# A Computational Study of Genetic Crossover Operators for Multi-Objective Vehicle Routing Problem with Soft Time Windows

Martin Geiger

University of Hohenheim, Production and Operations Management Department, 70593 Stuttgart, Germany

**Abstract.** The article describes an investigation of the effectiveness of genetic algorithms for multi-objective combinatorial optimization (MOCO) by presenting an application for the vehicle routing problem with soft time windows. The work is motivated by the question, if and how the problem structure influences the effectiveness of different configurations of the genetic algorithm. Computational results are presented for different classes of vehicle routing problems, varying in their coverage with time windows, time window size, distribution and number of customers. The results are compared with a simple, but effective local search approach for multi-objective combinatorial optimization problems.

## 1 Introduction

The requirements of modern logistics are widespread. It seems natural that for practical problems several objectives have to be optimized. The quality of service and products and the corresponding cost are only a small subset of necessary objectives that should be considered [25].

The distribution of products, a part of the supply chain in logistics, can be modelled as a vehicle routing problem (VRP). The objective is to find a cost-minimizing set of routes from a depot to serve a number of customers with known demands. Each customer is serviced by exactly one of the vehicles which capacities are not allowed to be exceeded.

The model of the vehicle routing problem with time windows (VRPTW) generalizes the basic problem by introducing time windows for each customer, defining an earliest and a latest possible time of service. In the case of the vehicle routing problem with soft time windows (VRPSTW), violation of these time windows is possible and does not lead to infeasibility of the solution [1]. In our approach, the possible violations of the time windows are modelled by introducing objective functions, leading to a multi-objective model that is able to describe the requirements of practical problems more detailed compared to single-criterion optimization approaches. The goal is to find all solutions minimizing the defined objective functions. As these functions are often conflicting, the concept of Pareto optimality is used to determine appropriate solutions.

The VRPTW [31], as well as the VRP [23], has been shown to be NP-hard in terms of complexity. In combination with the practical relevance, this is one of the reasons why there is still a lot of ongoing research. Earlier approaches include heuristics and optimization algorithms. Heuristics are focused on tour construction [31], tour improvement [28] or both in parallel [26,27].

Exact optimization methods successfully apply branch-and-bound methods [22] or integer linear programming based on Lagrangian relaxations [11,12,13]. However, only smaller problem instances up to 50 customers can be solved reliable [21].

More recent, metaheuristic strategies have been applied to the VRP and the VRPTW. They include Simulated Annealing [5,24], Tabu Search [30,34], Genetic Algorithms (GAs) [35,36], Ant Colony Optimization [4], hybrid approaches [37,38] and newer local search concepts like e.g. Guided Local Search [20].

Nevertheless, the focus of the treated models is in the observed cases on minimizing a single objective, the cost of the solution, by minimizing the distances travelled by the used vehicles, they do not address a more practical, multi-objective formulation with a relaxation of the restrictive time windows.

On the other side, there is an increasing interest on applying evolutionary based optimization techniques to multi-objective problems. Evolutionary algorithms are regarded to be well suited for multi-objective problems since a set of alternatives is used in the optimization process and the goal is the approximation of a set of Pareto optimal solutions. The idea is the convergence of the whole population towards the efficient frontier.

Since a first sketch of an idea by Goldberg [17], most of the present work on multi-objective optimization using evolutionary algorithms are focused on the integration of the objective functions in the calculation of the fitness values of the solutions, ranging from scalarizing functions to Pareto-based techniques. Here to mention are VEGA of Schaffer [29], MOGA of Fonseca and Fleming [14], NPGA of Horn et al. [19], NSGA of Srinivas and Deb [32] and SPEA of Zitzler [41]. These approaches include the ideas of fitness functions/fitness sharing, niching, mating restrictions and elitism to define algorithms that maintain diversity within the population, overcome local optima and finally converge towards the set of efficient alternatives. For a detailed overview concerning these topics, the interested reader is referred to Coello and Van Veldhuizen [7,39]. Although big progresses were made and a huge variety of applications were already successfully developed [6], an application to the vehicle routing problem under multiple objectives is still missing. To define an appropriate algorithm, the interchange between the problem structure, the configuration and the behavior of the algorithm has to be studied. Especially for the correct use of crossover operators in multi-objective optimization problems, computational results are only for a few examples available [3].

This paper fills the gap in the described field of research. A practical, multi-objective model of a VRPSTW is presented and a genetic algorithm is defined to solve different problem instances, varying in their time window coverage, time window size, number and distribution of customers. Several crossover operators are tested and computational results are given for the different instances. To compare the results obtained by the GA, a simple local search approach for multi-objective combinatorial optimization is presented.

## 2 The multi-objective vehicle routing problem with soft time windows

### 2.1 Multi-objective optimization

The goal of a multi-objective optimization is to

$$\text{“}\min\text{”}\, G(R) = \big(g_1(R), ..., g_k(R)\big) \tag{1}$$

$R \in \Omega$ is a solution of the problem and belongs to the set of all feasible solutions $\Omega$. The objective functions $g_1(R), ...g_k(R)$ map the decision $R$ into the objective space, leading to a objective vector $G(R)$. As often conflicting objective functions $g_k(R)$ are considered, minimization does not lead to a single optimal solution but is understood in the sense of efficiency (or Pareto optimality) [39].

**Definition 1.** An objective vector $G(R)$ is said to dominate $G(R')$, if $g_i(R) \leq g_i(R') \forall i \in \{1, ..., k\} \wedge \exists i \in \{1, ..., k\} | g_i(R) < g_i(R')$. We denote the domination of a vector $G(R)$ to the vector $G(R')$ with $G(R) \prec G(R')$.

**Definition 2.** A solution $R \in \Omega$ is said to be efficient or Pareto optimal, if $\neg \exists R' \in \Omega | R' \prec R$. Set set of all solutions fulfilling this property is called the Pareto set $P$.

From the description of the multi-objective optimization problem in equation 1 we derive in combination with the definitions 1 and 2 the goal to find all $R \in P$.

### 2.2 Problem description

The vehicle routing problem with soft time windows can be described as follows:
A known number of customers have to be delivered from a depot with a known amount of goods for which an unlimited number of homogeneous vehicles is available. It is assumed that each customer is visited by exactly one vehicle and a loading and a travelling constraint exists for the vehicles. A soft time window is associated with each customer, defining a desired earliest and a latest time of service. Violation of these time windows does not lead to infeasibility of the solution. With respect to the soft nature of the time windows, it is assumed that service is done immediately after the arrival of

the vehicle.
The objective of the problem is to maximize quality of service and to minimize cost, such that the requirements of the customers and the side-constraints are met. It is obvious, that the violation of the time windows has to be minimized in order to achieve a high quality of service. This can be done by minimizing the number of time window violations and the time window violations itself, measured in time units.
The cost consist of a fixed part, induced by the number of used vehicles and a variable part, caused by the route length and the travel time.

### 2.3 Notation of the model

The customers and the depot are modelled as vertices $V = \{v_0, v_1, ..., v_N\}$ in a graph $G = (V, A)$, connected by a set of arcs $A$. In $V$, the depot is indexed by 0 and the total number of customers is $N$.
With each arc $(v_i, v_j)$ from node $v_i$ to $v_j$, a nonnegative travel time $t_{v_i,v_j}$, a distance $d_{v_i,v_j}$ and a cost $c_{v_i,v_j}$ is associated. It is assumed that $t_{v_i,v_j} = d_{v_i,v_j} = c_{v_i,v_j} \quad \forall (v_i, v_j) \in A$.
$u_{v_i}$: unloading time at customer $v_i, v_i \in V$.
$d_{v_i}$: demand of customer $v_i, v_i \in V$.
$d^{max}$: capacity constraint of the vehicles.
$[a_{v_i}, b_{v_i}]$: time window at customer $v_i, v_i \in V$. $a_{v_i}$ defines the desired earliest, $b_{v_i}$ the latest time of service. It is necessary that $b_{v_i} \geq a_{v_i} \quad \forall v_i \in V$.
$t^a_{v_i}$: arrival time of the vehicle at customer $v_i, v_i \in V$.
$r_k = \big[[1]_k, [2]_k, ..., [n_k]_k\big]$: route $r_k$, representing $n_k$ customers in a given sequence. To assure, that each customer is included in exactly one route, it is necessary that $r_k \cap r_{k'} = \emptyset \quad \forall k \neq k'$ and $\mid \cup r_k \mid = \mid V \setminus \{v_0\} \mid$.
$R = \{r_1, ..., r_m\}$: set of routes of the solution. The number of routes $m$ can be greater than one and does not have to be equal in every solution.
The time $t(r_k)$ to travel route $r_k$ can now be obtained by calculating

$$t(r_k) = a_{v_0} + t_{v_0,[1]_k} + t_{[n_k]_k,v_0} + \sum_{[i]_k=1}^{n_k-1} (t_{[i]_k,[i+1]_k} + u_{[i]_k}) + u_{[n_k]_k} \tag{2}$$

As the time window $[a_{v_0}, b_{v_0}]$ defines the interval in which the vehicles are available, it is necessary that

$$t(r_k) \leq b_{v_0} \quad \forall r_k \in R \tag{3}$$

The quantity $q(r_k)$ of transported goods on route $r_k$ is given by

$$q(r_k) = \sum_{i=1}^{n_k} d_{v_i} \tag{4}$$

The capacity constraint must be met, so

$$q(r_k) \leq d^{max} \quad \forall r_k \in R \tag{5}$$

$w(v_i)$ measures the time window violation of customer $v_i, v_i \in V$.

$$w(v_i) = \max\left(\max(0; a_{v_i} - t^a_{v_i}); \max(0; t^a_{v_i} - b_{v_i})\right) \tag{6}$$

The fact, that the time window of customer $v_i$ is violated is indicated by $u(v_i)$:

$$u(i) = \begin{cases} 1 & : \quad w(i) > 0 \\ 0 & : \quad w(i) = 0 \end{cases} \tag{7}$$

The objective functions are given by:

$$g_1(R) = \sum_{k=1}^{|R|} t(r_k) \tag{8}$$

$$g_2(R) = \mid R \mid \tag{9}$$

$$g_3(R) = \sum_{i=1}^{N} w(v_i) \tag{10}$$

$$g_4(R) = \sum_{i=1}^{N} u(v_i) \tag{11}$$

The objective functions (8) and (9) minimize the costs associated with the solution. Function (8) minimizes the length of the routes, travelled by the vehicles. In addition to that, function (9) minimizes the number of used vehicles. To maximize the provided service it is desired to minimize occurring time window violations. The objective functions (10) and (11) express this circumstance by the measured deviations in time units and by the number of violated time windows.

According to the definitions (1) and (2) given in section 2.1, the goal is to find all $R \in P$.

## 3 A genetic approach

### 3.1 Encoding technique, initialization

We use a string of $N$ genes to encode the solutions of the problem. Each gene represents a customer of the problem and the sequence of them corresponds to the sequence of visiting the customers. A possible string, e.g. `"5 2 1 3 7 9 8 4 6 10"` is then partitioned to a set of tours, e.g. $r_1 = [v_5, v_2, v_1], r_2 = [v_3, v_7, v_9, v_8], r_3 = [v_4, v_6, v_{10}]$, assuring feasibility with respect to the defined side constraints (3) and (5) in section 2.3. The decoding technique inserts the next stored customer of the representation in the current tour as long as the side constraints are not being violated.
The initial population inherits $n^{pop}$ individuals $i$, consisting of randomly generated strings. To assure a sufficient potential of genetic information and avoid premature convergence, $n^{pop}$ is chosen equal to 500.

### 3.2 Fitness function

Most investigations on genetic algorithms for multi-objective optimization are based on examining the effectiveness of certain approaches of fitness functions. As we have a different focus, we restrict ourselves to the use of a single function, similar to the the well known fitness function proposed by Fonseca and Fleming [14].
For individual $i$ of the current population, the number of individuals whose corresponding objective vectors dominate the objective vector of $i$ is indicated by $\xi_i$. All currently not being dominated alternatives receive $\xi_i = 0$. The fitness $f(i)$ of individual $i$ is derived using a linear transformation with $\xi^{max} = \max_i(\xi_i)$ and two external given parameters $f^{min}$ and $f^{max}$:

$$f(i) = f^{max} - \frac{\xi_i(f^{max} - f^{min})}{\xi^{max}} \tag{12}$$

The parameters $f^{min}$ and $f^{max}$ are set to $f^{min} = 1$ and $f^{max} = 100$.

### 3.3 Genetic recombination operators

As we want to study the effectiveness of different crossover operators, several of them are implemented and tested. From the first approaches of genetic crossover operators, improvements were made by incorporating problem specific knowledge into the recombination process. This is in the case of the travelling salesman problem as well as in the case of the vehicle routing problem e.g. done by considering the distances of the customers to each other and selecting between two possible genes the route minimizing part.
As we deal with more than one objective, problem specific orientation during the crossover process is not as obvious. We therefore restrict the investigation

in this first approach on problem independent working operators.
Among the best studied crossover operators for permutation based encodings are the "partially mapped crossover" (PMX) [16], the "order based crossover" (OBX) [33] and the "uniform order based crossover" (UOBX) [9]. We apply them with a crossover probability of $p^{cross} = 1$.
Crossover operators introduce implicit mutations to the offspring. To distinguish between the true results obtained by crossover, no mutation operator is used in the basic test runs. However, for comparison reason a simple "swap mutation" exchanging two randomly picked genes is examined in a separate test run in combination with the UOBX as this crossover operator is being described as the best among the problem independent operators for the considered problem type. In this configuration (UOBX$\wedge$2EX) , it is applied with a probability of $p^{mut} = 0.1$ for the whole individual. Thus, the probability of an exchange of a single gene is $\frac{2p^{mut}}{N}$.

### 3.4 Elitism

It has been shown, that elitism plays in general an important role in genetic algorithms and in particular in multi-objective GAs, see e.g. [41]. Various approaches exist to implement an elitist strategy, ranging from preserving the $k$ best individuals in the population to storing the best individuals in an external set.
We use a simple, but effective strategy to obtain an elitist genetic algorithm by using overlapping populations. This principle has been introduced as the so called "steady state" genetic algorithm. From each generation to the next, the population is preserved and new individuals are only inserted if they improve the average quality of the population. Duplicates are not allowed, including duplicates of encoded strings as well as duplicates of the corresponding solutions in the decision space. The improvement of the population is in our case measured by the $\xi_i$-values. This means that the complexity of the necessary calculations to obtain the $\xi_i$-values by pairwise comparison is $2n^{pop}$.

### 3.5 Termination criterion

The algorithm can terminate after a defined time, a given number of calculations or after the algorithm has converged to a given goal and does not produce any improvements for a defined time or number of calculations. It is certainly interesting to perform calculations until a convergence to a local or global optimal front is reached. The termination criterion is chosen according to this goal. The genetic algorithm stops after 10,000 iterations with no found new individual $i$ having a $\xi_i = 0$.

## 4 A local search approach

A simple local search approach is used to compare the results obtained by the genetic algorithm. It is based on the encoding scheme of section 3.1 and can be described as follows:

1 initialize
  1.1 generate starting solution $R$
  1.2 encode starting solution $R$
  1.3 set $P^{approx} = \{R\}$
2 repeat until neighborhood of $R \in P^{approx}$ is investigated $\forall R \in P^{approx}$
  2.1 select new solution $R \in P^{approx}$
  2.2 generate neighborhood of $R$
  2.3 update $P^{approx}$
  2.4 set neighborhood of $R$ as investigated

The algorithm starts from a random generated solution. The calculation of the initial solution is done as in the initialization procedure of the genetic algorithm to assure proper comparison results. A set $P^{approx}$ is used to store the set of currently not dominated solutions. According to the given definitions (1) and (2) in section 2.1 it has to be noticed that these solutions do not necessarily fulfill the requirements of Pareto optimality. However, as we do not know the real efficient solutions and the true Pareto set $P^{true}$, we consider the nondominated solutions as an approximation of the Pareto set $P^{true}$.

The algorithm then starts in step 2 to investigate the neighborhood of the solutions in $P^{approx}$. A solution $R$ is randomly selected from the set $P^{approx}$ and the neighboring solutions are calculated. Procedure 2.3 then inserts new nondominated solutions of the neighborhood of $R$ into the set $P^{approx}$ and removes dominated ones.

The local search procedure terminates when the neighborhood all solutions $R \in P^{approx}$ has been investigated and as result a set of locally optimal solutions with respect to the used neighborhood is obtained.

An approach based on the neighborhood proposed by Lin is used to perform local search. Every possible substring of the encoding is extracted, reversed and reinserted, resulting in $\frac{N(N-1)}{2}$ calculations. The basic ideas of this method come from the local search descent method, incorporating multi-objective storage of nondominated solutions. We call this approach therefore multiple-objective local search descent (MOLSD).

# 5 Computational experiments

## 5.1 Test problems

To investigate the influence of the problem structure on the effectiveness of different configurations of the algorithms proposed above, a set of test instances incorporating different characteristics of the problem is necessary.
For that purpose, 40 test instances based on the data given from Solomon [31] were created. Each instance can be described using an $\alpha; \beta; \gamma; \delta$-classification scheme, representing the following configurations:

- $\alpha$: Distribution of the customers.
  We consider test instances with clustered data sets ($\alpha = C$) and with random distributed customers ($\alpha = R$).
- $\beta$: Number of customers.
  Problems with 20 and 30 customers were defined.
- $\gamma$: Coverage with time windows.
  This attribute measures the relation of the number of customers having time windows to the number of customer without time windows.
- $\delta$: Average time window size.
  The size of the time window at customer $v_i$ is defined as $b_{v_i} - a_{v_i}$.

Example: The string "C;20;0.70;60" describes the instance with a clustered distribution of 20 customers, having a density of 70% with time windows and an average time window size of 60 time units.
We expect the problems with random distribution of the customers and a high density of small time windows to be more complex to solve than problems with less and bigger time windows. This should affect the speed of convergence of the algorithms and the quality of the obtained solutions.
The neighborhood of the defined problems with 30 customers contains 2.29 times more alternatives than the one of the problems with 20 customers. An increase of the necessary evaluations of at least this factor is being expected.

## 5.2 Measures of effectiveness

Each of the performed test runs leads to an approximation $P^{approx}$ of the set of efficient alternatives $P$. Due to the complexity of the problem, we do not know the true Pareto set $P$. However, to obtain a measure of effectiveness, we compute a reference set $P^{ref}$ including all the found nondominated solutions during the test runs. The effectiveness of the approximations is then oriented on the approximation of this reference set $P^{ref}$.
To determine the distance between two sets of alternatives, a distance function measuring the distance of the underlying elements according to the the proposal of Czyzak and Jaszkiewicz [8] is used. The distance measure can be described as a weighted achievement scalarizing function, using $w_j = \frac{1}{\Delta_j}$ as a weight for the goals based on the spread $\Delta_j$ of the objective function values

of objective $j$ in the reference set $P^{ref}$. For an alternative $x \in P^{approx}$ to an alternative $y \in P^{ref}$, the distance is measured by equation (13).

$$c(x,y) = \max_{j=1,\ldots,k} \Big\{ 0; w_j \big( g_j(x) - g_j(y) \big) \Big\} \tag{13}$$

Two approaches are used to obtain the distance of the approximation $P^{approx}$ to the reference set $P^{ref}$. We measure the average distance by equation (14) and the worst-case distance by equation (15).

$$d_1 = \frac{1}{\mid P^{ref} \mid} \sum_{y \in P^{ref}} \min_{x \in M} \{ c(x,y) \} \tag{14}$$

$$d_2 = \max_{y \in P^{ref}} \Big\{ \min_{x \in M} \{ c(x,y \} \Big\} \tag{15}$$

As our goal is to minimize the distance between the obtained approximations $P^{approx}$ and the reference set $P^{ref}$, the distances $d_1$ and $d_2$ are to be minimized.

To come to stable and reliable conclusions, average values of $d_1$ and $d_2$ of several test runs with the same configuration are computed.

### 5.3 Results

The test runs were performed on a personal computer, equipped with an 1 GHz Intel Pentium III Processor and 384 MB RAM. Each algorithmic configuration was executed 100 times for each test problem in order to obtain stable results. The average number of evaluations of the test runs and the performance of the implemented heuristics is given in table 1 in the appendix. Resulting from the lower computational effort, the MOLSD is able to evaluate about 4.70 times more solutions per second than the GAs. Each second, 1,385 evaluations were computed by the MOLSD.

As expected, all algorithms tend to converge slower in instances with dense and small time windows and random distributed customers. It is worth to mention, that the increase of the necessary evaluations of the MOLSD from a $\beta = 20$ to a $\beta = 30$ was about factor 9.14, a huge increase compared to the additional computations of the GAs, having factors of 1.80 for the PMX, 1.96 for the OBX, 1.78 for the UOBX and 2.63 for the UOBX$\wedge$2EX.

The results, given in table 2 and 3 in the appendix indicate a strong dependency of the approximation quality on the distribution of the customers $\alpha$. The examined crossover operators PMX, OBX, UOBX itself as well as the genetic algorithm including the swap mutation UOBX$\wedge$2EX were in most instances with an $\alpha = C$ not able to outperform the simple local search approach MOLSD. Both defined measures $d_1$ and $d_2$ are affected by this property and the distance between the values of the MOLSD to the values of the GAs is increasing with falling $\gamma$ or rising $\delta$. We conclude, that VRPSTW with a structured distribution of the customers are more easy to be solved with simple local search while the examined configurations of the genetic algorithm tend to get stucked in local optima. This effect grows with a declining complexity of the underlying problem, expressed by a low coverage with time windows $\gamma$ or wide time windows $\delta$. Similar results have already been reported for the single objective VRPTW [2] and can be confirmed for the multi-objective formulation.

For VRPSTW with a random distribution of the customer we observe the contrary. In most test instances, the best results for the average distance $d_1$ are obtained by the UOBX$\wedge$2EX. However, the advantage of the UOBX$\wedge$2EX to the MOLSD is decreasing with falling $\gamma$ or increasing $\delta$. For some problems with large time windows $\delta$ and a small coverage $\gamma$, e.g. the "R;30;1.00;80/95/115", MOLSD produces better results.

The results of $d_2$ indicate, that the average worst-case deviation of the approximation $P^{approx}$ to the reference set $P^{ref}$ is superior in the approach of the MOLSD, except for some smaller instances with $\beta = 20$ and the hard "R;30;1.00;10".

A closer investigation of the results shows, that the superior results of the UOBX$\wedge$2EX is due to the use of the swap mutation operator. Comparing the results for the crossover-only GAs, it can be seen that with respect to the distances $d_1$ and $d_2$ OBX and UOBX perform in most instances better than PMX. On the other hand, a clear distinction between OBX and UOBX can't be concluded.

The results show a strong influence of the mutation operator on the effectiveness for the genetic algorithm. Although the GAs lead to weaker results in clustered data sets than the proposed local search approach, this behavior seems to be typical for the presented multi-objective optimization problem.

## 6 Conclusions

The goal of the paper was to investigate the influence of the problem structure on the effectiveness of different configurations of a genetic algorithm. A multi-objective formulation of the vehicle routing problem with soft time windows has been presented, describing the requirements of practical logistics more detailed than single criterion models. For comparison reasons, an approach of a local search approach incorporating storage of multiple non-dominated solutions was proposed.

The results show, that the problem structure, especially the distribution of the customers $\alpha$, is crucial for the behavior of the investigated algorithms. The genetic algorithms outperform the local search approach especially well in complex problems with dense and small time windows and a random distribution of the customers. However, the influence of the mutation operator seems to be stronger than expected and described in other publications on GAs.

An important role of simple (local search based) mutations can indicate, that the studied crossover operators themselves are comparable weak for the multi-objective formulation of the problem as they do not recombine the desirable structures of the underlying model. Nevertheless, specific formulations of particular multi-objective operators are still missing. A combination of genetic operators with local search heuristics is consequently a logical conclusion of the obtained results.

## References

1. Nagraj Balakrishnan. Simple heuristics for the vehicle routeing problem with soft time windows. *Journal of the Operational Research Society*, 44(3):279–287, 1993.
2. Joe L. Blanton Jr. and Roger l. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In Forrest [15], pages 452–459.
3. C. Brizuela, N. Sannomiya, and Y. Zhao. Multi-objective flow-shop: Preliminary results. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, and Carlos A. Coello Coello, editors, *Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001*, pages 443–457, 2001.
4. Bernd Bullnheimer, Richard F. Hartl, and Christine Strauss. Applying the ant system to the vehicle routing problem. In Voss et al. [40], chapter 20, pages 285–296.
5. Wen-Chyuan Chiang and Robert A. Russel. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63:3–27, 1996.
6. Carlos A. Coello Coello. List of references on evolutionary multiobjective optimization. http://www.lania.mx/˜ccoello/EMOO/EMOObib.html.
7. Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjetive optimization techniques. *Knowledge and Information Systems*, 3:269–308, 1999.
8. Piotr Czyzak and Andrej Jaskiewicz. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.
9. Lawrence Davis. A genetic algorithms tutorial. In *Handbook of genetic algorithms* [10], chapter I, pages 1–101.
10. Lawrence Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
11. Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.
12. M.L. Fisher. Optimal solution of vehicle routing problems using K-trees. *Operations Research*, 42(4):626–642, 1994.
13. M.L. Fisher and K.O. Jornsten. Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45(3):488–492, 1997.
14. Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Forrest [15], pages 416–423.

15. Stephanie Forrest, editor. *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann Publishers.
16. David E. Goldberg and Robert Lingle, Jr. Alleles, loci, and the traveling salesman problem. In Grefenstette [18], pages 154–159.
17. D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Reading, Massachusetts, 1989.
18. John J. Grefenstette, editor. *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, Hillsdale, NJ, 1985. Lawrence Erlbaum Associates.
19. J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In Z. Michalewicz, editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87, Piscataway, New Jersey, 1994. IEEE Press.
20. Philip Kilby, Patrick Prosser, and Paul Shaw. Guided local search for the vehicle routing problem with time windows. In Voss et al. [40], chapter 32, pages 473–486.
21. N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis. k-path cuts for the vehicle routing problem with time windows. Technical Report G-97-19, Les Cahiers di GERAD, 1997.
22. A.W.J. Kolen, A.H.G. Rinnooy Kan, and H.W.J.M. Trienekens. Vehicle routing with time windows. *Operations Research*, 35(2):266–273, 1987.
23. J. Lenstra and A. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, pages 221–228, 1981.
24. Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.
25. J.G. Partyka and R.W. Hall. On the road to service. *OR/MS Today*, pages 26–35, August 2000.
26. J.-Y. Potvin and J.M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66:331–340, 1993.
27. R. Russel. An effective heuristic for the $M$-tour traveling salesman problem with some side conditions. *Operations Research*, 25:517–524, 1977.
28. M. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.
29. J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In Grefenstette [18], pages 93–100.
30. Frédéric Semet and Eric Taillard. Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, 41:469–488, 1993.
31. Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, March-April 1987.
32. N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.
33. Gilbert Syswerda. Schedule optimization using genetic algorithms. In Davis [10], chapter 21, pages 332–349.
34. E.D. Taillard, F. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186, 1997.

35. Sam R. Thangiah. Vehicle routing with time windows using genetic algorithms. In Lance Chambers, editor, *Application Handbook of Genetic Algorithms: New Frontiers*, volume 2, pages 253–277. CRC Press, 1995.
36. S.R. Thangiah. Genetic algorithms, tabu search and simulated annealing methods for vehicle routing problems with time windows. In Lance Chambers, editor, *Practical Handbook of Genetic Algorithms: Complex Structures*. CRC Press, 1998.
37. S.R. Thangiah, I.H. Osman, and T. Sun. Hybrid genetic algorithms, simulated annealing and tabu search methods for vehicle routing problems with time windows. Technical report, Institute of Mathematics and Statistics, University of Kent, United Kingdom, 1994.
38. S.R. Thangiah, I.H. Osman, and T. Sun. Metaheuristics for vehicle routing problems with time windows. Technical report, Institute of Mathematics and Statistics, University of Kent, United Kingdom, 1995.
39. David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.
40. Stefan Voss, Silvano Martello, Ibrahim H. Osman, and Catherine Roucairol, editors. *META-HEURISTICS: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1999.
41. Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zürich, Switzerland, 1999.

# A Appendix

**Table 1.** Average number of evaluations.

| $\alpha;\beta;\gamma;\delta$ | MOLSD | PMX | OBX | UOBX | UOBX∧2EX |
|---|---|---|---|---|---|
| C;20;1.00;60 | 38,730 | 108,750 | 54,375 | 75,625 | 80,000 |
| C;20;0.70;60 | 43,643 | 98,125 | 80,625 | 98,125 | 95,625 |
| C;20;0.45;60 | 35,488 | 103,750 | 82,500 | 89,375 | 96,250 |
| C;20;0.30;60 | 26,938 | 71,875 | 56,875 | 64,375 | 76,875 |
| C;20;1.00;120 | 36,814 | 105,625 | 50,625 | 66,875 | 48,125 |
| C;20;1.00;180 | 42,047 | 175,625 | 63,750 | 45,000 | 138,750 |
| C;20;1.00;240 | 26,125 | 73,750 | 55,625 | 46,875 | 58,125 |
| C;20;1.00;360 | 23,978 | 74,375 | 36,250 | 35,000 | 34,375 |
| R;20;1.00;10 | 90,060 | 425,625 | 381,875 | 438,125 | 474,375 |
| R;20;0.70;10 | 87,199 | 385,625 | 398,125 | 353,125 | 411,875 |
| R;20;0.45;10 | 90,011 | 386,875 | 301,875 | 303,750 | 433,125 |
| R;20;0.30;10 | 46,364 | 196,250 | 204,375 | 238,750 | 231,875 |
| R;20;1.00;30 | 90,011 | 440,625 | 350,625 | 404,375 | 426,875 |
| R;20;0.70;30 | 74,997 | 361,250 | 251,875 | 345,000 | 367,500 |
| R;20;0.45;30 | 59,113 | 211,875 | 201,875 | 178,125 | 266,250 |
| R;20;0.30;30 | 22,067 | 106,250 | 98,125 | 115,000 | 142,500 |
| R;20;1.00;60 | 61,191 | 315,625 | 260,000 | 259,375 | 278,750 |
| R;20;1.00;80 | 44,650 | 233,125 | 171,875 | 225,000 | 226,250 |
| R;20;1.00;95 | 51,897 | 231,875 | 206,250 | 196,875 | 286,875 |
| R;20;1.00;115 | 24,271 | 144,375 | 118,125 | 115,625 | 145,000 |
| C;30;1.00;60 | 510,995 | 177,500 | 155,000 | 185,000 | 301,250 |
| C;30;0.70;60 | 367,880 | 156,250 | 141,250 | 127,500 | 220,000 |
| C;30;0.45;60 | 372,534 | 220,000 | 261,250 | 240,000 | 541,250 |
| C;30;0.30;60 | 211,845 | 147,500 | 178,750 | 160,000 | 230,000 |
| C;30;1.00;120 | 515,736 | 143,750 | 156,250 | 203,750 | 180,000 |
| C;30;1.00;180 | 532,005 | 271,250 | 120,000 | 146,250 | 246,250 |
| C;30;1.00;240 | 335,864 | 188,750 | 82,500 | 82,500 | 186,250 |
| C;30;1.00;360 | 253,083 | 128,750 | 71,250 | 73,750 | 101,250 |
| R;30;1.00;10 | 754,551 | 958,750 | 466,250 | 585,000 | 955,000 |
| R;30;0.70;10 | 642,278 | 650,000 | 503,750 | 453,750 | 862,500 |
| R;30;0.45;10 | 550,319 | 673,750 | 392,500 | 460,000 | 1,383,750 |
| R;30;0.30;10 | 276,878 | 391,250 | 435,000 | 315,000 | 403,750 |
| R;30;1.00;30 | 809,535 | 666,250 | 445,000 | 321,250 | 1,025,000 |
| R;30;0.70;30 | 583,814 | 658,750 | 478,750 | 326,250 | 1,023,750 |
| R;30;0.45;30 | 471,105 | 405,000 | 381,250 | 256,250 | 503,750 |
| R;30;0.30;30 | 155,643 | 236,250 | 246,250 | 160,000 | 330,000 |
| R;30;1.00;60 | 581,726 | 536,250 | 278,750 | 502,500 | 448,750 |
| R;30;1.00;80 | 486,374 | 403,750 | 357,500 | 287,500 | 587,500 |
| R;30;1.00;95 | 341,693 | 331,250 | 306,250 | 373,750 | 571,250 |
| R;30;1.00;115 | 150,554 | 201,250 | 210,000 | 170,000 | 245,000 |

**Table 2.** Average values of $d_1$. For each instance, the best obtained value is marked with †.

| $\alpha;\beta;\gamma;\delta$ | MOLSD | PMX | OBX | UOBX | UOBX∧2EX |
|---|---|---|---|---|---|
| C;20;1.00;60 | † 0.0200 | 0.0426 | 0.0219 | 0.0394 | 0.0330 |
| C;20;0.70;60 | † 0.0614 | 0.3471 | 0.0769 | 0.2320 | 0.0632 |
| C;20;0.45;60 | † 0.0983 | 2.2945 | 1.2784 | 0.4368 | 0.3701 |
| C;20;0.30;60 | † 0.1725 | 2.1288 | 0.8188 | 0.7016 | 0.7386 |
| C;20;1.00;120 | † 0.0312 | 0.0494 | 0.0554 | 0.0572 | 0.0478 |
| C;20;1.00;180 | † 0.0449 | 0.0510 | 0.0671 | 0.0527 | 0.0655 |
| C;20;1.00;240 | † 0.0111 | 0.0793 | 0.0478 | 0.0590 | 0.0441 |
| C;20;1.00;360 | † 0.5074 | 3.3814 | 2.6878 | 3.2138 | 2.6519 |
| R;20;1.00;10 | 0.0950 | 0.0595 | 0.0655 | 0.0631 | † 0.0478 |
| R;20;0.70;10 | 0.0768 | 0.0596 | 0.0542 | 0.0573 | † 0.0487 |
| R;20;0.45;10 | 0.0701 | 0.0822 | 0.0654 | 0.0602 | † 0.0546 |
| R;20;0.30;10 | 0.1396 | 0.2071 | 0.1585 | 0.1527 | † 0.1284 |
| R;20;1.00;30 | 0.1025 | 0.0852 | 0.0700 | 0.0698 | † 0.0571 |
| R;20;0.70;30 | 0.0841 | 0.0680 | 0.0716 | 0.0663 | † 0.0567 |
| R;20;0.45;30 | † 0.1015 | 0.1502 | 0.1408 | 0.1233 | 0.1089 |
| R;20;0.30;30 | 0.5280 | 0.8925 | 0.5055 | 0.4632 | † 0.4859 |
| R;20;1.00;60 | 0.1137 | 0.1284 | † 0.0767 | 0.0931 | 0.0770 |
| R;20;1.00;80 | 0.1238 | 0.1257 | 0.0834 | 0.0986 | † 0.0775 |
| R;20;1.00;95 | 0.1029 | 0.1806 | 0.1196 | 0.1135 | † 0.0935 |
| R;20;1.00;115 | 0.1913 | 0.2984 | 0.1986 | 0.2170 | † 0.1379 |
| C;30;1.00;60 | † 0.0497 | 0.2773 | 0.2727 | 0.2824 | 0.1825 |
| C;30;0.70;60 | † 0.3522 | 1.7077 | 1.8106 | 1.6318 | 0.4321 |
| C;30;0.45;60 | † 0.1827 | 3.3573 | 1.5968 | 1.4822 | 0.8985 |
| C;30;0.30;60 | † 0.2526 | 2.7933 | 1.4744 | 1.6240 | 0.9231 |
| C;30;1.00;120 | † 0.1395 | 0.2371 | 0.2716 | 0.2172 | 0.1863 |
| C;30;1.00;180 | † 0.1459 | 0.8585 | 0.2958 | 0.5522 | 0.4910 |
| C;30;1.00;240 | † 0.1516 | 0.7634 | 0.3881 | 0.6771 | 0.2882 |
| C;30;1.00;360 | † 0.1818 | 1.8217 | 1.5078 | 1.1559 | 0.9665 |
| R;30;1.00;10 | 0.1031 | 0.1505 | 0.1021 | 0.1065 | † 0.0638 |
| R;30;0.70;10 | 0.0991 | 0.1876 | 0.1123 | 0.1276 | † 0.0742 |
| R;30;0.45;10 | 0.0999 | 0.2734 | 0.1349 | 0.1387 | † 0.0705 |
| R;30;0.30;10 | † 0.1634 | 0.5062 | 0.2881 | 0.4721 | 0.1880 |
| R;30;1.00;30 | 0.0826 | 0.1450 | 0.1097 | 0.0962 | † 0.0661 |
| R;30;0.70;30 | 0.0858 | 0.1986 | 0.1220 | 0.0956 | † 0.0458 |
| R;30;0.45;30 | † 0.1468 | 0.5630 | 0.2426 | 0.2039 | 0.1472 |
| R;30;0.30;30 | † 0.2486 | 0.8227 | 0.4819 | 0.3566 | 0.3642 |
| R;30;1.00;60 | 0.0960 | 0.3288 | 0.1678 | 0.0985 | † 0.0744 |
| R;30;1.00;80 | † 0.1147 | 0.4261 | 0.1841 | 0.2325 | 0.1229 |
| R;30;1.00;95 | † 0.0911 | 0.4424 | 0.1438 | 0.1450 | 0.0971 |
| R;30;1.00;115 | † 0.1865 | 0.9801 | 0.2856 | 0.3311 | 0.2580 |

**Table 3.** Average values of $d_2$. For each instance, the best obtained value is marked with †.

| $\alpha;\beta;\gamma;\delta$ | MOLSD | PMX | OBX | UOBX | UOBX∧2EX |
|---|---|---|---|---|---|
| C;20;1.00;60 | † 0.0627 | 0.0770 | 0.0647 | 0.0857 | 0.0772 |
| C;20;0.70;60 | 0.2312 | 0.5312 | 0.2436 | 0.4079 | † 0.1927 |
| C;20;0.45;60 | † 0.2369 | 2.5922 | 1.5318 | 0.6826 | 0.5644 |
| C;20;0.30;60 | † 0.3744 | 2.5031 | 1.1377 | 1.0032 | 1.0189 |
| C;20;1.00;120 | † 0.0837 | 0.1059 | 0.1111 | 0.1184 | 0.1022 |
| C;20;1.00;180 | 0.1575 | † 0.1034 | 0.1680 | 0.1662 | 0.1643 |
| C;20;1.00;240 | † 0.0904 | 0.1172 | 0.1127 | 0.1306 | 0.1153 |
| C;20;1.00;360 | † 0.6995 | 3.7799 | 3.1016 | 3.6282 | 3.0673 |
| R;20;1.00;10 | 0.3134 | 0.1921 | 0.2469 | 0.2344 | † 0.2061 |
| R;20;0.70;10 | 0.2937 | † 0.1687 | 0.2000 | 0.2190 | 0.1950 |
| R;20;0.45;10 | 0.2757 | 0.2075 | † 0.2028 | 0.2252 | 0.2093 |
| R;20;0.30;10 | 0.2814 | 0.4148 | 0.3113 | 0.2997 | † 0.2636 |
| R;20;1.00;30 | 0.2735 | 0.1790 | 0.1792 | 0.1838 | † 0.1743 |
| R;20;0.70;30 | 0.2721 | † 0.1862 | 0.2355 | 0.2535 | 0.2255 |
| R;20;0.45;30 | † 0.2342 | 0.3502 | 0.2829 | 0.2589 | 0.2538 |
| R;20;0.30;30 | 0.9088 | 1.3208 | 0.8674 | † 0.8331 | 0.8422 |
| R;20;1.00;60 | 0.2517 | 0.2660 | 0.2001 | 0.2203 | † 0.1942 |
| R;20;1.00;80 | 0.3326 | 0.2609 | 0.2530 | 0.2553 | † 0.2287 |
| R;20;1.00;95 | 0.2302 | 0.3510 | 0.2472 | 0.2423 | † 0.2273 |
| R;20;1.00;115 | 0.3660 | 0.5899 | 0.4476 | 0.4894 | † 0.3303 |
| C;30;1.00;60 | † 0.1887 | 0.3939 | 0.3600 | 0.3769 | 0.2674 |
| C;30;0.70;60 | † 0.7243 | 2.2233 | 2.3094 | 2.1416 | 0.7733 |
| C;30;0.45;60 | † 0.4626 | 3.9027 | 2.0615 | 1.9792 | 1.3238 |
| C;30;0.30;60 | † 0.5225 | 3.1728 | 1.8258 | 2.0087 | 1.2447 |
| C;30;1.00;120 | † 0.2944 | 0.3682 | 0.4176 | 0.3649 | 0.3124 |
| C;30;1.00;180 | † 0.2499 | 1.1020 | 0.4593 | 0.7660 | 0.7074 |
| C;30;1.00;240 | † 0.2251 | 0.9629 | 0.5415 | 0.8705 | 0.4282 |
| C;30;1.00;360 | † 0.3634 | 2.2197 | 1.9335 | 1.5391 | 1.3466 |
| R;30;1.00;10 | 0.3281 | 0.4167 | 0.3338 | 0.3596 | † 0.3279 |
| R;30;0.70;10 | † 0.3200 | 0.4613 | 0.3667 | 0.3417 | 0.3417 |
| R;30;0.45;10 | † 0.4423 | 0.6875 | 0.5080 | 0.5000 | 0.5000 |
| R;30;0.30;10 | † 0.7625 | 1.1250 | 1.0000 | 1.1299 | 1.0000 |
| R;30;1.00;30 | † 0.2323 | 0.2537 | 0.2589 | 0.2663 | 0.2500 |
| R;30;0.70;30 | † 0.3103 | 0.4167 | 0.3338 | 0.3333 | 0.3333 |
| R;30;0.45;30 | † 0.7619 | 1.1358 | 1.0000 | 1.0000 | 1.0000 |
| R;30;0.30;30 | † 0.8483 | 1.3313 | 1.0411 | 1.0000 | 1.0184 |
| R;30;1.00;60 | † 0.4700 | 0.6415 | 0.5089 | 0.5000 | 0.5000 |
| R;30;1.00;80 | † 0.4056 | 0.7595 | 0.5000 | 0.5280 | 0.5000 |
| R;30;1.00;95 | † 0.4362 | 0.7333 | 0.4609 | 0.5000 | 0.5000 |
| R;30;1.00;115 | † 0.8460 | 1.5504 | 1.0000 | 1.0285 | 1.0000 |