Journal of King Saud University (Science) (2010) **22**, 123–131

King Saud University
Journal of King Saud University (Science)
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods

Nasser A. El-Sherbeny

Mathematics Department, Faculty of Science, Al-Azhar University, Nasr City 11884, Cairo, Egypt

Received 1 July 2009; accepted 27 March 2010
Available online 7 April 2010

KEYWORDS
VRPTW;
Exact methods;
Heuristic;
Metaheuristic

Abstract In this paper, we present a review with some limited of exacts, heuristics and metaheuristics methods for the vehicle routing problem time windows (VRPTW). Over the past 20 years vehicle routing problem with time windows has been an area of research that has attracted many researchers. In this period a number of papers have been published on the exact, heuristics and metaheursitics methods of the routing problem with time windows. This problem has model character in many branches of mathematics, computer science, and operations research. Metaheuristics support managers in decision-making with robust that provide high-quality solutions to important applications in business, engineering, economic and science in reasonable time horizons.

1. Introduction

The vehicle routing problem with time windows is an extension of the well-known vehicle routing problem (Crainic and Laporte, 2000; Toth and Vigo, 2002). The problems are often more simple than real-life problems. But even though a number of real-life constraints are left out the research models typically model the basic properties and thereby provide the core results used in the analysis and implementation of systems in real-life problems. One of the best-known routing problem is at the same times the simplest one namely the travelling salesman problem (TSP). A number of cities have to be visited by a salesman who must return to the same city where he started. The route has to be constructed in order to minimize the distance to be travelled. The vehicle routing problem (VRP) is the m-TSP where a demand is associated with each city, and vehicle has a certain capacity.

If we add a time window to each customer we get the vehicle routing problem with time windows. In addition to the capacity constraint, a vehicle now has to visit a customer within a certain time frame. The vehicle may arrive before the time window opens but the customer cannot be serviced until the time windows open. It is not allowed to arrive after the time window has closed. According to Lenstra and Rinnooy (1981) the VRP and VRPTW belong to the class of the NP-hard combinatorial optimization problems.

Many decision problems in business and economics, notably including those in manufacturing, location, routing and scheduling may be formulated as optimization problems. Typically these problems are too difficult to be solved exactly

E-mail address: nasserelsherbeny@yahoo.com

1018-3647 © 2010 King Saud University. All rights reserved. Peer-review under responsibility of King Saud University.
doi:10.1016/j.jksus.2010.03.002

Production and hosting by Elsevier

within a reasonable amount of time and heuristics become the methods of choice. In cases where simply obtaining a feasible solution is not satisfactory, but where the quality of solution is critical, it becomes important to investigate efficient procedures to obtain the best possible solutions within time limits deemed practical.

Often the number of customers combined with the complexity of real-life data does not permit solving the problem exactly. In these situations one can apply approximation algorithms or heuristics. Both approximation algorithm and heuristics produce are feasible but not necessarily optimal solution. Whereas a worst-case deviation is known for approximation algorithms nothing a priori is known for heuristics, but typically they can tuned to perform very well. The be success of metaheuristics, simulated annealing, tabu search, genetic algorithms on hard single and multi-objective vehicle routing problem with time windows is well recognized today (Tuyttens et al., 2004).

In El-Sherbeny (2001), the vehicle routing problem with time windows has been solved with a multi-objective simulated annealing (MOSA) method. Three categories of objectives are discussed: concerning the vehicle used (number of vehicles, number of covered/uncovered vehicles), concerning time (total duration of the routes, the homogeneity of the duration of the routes, working time not used, total waiting times due to time-windows constraints), and concerning the flexible duration of the routes (longer duration of the routes is preference).

In this paper we present a review with some limited of exacts, heuristics and metaheuristics methods for the vehicle routing problem with time windows. The paper is organized as follows. In Section 2, we provide the basic definition and some generalizations of the vehicle routing problem with time windows. In Section 3, we briefly discuss some issues related to the exact methods for the VRPTW. In Section 4, the non-exact methods for the VRPTW will be reviewed. In Section 5, we review the modified of the routing problem with the artificial intelligence. In Section 6, cover of some methods of metaheuristics, simulated annealing, tabu search and genetic algorithms.

2. The vehicle routing problem with time windows

The VRPTW is an important generalization of the VRP and a basic distribution management problem that can be model many real-world problems and which are consists of designing a set of minimum cost routes, originating and terminating at a central depot, for a fleet of vehicles which services a set of customers with known demands. The customers must be assigned exactly once to vehicles such that the vehicle capacities are not exceeded. The service at a customer must begin within the time window defined by earliest time and the latest time when the customer permits the start of service. Some of the more useful applications of the VRPTW include bank deliveries, postal deliveries, industrial refuse collection, national franchise restaurant deliveries, and school bus routing and security patrol services.

2.1. Mathematical model of the VRPTW

The VRPTW is given by a fleet of homogeneous vehicles denoted by V, a set of customers C and a directed graph $G = (V, C)$. The graph consists of $|C| + 2$ vertices, where the customers are denoted 1, 2, …, n and the depot is represented by the vertex 0 (the driving-out depot) and $n + 1$ (the returning depot).

The VRPTW has multiple objectives in that the goal is to minimize not only the number of vehicles required, but also the total travel time, waiting time, and total travel distance incurred by the fleet of vehicles. The set of arcs denoted by A represents connections between the depot and the customers and among the customers. No arc terminates in vertex 0, and no arc originates from vertex $n + 1$. With each arc (i, j), where $i \neq j$, we associate a cost c_{ij} and a time t_{ij}, which may include service time at customer i.

Each vehicle has a capacity q and each customer i a demand d_i. Each customer i has a time window $[a_i, b_i]$. A vehicle must arrive at the customer before b_i. It can arrive before a_i but the customer will not be serviced before. The depot also has a time window $[a_0, b_0]$. Vehicles may not leave the depot before a_0 and must be back before or at time b_{n+1}.

It is assumed that q, a_i, b_i, d_i, c_{ij} are non-negative integers, while the t_{ij} are assumed to be positive integers. It is assumed that the triangular inequality is satisfied for both the c_{ij} and the t_{ij}. The model contains two sets of decision variables x_{ijk} and s_{ik}. For each arc (i, j), where $i \neq j$, $i \neq n+1$, $j \neq 0$, and each vehicle k we define $x_{ijk} = 1$, if and only if the optimal solution, arc (i, j) is traversed by vehicle k and equal 0, otherwise.

The decision variable s_{ik} is defined for each vertex i and each vehicle k and denotes the time vehicle k starts to service customer i. In case the given vehicle k does not service customer i, s_{ik} does not mean anything. We assume $a_0 = 0$ and therefore $s_{0k} = 0$, for all k. We want to design a set of minimal cost routes, one for each vehicle, such that each customer is visited exactly once, are every route originates at vertex 0 and ends at vertex $n + 1$, and the time windows and capacity constraints observed. We can state the VRPTW mathematically as follows:

$$\min \sum_{k\in V}\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ijk} \tag{2.1}$$

subject to,

$$\sum_{k\in V}\sum_{j\in N} x_{ijk} = 1 \quad \forall i \in C \tag{2.2}$$

$$\sum_{i\in C} d_i \sum_{j\in N} x_{ijk} \leqslant q \quad \forall k \in V \tag{2.3}$$

$$\sum_{j\in N} x_{0jk} = 1, \quad \forall k \in V \tag{2.4}$$

$$\sum_{i\in N} x_{ihk} - \sum_{j\in N} x_{hjk} = 0, \quad \forall h \in C, \quad \forall k \in V \tag{2.5}$$

$$\sum_{i\in N} x_{i,n+1,k} = 1, \quad \forall k \in V \tag{2.6}$$

$$s_{ik} + t_{ij} - K(1 - x_{ijk}) \leqslant s_{jk}, \quad \forall i,j \in N, \quad \forall k \in V \tag{2.7}$$

$$a_i \leqslant s_{ik} \leqslant b_i, \forall i \in N, \quad \forall k \in V \tag{2.8}$$

$$x_{ijk} \in \{0,1\}, \forall i,j \in N, \quad \forall k \in V \tag{2.9}$$

The constraints (2.2) states that each customer is visited exactly ones, and (2.3) means that no vehicle is loaded with more than its capacity allows it to. The next three equations (2.4), (2.5), and (2.6) ensures that each vehicle leaves the depot 0, after arriving at a customer the vehicle leaves again, and finally arrives at the depot $n + 1$. The inequalities (2.7) states that a vehicle k cannot arrive at j before $s_{ik} + t_{ij}$ if it is travelling from

i to j. Here K is a large scalar. Finally constraints (2.8) ensure that time windows are observed, and (2.9) are the integrality constraints. Note that an unused vehicle is modeled by driving the empty route $(0, n+1)$.

2.2. Generalizations of the VRPTW

A number of additional constraints or properties of more complex routing problems can be modelled using the framework just developed. In this section we will briefly discuss how to allow non-identical vehicles; work with more than one depot, multi-compartment vehicles, the use of multiple time windows or soft time windows, and pick-up and delivery.

2.2.1. Non-identical vehicles

Vehicles can be non-identical in several ways. The typical way a heterogeneous fleet of vehicles are characterized by the capacity, but it could also be different due to different arc costs for each vehicle, different types (covering and non-covering) vehicles, different travel times, time windows or other characteristics (Tuyttens et al., 2004; El-Sherbeny, 2001; El-Sherbeny and Tuyttens, 2001).

2.2.2. Multiple depots

In real-life problems there might be more than one depot. The VRPTW can be used to model situations where multiple depots exist. The customers are serviced by several depots, each depot having their own fleet of vehicles (Lie and Simchi-Levi, 1990; Desaulniers et al., 1997). Usually one assumes that vehicles must return to the same depot as they started form. In a relaxed form we only demand that the number of vehicles arriving at the depot equals the number of vehicles leaving it. In a further relaxation seldom used there are no constraints on which depots the vehicles should return to.

2.2.3. Multiple time windows

In the VRPTW each customers has one time window where the service must take place. Allowing customers to have multiple and disjoint time windows. A vehicle that arrives between two time windows must wait until the beginning of the next time window.

2.2.4. Multiple compartments

If the vehicles have two or more compartments the routing problem is known as a Multiple Compartment VRPTW (MCVRPTW). The use of multiple compartments is relevant, when the vehicles transports several commodities which must remain separated during transportation. An example is the distribution of oil products to service stations where the tank trucks are divided into a number of compartments in order to transport the different kinds of petrol.

In the same way we can extend the VRPTW model in handle multi-dimension capacity constraints. In VRPTW the capacity is one dimensional. This dimension can be the weight, volume, value or pieces. However, the capacity constraints can be multi-dimensional, for instance weight and volume in order to be able to handle cases where many large boxes do not violate the weight constraint but their volume is to large for one vehicle, or the other way around.

2.2.5. Soft time windows

Sometimes a cost $p(s_i)$ depending on the service time s_i of a customer i is introduced in order to penalize arrivals that are feasible but undesirable. The time window is then said to be soft, if the cost is non-decreasing with time, i.e., $s_i^1 \leq s_i^2 \Rightarrow p(s_i^1) \leq p(s_i^2)$. The dominance criterion remains valid and the soft time windows can be incorporated in our VRPTW (Koskosidis et al., 1992). The case where the penalty function $p()$ is a general function is not efficiently solvable.

2.2.6. Pick-up and delivery

In VRPTW we either pick-up goods at the customers or goods are delivered to the customers. In pick-up and delivery the vehicles, can perform both tasks. In the simple Backhauling version (VRPBTW) of pick-up and delivery the vehicles must be completely empty before the pick-up phase starts (Dumes et al., 1991).

In this simple case the customers can be divided into two classes of customers: a set of delivery customers and a set of pick-up customers. Now by removing all arcs from pick-up customers to delivery customers we ensure that it is not possible to service a delivery customer after a pick-up customer.

3. Exact methods for the VRPTW

The exact methods for the VRPTW can be classified into three categories: lagrange relaxation-based methods, column generation and dynamic programming. Exact methods often perform very poorly (in some cases taking days or more to find moderately decent, let alone optimal, solutions even to fairly small instances).

3.1. Lagrange relaxation-based methods

There are a number of papers using slightly different approaches. There is variable splitting followed by lagrange relaxation, K-tree approach followed by lagrange relaxation (Fisher et al., 1997; Holland, 1975, and in Kohl and Madsen (1997) presented shortest path with side constraints approach followed by lagrange relaxation. The relaxes of the constraints ensuring that every customer is served exactly once, that is:

$$\sum_{k\in V}\sum_{j\in N} xijk = 1, \quad \forall i \in C$$

is relaxed and the objective function with the added penalty term then becomes

$$\min \sum_{k\in V}\sum_{i\in N}\sum_{j\in N}(cij - \lambda j)xijk + \sum_{j\in V}\lambda j$$

Here λ_j is the lagrange multiplier associated with the constraint that ensures that customer j is serviced.

In Fisher et al. (1997) are presents an algorithm for solving the VRPTW optimally where the problem is formulated as a K-tree problem with degree $2K$ on the depot. A K-tree for a graph containing $n+1$ vertices is a set of $n+K$ edges spanning the graph. Informally, the VRPTW could be described as finding a K-tree with degree $2K$ on the depot, degree 2 on the customers and subject to time and capacity constraints. A K-tree with degree $2K$ on the depot therefore becomes equal to K routes.

The problem is defined and all constraints except the constraints ensuring that at most one arc is joining customers i and j are then lagrangian relaxed. The problem is then solved with a minimum degree-constrained K-tree problem as subproblem and the lagrange multipliers are set using the sub-gradient approach.

3.2. Column generation

In Agarwal et al. (1989) is used for the first column generation to solve the VRP problem, while (Desrosiers et al., 1984) used the column generation approach to solve the m-TSP with time windows. In Desrosiers et al. (1984), the column generation approach is used for the first time for solving the VRPTW, and more effective version of the same model with addition of valid inequalities solves more instances to optimality in Desrosiers et al. (1992).

If a linear program contains too many variables to be solved explicitly, then we can initialize the linear program with a small subset of the variables and compute a solution of this reduced linear program. Afterwards, we check if the addition of one or more variables, currently not in the linear program, might improve the linear program solution. This check can be done by the computation of the reduced costs of the variables. In this case, a variable of negative reduced cost can improve the solution.

3.3. Dynamic programming

The dynamic programming approach for VRPTW is presented for the first time in Kolen et al. (1987) and Christofides and Beasley (1984) are uses the dynamic programming paradigm to solve the VRP.

The algorithm of Kohl and Madsen (1997) use branch-and-bound to achieve optimality. Each node α in the branch-and-bound tree corresponds to three sets: $F(\alpha)$ which the set is of fixed feasible routes starting and finishing at the depot, $P(\alpha)$ which is a partially build route starting at the depot and $C(\alpha)$ denotes the set of customers forbidden to be next on $P(\alpha)$.

Branching is done by selecting a customer i that is not forbidden, that is $iC(\alpha)$, and that does not appear on any route, that is $iF(\alpha) \cup P(\alpha)$. Branching decisions are taken on route-customer allocations. Then two branches are generated: one in which the partially build route $P(\alpha)$ is extended by i and one where i is forbidden as the next customer on the route that is added to $C(\alpha)$. Customer i is chosen as the customer the partial route $P(\alpha)$ was extended with in the calculation that lead to the lower bound of node α. At each branch-and-bound node dynamic programming is used to calculate a lower bound on all feasible solution defined by $F(\alpha)$, $P(\alpha)$ and $C(\alpha)$.

First we discuss the case of the root node ($F(\alpha) = \phi$, $C(\alpha) = \phi$ and $P(\alpha) =$ depot). Here we construct a directed graph with vertices $v(i, q, k)$ for $i = 0, 1, \ldots, n$; $q = 0, 1, \ldots, Q$ and $k = 0, 1, \ldots, m$, where n is the number of customers, m is the number of vehicles and Q is the sum of all customer demands q_i. Hence, associated with each branch-and-bound node is a set of routes.

A directed path from $v(0, 0, 0)$ to $v(i, q, k)$ in the graph corresponds to a set of k routes with a total load of q and with different last visited customers (each one in$\{1, 2, \ldots, i\}$). The arc lengths in the directed graph will be defined as the total length of the corresponding routes. The lower bound is then given by the minimum over $k = 1, 2, \ldots, m$ of the shortest paths lengths from $v(0, 0, 0)$ to $v(n, Q, k)$. Note that there are no constraints enforcing customers to be visited by any of the routes generated. Therefor the resulting minimum is a lower bound. Dynamically we try to extend a set of k routes with load q and last customers $\{1, 2, \ldots, i\}$ to last customers $\{1, 2, \ldots, i + 1\}$. Here there are two possibilities:

- Customer $i + 1$ is not included as endpoint of any routes. This results in an arc from $v(i, q, k)$ to $v(i + 1, q, k)$ with length 0. Note that a customer $i + 1$ that is not an endpoint might still be member of one of the other routes generated by the function $F(i, q)$ described below.
- Insert customer $i + 1$ as the last customer on a route of load q'. This generates an arc from $v(i, q, k)$ to $v(i + 1, q + q', k + 1)$ of length $F(i + 1, q')$ for each possible value of q'. Generally, $F(i, q)$ is defined as the minimum length of a feasible route with total load q and last customer i. The routes associated with a given vertex $v(i, q, k)$ are the routes given from the computation of $F(i, q)$ and the extension made using the arcs.

If we are at an arbitrary node in the branch-and-bound tree we distinguish between two cases: $P(\alpha) = \phi$ and $P(\alpha) \neq \phi$. If $P(\alpha) = \phi$ we just adjust the problem for the number of already generated routes $\hat{k}$, their load $\hat{q}$ and the set of customers already used in these routes $\hat{I}$.

In the case of $P(\alpha) \neq \phi$ exactly one of the routes in the lower bound is an extension of $P(\alpha)$. Now, let $F'(i, q)$ be the minimum length of such an extension ($F'(i, q)$ is calculated in the same way as $F(i, q)$). As before, the problem can be reduced according to $\hat{k}$, $\hat{q}$ and $\hat{I}$. The directed graph is now extended to contain vertices $v(i, q, k)$ and $v'(i, q, k)$ and the following arcs:

- Arcs of length 0 from $v(i, q, k)$ to $v(i + 1, q, k)$ and from $v'(i, q, k)$ to $v'(i + 1, q, k)$. These correspond to not using the customer $i + 1$ in the routes.
- Arcs of length $F(i + 1, q')$ from $v(i, q, k)$ to $v(i + 1, q + q', k + 1)$ and from $v'(i, q, k)$ to $v'(i + 1, q + q', k + 1)$ for each possible value of q'.

In Kolen et al. (1987) problems up to 15 customers are solved by this method.

4. Heuristic algorithms for the VRPTW

The non-exact algorithms for the VRPTW have been very active-far more active than that of exact algorithms. The field of approximation algorithms and heuristics one sometimes classifies an algorithm as sequential and parallel. In a sequential algorithm one route at a time is constructed, while a parallel algorithm may build more routes at the same time. This conflicts with the notion of a sequential algorithm and a parallel algorithm. We will therefore avoid the use of this classification of heuristics.

Heuristic algorithms that a set of routes from scratch are typically called route-building heuristics, while an algorithm that tries to produce an improved solution on the basis of an already available solution is denoted route-improving.

A heuristic is defined by Reeves (1995) as a technique which seek good (near-optimal) solutions at a reasonable computa-

tional cost without being able to guarantee optimality, to state how close to optimality a particular feasible solution is or, in some cases, even to guarantee feasibility. Often heuristics are problem-specific, so that a method which works for one problem cannot be used to solve a different one.

4.1. Route-building heuristics

The first paper on route-building heuristics for the VRPTW is (Baker and Schaffer, 1989). Their algorithm is an extension of the saving heuristic of the VRP (Clarke and Wright, 1964). The algorithm begins with all possible single-customer route (depot-*i*-depot). In every iteration we calculate which two routes can be combined with the maximum saving, where the saving between customers i and j are given by:

$$s_{ij} = t_{i0} + t_{0j} - Gt_{ij}, i \neq j, i, j = 1, 2, \ldots, n \quad (4.1)$$

where G is sometimes referred to as the route form factor. A time oriented nearest-neighbourhood algorithm is developed by defining the savings as a combination of distance, time and time until feasibility. A similar heuristic based on the savings algorithm is developed in Solomon (1987), but here the time aspect is not part of the savings function. Instead the arcs that can be used are limited by how large the waiting times get if they are used. Due to the existence of time windows we have to take account of route orientation. Additionally we have to check for violation of the time windows when two routes are combined. These heuristics have time complexity of $O(n^2 \log n^2)$.

Also Landeghem (1988) has presented a heuristic based on the saving heuristic. His bi-criteria heuristic uses the time windows in (4.1) in order to get a measurement of how good a link between customers is in terms of timing.

Another heuristic described in Solomon (1987) is a time-oriented, nearest-neighbourhood heuristic. Every route in this heuristic is started by finding the unrouted customer that is closest to the depot. The closeness relation tries to take both geographical and temporal closeness of the customers into account. At every subsequent iteration the customer closest to the last customer added to the route is considered for insertion to the end of the route presently generated. When the search fails a new route is started.

Assigning customers to clusters is done by using a technique introduced in Gillett and Miller (1974) for the VRP. Here a centre of gravity is computed and the clusters are partitioned according to their polar angle. Scheduling the customers, one of the previously developed route-building heuristics is used to build a 1-route solution. Due to the time windows and/or capacity constraints some customers may now be unscheduled. In order to schedule these scheduled customers are removed and the process is repeated. The sweep heuristic typically performs better than other heuristics in cases where many customers can be assigned to each route.

In general the heuristics of Solomon (1987) and Landeghem (1988) return a solution fast. Their solution does however generally lack in quality. Most of the time the solutions are more than 10% from optimum.

A problem of building one route at a time is usually that the routes generated in the latter part of the process are of poor quality as the last unrouted customers tends to be scattered over the geographic area. In Potvin and Rousseau (1993) tries to overcome this problem of the insertion heuristics by building several routes simultaneously. The initialization of the routes is done by using the insertion heuristic of Solomon (1987). On each route the customer farthest away from the depot is selected as a seed customer. Then we proceed by computing the best feasible insertion place for each unserviced customer and insert the one with the largest difference between the best and the second best insertion place. This method is better than the Solomon heuristics but still the solutions are quite far away from optimum. In Russell (1995) elaborates further on the insertion approach.

In Antes and Derigs (1995) another approach builds upon the classical insertion idea. Here every unrouted customer requests and receives from every route in the schedule a prize for insertion, defined in a similar way as in the Solomon heuristics. Then the unrouted customers send a proposal to the route with the best offer, and each route accepts the best proposal among those customers with the fewest number of alternatives. Note that more customers can be inserted in each iteration. If a certain threshold of routes is violated a certain number of customers are removed and the process is initiated again. The results of Antes and Derigs (1995) are comparable to those presented in Potvin and Rousseau (1993). Generally building several routes in parallel results in better solutions than building the routes one by one.

4.2. Route-improving heuristics

The basis of almost every route-improving heuristic is the notion of a neighbourhood. The neighbourhood of a solution S is a set $N(S)$ of solutions that can be generated with a single modification of S.

Checking some or all of the solutions in a neighbourhood might reveal solutions that are better with respect to objective function. This idea can be repeated from the better solution. At some point no better solution can be found an optimum has been reached. It is definitely a local optimum but it might even be global. This algorithm is called local search. The neighbourhood structures used in various VRPTW heuristics will be introduced, and then the algorithms in which they are used are described.

4.3. The neighbourhoods for the VRPTW

One of the most used improvement heuristics in routing and scheduling is the r-Opt heuristic. Here r arcs removed and replaced by r other arcs. A solution obtained using an r-Opt neighbourhood that cannot be improved further is called r-Optimal. Usually r is at most 3. Using the 3-Opt on the routes of a solution to the VRPTW problem is not without problems. For all possible 2-Opt interchanges and some of the exchanges in the 3-Opt neighbourhood, parts of the route is reversed. This may very likely lead to a violation of the time windows. In Potvin and Rousseau (1995), are presents two variants 2-Opt* and Or-Opt that maintain the direction of the route.

It is quite easy to see that Or-Opt's are a subset of 3-Opt's as we exchange three special arcs with three others. The size of the neighbourhood is although reduced from $O(n^3)$ to $O(n^2)$. Generally the size of a r-Opt neighbourhood is $O(n^r)$. The 2-Opt* is exchanging one segment of one route with a segment of another route. These neighbourhood-operators are sometimes denoted crossover or simply cross.

The exchange operator swaps customers in different routes, thereby interchanging two customers simultaneously into the other routes. The k-node interchange by Christofides and Beasley (1984) is modified by several authors to take time windows into account. Sequentially each customer i is considered and the sets M_1 and M_2 are identified. M_1 is defined as the customer i and its successor j. Then the elements of M_2 are found as the two customers closest to i and j but not on the same route as i and j (found by minimizing insertion cost calculated by the Euclidean distance). A neighbourhood is then defined by removing the elements of M_1 and M_2 and inserting them in any other possible way. As this neighbourhood is quite large, only the k most promising candidates are checked.

Another neighbourhood is the λ-interchange. In Osman (1993) originally developed for the VRP. It is a generalization of the relocate operator. Here a subset of customers of size $\leqslant\lambda$ in one route is exchanged with a subset of size $\leqslant\lambda$ from another route. Typically there is also given an ordering in which the different set sizes are tested. For example using a 2-interchange scheme we would first try to move one element from one route to another, and one the other way. Then we would try the reverse situation; and then try to exchange one element from one route with one from the other. This would be written as (1, 0), (0, 1), (1, 1), (0, 2), (2, 0), (2, 1), (1, 2) and (2, 2).

Finally, a neighbourhood denoted shift-sequence is used in Schulze and Fahle (1999). A customer is moved from one route to another then checking all possible insertion positions. If an insertion is made feasible by removing another customer j, it is removed and inserted in another route. This procedure is repeated until feasibility is restored.

4.3.1. Review of neighbourhood-operator

- Relocate operator: a move one customer from one route to another.
- Exchange operator: the interchange two customers between two routes.
- 2-Opt* operator: a changing one segment of a route with another segment from another route.
- Or-Opt operator: a continuous segment of customers is moved from one position on a route to another.
- K-node interchange operator: a sequentially each customer i is considered. Customer i and its successor j and the two customers closest to i and j but not on the same route are removed. The neighbourhood is defined by trying to insert these four vertices in any other possible way. As this neighbourhood is quite large, only the k most promising candidates are checked.
- λ-interchange operator: a subset of customers of size $\leqslant\lambda$ is exchanged with a subset of customers of size $\leqslant\lambda$ from another route.
- Shift-sequence operator: a customer is moved from one route to another checking all possible insertion positions. If an insertion is feasible by removing another customer j, it is removed and inserted in another route. This procedure is repeated until feasibility is restored.

5. Routing problem with artificial intelligence

The fields of artificial intelligence and operations research are closely related. Artificial intelligence techniques have been applied to routing problems in two ways. First, expert systems are being developed to assist a user faced with a particular application in choosing an appropriate algorithm and tuning the parameters of that algorithm to the problem at hand. Second, successful new algorithms have recently been developed using artificial intelligence search techniques like for example simulated annealing, tabu search.

A generic search procedure for the routing problems begins with a starting solution S and a rule for constructing a neighborhood $N(S)$ of alternative solutions that are near to S in some sense. The search procedure then chooses a new solution from $N(S)$ and iterates until some stopping criterion is reached.

Traditional local improvement methods choose a solution from $N(S)$ with a strictly improved objective function value and stop when $N(S)$ contains no such improving solution. Simulated annealing and tabu search allow the selection of non-improving solutions under certain conditions to be defined below.

An initial solution for a search algorithm can be obtained by applying any of the heuristics defined in (4.1). Usually the neighborhood $N(S)$ is defined to be all solutions by exchanging n_1 customers on a given route with n_2 customers on another route. Typically, $n_1 \leqslant 1$ and $n_2 \leqslant 1$, so the exchanges permitted consist of moving a single customer from one route to another route or exchanging two customers on different routes.

Different approaches can be used to compute the change in cost resulting from the addition or deletion of a customer on a route. Ideally, one would evaluate this change by optimally solving a travelling salesman problem (TSP) over the depot plus the customers on the route before and after the addition or deletion of the new customer. However, this approach can be computationally expensive so a less accurate but faster method is generally used. When a customer is deleted, the cost of the new route is computed without changing the sequence of the customers that remain on the route. Similarly, the customer added to a route is inserted between two existing customers without changing the sequence of the original customers on the route. The choice of where to insert the new customer is made to minimize the increase in cost.

With a traditional local improvement method that only accepts solutions that strictly improve the objective function, one can choose whether to accept the first solution in $N(S)$ that improves the objective function or to examine all solutions in $N(S)$ and choose the one producing the greatest reduction in the objective function.

- Simulated annealing searches the neighbourhood of $N(S)$ in a defined order. Let Δ denote the increase in object value for some $S' \in N(S)$. Then S' is accepted as the new incumbent solution either if $\Delta \leqslant 0$ or $\Delta > 0$ and $e^{-\Delta/T} \geqslant \theta$ where θ is a uniform random parameter $0 < \theta <$ and T is a control parameter of the search called the temperature. Typically, T is gradually lowered during the search procedure, steadily reducing the probability a non-improving solution will be accepted. If a complete search of $N(S)$ fails to produce a new incumbent, the value of T is raised by some amount and the search repeated. The simulated annealing algorithm stops when a fixed consecutive number of searches of $N(S)$ fail to produce a new incumbent solution.
- Tabu search chooses the best solution contained in $N(S)$ that does not violate certain restrictions designed to prevent the algorithm from cycling. Typically, these restrictions pre-

vent for the next t iterations a movement of customers that would "undo" a previous movement. For example, if customer c is moved from route A to some other route on iteration k, then on iterations $k + 1$ through $k + 1$ we prohibit any exchange that would move customer c back to route A. Similarly, if on iteration k customer c_1 on route A and c_2 on route B are exchanged, then on iterations $k + 1$ through $k + t$ we prohibit any movement that would return customer c_1 to route A and customer c_2 to route B. Tabu search stops after some fixed iteration limit has been reached.

6. Metaheuristics

A metaheuristics are powerful techniques applicable generally to a large number of problems. A metaheuristic refers to an iterative master strategy that guides and modifies the operations of subordinate heuristics by combining intelligently different concepts for exploring and exploiting the search space. A metaheuristic is a solution concept. The adaptation to a specific problem uses heuristics as solution methods. A metaheuristics may manipulate a complete or incomplete single solution or a collection of solutions at each iteration. The family of metaheuristics includes, but is not limited to, simulated annealing, tabu search and genetic algorithms. The success of these methods is due to their capacity "to solve in practice" some hard combinatorial problems.

6.1. Simulated annealing

The name "simulated annealing" is due to the fact that conceptually it is similar to a physical process, known as annealing, where a material is heated into a liquid stat then cooled back into a recrystallized solid state. The simulated annealing was one of the first metaheuristics developed. When using simulated annealing one does not search for the best solution in the neighbourhood of the current solution. Instead one simply draws at random a solution from the neighbourhood. If the solution is better it is always accepted as a new current solution, but if the solution is worse than the present current solution it is only accepted with a certain probability. The acceptance probability is determined by a temperature which is gradually decreased. By reducing the temperature the selection becomes more and more selective in accepting new solution. The idea of simulated annealing comes from thermodynamics and metallurgy: when a metal in fusion is cooled slowly enough it tends to solidify in a structure of minimal energy.

In Chiang and Russell (1996) develop the three different simulated annealing methods: one using a modified version of the k-node interchange mechanism and the second one using the λ-interchange mechanism as proposed by Osman (1993), with $\lambda = 1$. The third algorithm borrows the concept of a tabu list from the tabu search metaheuristic. Using simulated annealing with the λ-interchange mechanism the tabu list contains moves that will not be allowed for the time being. The last two methods have faster convergence than the first, although the first yields slightly better results. The travel distances obtained by the three simulated annealing were between 7% and 11.5% from optimum.

In Thamgiah et al. (1995) a non-monotone probability function is used and are using the λ-interchange with $\lambda = 2$ scheme to define the neighbourhood. The temperature is decreased after every iteration. In case the entire neighbourhood has been explored without finding any accepting moves the temperature is increased. This is called a reset. The temperature is increased to the maximum of the temperature at which the best solution was found and half of the temperature at the last reset. After R resets without improving the best solution the algorithm terminates. The quality of the solutions obtain in Thamgiah et al. (1995) is about the same as those obtained by Chiang and Russell (1996).

6.1.1. Remark

The comparison between simulated annealing and the best existing algorithms strongly depends on the problem treated and the implementation. For some problems, the superiority of simulated annealing is observed only for examples of large dimensions and at the price of a high calculating time. Certain current studies propose to improve the effectiveness of the algorithm and its computation time: by parallelization or by hybridization with other algorithms such as tabu search or genetic algorithms.

6.2. Tabu search

The tabu search is one of the old metaheuristics. It was introduced by Glover (1989) and Fisher et al. (1997). At each iteration the neighbourhood of the current solution is explored and the best solution in the neighbourhood is selected as the new current solution. In order to allow the algorithm to escape from a local optimum the current solution is set to the best solution in the neighbourhood even if this solution is worse than the current solution. To prevent cycling visiting recently selected solutions is forbidden. This is implemented using a tabu list. Often, the tabu list does not contain illegal solutions, but forbidden moves. It makes sense to allow the tabu list to be overruled if this leads to an improvement of the current over-all best solution. Criteria such as this for overruling the tabu list are called aspiration criteria. The most used criteria for stopping a tabu search are a constant number of iterations without any improvement of the over-all best solution or a constant number of iteration in all.

In Solomon (1987) is used to generate the initial solution. The algorithm shifts between the two strategies. When one has not made improvement for a certain number of iterations the other strategy is used instead and vice versa. In order to minimize the number of routes the algorithm tries to move customers from routes with few customers to other routes. This is done using Or-Opt.

In Badeau et al. (1995), was first generate a series of solutions and then new solutions are composed by randomly selecting from the already generated routes. The selection is done based to the good routes. When one route is selected the remaining routes servicing customers from this route is excluded. This process is continued until all customers are serviced by a route, or the algorithm runs out of routes.

The solution is now decomposed into groups of routes. For each group a tabu search is performed using the exchange operator on segments. Furthermore segments of customers are moved around within each route. In order to force the

algorithm to make a thorough exploration of the search space, frequently performed crossovers are penalized.

Another tabu search algorithm with similarities to Garcia et al. (1994) is developed in Schulze and Fahle (1999). It is based upon the local search methods discussed in Potvin and Rousseau (1995).

In Garcia et al. (1994) and Schulze and Fahle (1999) have been written about parallelization of the tabu search heuristic. Their efforts can be divided into three groups, partitioning of the neighbourhood, run parallel threads of tabu searches and decompose the problem into subproblems each solved by parallel tabu searches.

In Badeau et al. (1995) the tabu search heuristic for the VRPTW is essentially a parallelization of the tabu search. Here a combination of strategy 2 and 3 is used. After the generation of a solution at the master processor, the solution is decomposed into groups of routes. At each slave processor an independent tabu search tries to improve the over-all solution by improving the solution for the routes it received from the master processor.

Among the parallel implementations the algorithm developed in Schulze and Fahle (1999) performs slightly worse that the other parallel algorithms when working in problems with tight windows. As time windows get larger all algorithms perform equal with respect to the quality of the solution. As different parallel computers or network of computers are used it is difficult to compare the algorithms with respect to running time.

The reactive tabu search metaheuristic is applied to the parallel construction approach of Russell (1995). The tabu search route improvement procedure is invoked each time another 10% of the customers have been added to the emerging routes using the λ-interchange of Osman (1993) as the neighbourhood. If the same solution defined with the number of vehicle, the total accumulated distance, and the total accumulated travel time. Occurs to often the tabu list is increased by a constant factor. If waiting time is eliminated, the customer is fixed at its position for a number of iterations, and as in Badeau et al. (1995) too frequent switches of customers is penalized. On the other hand if no feasible solution is found by the tabu search, the size of the tabu list is decreased by constant factor. Generally it is among the best heuristics for the VRPTW. One of the conclusions in Badeau et al. (1995) is that diversification/intensification is just as important in obtaining good solutions as variable length tabu list.

6.2.1. Remark

Tabu search is an approximate method, proposed for combinatorial optimization hard problems. The applications show that this method is very flexible and efficient, and gives results, which compete or exceed those of the best heuristic known. However, in spite of the success it has, there is no support theoretic available to date. No study made it possible to prove the convergence of the method. In order to improve the procedure, techniques of parallelization and combination with other approaches are proposed. We quote on this subject the hybridization of tabu search and of simulated annealing in Osman (1993) and Tuyttens et al. (1994), and that tabu search and genetic algorithms in Glover et al. (1995).

- Some similarities between simulated annealing and tabu search:

Both start with an initial complete feasible solution and iteratively generate additional solutions, both can exactly or approximately evaluate candidate solutions, both maintain a record of the best solution obtained so far, and both must have a mechanism for termination. However, there are two important differences between the methods. Tabu search permits away from a local optimum (i.e., diversifying by an essentially deterministic mechanism, where as we well see, a probabilistic device is used in simulated annealing). Second, tabu search tends to temporarily permit moving to poorer solutions only when in the vicinity of a local optimum, where as this can happen at any time in simulated annealing.

6.3. The genetic algorithms

The genetic algorithms are probabilistic procedures of search, which took as a starting point the genetic evolution of a species. The principal idea is to reproduce the natural evolution of organisms, generation after generation, by respecting the phenomena's heredity and law of survival stated by Darwin.

The first use of the genetic algorithms goes bake to the year 1950 when biologists simulated the evolution of the organisms. Later Holland (1975), Goldbeerg (1989) and Fisher (1994) adapted the genetic algorithms to the resolution of combinatorial optimization problems.

Contrary to simulated annealing and tabu search, the genetic algorithms operate in a population of solution and not only one solution (Pirlot, 1996). A population of structures, each one corresponding to a possible solution, represents the space of the solution of the problem. The new structures are generated by application of genetic operators (selection, crossing and mutation) on the potential parents chosen inside the population. The genetic algorithms are based on the principle that best parents produce best children; so the strongest members of the population have strong probability to be selected as the parents. The best parents are crossed to give the place to new children who replace the parents or the week elements of the population. The procedure is repeated until a population is obtained where all the individuals are very strong, corresponding to optimal or almost optimal solution of the problem.

7. Conclusion

The VRPTW occupies the heart of distribution management. There exist some limited versions of the variety of exact, heuristic and metaheuristic to approximate algorithms have been proposed for its solution. A number of powerful metaheuristics have also been proposed simulated annealing, Tabu search and genetic algorithm. Over the past 20 years VRPTW has been an area of research that has attracted many researchers.

Acknowledgment

The author would like to thank an anonymous referee for some useful comments.

References

Agarwal, Y., Mathur, K., Salki, H., 1989. A set-partitioning-based exact algorithm for the vehicle routing problem. Networks 19, 731–749.

Antes, J., Derigs, U., 1995. A New Parallel Tour Construction Algorithm for the Vehicle Routing Problem with Time Windows. Technical Report, Lehrstuhl fur Wirtschaftsinformatik und Operations Research, Universitat zu Koln.

Badeau, P., Gendreau, M., Guertin, F., Potvin, J., Taillard, E., 1995. A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. Technical Report, CRT-95-84, Centre de recherche sur les transports, Montréal.

Baker, E., Schaffer, J., 1989. Solution improvement heuristics for the vehicle routing and scheduling problem with time windows constraints. American Journal of Mathematics and Management Sciences 6 (3,4), 261–300.

Chiang, W., Russell, R., 1996. Simulated annealing metaheuristics for the vehicle routing problem with time windows. Annals of Operations Research 63, 3–27.

Christofides, N., Beasley, N., 1984. The period routing problem. Networks 14, 237–241.

Clarke, G., Wright, W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581.

Crainic, G., Laporte, G., 2000. Fleet Management and Logistics. Kluwer Academic Publishers.

Desaulniers, G., Lavigne, J., Soumis, F., 1997. Multi-depot vehicle scheduling problem with time windows and waiting costs. European Journal of Operation Research 111, 479–494.

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40 (2), 342–354.

Desrosiers, J., Soumis, F., Desrosiers, M., 1984. Routing with time windows by column generation. Networks 14 (4), 545–565.

Dumes, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operation Research 54, 7–21.

El-Sherbeny, N., 2001. Resolution of a Vehicle Routing Problem with Multiobjective Simulated Annealing Method. Ph.D. Dissertation, Faculte Polytechnique de Mons, Belgique.

El-Sherbeny, N., Tuyttens, D., 2001. Optimization multicriteria of routing problem. Troisieme Journee de Travail sur la Programming Mathematique Multi-objective, Faculte Polytehnique de Mons, Belgique.

Fisher, M., 1994. Optimal solution of vehicle routing problem using minimum k-trees. Operations Research 42 (4), 626–642.

Fisher, M., Jornsteen, K., Madsen, O., 1997. Vehicle routing with time windows: two optimization algorithms. Operations Research 45 (3), 488–492.

Garcia, B., Potvin, J., Rousseau, J., 1994. A parallel implementation of the tabu search heuristic for vehicle routing problems with time windows constraints. Computers and Operations Research 21 (9), 1025–1033.

Gillett, B., Miller, L., 1974. A heuristic algorithm for the vehicle dispatch problem. Operation Research 22, 340–349.

Glover, F., 1989. Tabu search. Part 1, ORSA. Journal on Computing 1 (3), 190–206.

Glover, F., Kelly, P., Laguna, M., 1995. Genetic algorithms and Tabu search: hybrids for optimizations. Computer Operation Research 22 (1), 111–134.

Goldbeerg, E., 1989. Genetic Algorithms in Search. Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Holland, H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI.

Kohl, N., Madsen, O., 1997. An optimization algorithm for the vehicle routing problem with time windows based on lagrangean relaxation. Operations Research 45 (3), 395–406.

Kolen, A., Rinnooy Kan, A., Trienkens, H., 1987. Vehicle routing problem with time windows. Operation Research 35, 266–273.

Koskosidis, Y., Powell, W., Solomon, M., 1992. An optimization-based heuristic for vehicle routing and scheduling with soft time windows constraints. Transportation Science 26, 57–69.

Landeghem, A., 1988. A bi-criteria heuristic for the vehicle routing problem with time windows. European Journal of Operational Research 36, 217–226.

Lenstra, J., Rinnooy, K., 1981. Complexity of vehicle routing and scheduling problems. Networks 11, 221–227.

Lie, C., Simchi-Levi, D., 1990. Worst-case analysis of heuristics for multi-depot capacitated vehicle routing problems. Journal on Computing 2, 64–73.

Osman, I., 1993. Metastrategy simulated annealing and tabu search heuristic algorithms for the vehicle routing problem. Annals of Operations Research 41, 421–434.

Pirlot, M., 1996. General local search methods. European Journal of Operational 92, 493–511.

Potvin, J., Rousseau, J., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66, 331–340.

Potvin, J., Rousseau, J., 1995. An exchange heuristic for routing problems with time windows. Journal of Operational Research Society 46 (12), 1433–1446.

Reeves, C., 1995. Modern Heuristic Techniques for Combinatorial Problems. Mc-Greenhill.

Russell, R., 1995. Hybrid heuristics for the vehicle routing problem with time windows. Transportation Science 29 (2), 156–166.

Schulze, J., Fahle, T., 1999. A parallel algorithm for the vehicle routing problem with time windows constraints. Annals of Operations Research 86, 585–607.

Solomon, M., 1987. Algorithms for the vehicle routing and scheduling problems with time windows constraints. Operations Research 35 (2), 254–265.

Thamgiah, S., Osman, I., Sun, T., 1995. Metaheuristics for the Vehicle Routing Problem with Time Windows. Technical Report, SUR-CpSc-TR-95-32, Artificial Intelligence and Robotics Laboratory, Computer Science Department, Slippery Rock University, PA.

Toth, P., Vigo, D., 2002. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications.

Tuyttens, D., Liegeois, B., Pirlot, M., Teghem, J., Trauwaert, E., 1994. Homogeneous grouping of unclear fuel cans through annealing and tabu search. Annals of Operation Research 50, 575–607.

Tuyttens, D., Teghem, J., El-Sherbeny, N., 2004. A particular multiobjective vehicle routing problem solved by simulated annealing. Lecture Notes in Economics and Mathematical Systems, vol. 535. Springer-Verlag, Germany, pp. 133–152.