles.

φ = weight factor for the travel total time of a vehicle.

η = penalty weight factor for an overloaded vehicle.

κ = penalty weight factor for the total tardy time in a vehicle route.

Principle decision variable: $x_{ijk} = \{0,1\}$: 0 if there is no arc between node *i* and *j* and 1 otherwise.

$$Min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=0}^{K-1} c_{ij} x_{ijk} \tag{1}$$

Subject to:

$$\sum_{k=0}^{K-1} \sum_{j=1}^{N} x_{ijk} = K \ for \ i = 0 \tag{2}$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{jik} \le 1 \quad for\ i = 0; k \in [0, K-1] \tag{3}$$

$$\sum_{k=0}^{K-1} \sum_{j=1}^{N} x_{ijk} = 1 \quad for\ i = 1,2..N \tag{4}$$

$$\sum_{i=0, i \ne h}^{N} x_{ihk} - \sum_{j=1, j \ne h}^{N} x_{hjk} = 0 \quad \forall h \in [1, N]; k \in [0, K-1] \tag{5}$$

$$u_i - u_j + N x_{ij} \le N - 1 \quad for\ i \in [1, N]; j \in [1, N]; i \ne j \tag{6}$$

$$\sum_{i=0}^{N} m_i \sum_{j=0, j \ne i}^{N} x_{ijk} \le q_k \quad \forall k \in [0, K-1] \tag{7}$$

$$\sum_{i=0}^{N} \sum_{j=0, j \ne i}^{N} x_{ijk} (t_{ij} + f_i + w_i) \le r_k \quad \forall k \in [0, K-1] \tag{8}$$

- The objective function of the problem is given in (1).
- Constraint (2) specifies that there are exactly K routes going out of the depot.
- The third constraint (3) makes sure that each route leaves the depot and return to the depot
- Constraints (4) and (5) make sure exactly one vehicle goes to and leaves a customer.
- Constraint (6) ensures that there are no sub-tours in the solution. A sub-tour is a route that does not pass through the depot.

- (7) is the capacity constraint.

- Maximum travel time for each vehicle is assured in Eq. (8).

The model described in this section is a standard mathematical model for a basic VRP problem. When additional constraints are needed, they must be added to the existing constraints in the model or some of the existing constraints must be relaxed.

The industry requires additional constraints on the basic VRP. Additional constraints that we will address include:

- The limitation of the length, duration or cost of each individual tour. This restricts a route for running too long, which can result in overtime costs, insufficient fuel, etc.

- The addition of a service time for each customer. The volume of the stock to be delivered can have an influence on the service time at a customer. The delivery time will have an influence on the total route time and must be taken into account.

- The addition of time windows during which the customers have to be visited. The problem we will discuss is the use of multiple time windows, i.e. the customer can specify more than one time period available for delivery.

- The vehicle can return to the depot and have enough time for another route before the maximum allowed time is up. This will allow double scheduling, which will result in a cost saving, as the second route utilize the same vehicle and reduce the number of vehicles required to service all the customers.

- The travel time can vary between customers depending on the time of day. This implies peak and off-peak travel times.

- The fleet is not necessarily homogeneous, i.e. vehicles can differ in capacity and cost. This might result in a good solution to use the vehicles with a large capacity to pick up customers that is far away form the depot.

- A vehicle can have a specified available time. This allows for certain vehicles to be out in the field longer to cater for long routes. The implementation will add time window constraints to a vehicle.

We need to redefine the mathematical model for our problem. We will make use of the base model with the following changes:

- Constraint (2) is now invalid and will be replace by

$$\sum_{j=1}^{N} x_{ijk} \le p_k \quad for\ i = 0; k \in [0, K-1] \tag{2}$$

where p_k is the maximum number of routes allowed for vehicle k.

The number of routes going out of the depot for a specific vehicle are constrained to a maximum of p_k , which implies that a vehicle can now have multiple routes done in a day.

- We impose time windows at a stop

$$t_0 = 0 \tag{9}$$

$$t_i + x_{ijk}(t_{ij} + f_i + w_i) \le t_j \quad i, j \in [1, N]; i \ne j; k \in [0, K-1] \tag{10}$$

$$e_i \le t_i \le l_i \tag{11}$$

- We redefine the service time at each stop as

  f_i = Fixed Time + (Variable Time * m_i)

- We also redefine the meaning of travel time

  t_{ij} = Travel Time at ($t_i + f_i + w_i$)

  which calculates the travel time from i to j depending on the departure time at i.

- We just make a note that q_k is not necessarily the same for each vehicle.

- The monetary cost of a route can be calculated as follows

  $$C(R_{ki}) = (F_k / \sum_{j=1}^{N} x_{ijk}) + (D_k * V_k) \quad for\ i = 0; k \in [0, K-1]$$

  where the first term is the fixed cost of the vehicle divided into the number of routes and the second term is the distance of the route multiplied by the running cost of the vehicle.

2.2. Meta Heuristics

The implementation of an algorithm that can efficiently and in reasonable time solves the aforementioned problem has not been successfully implemented before. To embark on a journey to find a sufficient algorithm requires investigation of existing problems and solutions as well as inventing new methods. Several papers have been presented that solve the VRP with additional side constraints. They mainly focus on solving the basic VRP with one or two additional side constraints. Some of the most popular problems include the VRP with time windows and the VRP with pickup and delivery.

Heuristic methods play an important role in solving problems with this complexity. Most solutions include a heuristic method, or a hybrid of heuristic methods at the heart of the solution. In the next section, we will discuss some of the more popular heuristic methods.

Meta-heuristics, or global optimization heuristics, have a common feature: they guide a subordinate heuristic in accordance with a concept derived from artificial intelligence, biology, nature or physics to improve their performance.

Meta-heuristics succeed in leaving the local optimum by temporarily accepting exchanges that decrease the objective function value. Meta-heuristics use information of the problem environment and the nature of the objective function to direct the search process to regions that promise better solutions. It is possible that the meta-heuristic will return to the local optimum without finding a better solution. This is called cycling and can be avoided by adjusting the heuristic's settings to allow more degrading moves for longer.

The concept of a heuristic being trapped at a local optimum can be demonstrated in Figure 2. If a heuristic finds a solution S, with objective function F(S), where S is close to point C, then it will only improve until it gets to local optimum C. No further improvements in the objective function will be achievable, because all moves will reduce the objective function. However, if a meta-heuristic finds a solution close to point B, degrading moves will be allowed that may direct the search to the global optimum, point A.

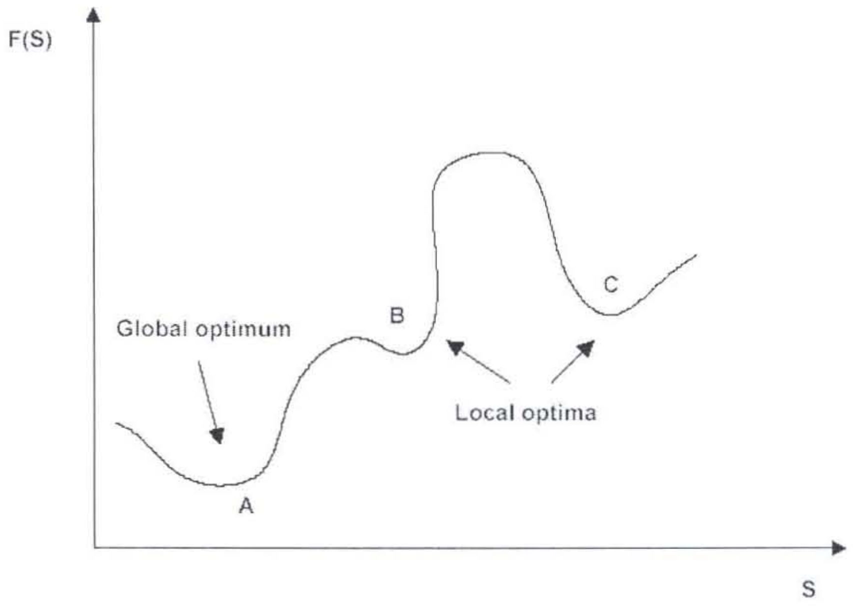

Figure 2: Global and Local Optima

Meta-heuristics will be successful on a given optimisation problem if it can provide a balance between the exploitation of the accumulated search experience and the exploration of the search space to identify regions with high quality solutions in a problem specific, near optimal way. The various meta-heuristics are classified according to the following criteria:

- **Trajectory methods vs. discontinuous methods**: Trajectory methods like SA and TS follow one single search trajectory corresponding to a closed walk on the neighbourhood graph. Discontinuous methods allows larger jump in the neighbourhood graph.

- **Populated-based vs. single-point search**: In single-point search only one single solution is manipulated at each iteration of the algorithm. TS and SA are single-point search methods. GA and ant colony algorithms is Population-based.

- **Memory usage vs. memoryless methods**: Meta-heuristics with memory are the TS, GA, SS and ant systems. According to Taillard et al. [48] these meta-heuristics with memory can be viewed as adaptive memory programming (AMP) heuristics. The term "memory" was used explicitly for TS, but other meta-heuristics use mechanisms that can be considered as memories. There are meta-heuristics that cannot be entered into the AMP methods, such as SA. However they may be included in the improvement procedure of AMP.

- **One vs. various neighbourhood structures**: SA and TS algorithms are based on one single neighbourhood structure. Other algorithms such as Iterated Local Search typically use at least two different neighbourhood structures.

- **Dynamic vs. static objective function**: Some algorithms modify the evaluation of the single search states during the run of the algorithm. In the use of a dynamic objective function penalties for the inclusion of certain solution attributes that modify the objective function are introduced. TS may be interpreted as using dynamic objective function, as some point in the search is forbidden, corresponding to infinitely high objective function values. The other algorithms use static objective functions.

Evaluation of heuristic methods consists of comparing criteria such as running time, quality of solution, ease of implementation, flexibility and robustness. For the purpose of our algorithm, flexibility is an important consideration. The algorithm should be able to handle changes in the data patterns, side constraints and objective function, as each client has his own specific requirement. We are not working on a predetermined set of data with a specified objective function. Working in such an environment make it possible to find a method that is effective for that specific environment by making use of the knowledge about the problem.

Because the heuristic methods are non deterministic, i.e. we cannot predict the result even if we apply the same algorithm on the same data with the same number of iterations, the algorithm should not perform poorly on any instance, as well as being able to produce a good solution each time it is applied to the same instance.

We will also try to validate the applicability of the method on our problem by discussing the design of the method is well as what we see as its advantages and disadvantages. With this approach we will filter out certain methods. Comparisons discussed in this paper are from existing papers, which mainly present the best results found for the method. Comparison is also made difficult because solutions were not all implemented on the same computer (running time), and have not all use the same number of iterations. Existing methods is also not designed for our specific problem and thus we cannot really compare methods outright to decide on a method to implement for our problem.

Using only the best results of a non-deterministic heuristic, as is often done in the literature, may create a false picture of its real performance. We considered average results based on multiple executions on each problem an important basis for the comparison of non-deterministic methods. Furthermore, it would also be important to report the worst-case performance.

Moreover, an algorithm should be able to produce good solutions every time it is applied to a given instance. This is to be highlighted since any heuristics are non-deterministic, and contain some random components such as randomly chosen parameter values. The output of separate executions of these non-deterministic methods on the same problem is in practice never the same. This makes it difficult to analyze and compare results.

Heuristic methods

An algorithm is said to be efficient when it runs in polynomial time, i.e., its running time is not longer than a polynomial function of the size of the problem. An algorithm is said to be effective if it produces high-quality solutions, preferably in less time than any efficient algorithm for the problem. The most preferred algorithms are both efficient and effective. If the algorithm produces the mathematically best solution it is called optimal (or exact) if it produces a good but not necessarily best solution it is called heuristic. A construction algorithm constructs a solution to a problem, whereas an improvement algorithm works on an existing solution to obtain better levels performance measures.

According to Laporte [33], heuristics belong to two broad classes: classical heuristics and modern heuristics (or metaheuristics). Classical heuristics can be broadly classified into three categories. Constructive heuristics gradually build a feasible solution while keeping an eye on solution cost, but do not contain an improvement phase per se. In two-phase heuristics, the problem is decomposed into its two natural components: clustering of vertices into feasible routes and actual route construction, with possible feedback loops between the two stages. Two-phase heuristics can be divided into two classes: cluster-first, route-second methods and route-first, cluster-second methods. In the first case, vertices are first organized into feasible clusters, and a vehicle route is constructed for each of them. In the second case, a tour is first built on all vertices and is then segmented into feasible vehicle routes. Finally, improvement methods attempt to upgrade any feasible solution by performing a sequence of edge or vertex exchanges within or between vehicle routes. The distinction between constructive and improvements methods is, however, often blurred since most constructive algorithms incorporate improvements steps at various stages.

As far as we are aware, six main types of metaheuristics have been applied to the VRP:

1) Simulated Annealing (SA),

2) Deterministic Annealing (DA),

3) Tabu Search (TS),

4) Genetic Algorithms (GA),

5) Ant Systems (AS), and

6) Neural Networks (NN).

The first three algorithms, SA, DA and TS, start from an initial solution x_t , and move at each iteration *t* from x_t to a solution x_t+1 in the neighborhood N(x_t) of x_t, until a stopping condition is satisfied. If *f(x)* denotes the cost of *x*, then *f*(x_t+*1*) is not necessarily less than *f*(x_t). As a result, care must be taken to avoid cycling. Put paragraph in bullets

GA examines at each step a population of solutions. Each population is derived from the preceding one by combining its best elements and discarding the worst. AS is a constructive approach in which several new solutions are created at each iteration using some of the information gathered at previous iterations. As was pointed out by Taillard et al. [48], TS, GA and AS are methods that record, as the search proceeds, information on solutions encountered and use it to obtain improved solution. NN is a learning mechanism that gradually adjusts a set of weights until an acceptable solution is reached. The rules governing the search differ in each case and these must also be tailored to the shape of the problem at hand. Also, a fair amount of creativity and experimentation is required.

The following sections discuss the most applicable methods.

2.2.1. Simulated Annealing (SA)

Simulated Annealing searches the solution space by simulating the annealing process in metallurgy (Qili et al [39]). The algorithm jumps to distant location in the search space initially. The size of the jumps reduces as time goes on or as the temperature "cools" down. Eventually the process will turn into local search descent method.

One of its characteristics is that for very high temperatures, each state has almost equal change to be the current state. At low temperatures only states with low energy have a high probability of being the current state. These probabilities are derived for a never ending executing of the metropolis loop. The advantages of the scheme is:

- SA can deal with arbitrary systems and cost functions.
- SA statistically guarantees finding an optimal solution
- SA is relatively easy to code, even for complex problems.
- SA generally gives a good solution.

However this original version from SA has some drawbacks

- Repeated annealing with a 1/log k schedule is very slow, especially if the cost function is expensive to compute, which will be the case for our problem.

- For problems where the energy landscape is smooth, or there are few local minima, SA is an overkill – simpler faster methods works better. But usually one does not know what the energy landscape is.

- Normal heuristic methods, which are problem specific or take advantage of extra information about the system, will often be better than general methods. But SA is often comparable to heuristics.

- The method cannot tell if it has found and optimal solution.

2.2.2. Tabu Search (TS)

The word Tabu (or taboo) comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga Island to indicate things that cannot be touched because they are sacred.[3] According to Webster's Dictionary, the word now also means "a prohibition imposed by social custom as a protective measure" or of something "banned as constituting a risk." These current more pragmatic senses of the word accord well with the theme of Tabu search. The risk to be avoided in this case is that of following a counter-productive course, including one, which may lead to entrapment without hope of escape. On the other hand, as in the broader social context where "protective prohibitions" are capable of being superseded when the occasion demands, the "taboos" of Tabu search are to be overruled when evidence of a preferred alternative becomes compelling.

[3] Source: Tabu Search Network [31]

Tabu Search (TS) is a local search metaheuristic introduced by Glover (1986). TS explores the solution space by moving at each iteration from a solution to the best solution in a subset of its neighbourhood N(s). Contrary to classical descent methods, the current solution may deteriorate from one iteration to the next. Thus, to avoid cycling, solutions possessing some attributes of recently explored solutions are temporarily declared Tabu or forbidden. The duration that an attribute remains Tabu is called its Tabu-tenure and it can vary over different intervals of time. The Tabu status can be overridden if certain conditions are met; this is called the aspiration criterion and it happens, for example, when a Tabu solution is better than any previously seen solution. Finally, various techniques are often employed to diversify or to intensify the search process.

The most important association with traditional usage, however, stems from the fact that taboos as normally conceived are transmitted by means of a social memory, which is subject to modification over time. This creates the fundamental link to the meaning of "taboo" in Tabu search. The forbidden elements of Tabu search receive their status by reliance on an evolving memory, which allows this status to shift according to time and circumstance.

TS is the only metaheuristic that has been explicitly developed with a memory. In a sense this method imitates the human being looking for a good solution of a combinatorial optimization problem. Glover proposed a number of strategies to guide the search and make it more efficient. TS is open for any strategy well adapted to the problem on which it is applied.

More particularly, Tabu search is based on the premise that problem solving, in order to qualify as intelligent, must incorporate adaptive memory and responsive exploration. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Since local choices are guided by information collected during the search, TS contrasts with memoryless designs that heavily rely on semi random processes that implement a form of sampling. Examples of memoryless methods include semi greedy heuristics and the prominent "genetic" and "annealing" approaches inspired by metaphors of physics and biology. Adaptive memory also contrasts with rigid memory designs typical of branch and bound strategies. (It can be argued that some types of evolutionary procedures that operate by combining solutions, such as genetic algorithms, embody a form of implicit memory. However, this form of memory is not sufficient to embrace many aspects of what we normally conceive to be a hallmark of 'intelligent' problem solving. Tabu search also has implicit memory features that offer opportunities for establishing more effective variants of evolutionary approaches.)

The emphasis on responsive exploration in Tabu search, whether in a deterministic or probabilistic implementation, derives from the supposition that a bad strategic choice can yield more information than a good random choice. In a system that uses memory, a bad choice based on strategy can provide useful clues about how the strategy may profitably be changed. (Even in a space with significant randomness a purposeful design can be more adept at uncovering the imprint of structure.)

Responsive exploration integrates the basic principles of intelligent search, i.e., exploiting good solution features while exploring new promising regions. Tabu search is concerned with finding new and more effective ways of taking advantage of the mechanisms associated with both adaptive memory and responsive exploration. The development of new designs and strategic mixes makes TS a fertile area for research and empirical study.

The main advantage of the basic version is its aggressiveness: the search converges toward the local optimum and examines the neighbourhood of this local optimum very quickly. However, it can easily get trapped in a sub-space containing only solutions of poor quality. To diversify the search and force it to visit solutions with different characteristics, one basic idea was to increase the number of forbidden components when performing local modifications to a solution. So, the discussion quickly turned around the optimum tabu list size, since the short list allows a thorough examination of the neighbourhood of a good solution while a long list facilitates the escape from a local optimum to explore new regions of the search space. The reactive Tabu search proposed by Battiti and Tecchiolli (1994) (in Bräysy [5], p. 4) was designed to automatically adapt the Tabu list size and avoid the fastidious task of explicitly managing the Tabu list.

The main difficulty with TS is thus to efficiently incorporate diversification and intensification mechanisms. The use of a memory that stores good solutions visited during the search and the design of a procedure to create provisional solutions from it is a way to achieve this goal. Indeed, solutions contained in memory during the initial search phase present different characteristics, thus leading to a diversified search. Later, solutions contained in memory are mostly representative of one or a few good regions of a solution space. The result is that the search gradually shifts from diversification to intensification.

2.2.3. Genetic Algorithms (GA)

The Genetic Algorithm (GA) is an adaptive heuristic search method based on population genetics. The basic concepts were developed by Holland (1975) (in Ombuki et al, [39], p.3), while the practicality of using the GA to solve complex problems was demonstrated in De Jong (1975) and Goldberg (1989) (in Bräysy and Gendreau, [8], p. 10).

GA evolves a population of individuals encoded as chromosomes by creating new generations of offspring through an iterative process until some convergence criteria are met. Such criteria might, for instance, refer to a maximum number of generations, or the convergence to a homogeneous population composed of similar individuals. The best chromosome generated is then decoded, providing the corresponding solution.

The creation of a new generation of individuals involves three major steps or phases: selection, recombination and mutation. The selection phase consist of randomly choosing two parent individuals from the population for mating purposes. The probability of selecting a population member is generally proportional to its fitness in order to emphasize genetic quality while maintaining genetic diversity. Here, fitness refers to a measure of profit, utility or goodness to be maximized while exploring the solution space. The recombination or reproduction process makes use of genes of selected parents to produce offspring that will form the next generation. As for mutation, it consists of randomly modifying some gene(s) of a single individual at a time to further explore the solution space and ensure, or preserve, genetic diversity. The occurrence of mutation is generally associated with a low probability. A new generation is created by repeating the selection, reproduction and mutation processes until all chromosomes in the new population replace those from the old one. A proper balance between genetic quality and diversity is therefore required within the population in order to support efficient search.

Although theoretical results that characterize the behaviour of the GA have been obtained for bit-string chromosomes, not all problems lend themselves easily to this representation. This is the case, in particular, for sequencing problems, such as the vehicle routing problem, where an integer representation is more often appropriate. Therefore, in most applications to VRPTW, the genetic operators are applied directly to solutions, represented as integer strings, thus avoiding coding issues. In most cases the authors use delimiters to separate customers served by different routes.

The genetic algorithm is very simple, yet it performs well on many different types of problems. There are many ways to modify the basic algorithm, and many parameters that can be \tweaked". Basically, if the objective function, the representation and the operators are all right, then variations on the genetic algorithm and its parameters will result in only minor improvements in the overall results.

For any GA, there are five important parameters that determine the performance of its application: representation of solution, initial population, selection, reproduction, and population improvements (Qili, [39], p. 72).

Advantages

- GA is very flexible with a lot of parameters to adjust for different needs;
- GA generally explores a larger neighbour hood than local search heuristics;
- With proper parameters, GA practices a global optimization that bypasses the local optimum problem;
- Given enough time, GA usually gives good solution.

Disadvantages

- GA is one of the slowest algorithms in finding the optimum;
- It has no termination criteria other than a number of generations;

- GA can be trapped in a local plateau, as the movement of the population is limited by the crossover operations, if that plateau is big and at enough.

Coding a solution with a binary vector is not natural and can significantly impact the performance. Hence, binary coding was replaced by a more natural representation of solutions. The classical cross over operators does not correspond to logical operations on solutions. Furthermore, the use of other representations and binary vectors naturally led to the design of specialized operators, well adapted to the solution representation and capable of generating new feasible solutions. GA can easily identify different solution sub spaces with good characteristics, but they lack the "killer instinct" that would allow them to intensify the search into these areas. To alleviate this weakness, the mutation operation was replaced by repair procedures and local search.

2.2.4. Ant Systems (AS)

The idea of imitating the behaviour of ants to find solutions to combinatorial optimization problems was initiated by Colorni, Dorigo and Maniezzo (in Bullnheimer et al, [12], p. 1). The metaphor comes from the way ants search for food and find a way back to the nest. Initially ants explore the area surrounding their nest in a random manner. As soon as an ant finds a source of food, it evaluates the interest of the source (quantity and quality) and carries some of food to the nest. During the return trip, the ant leaves on the ground a chemical pheromone trail whose quantity depends on the quality of the source. The role of this pheromone trail is to guide other ants toward the source. After a while, the path to a good source of food will be indicated by a large pheromone trial, as a trial grows with number of ants that reach the source. Since source is that are close to the nest are visited more frequently than those that are far way, pheromone trials leading to the nearest sources grow faster. The final result of this process is that ants are able to optimize their work.

The transposition of this food searching the area into an algorithm framework for solving combinatorial optimization problems is octane through an analogy between:

- the search area of the real ants and the set of feasible solutions to the combinatorial problem;
- the amount of food associated with the source and the objective function;
- the pheromone trial and an adaptive memory.

The most important component of an ant system is the management of the pheromone trials. In a standard ant system, pheromone trials are used in conjunction with the objective function to guide the construction of new solutions. Once a solution has been produced, a standard ant system updates the pheromone trials as follows: first all trials are a weakened to simulate the evaporation of pheromone; then, pheromone trials that correspond to components that were used to construct the resulting solution are reinforced, taking into consideration the quality of this solution.

Based on the previous general scheme different AS implementations have been proposed where pheromone updating is performed in different ways. Different ways of modifying pheromone values generate different types of search mechanisms. Recently it has been shown that AS based algorithms are being powerful in combination with local search procedures. In these situations pheromone information is used to produce solutions (diversification phase) that are optimized by a local search (intensification phase). Optimize solutions are then used to update pheromone information and new solutions are successively generated by the ants.

Like GA, early implementations of the ant system converged too slowly toward high-quality solutions. Therefore, intensification mechanisms were gradually introduced. The most recent implementations incorporate local search mechanisms to improve the solutions produced by the ants.

2.3. Existing Methods and Implementations

The vehicle routing problem has many variants that have been attempted by many people with different criteria and different methods. The question arises on how could another study on the problem be feasible. In the following section we will discuss some of the existing implementations of the VRP. This section will discuss some implementations which will enable us to derive methods already tested, or show incompleteness in there implementation for our use. It must be noted that certain methods were not considered as feasible because it was deemed too slow. We can reconsider these methods because of the improvement in computing power in recent years.

Table 1 is a present state of work done of the study to derive a feasible solution for our problem. The model indicates the model implemented by the author that is of interest to us. The following section will discuss the methods in detail.

Present State			
Author	*Year*	*Problem*	*Model*
Amberg, Domschke and Voβ	2000	M-CARP	Cluster First Route Second
Taillard, Laporte and Gendreau	1995	VRPM	Tabu Search, generating and combining promising solutions.
Lau and Liang	2000	PDPTW	Two-staged heuristic, Construction and Tabu Search, working with job pairs
Salhi et al	1992	VFM	Unlimited vehicles, best vehicle selection
Taillard	1996	VRPHE	Column generation method
De Backer and Furnon	1997	VRPTW	Constraint programming, routestop has next stop
Xu and Kelly	1999	VRPTW	TS with independent tabu tenure per operation
Ombuki, Nakamura and Osamu	2002	VRPTW	Hybrid GA and TS
Van Schalkwyk	2002	VRPTW	Time Window Compatibility, selective neigbour list

Table 1: Present State

2.3.1. Multiple depot

Although we do not focus on a multiple depot implementation of the VRP, it is important to understand the methods available for solving this problem. In our problem we make use of the cluster first route second (CFRS) method. CFRS methods are more suitable for node routing problems. The clustering method is left to the specific client, i.e. the nodes will be clustered with an algorithm selected by the client before we receive the data.

In the capacitated arc routing problem with multiple centres the objective is to find routes starting from the given depots or centres such that each required arc is served, capacity and usually additional constraints are satisfied and total travel cost is minimised. The paper of Amberg et al, [1] consider a heuristic transformation of the multiple centre arc routing problems into a multiple centre capacitated minimum spanning tree problem with arc constraints. Arc routing applications referred to problems where the distribution or collection of goods is bound up with traversing a distance such as mail delivery, snow removal, garbage disposal, street sweeping and police patrols. Thus, the customers are modelled as arc or edges, whereas in node routing problems the customers correspond with the nodes as, e.g. in the travelling salesman problem. The well-known Chinese postman problem (CPP) is the basic arc routing problem was named after the Chinese scientist Mei-Ko Kwan (1962) who was the first to publish on this problem.

Introducing additional constraints even in undirected or directed graphs usually yields NP-hard problems such as the capacitated Chinese postman problem, where the capacity of the postman is restricted, or the rural postman problem (RPP) where the set of required arcs (i.e. those arcs which need serving) need not be connected and has to be linked using non-required arcs. With respect to developing solution methods, it is important to note that capacitated arc routing problems consist of two interdependent sub problems: The assignment problem which forms subsets or clusters of required arcs served by the same vehicle and the sequencing or routing problem which determines the sequence of serving the arcs.