European Journal of Operational Research 202 (2010) 223–231

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Approximative solutions to the bicriterion Vehicle Routing Problem with Time Windows

Juliane Müller [*]

Tampere University of Technology, Department of Mathematics, Korkeakoulunkatu 1, 33720 Tampere, Finland

ARTICLE INFO

Article history:
Received 16 August 2008
Accepted 29 April 2009
Available online 8 May 2009

Keywords:
Logistics
Heuristics
Bicriterion optimization
Vehicle Routing with Soft Time Windows

ABSTRACT

The Vehicle Routing Problem with Time Windows (VRPTW) is a combinatorial optimization problem. It deals with route planning and the distribution of goods from a depot to geographically dispersed customers by a fleet of vehicles with constrained capacities. The customers' demands are known and each customer has a time window in which it has to be supplied. The time windows are assumed to be soft, that means, violations of the time windows are allowed, but associated with penalties. The problem is to organize the vehicle routes optimally, i.e. to minimize the total costs, consisting of the number of used vehicles and the total distance, and the penalties simultaneously. Thus, the problem is formulated as a bicriterion minimization problem and heuristic methods are used to calculate approximations of the Pareto optimal solutions. Experimental results show that in certain cases the allowance of penalties leads to significant savings of the total costs.

1. Introduction

The Vehicle Routing Problem (VRP) goes back to Dantzig and Ramser (1959) who introduced it in 1959. The aim of the VRP is to determine an optimal set of routes for a fleet of vehicles that are used to distribute goods to customers with known demands. The routes have to start and end at a central depot, and each customer must be supplied only once by only one vehicle. The overall demands of the customers that are served by the same vehicle must not exceed the vehicle's capacity. The problem is to minimize the total costs, consisting of the number of used vehicles and the total distance, subject to the above constraints.

The VRP is an integer programming problem which belongs to the category of NP-hard problems. Since exact algorithms are usually computationally expensive, approximate solutions with sufficient accuracy that can be obtained fast are often desired. For that purpose, heuristic methods are used. In this paper the Vehicle Routing Problem with Time Windows (VRPTW) is considered since customers often have time windows in which the vehicle has to arrive and start the service. The time windows can be soft (VRPSTW) or hard (VRPHTW). Soft means, that violating the time windows is allowed, but associated with penalties. In contrast, hard time windows do not allow any delay.

Several techniques have been developed for solving the VRPTW. There are four general types of exact solution methods: dynamic programming (Kolen et al., 1987), Dantzig–Wolfe (column generation) (Desrochers et al., 1992; Kohl et al., 1999), Lagrange decomposition (Kohl and Madsen, 1997) and solving the classical model formulation directly. However, exact solution methods are very time consuming, and therefore not suitable for real life applications when results must be obtained quickly. Thus, approximation methods, namely, heuristics and metaheuristics are used.

Simulated annealing (Arbelaitz et al., 2001; Czech and Czarnas, 2002; Oliveira et al., 2006; Zhong and Pan, 2007), tabu search (Osman, 1993; Gendreau et al., 1994; Taillard et al., 1997; Toth and Vigo, 1998), ant algorithms (Colorni et al., 1991; Bullnheimer et al., 1999; Barán and Schaerer, 2003), and genetic algorithms Holland, 1975; Jong, 1975; Goldberg and Holland, 1988, for example, range amongst the metaheuristic methods. Heuristics can be classified into route building and route improvement heuristics. Classical examples of route building heuristics are the Clarke and Wright (1964) savings heuristic, and the I1 route construction heuristic by Solomon (1987). Route improvement heuristics, as for example the Or-opt procedure by Or (1976) and the 2-opt*-procedure by Potvin and Rousseau (1995), aim to improve an initial solution by exchanging customers within and between routes.

This paper focuses on computing approximate solutions to the VRPSTW. By allowing violations of the customers' time windows, solutions that cause less total costs might be obtained, but at the same time penalties arise. The problem is to minimize the total transportation costs, which consist of vehicle fixed costs and distance costs, and the penalties simultaneously. However, the more the total costs are reduced, the higher the penalties. These two conflicting objectives do not allow a unique optimal solution. The solutions are trade-offs between the total costs and the penalties

* Tel.: +358 46 8921 445.
E-mail address: juliane.mueller@tut.fi

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.04.029

and form a so-called Pareto front. From these Pareto solutions the decision maker has to choose the compromise that suits the problem at hand best.

This paper is organized as follows: the theoretical background of multi-objective optimization problems and the ϵ-constraint technique which is used to solve the VRPSTW are described in Section 2. In Section 3 the applied heuristic methods and the assumptions concerning the penalties are depicted. The used problem sets for testing the algorithm, as well as computational results are discussed in Sections 4 and 5. Section 6 concludes the paper and future research topics are suggested.

2. The Vehicle Routing Problem with Soft Time Windows

2.1. Prerequisites of multi-objective optimization problems

In this section the necessary concepts of multi-objective optimization problems, as well as the related fundamental terms are provided. The notations and definitions are taken from Coello et al., 2002. For more detailed information on the topic, especially on methods for solving multi-objective optimization problems, the reader is referred to Miettinen (1999).

Let $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ be the vector of decision variables, $g_i(\vec{x}) \geqslant 0,\ i = 1, \ldots, m$, and $h_i(\vec{x}) = 0,\ i = 1, \ldots, p$, are the inequality and equality constraints and $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T$ is the vector of the objective functions that are to be minimized.[1] Then problems of the form

$$\begin{aligned}
&\vec{f}(\vec{x}) \to \text{“}min\text{”},\\
&\vec{f} : \Omega \to \mathbb{R}^k, \qquad (1)\\
&\Omega = \{\vec{x} \in \mathbb{R}^n : \vec{g}(\vec{x}) \geqslant 0, \vec{h}(\vec{x}) = 0\}
\end{aligned}$$

are called multi-objective optimization problems. In this formulation the notation in the first line indicates the optimization of the vector function. However, the meaning of the term “optimum” changes in the multi-objective case because there are more than one objective function which usually conflict with each other. This means that a decrease in some criteria leads to a simultaneous increase in at least one other criterion. The goal is to find a feasible solution $\vec{x}^*$ that represents the best compromise (or trade-off) between the single objective functions. The optimum in this case is called Pareto optimum.

Definition 2.1.1 (*Pareto optimality*). A point $\vec{x}^* \in \Omega$ is Pareto optimal if for every $\vec{x} \in \Omega$ and $I = \{1, 2, \ldots, k\}$ either, $f_i(\vec{x}) = f_i(\vec{x}^*), \forall i \in I$ or, there is at least one $i \in I$ such that $f_i(\vec{x}) > f_i(\vec{x}^*)$.

In other words, there is no feasible $\vec{x}$ that is at least as good as $\vec{x}^*$ for all criteria and strictly better for at least one. These Pareto optimal solutions are considered to be the optimal solutions in the sense of multi-objective optimization problems.

Definition 2.1.2 (*Pareto dominance*). A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate $\vec{v} = (v_1, \ldots, v_k)$ if and only if $\forall i \in \{1, \ldots, k\} : u_i \leqslant v_i \wedge \exists i \in \{1, \ldots, k\}$ with $u_i < v_i$. The Pareto dominance of $\vec{u}$ over $\vec{v}$ is denoted by $\vec{u} \preceq \vec{v}$.

Definition 2.1.3 (*Pareto optimal set*). For the problem of the form (1) the Pareto optimal set (P^*) is defined as $P^* := \{x \in \Omega : \neg\exists x' \in \Omega \text{ with } \vec{f}(x') \preceq \vec{f}(x)\}$.

[1] Note that maximization problems can be handled analogously with $\max f_i(\vec{x}) = -\min(-f_i(\vec{x}))$ and converting the inequality constraints $g_i(\vec{x}) \leqslant 0$ into $-g_i(\vec{x}) \geqslant 0$.

Definition 2.1.4 (*Pareto front*). For the problem of the form (1) and the Pareto optimal set P^*, the Pareto front (PF^*) is defined as $PF^* := \{\vec{u} = \vec{f} = (f_1(x), \ldots, f_k(x)), x \in P^*\}$.

Pareto optimal solutions are also called non-inferior, admissible or efficient solutions. Their corresponding vectors $\vec{f}(\vec{x}^*) = [f_1(\vec{x}^*), f_2(\vec{x}^*), \ldots, f_k(\vec{x}^*)]^T$ are termed non-dominated. The Pareto optimal set consists of the solutions whose corresponding vectors are non-dominated with respect to all other comparison vectors. If the non-dominated vectors are plotted in the objective space, they are known as Pareto front.

Solution techniques for multi-objective optimization problems consist in general of two stages: the optimization of the multiple objective functions, and the decision what kind of “trade-offs” are most appropriate from the decision maker’s point of view. The techniques can be divided into generating techniques (a posteriori articulation of preferences), techniques which rely on prior articulation of preferences (non-interactive methods) and techniques which rely on progressive articulation of preferences (interaction with the decision maker) (Coello et al., 2002).

In this paper, the ϵ-constraint method, which belongs to the a posteriori preference articulation techniques, is applied. The multi-objective optimization problem is solved with a single objective strategy by transformation into a single objective problem. The idea of the ϵ-constraint method is to minimize one objective function (the most preferred one) at a time and to take the other objectives as constraints which are bounded by a value ϵ_l. Assuming that the rth objective function is minimized and the other $k-1$ objectives are turned into constraints, the problem can be stated as follows:

$$\min \quad f_r(\vec{x}) \qquad (2)$$

subject to $f_l(\vec{x}) \leqslant \epsilon_l$ for $l = 1, 2, \ldots, k$ and $l \neq r$, where ϵ_l are limits of the objective functions that must not be exceeded. If the optimal solution of (2) is unique, it is efficient. By varying the right hand side values ϵ_l, all efficient solutions can be found: x^* is efficient, if and only if it is an optimal solution of (2) for all $r = 1, \ldots, k$, where $\epsilon_l = f_l(\vec{x}), l \neq r$.

2.2. Problem formulation

The VRPSTW can be described as a graph theoretic problem. The graph $G = (V, A)$ represents a network, where $V = \{0, \ldots, n+1\}$ is the set of nodes and A is the set of arcs. The nodes $\{1, \ldots, n\}$ stand for the customers. It is assumed that there is only one central depot, which is associated with the nodes 0 and $n+1$. The arcs $(i,j) \in A$ are the connections between the customers. Each arc $(i,j) \in A$ is assigned a cost c_{ij}. Loop arcs (i,i), i.e. arcs that connect a node to itself, are not allowed, and therefore the costs for these arcs are set to $c_{ii} = +\infty, \forall i \in V$. The set of customers that are directly reachable from customer i are denoted by $\Delta^+(i)$, the so-called forward star of i. Analogously, $\Delta^-(i)$ is the backward star of customer i, i.e. the set of customers from which customer i is directly reachable.

The time windows of the customers are denoted by $[a_i, b_i]$. The set of identical vehicles is denoted by K. Every vehicle starts and ends its route at the depot whose time window is denoted by $[a_0, b_0] = [a_{n+1}, b_{n+1}]$ where the lower bound represents the earliest possible departure from the depot. When a vehicle is serving a customer, it has to stay there for the time period $s_i > 0, i \in V$ (service time). Every customer has a certain demand $d_i, i \in V$. Since each customer may be visited only once, its demand has to be fulfilled at that visit. The service time and the demand of the depot are set to $s_0 = s_{n+1} = 0$ and $d_0 = d_{n+1} = 0$, respectively.

It is assumed that the travel times t_{ij} from customer i to customer j, the service times s_i, the intervals $[a_i, b_i]$, the demands d_i

and the capacity C of the vehicles are given. Furthermore, it is assumed that the travel times fulfill the triangle inequality $t_{ik} + t_{kj} \geqslant t_{ij}$, and that the travel time matrix and distance matrix coincide. Therefore, distance and travel time, are used interchangeably.

The mathematical model for the VRPSTW contains two types of decision variables, namely, flow and time variables. The flow variables $x_{ijk}, (i,j) \in A, k \in K$, are equal to 1 if vehicle k uses the arc $(i,j) \in A$, and 0 otherwise. The time variables $w_{ik}, i \in V, k \in K$, specify the service start at customer i when it is served by vehicle k. If i is not served by vehicle k, w_{ik} has no meaning.

The objective is to minimize the total costs for serving all customers. Therefore, the number of used vehicles and the overall distance should be as small as possible. By allowing violations of the customers' time windows it is possible to save distance and vehicle fixed costs, but penalties, which are measured according to the delayed delivery in time units, must be accepted. It is important to know in which "currency" these penalties have to be paid. This means, the penalties cannot always be expressed as monetary losses; the effects of the possible decrease in the company's esteem, and the loss of customers due to unreliable schedules might not be easily quantified in the value of money. These are crucial aspects that have to be taken into consideration when different routing plans are evaluated.

A vehicle that arrives too late at customer j, has to start the service immediately and gets the penalty time t_{l_j} assigned which is calculated as

$$t_{l_j} = \max\{(w_{ik} + s_i + t_{ij}) - b_j, 0\}. \tag{3}$$

If a vehicle arrives too early, it has the possibility to start the service immediately. In this case t_{e_j} is the assigned penalty:

$$t_{e_j} = \max\{a_j - (w_{ik} + s_i + t_{ij}), 0\}. \tag{4}$$

On the other hand, the vehicle could also wait until a_j and the corresponding penalty is zero. Therefore, the time

$$t_p = \sum_{(i,j)\in A} \max\{a_j - (w_{ik} + s_i + t_{ij}), (w_{ik} + s_i + t_{ij}) - b_j, 0\} x_{ijk} \tag{5}$$

must be minimized for every vehicle $k \in K$ that serves customer j after customer i in order to minimize the cumulative penalty time. Therefore, the two conflicting objective functions that must be minimized simultaneously are:

- the total costs consisting of vehicle fixed costs and distance costs:
$$f_1(x_{ijk}) = \sum_{k\in K} \sum_{(i,j)\in A} c_{ij} x_{ijk}; \tag{6}$$
- the penalty time which is calculated from the delayed delivery:
$$f_2(x_{ijk}) = \sum_{k\in K} \sum_{(i,j)\in A} t_{p_k} x_{ijk}. \tag{7}$$

The problem can be stated mathematically as follows:

$$\min \quad f_1(x_{ijk}) \tag{8}$$
$$\min \quad f_2(x_{ijk}) \tag{9}$$

subject to

$$\sum_{k\in K} \sum_{j\in \Delta^+(i)} x_{ijk} = 1 \quad \forall i \in N, \tag{10}$$
$$\sum_{j\in \Delta^+(0)} x_{0jk} = 1 \quad \forall k \in K, \tag{11}$$
$$\sum_{i\in \Delta^-(j)} x_{ijk} - \sum_{i\in \Delta^+(j)} x_{jik} = 0 \quad \forall k \in K,\ j \in N, \tag{12}$$
$$\sum_{i\in \Delta^-(n+1)} x_{i,n+1,k} = 1 \quad \forall k \in K, \tag{13}$$
$$\sum_{k\in K} \sum_{i\notin S} \sum_{i\in S} x_{ijk} \geqslant 1 \quad \forall S \subset V,\ |S| \geqslant 2, \tag{14}$$
$$\sum_{i\in N} d_i \sum_{j\in \Delta^+(i)} x_{ijk} \leqslant C \quad \forall k \in K, \tag{15}$$
$$x_{ijk}(w_{ik} + s_i + t_{ij} - w_{jk}) \leqslant 0 \quad \forall k \in K,\ (i,j) \in A \tag{16}$$

and

$$x_{ijk} \in \{0,1\} \quad \forall k \in K,\ (i,j) \in A. \tag{17}$$

Let $N = V \setminus \{0, n+1\}$ be the set of customers. The constraints (10) state that each customer $i \in N$ is assigned to one vehicle route. The constraint sets (11), (12) and (13) are the flow constraints. They condition that each vehicle k has to leave the node 0 once, that it leaves customer $i, i \in N$, if and only if it enters that customer, and that it has to return to the node $n+1$. To prevent subtours, the constraints (14) are required. Constraints (15) guarantee that no vehicle can serve more customers than its capacity allows. In case the demand of a customer is larger than the vehicle capacity, the customer's demand can be divided into multiple parts that do not exceed the vehicle capacity. Accordingly, copies of this customers must be created at the same position with the same time window. If the sum of the demands of the customers delivered by vehicle k is less than the capacity, the vehicle carries only the sum of the demands. Constraints (16) express that a vehicle k traveling from customer i to j, cannot arrive at customer j before $w_{ik} + s_i + t_{ij}$. The set of integrality constraints is given by (17).

3. Methodology

In this paper the minimization of the total costs, i.e. the number of used vehicles and the total traveled distance, is regarded as the primary minimization preference. Every additional vehicle causes fixed costs (purchase, follow-up costs for insurance and maintenance) and requires an additional driver. The total traveled distance is also an important cost driver because pollution is a crucial topic and the prices for gasoline are perpetually increasing.

Thus, the overall goal is to minimize the total costs for supplying all customers. Allowing violations of the time windows may result in more possibilities for choosing the next customer to visit. Especially in cases where violations cause the usage of less vehicles or substantial savings in the total distance, penalties may become an acceptable burden. According to the ϵ-constraint method one of the objective functions f_1 and f_2 turns into an additional constraint which is bounded by a certain value ϵ. Since the emphasis is on the minimization of f_1, the objective f_2 turns into this extra constraint.

It is assumed that each vehicle is assigned the same allowed penalty limit so that the overall penalty of the solution is bounded to K times the penalty limit per vehicle. The penalty limit is denoted by $t_{\text{p\_limit}}$,[2] and given by the decision maker.

Approximate solutions to the problem are obtained in two stages using heuristic methods. At first Solomon's route construction heuristic I1 is applied to obtain an initial (current) solution. Then, route improvement heuristics, namely, an ejection chain, Or-opt exchange and 2-opt* procedure are repeatedly applied to the current solution to decrease the number of used vehicles and to obtain savings in distance. These procedures have the advantage that they preserve the sequence in which the customers are visited. Many k-opt exchange operators developed for solving the VRP reverse this sequence which is inappropriate for applications to problems with time windows. In solutions to the VRPTW, custom-

[2] This bound corresponds to the value of ϵ. If $\epsilon = 0$, the vehicles are not allowed to cause any penalty and the VRPHTW results.

ers with higher time window upper bounds must be served after those with lower time window upper bounds. If the sequence is reversed, the resulting solution is likely to be infeasible with respect to the penalty constraints.

Heuristic methods have the advantage that they can often create solutions very fast. This is important for applications in real life when the time for scheduling is highly bounded. Since real life problems are usually larger and more complicated than the theoretical test problems, solutions that can be obtained sufficiently fast are required. However, quickness is not the only matter of importance. Also the solution quality should be acceptable. Compared to solutions obtained from metaheuristics, simple heuristics are often inferior with respect to the solution quality. Metaheuristics, in contrast, have the disadvantage that they are usually computationally expensive, which constitutes a drawback in their application to real life problems. The computation time tends to become rather high so that it might sometimes seem unfavorable to use these kinds of algorithms. Another disadvantage is their reliability. Although metaheuristics may deliver better solutions than simple heuristics, metaheuristic algorithms usually have to be run several times in order to find the best solution. Thus, in order to find approximations of the exact Pareto front within an acceptable time frame, heuristics were used.

3.1. Route building heuristic

The VRPSTW is approached using different penalty limits. During the first step the penalty limit is zero. If no improvements can be found anymore, the allowed penalty limit per vehicle is raised to give the algorithm the chance to find solutions that cause less total costs, but in turn penalties have to be accepted.

The initial routes were constructed using Solomon's I1 insertion heuristic. A "seed" customer, which initializes the first route, is chosen according to three possibilities:

1. The unrouted customer farthest away from the depot;
2. The unrouted customer with the smallest time window's upper bound;
3. The unrouted customer which is farthest away from the depot becomes the first seed customer seed$_1$, the second seed customer seed$_2$ is the one farthest away from the depot and farthest away from seed$_1$, the third seed customer seed$_3$ is the one farthest away from the depot and farthest away from seed$_2$, and so on.

Each initial route contains at first only the seed customer and is then filled with unrouted customers. Two cost measures $c_1(i,u,j)$ and $c_2(i,u,j)$ are used to identify the customer to be inserted into the partial route and to find the best feasible place between the two consecutive nodes i and j. The insertion procedure consists of three steps. At first, the minimum feasible insertion cost $c_1^*(i,u,j)$ for each unrouted customer u is computed. The cost c_1 for inserting customer u between the two adjacent nodes i and j are calculated as $c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j)$, with $\alpha_1 + \alpha_2 = 1$, $\alpha_1 \geqslant 0$, $\alpha_2 \geqslant 0$. The additional distance a vehicle has to accept when it serves customer u in the same route as j is represented by $c_{11}(i,u,j)$. The component $c_{12}(i,u,j)$ is the modification of the service time at customer j when u is inserted before j in the tour. These two components are calculated as $c_{11}(i,u,j) = t_{iu} + t_{uj} - \mu t_{ij}, \mu \geqslant 0$ and $c_{12}(i,u,j) = w_{u,jk} - w_{jk}$, where t_{iu}, t_{uj} and t_{ij} represent the distances in time units between customers i and u, u and j, and i and j, respectively. The service starting time at customer j is denoted by w_{jk}, and $w_{u,jk}$ stands for the new time of the service start at customer j with u inserted before j.

Secondly, the unrouted customer u^* is selected among the customers that can be feasibly inserted into the route. This customer maximizes the criterion $c_2(i,u,j)$, the generalized saving measure,[3] which is obtained from $c_2(i,u,j) = \lambda t_{0u} - c_1^*(i,u,j)$, $\lambda \geqslant 0$. In the last step the customer u^* is inserted at its best place. The insertion of customers is executed as long as the capacity and time window constraints are not violated. If unrouted customers that could not be feasibly inserted remain, a new route is initialized and the insertion process is repeated. The procedure ends when each customer is assigned to a vehicle.

Four different parameter settings are used to tune the insertion procedure. These are $(\alpha_1, \alpha_2, \mu, \lambda) = (1,0,1,1), (1,0,1,2), (0,1,1,1)$ and $(0,1,1,2)$. The three possibilities of choosing the seed customers and the four parameter settings result in 12 (different) initial solutions. Out of these the solutions which require the lowest number of vehicles are selected and from this set the solution with the shortest distance is chosen to be improved.[4] Denote the number of vehicles in the initial solution by K^{init}, and the initial distance by d^{init}. The initial solution is denoted by $S^{\text{init}} = \{R_1, \ldots, R_p\}$, where $R_p = (v_{R_p,1}, v_{R_p,2}, \ldots, v_{R_p,\eta_p})$ is the set of customers served by vehicle p. The sequence of the customers in the route R_p is equivalent to the order in which they are served by the vehicle.

3.2. Route elimination procedure

The ejection chain procedure (ECP) attempts to reduce the number of used vehicles in the current solution. Routes can only be eliminated if more than the minimum number of necessary vehicles $K_{\min} = \lceil (\sum_{i \in N} d_i)/C \rceil$ is used, where d_i is the demand of customer i and assuming that all vehicles have the same capacity C.

The ECP examines all routes successively. One customer after the other of the route under consideration is reinserted in another route. Assumed that route R_s is to be eliminated. At first the customer $v_{R_s,1}$ is inserted in route $R_q, q \neq s$. The insertion trials are performed consecutively, i.e. $v_{R_s,1}$ is inserted from the first to the last position of R_q and in all positions of all other routes of the current solution. The resulting route is always checked for feasibility. If insertion is possible without removing (ejecting) any customer of R_q in order to retain feasibility, the procedure continues with the insertion of customer $v_{R_s,2}$. If the insertion of $v_{R_s,1}$ results in an infeasible solution, it is checked which constraint is violated:

1. In case the insertion of the customer at a certain position in R_q would cause a violation of its own time window (i.e. the service would start later than the penalty limit of the corresponding vehicle allows), the next route is taken into account for insertion.
2. In case customers of route R_q that are arranged behind $v_{R_s,j}$ are served too late and the penalty limit is exceeded, those customers causing penalty are ejected one after another and stored in a list L_{eject}, until the penalty constraint is satisfied and a feasible route emerges.
3. In case the capacity constraint is exceeded, customers from the route R_q are ejected and stored in a list L_{eject} until the insertion of customer $v_{R_s,j}$ is feasible.

For each route and each insertion place, the altered route and L_{eject} are stored. When every position has been examined, the solution which causes the least number of ejected customers becomes the interim solution S^{inter}. In the next step, the ejected customers are reinserted in the routes of $S^{\text{inter}} \setminus (R_s \setminus \{v_{R_s,1}\})$ analogously to the insertion of $v_{R_s,1}$. The list of ejected customers is updated, the customer that has been successfully rerouted leaves the list and

[3] The classical savings are calculated by setting $\alpha_1 = 1, \alpha_2 = 0, \mu = 1$ and $\lambda = 2$.

[4] The case that solutions with different routes result in the same total distance can be treated by choosing the initial route randomly. This, however, did not occur in any of the test problems.

the customers that might have been ejected by this insertion join it. Additionally, L_{eject} as well as the total distance of the new interim solution are stored. The process of reinserting ejected customers is repeated until either the list L_{eject} is empty and the whole procedure starts anew with the next customer $v_{R_s,2}$, or the current list L_{eject} and the corresponding distance equal a previously computed result. In this case the elimination trial of the current route under consideration terminates and continues with the next route, thus avoiding the recalculation of the same results. The elimination trial also terminates if the number of iterations of reinserting customers from L_{eject} exceeds a certain limit.

As soon as it turns out that the elimination of a route is possible, the ECP terminates and a new current solution S^{curr} is obtained to replace the former solution. The ejection chain procedure is repeated until no improvement can be found anymore or $K_{\min}$ is reached.

3.3. Or-opt procedure

The used Or-opt exchange procedure takes the first customer $v_{R_1,1}$ in the first route R_1 out of its current position. The customer $v_{R_1,1}$ is inserted into all other places of R_1 as well as into all positions of every other route of $S \setminus R_1$ one after another. More precisely, $R_1 = (v_{R_1,1}, v_{R_1,2}, \ldots, v_{R_1,\eta_1})$ is modified to $R_1' = (v_{R_1,2}, v_{R_1,1}, \ldots, v_{R_1,\eta_1}), R_1'' = (v_{R_1,2}, v_{R_1,3}, v_{R_1,1}, \ldots, v_{R_1,\eta_1}), \ldots, R_1^{(\eta_1-1)} = (v_{R_1,2}, \ldots, v_{R_1,\eta_1}, v_{R_1,1}), R_2 = (v_{R_2,1}, v_{R_2,2}, \ldots, v_{R_2,\eta_2})$ is modified to $R_2' = (v_{R_1,1}, v_{R_2,1}, v_{R_2,2}, \ldots, v_{R_2,\eta_2}), R_2'' = (v_{R_2,1}, v_{R_1,1}, v_{R_2,2}, \ldots, v_{R_2,\eta_2}), \ldots, R_2^{(\eta_2)} = (v_{R_2,1}, \ldots, v_{R_2,\eta_2}, v_{R_1,1})$, and so on.

If the insertion of the customer $v_{R_1,1}$ is feasible, for example after the second customer of route q (i.e. $R_q'' = (v_{R_q,1}, v_{R_q,2}, v_{R_1,1}, v_{R_q,3} \ldots, v_{R_q,\eta_q})$), the distances of the altered routes are calculated and the number of used vehicles K^{new} as well as the total distance d^{new} are compared to the values K^{curr} and d^{curr}. If any of the values is better, i.e. smaller, the solution is stored in a list L_{im} of possible improvements.

Next, the customer sequence $(v_{R_1,1}, v_{R_1,2})$ is inserted at the same place where already the single customer $v_{R_1,1}$ fitted in feasibly (in the latter example R_q becomes $R_q'' = (v_{R_q,1}, v_{R_q,2}, v_{R_1,1}, v_{R_1,2}, v_{R_q,3} \ldots, v_{R_q,\eta_q})$). If this solution turns out to be feasible again, the distance and number of vehicles are recalculated, compared with the current values, and in case any of the values is better, the solution is stored in L_{im}. The process is repeated with the customer sequence $(v_{R_1,1}, v_{R_1,2}, v_{R_1,3}), (v_{R_1,1}, v_{R_1,2}, v_{R_1,3}, v_{R_1,4})$, etc. until no feasible insertion of the whole sequence is possible anymore or the sequence cannot be extended due to the length of R_1.

In general, if the insertion of the customer $v_{R_i j}$ into the new place is infeasible with respect to the current penalty limit, it is checked if the customer itself causes the excess of the limit or if it is caused by the following customers. In the first case no further attempts to insert the customer $v_{R_i j}$ into this route are made because an insertion farther back in the route would only cause an increase of penalties. As long as there are other routes left where insertion has not been considered yet, $v_{R_i j}$ is inserted into the first position of the next route and the process of inserting sequences, starting with $v_{R_i j}$, and checking feasibility is repeated analogously. In the latter case the customer $v_{R_i j}$ is inserted into the next place of the same route, the feasibility is tested and the further steps are carried out analogously.

If the insertion of a sequence $(v_{R_i j}, v_{R_i j+1}, \ldots, v_{R_i,\nu}),\ \nu \leqslant \eta_j$, fails the feasibility test, the insertion of the sequence is terminated and the single customer $v_{R_i j}$ is inserted into the next position of the same route.[5] Feasibility tests, calculations of distance and used vehicles as well as extensions of the customer sequences are repeated until the whole procedure has been performed for every customer of every route.

As soon as a customer sequence $r_r \subseteq R_r, r \in \{1, \ldots, p\}$ is taken out of the route R_r, and tried to be inserted into route R_s, $s \in \{1, \ldots, p\}, s \neq r$, it is checked if the capacity constraints of vehicle s serving the customers of route $R_s \cup r_r$ are violated. In case they are, the insertion trial of the considered customer sequence into route R_s terminates.

All solutions, that yield a better total distance or number of used vehicles, are gathered in the list $L_{\text{im}} = \{\text{Im}_1,\ \text{Im}_2, \ldots, \text{Im}_\kappa\}$. From those possibilities the solutions that require the smallest number of vehicles are selected for further exploration. They are denoted by $L_v, L_v \subseteq L_{\text{im}}$. From the remaining solutions in L_v the solution with the shortest total distance becomes the first improvement of the current solution, say I_1. The Or-opt method modifies at most two routes of S^{curr} at a time (the route where the customer is taken out from and the route where the customer is inserted in). Therefore, a further improvement, say $I_2 \in L_v$, with the solution that changes other routes than I_1 and that causes the next smallest total distance value is realized. It is checked if a further improvement I_3 exists which alters routes different from those changed by I_1 and I_2. If this is the case, it is carried out and the process is repeated until there are no more improvements left, that affect routes that have not already been changed. The result of the procedure is a new solution S^{new} with smaller values of total distance or vehicle number.

3.4. 2-opt* procedure

In the 2-opt* procedure, introduced by Potvin and Rousseau (1995), the fact that a multiple Traveling Salesman Problem (mTSP) can be transformed into an equivalent TSP by creating m copies of the depot is used. The applied 2-opt* procedure joins at first the single routes of the current solution $S^{\text{curr}} = \{R_1, \ldots, R_p\}$ and inserts p copies of the depot. The single routes in this "loop" are distinguished from each other by the places of the depots. All the customers between two depots belong to the same original route.

The loop is denoted by the sequence $(D_1, R_1, D_2, R_2, \ldots, D_p, R_p, D_1)$, where D_1 appears twice to symbolize that the loop is closed. The algorithm starts with the starting customer sequence of the route R_1, i.e. the customer $v_{R_1,1}$, and tries to exchange the end part after $v_{R_1,1}$, i.e. the sequence $(v_{R_1,2}, v_{R_1,3}, \ldots, v_{R_1,\eta_1})$, with the end part of another route of $S \setminus R_1$. This is executed consecutively: At first $v_{R_1,1}$ is connected to the first customer of the second route $v_{R_2,1}$ and the depot before $v_{R_2,1}$ is connected to $v_{R_1,2}$. Accordingly, the new routes are $R_{\text{new}_1} = (v_{R_1,1}, v_{R_2,1}, v_{R_2,2} \ldots, v_{R_2,\eta_2})$ and $R_{\text{new}_2} = (v_{R_1,2}, v_{R_1,3}, \ldots, v_{R_1,\eta_1})$. Then, $v_{R_1,1}$ is connected to the second customer $v_{R_2,2}$ of route R_2 and the first customer of R_2 is connected to the second customer of R_1. Thus, the resulting modified routes are $R_{\text{new}_1} = (v_{R_1,1}, v_{R_2,2}, v_{R_2,3} \ldots, v_{R_2,\eta_2})$ and $R_{\text{new}_2} = (v_{R_2,1}, v_{R_1,2}, v_{R_1,3} \ldots, v_{R_1,\eta_1})$. In general, if routes R_q and R_s are taken into account, the new route R_{new_1} starts at depot D_q and ends in the depot $D_{s+1}, R_{\text{new}_2}$ starts at depot D_s and ends in the depot D_{q+1}.

After all connections to route R_2 have been examined, the process of exchanging end parts is repeated analogously with routes $R_3, \ldots, R_p$. After all these possibilities have been explored, the starting customer sequence of route R_1 is extended to $(v_{R_1,1}, v_{R_1,2})$ and the linking procedure to the other routes' end parts starts anew. Finally, the sequence containing all customers $(v_{R_1,1}, \ldots, v_{R_1,\eta_1})$ is taken as a starting sequence. In case it is connected to the first customer of another route, the second newly constructed route contains only two consecutive depots. Therefore, this route is empty and if the other one is feasible, a solution with one vehicle less is obtained. The whole process is repeated using starting sequences from every route.

[5] If there is no such position left, i.e. the customer sequence was inserted after the last customer of the considered route, the customer $v_{R_i j}$ is inserted into the first position of the next route. If there is no unchecked route left, the process continues with customer $v_{R_i j+1}$ in the same manner as with $v_{R_i j}$.

After each modification, the single routes are checked for feasibility. If they are feasible with respect to the vehicle capacities and the allowed penalty time per vehicle, K^{new} and d^{new} are calculated and compared to the values K^{curr} and d^{curr} of the current solution S^{curr}. If any of these values is better, the solution goes into a list of improvement possibilities $L'_{\text{im}} = \{\text{Im}'_1, \text{Im}'_2, \ldots, \text{Im}'_\kappa\}$.

If a solution is infeasible because the customers of the end part of the second newly constructed route R_{new_2} exceed the given penalty limit, the examination of the route is terminated because further linkings would only insert more customers at the beginning of R_{new_2} and accordingly increase the penalty. The process continues by linking the current starting sequence to the first customer of the next route (in case routes that have not yet been examined exist). If there is no such route left, the current starting sequence is extended and the procedure repeated.

Finally, after all linking possibilities have been explored, the solutions in L'_{im} which require the smallest number of vehicles are selected and build the list L'_v. Since the 2-opt* alters only two routes within one improvement step, the improvement of S^{curr} is carried out analogously to the improvement in the Or-opt exchange procedure. A new solution S^{new} with a shorter total distance or a smaller vehicle number results.

3.5. Penalty assumptions

It is assumed that a vehicle arriving too early has to wait and that this time is not penalized. The cumulative penalty of each vehicle is at next limited to zero. The limit is gradually raised with an increment to generate different solutions. The smaller the increments, the more often the improvement procedures must be repeated to improve the solution. The penalty limit is always increased when none of the improvement procedures is able to reduce the total distance or the number of vehicles any further. The solutions that could not be improved anymore within a certain penalty limit are stored in a list L_{res}. After the maximum penalty limit per vehicle has been reached, the solutions in the list L_{res} are checked for Pareto dominance. Inferior solutions must be deleted so that only non-dominated solutions are left and the decision maker can choose the compromise between penalties and total costs that is considered to be most suitable.

In each step of the improvement procedures the arrival times of the vehicles at the customers' locations are calculated. If i is the first customer of the route served by vehicle k, it is assumed that the vehicle arrives at the time $t_{\text{ar}_{\text{first}}} = \max\{t_{0i}, a_i\}$. If the lower bound of the customer's time window a_i is bigger than the time t_{0i} the vehicle needs to drive from the depot to customer i, i.e. $a_i > t_{0i}$, the waiting time at the first customer can be reduced to zero. During each improvement step the rearranged route R_p is examined with respect to time window feasibility. Assuming customer $v_{R_p,j}$ is supplied too late, a back shifting of the starting times at the customers $v_{R_p,1}, v_{R_p,2}, \ldots, v_{R_p,j-1}$ is not possible because the vehicle starts the service at the first customer already at the earliest possible time.

At the end of the algorithm the optimal departure times of the vehicles from the depot are calculated for the given tours to minimize the overall waiting times. Assuming that the vehicle arrives at n customers too early and at none too late. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be their position in the route, e.g. $\alpha_4 = 19$ means, that the fourth customer at which the vehicle arrives too early is the 19th customer in the route. With the notation of Eq. (4), the time t_{post} that has to be added to the current starting times is calculated as follows:

$$t_{\text{post}} = \sum_{j=i}^{n} t_{e_{\alpha_j}} \tag{18}$$

for all customers that are situated within $[\alpha_{i-1}, \alpha_i - 1], i = 1, \ldots, n$ ($\alpha_0 := 1$ denotes the first customer in the route). Thus, if the vehicle arrives, for example, at the customers $v_{R_q,j}, v_{R_q,k}$ and $v_{R_q,l}$ too early, the starting times at the customers at positions 1 to $j-1$ and the time the vehicle leaves the depot are $t_{\text{post}} = t_{e_j} + t_{e_k} + t_{e_l}$ time units later, i.e. the sum of the times the vehicle arrives too early at customers j, k and l. Accordingly, the starting times at the customers at the positions j to $k-1$ are postponed for the time $t_{\text{post}} = t_{e_k} + t_{e_l}$, and the starting times at the customers at the positions k to $l-1$ are postponed for $t_{\text{post}} = t_{e_l}$. The starting times at the customers situated after $l-1$ are not changed. Such a shifting is possible without causing any additional penalty only if the time span between the starting time to be altered and the corresponding upper bound of the customer's time window is bigger or equal than the calculated t_{post}. It is assumed that the penalty time is not allowed to be increased by shifting the starting times. Therefore, if shifting for the whole time t_{post} is not possible, only the allowed fraction will be realized.

In case vehicle q arrives too late at customer $v_{R_q,j}$, it is checked if it arrived too early at any customer $v_{R_q,i}$, $i = 1, \ldots, j-1$. If this is the case, a shifting according to Eq. (18) can be performed for the customers served before $v_{R_q,j}$.[6] If $v_{R_q,k}, k \in \{1, \ldots, j-1\}$, is a customer at which the vehicle arrives before a_k, the service starts at a_k. The time the vehicle departs later from the depot results in an arrival at customer k latest at the time a_k. Thus, a postponed departure from the depot reduces the waiting time at the customers situated before $v_{R_q,j}$ without increasing the penalty. If the vehicle arrives too early at customers situated after $v_{R_q,j}$, a shifting of the starting times is not possible without the simultaneous increase of the penalty at customer $v_{R_q,j}$ by the shifted time span.

The procedures described above were arranged in the following order:

1. Set initial penalty to 0, compute initial solution with route building heuristic.
2. Repeat ECP until no more improvements possible.
3. Repeat Or-opt procedure until no more improvements possible.
4. If any improvement in 3., repeat ECP until no more improvements possible. Else, go to 5.
5. Repeat or-opt* procedure until no more improvements possible.
6. If any improvement in 4. or 5., go to 3.
7. Else, store current solution, raise penalty limit, go to 2.

4. Test setup

The algorithms were implemented in MATLAB R2007a and tested on the classical 56 benchmark problems designed by Solomon (http://web.cba.neu.edu/~msolomon/problems.htm). These problems can be divided into 6 data sets, denoted by R1, C1, RC1, R2, C2 and RC2. Each test problem contains 100 customers and one central depot. The geographical data (Cartesian coordinates), demands, time windows and service times of the customers and the depot are given. The depot's service time and demand are zero. The vehicle fleet is homogeneous and the capacity is defined according to the data set.

The customers of the classes R1 and R2 are uniformly distributed, whereas the customers of classes C1 and C2 are arranged in clusters. The problem sets RC1 and RC2 constitute a mixture of the classes R and C. The classes contain 8–12 different problem instances. The instances within each problem class differ only with respect to the customers' time windows. The customers' coordinates within one class are the same.

R1, C1 and RC1 problems have short scheduling horizons and the vehicles have only small capacities. Therefore, each vehicle

[6] Analogously to the case where none of the customers is served too late, the penalty times must not increase.

serves only a few customers. In contrast, R2, C2 and RC2 instances have longer scheduling horizons and the vehicles have higher capacities. Each vehicle supplies more customers and therefore, compared to the type 1 problems, fewer vehicles are needed. The distances d^{ij} between the nodes i and $j, i, j \in V$, were calculated from their Cartesian coordinates using euclidean distances. These distances were also used as travel times.

5. Simulation results

The objective behind raising the penalty limit gradually was to obtain solutions that cause as few costs as possible and penalties within a certain manageable range. The different characteristics of the problem classes influenced the behavior of the applied heuristics with respect to the computation time, and also the outlook of the calculated approximate Pareto fronts. The results for the single classes are analyzed in the following. The results were calculated under the assumption of a maximum penalty limit per vehicle of 500 and a step size of 20.

5.1. Classes C1 and C2

The results for the C1 and C2 problems had in general the form as in Fig. 1, where an approximate solution of the problem C102 is illustrated (the total costs are plotted against the penalties, the numbers next to the data points indicate the amount of used vehicles). The Pareto front consists of only few solutions. Improvements with respect to the number of used vehicles were not possible due to the capacity constraints. On the other hand, also the savings in total distance were very small compared to the arising penalties.

Similar statements hold for the problems of class C2. Improvements with respect to the number of used vehicles were not possible due to vehicle capacities. Also distance savings were rare and for the problem C206, for instance, only one Pareto point could be calculated and the raise of the allowed penalty limit to 500 did not yield any distance-improved solution.

Those results can be explained by the geometry of the customers. Since the C1 and C2 problem instances contain customers arranged in clusters, the distances between customers within one cluster are smaller compared to the distances between customers of different clusters. Thus, in a vehicle- and distance-optimal solution it can be expected that at most one vehicle serves one cluster of customers in case time window and capacity constraints allow it. If such an assignment of vehicles to clusters is possible, it is unlikely that the exchange of customers between routes would lead to distance improvements. The corresponding routes of solutions at different penalty levels proved this expectation. Most distance savings were obtained by interchanging customers within a route which indeed caused delays at other customers. However, the resulting distance savings were evanescently small compared to the arising penalty. Since the main cost driver, i.e. the number of used vehicles, could not be decreased because of the capacity constraints, and the achieved distance savings by allowing penalties were inefficient, it does not seem favorable to accept penalties in order to save distance units.

5.2. Classes R1, RC1, R2 and RC2

The obtained results for the classes R and RC were very similar to each other. The results of the RC1 and R1 problems left more opportunities to choose between different trade-offs between total costs and penalties. In these problems especially the setting of the unit costs had a big influence on the outlook of the Pareto front. According to the calculated solutions, the R1 and RC1 problems required more vehicles than conditioned by the capacity constraints. Thus, it can be expected that besides distance savings, also improvements with respect to the number of used vehicles are possible.

The customers are not arranged in clusters so that there are more chances to change the vehicles' routes in order to save distance units. The problem of causing higher distances by traveling from one cluster to another, as it was the case in the classes C1 and C2, cannot appear. On the other side, saving vehicle units can also cause an increase in total distance. In those cases it depends on the vehicle and distance unit costs which solution belongs to the Pareto front. Assuming the use of one vehicle less results in a decrease of the total costs of w_1 units, but simultaneously causes an increase of the total distance which, multiplied by the distance unit costs, gives a total of w_2 units. In case $w_2 > w_1$ the solution with one vehicle less and more total distance is dominated by the solution with more vehicles and less total distance. An example is illustrated in Fig. 2 (all unit costs were set to 1). In contrast, Fig. 3 shows the results for the same problem when the vehicle fixed costs are 1000 and the distance unit costs are 1.

As can be seen in Fig. 3, the Pareto solution using 11 vehicles caused at least an overall penalty of 119.6 units. Depending on the vehicle and distance unit costs, this may constitute an acceptable burden compared to the costs of four additional vehicles. The approximate Pareto optimal solutions in Fig. 3 can be classified into groups according to the number of used vehicles. The gaps between the single groups are bigger compared to the gaps between the single data points within one group. This characteristic of the Pareto front can be explained by the setting of the vehicle fixed costs and the distance unit costs. In case those costs would differ only slightly from each other, results as in Fig. 2 are obtained. However, this difference is much bigger in real life problems. Thus, saving one vehicle results in a very large cost reduction (see solutions marked with a circle), and compared to that, distance savings result in a significantly smaller reduction (see solutions marked with an asterisk). From this rationalization follows that the real Pareto

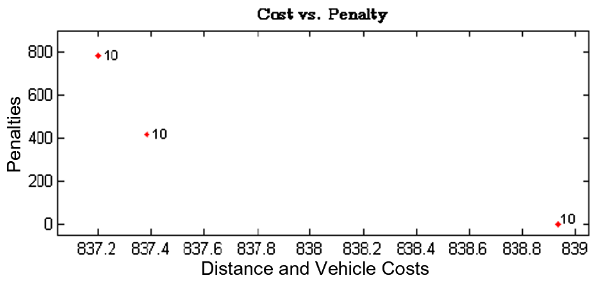

Fig. 1. The result of the algorithm for the problem C102. The maximum penalty limit was set to 500, the step size was set to 20, all unit costs were set to 1.

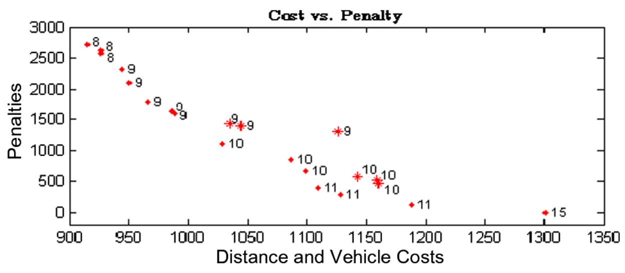

Fig. 2. The result of the algorithm for the Problem R103. The maximum penalty limit was set to 500, the step size was set to 20. All unit costs were set to 1. The data points marked with an asterisk illustrate the solutions in which the usage of less vehicles causes an increase in total distance, and thus dominated solutions.

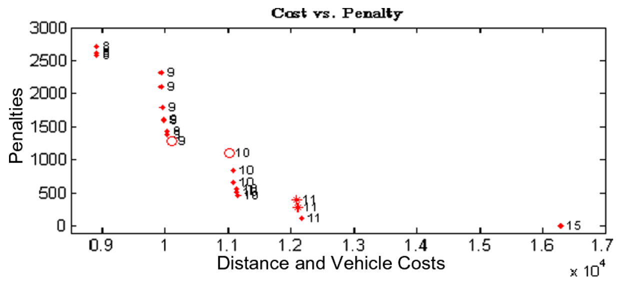

Fig. 3. The result of the algorithm for the Problem R103. The maximum penalty limit was set to 500, the step size was set to 20. The vehicle fixed costs were set to 1000. The distance unit costs were set to 1.

front must have a similar form like the approximate solutions obtained by the applied algorithm.

The obtained Pareto front for the problem RC104 is illustrated in Fig. 4. In this example all unit costs were set to 1. Noticeable is the penalty gap between the solutions requiring nine and ten vehicles. The decision whether a solution with nine vehicles is acceptable or not, is totally dependent on the unit costs. The higher the vehicle fixed costs are compared to the arising penalties, the more beneficial the 9-vehicles solution becomes.

The form of the approximated Pareto fronts for the problems of the class R2 turned out to be similar to those of the R1 problems with the difference that less groups of data points resulted. The minimum number of required vehicles for the R2 problems was two. The number of used vehicles in the initial solutions was already three and four, respectively, and accordingly less data point groups resulted.

The results for the problem R202 are illustrated in Fig. 5. Here the vehicle fixed costs were set to 1000 and the distance unit costs were set to 1. In Fig. 6 the same problem is illustrated with all unit costs set to 1. The figures emphasize how important the difference between distance unit costs and vehicle fixed costs is. The bigger this difference, the more beneficial the solutions with less vehicles become. However, savings were usually only achieved with respect to distance units. The results also showed that the ratio between the amount of saved distance and the amount of caused penalties varied drastically. Thus, the distance costs and the consequences of accepting penalties must be carefully analyzed in advance by the decision maker to decide which trade-off the most beneficial for the problem under consideration is.

Due to the customers' demands and the vehicle capacities, the solutions to the problems of type RC2 required at least two vehicles. Also in these cases vehicle savings were rarely possible. The results for the problem RC202 are shown in Figs. 7 and 8. Fig. 7 illustrates that several 3-vehicle solutions were dominated by the 4-vehicle solutions because using less vehicles caused more total distance. With an increasing difference between the vehicle fixed costs and the distance unit costs, the 3-vehicle solutions became non-dominated (Fig. 8). As in the R2 case, also the ratios between saved total costs and penalties varied. Remarkable vehicle savings as, for example, in the case R103, were not achieved because it was not possible to reduce the number of vehicles by more than one and two, respectively. Therefore, distance savings were usually the only improvement possibilities (see Fig. 9).

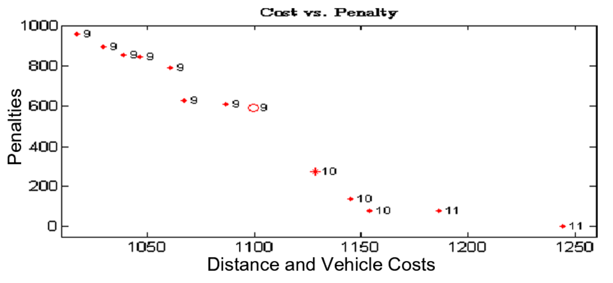

Fig. 4. The result of the algorithm for the Problem RC104. The maximum penalty limit was set to 500, the step size was set to 20. All unit costs were set to 1.

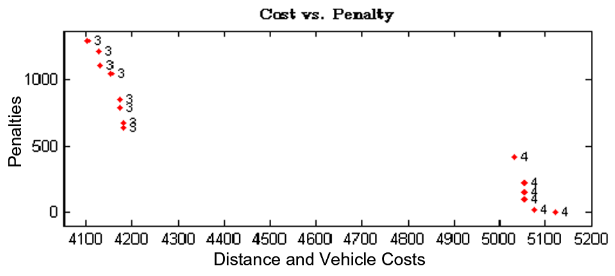

Fig. 5. The result of the algorithm for the Problem R202. The maximum penalty limit was set to 500, the step size was set to 20. The vehicle fixed costs were set to 1000 while the distance unit costs were set to 1.

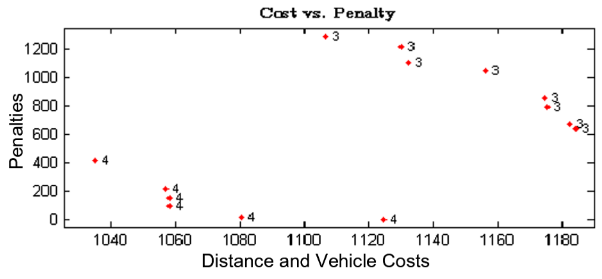

Fig. 6. The result of the algorithm for the Problem R202. The maximum penalty limit was set to 500, the step size was set to 20. All unit costs were set to 1.

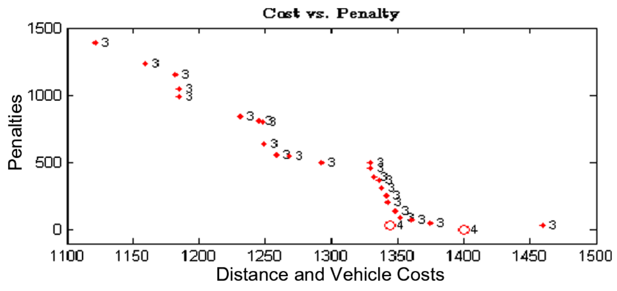

Fig. 7. The result of the algorithm for the Problem RC202. The maximum penalty limit was set to 500, the step size was set to 20. All unit costs were set to 1.

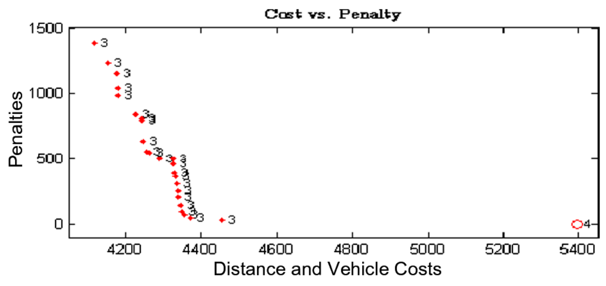

Fig. 8. The result of the algorithm for the Problem RC202. The maximum penalty limit was set to 500, the step size was set to 20. The vehicle fixed costs were set to 1000. The distance unit costs were set to 1.

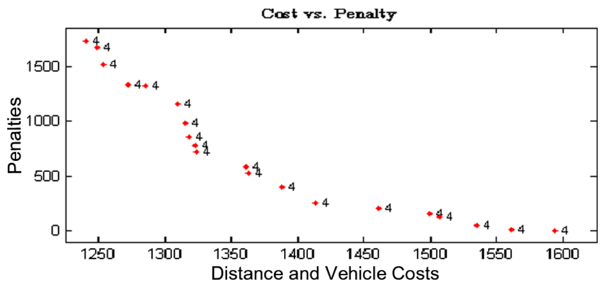

Fig. 9. The result of the algorithm for the Problem RC201. The maximum penalty limit was set to 500, the step size was set to 20. All unit costs were set to 1.

6. Conclusions

The purpose of this study was to examine the Vehicle Routing Problem with Soft Time Windows from a bicriterion point of view. The goal was to find trade-offs between the total traveling costs, consisting of vehicle fixed costs and total traveled distance, and the penalties which were caused by supplying customers too late for the sake of saving total transportation costs. Approximate solutions were obtained by first computing an initial solution using Solomon's I1 insertion heuristic and then optimizing these solutions by applying ejection chain, Or-opt and 2-opt* procedures.

The implemented procedures were tested on Solomon's 56 benchmark problems, and the results showed that the possibility of improving the solutions by allowing penalties as well as the form of the corresponding Pareto fronts mainly depended on the geographical distribution of the customers. In the clustered problem classes C1 and C2, improvements with respect to the number of vehicles and the total distance were rarely possible. In contrast, for the problems of classes R1 and RC1, significant savings in the amount of used vehicles and distance units were obtained. For the instances of classes R2 and RC2, mainly improvements with respect to the total distance were achieved due to the vehicles' capacities.

The results also indicate that the form of the Pareto front strongly depends on the setting of the unit costs. The higher the difference between distance unit costs and vehicle fixed costs were, the more attractive the solutions requiring less vehicles became. In real-world instances this difference can be assumed to be rather high, regarding all additional costs that are linked to the usage of an additional vehicle (insurance costs, drivers' wages, maintenance, etc.). Therefore, it is reasonable to think about savings in the number of used vehicles and to accept penalties in turn.

As mentioned before, it is important to carefully analyze the consequences of accepting delayed deliveries when evaluating routing plans. If the customers' characteristics and their importance to the company are known, i.e. purchasing behavior, expected sales, payment duration, it might also be advantageous to give priority to some customers and not allow any delays for them. These priority customers could be, for example, bulk buyers who are from the sales point of view more important than customers who buy only a small amount once in a while. However, the analysis of customer behavior and possible consequences for the delivering company go beyond the scope of this paper.

In conclusion it should be pointed out again that in this study approximations of the Pareto fronts were calculated using the described heuristic methods. However, the results suggest that also the exact Pareto fronts will have the characteristic outlooks as described in Section 5 due to the fact that the reduction of used vehicles results in more savings than a mere reduction of the total distance. The application of metaheuristics to the VRPSTW according to the described approach in order to find the exact Pareto fronts is left as a future research topic.

Acknowledgements

The helpful comments and suggestions by the anonymous reviewers are gratefully acknowledged.

References

Arbelaitz, O., Rodriguez, C., Zamakola, I., 2001. Low cost parallel solutions for the VRPTW optimization problem. In: International Conference on Parallel Processing Workshops.
Barán, B., Schaerer, M., 2003. A multiobjective ant colony system for vehicle routing problem with time windows. In: Proceedings of the 21st IASTED International Conference.
Bullnheimer, B., Hartl, R.F., Strauss, C., 1999. An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research 89, 319–328.
Clarke, G., Wright, J.W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581.
Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B., 2002. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers.
Colorni, A., Dorigo, M., Maniezzo, V., 1991. Distributed optimization by ant colonies. In: Proceedings of the European Conference on Artificial Life, pp. 134–142.
Czech, Z.J., Czarnas, P., 2002. Parallel simulated annealing for the vehicle routing problem with time windows. In: Proceedings of 10th Euromicto Workshop on Parallel Distributed and Network-Based Processing, Canary Islands, Spain, pp. 376–383.
Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Management Science 6, 80–91.
Desrochers, M., Desrosiers, J., Solomon, M.M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40, 342–354.
Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. Management Science 40, 1276–1290.
Goldberg, D.E., Holland, J.H., 1988. Genetic algorithms and machine learning. Machine Learning 3, 95–99.
Holland, J.H., 1975. Adaption in Natural and Artificial Systems. The University of Michigan Press.
Jong, K.A.D., 1975. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Thesis, University of Michigan.
Kohl, N., Madsen, O.B.G., 1997. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. Operations Research 45, 395–406.
Kohl, N., Desrosiers, J., Madsen, O.B.G., Solomon, M.M., Soumis, F., 1999. 2-path cuts for the vehicle routing problem with time windows. Transportation Science 33, 101–116.
Kolen, A.W.J., Kan, A.H.G.R., Trienekens, H.W.J.M., 1987. Vehicle routing with time windows. Operations Research 35, 266–273.
Miettinen, K.M., 1999. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers.
de Oliveira, H.C.B., Vasconcelos, G.C., Alvarenga, G.B., 2006. A multi-start simulated annealing algorithm for the vehicle routing problem with time windows. In: Proceedings of the Ninth Brazilian Symposium on Neural Networks.
Or, I., 1976. Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking. Ph.D. Thesis, Northwestern University, Evanston, III.
Osman, I.H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research 41, 421–451.
Potvin, J.-Y., Rousseau, J.-M., 1995. An exchange heuristic for routeing problems with time windows. Journal of the Operational Research Society 46, 1433–1446.
Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265.
Taillard, E., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science C, 109–122.
Toth, P., Vigo, D., 1998. The Granular Tabu Search (and its Application to the Vehicle Routing Problem).
Zhong, Y., Pan, X., 2007. A Hybrid Optimization Solution to VRPTW Based on Simulated Annealing.