European Journal of Operational Research 247 (2015) 441–458

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

The multi-objective generalized consistent vehicle routing problem

Attila A. Kovacs[*], Sophie N. Parragh, Richard F. Hartl

Department of Business Administration, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

ARTICLE INFO

Article history:
Received 20 August 2014
Accepted 11 June 2015
Available online 20 June 2015

Keywords:
Vehicle routing
Periodic distribution problems
Service consistency
Multi-objective optimization

ABSTRACT

More and more companies in the routing industry are providing consistent service to gain competitive advantage. However, improved service consistency comes at the price of higher routing cost, i.e., routing cost and service consistency are conflicting objectives. In this paper, we extend the generalized consistent vehicle routing problem (GenConVRP) by considering several objective functions: improving driver consistency and arrival time consistency, and minimizing routing cost are independent objectives of the problem. We refer to the problem as the multi-objective generalized consistent vehicle routing problem (MOGenConVRP). A multi-objective optimization approach enables a thorough trade-off analysis between the conflicting objective functions. The results of this paper should help companies in finding adequate consistency goals to aim for. Results are generated for several test instances by two exact solution approaches and one heuristic. The exact approaches are based on the ϵ-constraint framework and are used to solve small test instances to optimality. Large instances with up to 199 customers and a planning horizon of 5 days are solved by multi directional large neighborhood search (MDLNS) that combines the multi directional local search framework and the LNS for the GenConVRP. The solution quality of the heuristic is evaluated by examining five multi-objective quality indicators. We find that MDLNS is an eligible solution approach for performing a meaningful trade-off analysis.

Our analysis shows that a 70 percent better arrival time consistency is achieved by increasing travel cost by not more than 3.84 percent, on average; visiting each customer by the same driver each time is significantly more expensive than allowing at least two different drivers per customer; in many cases, arrival time consistency and driver consistency can be improved simultaneously.

1. Introduction

Vehicle routing with consistency considerations is a multi-objective process: Companies put large efforts in optimizing vehicle routes in order to decrease cost. At the same time, many companies are willing to increase routing cost to visit each customer at similar times of the day with drivers familiar to them. This is because service consistency increases customer satisfaction and, thus, the lifetime value of customers. As proposed in Kovacs, Golden, Hartl, and Parragh (2014b), we improve driver consistency by reducing the maximum number of different drivers that visit a customer. Arrival time consistency is achieved by minimizing the maximum difference between the earliest and the latest arrival time at each customer within the planning horizon. Decision makers are faced with the task of choosing between a minimal-cost routing plan that is poorly consistent, a high-cost routing plan with perfect consistency (i.e., visiting each customer at exactly the same time with his favorite driver), or any routing plan in between these extremes.

Several papers examine the trade-off between service consistency and routing cost, but the multi-objective nature of the problem is often oversimplified. For example, Coelho and Laporte (2013b), Francis, Smilowitz, and Tzur (2007), Groër, Golden, and Wasil (2009), Kovacs, Parragh, and Hartl (2014c), and Spliet (2013) compare the routing cost of solutions in which consistency is ignored to solutions in which consistency is enforced by hard constraints. In Coelho, Cordeau, and Laporte (2012), Coelho and Laporte (2013a), Kovacs, Golden, Hartl, and Parragh (2014a), and Smilowitz, Nowak, and Jiang (2013), the multi-objective problem is transformed into a single-objective problem by aggregating routing cost and consistency measurements into a single objective function. This approach involves a priori decisions about the importance of each objective, i.e., service consistency needs to be expressed in monetary gain. Feillet, Garaix, Lehuédé, Péton, and Quadri (2014) introduce the time-consistent vehicle routing problem

[*] Corresponding author. Tel.: +43 1 4277 38095.
E-mail addresses: attila.kovacs@univie.ac.at (A.A. Kovacs), sophie.parragh@univie.ac.at (S.N. Parragh), richard.hartl@univie.ac.at (R.F. Hartl).

http://dx.doi.org/10.1016/j.ejor.2015.06.030

with two objectives: minimizing routing cost and improving arrival time consistency. A general survey on consistency considerations in vehicle routing is given in Kovacs et al. (2014b).

Multi-objective optimization is a flexible approach for analyzing the trade-off between conflicting objective functions. With conflicting objectives, there may exist a large number of relevant solutions. These solutions are not optimal in the sense of single-objective optimization; rather, they are trade-off solutions, i.e., solutions that cannot be improved in one objective without deteriorating another. Among all solutions found, the one that maximizes the utility of the current decision maker will be implemented in the field.

The active interest by researchers and practitioners in multi-objective combinatorial optimization (MOCO) has generated a vast amount of literature. Survey papers are presented, e.g., by Ehrgott and Gandibleux (2002). Prominent examples of mathematical programming-based optimizers for multi-objective problems are the ϵ-constraint method (e.g., Srinivasan & Thompson, 1976) and the two-phase method (e.g., Przybylski, Gandibleux, & Ehrgott, 2008; Vise, Teghem, Pirlot, & Ulungu, 1998). Both approaches provide trade-off solutions by repeatedly solving a single-objective problem with changing parameters (e.g., tightness of constraints and coefficients of the objective function). Modified branch-and-bound and branch-and-cut algorithms solve the multi-objective problem in a single run (e.g., Jozefowiez, Laporte, & Semet, 2012; Parragh & Tricoire, 2014; Vincent, Seipp, Ruzika, Przybylski, & Gandibleux, 2013).

Heuristic multi-objective optimizers are often based on evolutionary algorithms (EAs). EAs maintain a pool of solutions (called population) during the search process. Therefore, EAs are particularly suited for finding many trade-off solutions in a single run. An overview of evolutionary solution approaches is presented, e.g., in Fonseca (1995), Konak, Coit, and Smith (2006) and Zitzler and Thiele (1998). Multi-objective extensions of single-objective heuristics such as variable neighborhood search, ant colony optimization, and simulated annealing are presented, e.g., in Jozefowiez, Semet, and Talbi (2002), Parragh, Doerner, Hartl, and Gandibleux (2009), Doerner, Gutjahr, Hartl, Strauss, and Stummer (2004), and Czyżak and Jaszkiewicz (1998). Paquete, Chiarandini, and Stützle (2004) and Tricoire (2012) present heuristic frameworks for MOCO problems that integrate various local search strategies.

In this article, we present the multi-objective generalized consistent vehicle routing problem (MOGenConVRP). The problem is based on the generalized consistent vehicle routing problem (GenConVRP, Kovacs et al., 2014a) that aggregates routing cost and arrival time consistency into a single objective function; the number of different drivers per customer is bounded. In the MOGenConVRP, routing cost, arrival time consistency, and driver consistency are independent objectives of the problem. Solving this multi-objective problem in practice is unrealistic: typically, the long computation time caused by the large search space and the high number of relevant solutions is prohibitive. The aim of our work is to find all trade-off solutions for different test instances; these solution sets enable a thorough analysis of the trade-off between arrival time consistency, driver consistency, and routing cost. Our results should help companies in the routing industry in finding adequate consistency goals to aim for. We devise two exact solution approaches and one heuristic. The exact approaches are based on the ϵ-constraint method (Haimes, Lasdon, & Wismer, 1971; Laumanns, Thiele, & Zitzler, 2006; Srinivasan & Thompson, 1976). For both exact approaches, the computation time is reduced by introducing new valid inequalities and by exploiting special properties of the problem. Optimal solutions for the MOGenConVRP are important for benchmarking tests that measure the quality of approximation methods. Our heuristic is a combination of the large neighborhood search algorithm (LNS) for the GenConVRP (Kovacs et al., 2014a) and the multi directional local search framework (MDLS, Tricoire, 2012). We refer to the heuristic algorithm as multi directional large neighborhood search (MDLNS). The quality of the MDLNS is validated by computing several unary quality measurements for multi-objective solution sets. The computation time of the algorithms is secondary for our analysis. For practical applications, our results provide a guideline for choosing proper bounds (in hard-constraint models) or weights (in soft-constraint models) on consistency measurements. The resulting single-objective problems can be solved quickly and provide a routing plan for a long period of time.

Our paper provides several contributions. In Section 2, we introduce and formally define the multi-objective generalized consistent vehicle routing problem with three objective functions: arrival time consistency, driver consistency, and travel cost. We propose two exact solution approaches with new valid inequalities and one heuristic; the algorithms are described in Section 3 and Appendix A. For analyzing the trade-off between conflicting objectives, we introduce a transformation scheme that allows aggregating results across several instances in a meaningful way. Computational experiments are presented in Section 4 and numerical results are summarized in Section 4.4.

2. Problem definition

The MOGenConVRP is based on the GenConVRP (Kovacs et al., 2014a) and is modeled on a complete directed graph $G = (N^0, A)$. $N^0 = \{0, 1, \ldots, n\}$ is the set of nodes representing customer locations and the depot 0, N is the set of customers ($N^0 \backslash \{0\}$). Customers are divided into AM customers who can only be visited in the morning ($N^{am} \subseteq N$) and PM customers who require a visit in the afternoon ($N^{pm} \subseteq N$, $N^{am} \cap N^{pm} = \{\}$, $N^{am} \cup N^{pm} = N$). This constraint is important for customers that are not available all-day, require service at specific times of the day (e.g., medical treatment in home-care services), or hire part-time workers to handle package deliveries (Wong, 2008). Splitting the day into two halves also models applications in which drivers perform pick-up operations in the morning and delivery operations in the afternoon. Our algorithms are applicable to problems with and without time windows. Yet, the problem can be solved faster if time windows are incorporated.

The planning horizon involves $|D|$ days where D is the set of days. Set $N^f \subseteq N$ contains all customers that require at least two visits during the planning period; consistency goals are only relevant for customers in N^f. $A = \{(i, j) : i, j \in N^0, i \neq j\}$ is the set of arcs. Arcs from customers $\in N^{pm}$ to customers $\in N^{am}$ are omitted from the graph. Each arc $(i, j) \in A$ is associated with travel time t_{ij}. The travel time matrix satisfies the triangle inequality. Customers are visited by a homogeneous fleet of vehicles in the set $K = \{0, 1, \ldots, v\}$. The number of vehicles is not restrictive (i.e., $v = n$). Each vehicle has a capacity of Q. Drivers and vehicles share a one-to-one relationship. We use drivers and vehicles interchangeably, depending on the context. Each route starts and ends at the depot. The departure time is flexible, but vehicles must return to the depot before time T. On each day $d \in D$, each customer $i \in N$ has demand q_{id} and service time s_{id}. Auxiliary parameters w_{id} are set to 1 if customer i requires service on day d ($q_{id} > 0$) and set to 0, otherwise. Each set of nodes is specified by index d to denote customers that require service on day d (e.g., N_d^0 contains customers that require service on day d and the depot). Additionally, we define set $A_d \subset A$ that contains arcs between customers in N_d; A_d^0 also contains the arcs from and to the depot. The model uses the following binary variables:

$$x_{ijkd} = \begin{cases} 1, & \text{if arc}(i, j) \in A_d^0 \text{ is traversed by vehicle } k \text{ on day } d, \\ 0, & \text{otherwise;} \end{cases}$$

$$y_{ikd} = \begin{cases} 1, & \text{if customer } i \text{ is assigned to vehicle } k \text{ on day } d, \\ 0, & \text{otherwise;} \end{cases}$$

$$z_{ik} = \begin{cases} 1, & \text{if customer } i \text{ is assigned to vehicle } k \text{ at least once} \\ & \quad \text{in the planning period,} \\ 0, & \text{otherwise.} \end{cases}$$

Variables a_{id} denote the arrival time at customer $i \in N_d$ on day d; z_{max} is the maximum number of different drivers that any customer encounters; l_{max} gives the maximum arrival time difference, i.e., largest difference between the latest and the earliest arrival time per customer among all customers. Vehicle idling to reduce the arrival time difference is not allowed. There are several reasons for adding this constraint: first, idling is unproductive working time that has to be paid for by the service provider; second, planned idle times are often ignored by the staff; third, Kovacs et al. (2014c) show that a high level of arrival time consistency can be achieved without idle times if vehicle departure times are flexible.

The objective (1) is to minimize a vector of conflicting objective functions. The vector f is composed of the total travel time (2), the maximum number of different drivers per customer (3), and the maximum arrival time difference (4).

$$min\ f = (f_1, f_2, f_3) \tag{1}$$

$$f_1 = \sum_{d \in D} \sum_{k \in K} \sum_{(i,j) \in A_d^0} t_{ij} x_{ijkd} \tag{2}$$

$$f_2 = z_{max} \tag{3}$$

$$f_3 = l_{max} \tag{4}$$

subject to

$$y_{0kd} = 1 \quad \forall\, k \in K, d \in D \tag{5}$$

$$\sum_{k \in K} y_{ikd} = 1 \quad \forall\, i \in N_d, d \in D \tag{6}$$

$$\sum_{i \in N_d} q_{id} y_{ikd} \le Q \quad \forall\, k \in K, d \in D \tag{7}$$

$$\sum_{i \in N_d^0} x_{ijkd} = \sum_{i \in N_d^0} x_{jikd} = y_{jkd} \quad \forall\, j \in N_d^0, k \in K, d \in D \tag{8}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \le a_{jd} \quad \forall\, (i,j) \in A_d, k \in K, d \in D \tag{9}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) + (1 - x_{ijkd})T \ge a_{jd} \quad \forall\, (i,j) \in A_d, k \in K, d \in D \tag{10}$$

$$z_{ik} \ge y_{ikd} \quad \forall\, i \in N_d^f, k \in K, d \in D \tag{11}$$

$$\sum_{k \in K} z_{ik} \le z_{max} \quad \forall\, i \in N^f \tag{12}$$

$$(a_{i\alpha} - a_{i\beta}) w_{i\alpha} w_{i\beta} \le l_{max} \quad \forall\, i \in N^f, \alpha, \beta \in D \tag{13}$$

$$\left[t_{0i}, min\left(\frac{T}{2}, T - t_{i0} - s_{id}\right)\right] \ni a_{id} \quad \forall\, i \in N_d^{am}, d \in D \tag{14}$$

$$\left[max\left(t_{0i}, \frac{T}{2}\right), T - t_{i0} - s_{id}\right] \ni a_{id} \quad \forall\, i \in N_d^{pm}, d \in D \tag{15}$$

$$x_{ijkd} \in \{0, 1\} \quad \forall\, (i,j) \in A_d^0, k \in K, d \in D \tag{16}$$

$$y_{ikd} \in \{0, 1\} \quad \forall\, i \in N_d^0, k \in K, d \in D \tag{17}$$

$$z_{ik} \in \{0, 1\} \quad \forall\, i \in N^f, k \in K \tag{18}$$

Equalities (5) ensure that each route starts from the depot. Constraints (6) guarantee that each customer is serviced on each day he requires service. Inequalities (7) limit the vehicle capacity to Q. Equalities (8) are flow conservation constraints. Inequalities (9) and (10) set the arrival times at the customers. Vehicle idling to improve time consistency is prohibited by (10). Inequalities (9) also prevent sub-tours. Driver consistency is defined in (11) and (12). Constraints (11) set the z variables and constraints (12) set the maximum number of drivers per customer. Constraints (13) define the maximum arrival time difference. Finally, constraints (14)–(18) define the domains of the decision variables. Time window feasibility is ensured by restricting the domains of the arrival time variables; this approach reduces the computation time of the applied integer linear programming optimizer.

With multiple objectives of equal rank, we cannot decide whether or not a solution is better than another for any pair of solutions, i.e., the multi-objective search space is partially ordered. Comparable solutions are ordered by a dominance relation defined as follows: solution s^1 dominates solution s^2 ($s^1 \succ s^2$) if and only if $\forall j \in \{1, 2, 3\}$: $f_j(s^1) \le f_j(s^2)$ and $\exists j \in \{1, 2, 3\}$: $f_j(s^1) < f_j(s^2)$.

A solution s is efficient if and only if there is no other solution s' that dominates s. Set E contains all efficient solutions, $E = \{s : \nexists\, s'$ such that $s' \succ s\}$. (Note: solutions in E are incomparable and equally good for a neutral decision maker.) The Pareto-front is defined as $P = \{f(s) : s \in E\}$. Elements of P are said to be non-dominated points in the objective space. Depending on the context, we use solution either to denote an element of E or an element of P.

Our goal is to generate P in order to study the trade-off between travel cost and service consistency. There are at least as many efficient solutions as non-dominated points, i.e., several solutions might give the same objective vector. For us, it suffices to find only one solution for each point in P.

3. Solution approaches

We propose two exact solution approaches (i.e., approaches that find all points in P) based on the ϵ-constraint method. In Section 4, we show that the exact algorithms are able to solve MOGenConVRP instances with up to 10 customers that require service over 5 days. In order to study realistic problem instances, we also devise a metaheuristic algorithm that provides an approximation of P, denoted by P_{approx}. MDLNS integrates the large neighborhood search algorithm for the GenConVRP (Kovacs et al., 2014a) into the multi directional local search framework (Tricoire, 2012). The LNS is an advanced approach for single-objective consistent vehicle routing problems; the performance of the MDLS is comparable to the best known solution approaches for three different combinatorial optimization problems: the multi-objective multi-dimensional knapsack problem, the bi-objective set packing problem, and the bi-objective orienteering problem (Tricoire, 2012). In this section and in Appendix A, we describe the algorithms in detail and present problem-specific enhancements.

3.1. Exact approaches

In the ϵ-constraint method, the multi-objective problem is transformed into a single-objective problem by optimizing one objective function and restricting the objective value of the remaining functions. By modifying the tightness of the constraints, we can identify different elements of the Pareto-front. The single-objective problem is solved by a procedure that we refer to as $opt(f, \epsilon, \epsilon')$; it is defined as follows (Laumanns et al., 2006):

$$lex\ min\ f(s) = (f_1, f_2, f_3) \tag{19}$$

subject to

$$\epsilon_j \le f_j(s) < \epsilon'_j \quad \forall\, j \in \{2, 3\} \tag{20}$$

$$s \in S \tag{21}$$

The objective is to minimize the three objective functions in lexicographic order (19): f_1 first, f_2 second, and f_3 third. Constraints (20) are ϵ-constraints that bound the objective value of f_2 and f_3. Set S contains all feasible solutions defined by constraints (5)–(18). Expression (21) restricts the search space to solutions from S.

Without lexicographic optimization, the procedure might return solutions that are not efficient. For example, a solution s^1 with minimal travel cost and objective vector $f(s^1) = (100, 3, 21)$ is dominated by solution s^2 with $f(s^2) = (100, 2, 20)$. The computation time for finding s^1 will be wasted, if s^1 and s^2 are both within the same search space. By performing a lexicographic optimization, we can guarantee that the provided solution is always non-dominated for the given search space. The output of the procedure is either an optimal solution for the constrained problem (if a feasible solution exists) or null (if the search space is empty).

In our implementation, $opt(f, \epsilon, \epsilon')$ is based on a branch-and-bound algorithm. The lexicographic optimization is performed in two phases: The single-objective problem with f_1 as the only objective is solved in the first phase; let s^1 be the provided solution. In the second phase, we restrict the f_1 value of the second solution s^2 to the previously obtained value ($f_1(s^2) \leq f_1(s^1)$) and solve the problem with the following objective function:

$$min\ f_2(s^2) + \frac{f_3(s^2)}{UB_3}. \tag{22}$$

Parameter UB_3 is an upper bound for f_3 such that $UB_3 > f_3$. (A trivial upper bound for f_3 is half the closure time at the depot $T/2$.) This approach optimizes f_2 and f_3 in lexicographic order since $f_2 \in \mathbb{N}$ and $\frac{f_3(s)}{UB_3} < 1$. Furthermore, the second phase can be solved with minimal effort: once the first phase solution is found, the second phase can be solved by exploring the unprocessed nodes of the branch-and-bound tree; all other nodes have already been pruned either by infeasibility or by the f_1-bound. The resulting solution is the correct output of $opt(f, \epsilon, \epsilon')$.

We propose two algorithms that repeatedly apply $opt(f, \epsilon, \epsilon')$ in order to obtain P. The algorithms differ in the scheme the ϵ-constraints (20) are modified. The first approach, denoted by three dimensional adaptive ϵ-constraint method, is described in the next section. The second approach is called two dimensional adaptive ϵ-constraint method; it performs slightly worse than the three dimensional method and is described in Appendix A. The total computation time of both algorithms depends mainly on the computation time of $opt(f, \epsilon, \epsilon')$. The optimization process can be speeded up by aborting the procedure when a certain optimality gap (i.e., difference between the best found solution and the best lower bound) is reached. Yet, this might prevent us from generating the optimal Pareto-front.

3.1.1. *Three dimensional adaptive ϵ-constraint method*

Our first exact algorithm is referred to as three dimensional (3D) adaptive ϵ-constraint method. It is based on the framework proposed by Laumanns et al. (2006). The authors present a general algorithm for solving multi-objective problems with an arbitrary number of objectives, h. The worst case time complexity of the algorithm, measured by the calls of $opt(f, \epsilon, \epsilon')$, depends on the number of points in P ($|P|$) and the number of objectives: $\mathcal{O}(|P|^{h-1})$.

The basic concept is an $h - 1$ dimensional hypergrid that divides the objective space into rectangular subspaces that are parallel to the axes, e.g., into stripes in bi-objective problems (Laumanns et al., 2006). Each cell in the grid represents one constrained search space for which $opt(f, \epsilon, \epsilon')$ is applied. The coordinates of the grid are given by solutions that have been found earlier; so, the number of cells in the grid grows to the power of $h - 1$ each time a new solution is found. In our case, the objective space is divided with respect to f_2 and f_3. For each efficient solution found during the search, we store the objective values f_2 and f_3 in vectors e_{f_2} and e_{f_3}, respectively. Matrix $e = (e_{f_2}, e_{f_3})$ defines the coordinates of the grid.

Algorithm 1 .

```
Require: $E = \{\}, F = \{\}, e = ((1, \infty), (0, \infty))$
1:  $m = 0$                                              ▷ iteration counter
2:  $c = 1$                                              ▷ number of cells
3:  loop
4:      loop
5:          if $m \geq c$ then
6:              return $E$
7:          end if
8:          $(\epsilon, \epsilon') = getConstraints(m, e, |E|)$   ▷ retrieve $\epsilon$-constraints
    from matrix of coordinates $e$
9:          if $[\epsilon, \epsilon'] \not\subset F$ then
10:             $s = opt(f, \epsilon, \epsilon')$
11:             if $s = null$ or $\exists s' \in E : s' \succ s$ then
12:                 $F = F \cup [\epsilon, \epsilon']$
13:             else
14:                 break
15:             end if
16:         end if
17:         $m = m + 1$
18:     end loop
19:     $E = E \cup \{s\}$
20:     $F = F \cup [f(s), \epsilon']$
21:     $updateConstraints(f(s), e)$        ▷ update matrix of coordinates $e$
    and divide grid
22:     $c = (|E| + 1)^2$
23:     $m = 0$
24: end loop
```

Fig. 1 illustrates the f_2–f_3 projection of the objective space for three iterations of the 3D method, respectively. For each iteration, the figure on the left shows the initial grid and the figure on the right shows the grid after an efficient solution has been found. In the first iteration, the grid consists of a single cell that contains an efficient solution denoted by a point. The new solution divides the grid into four cells. The numbering of the cells starts with 0 in the lower left corner and increases successively column by column. In the second iteration, we continue the search in cell 0. Cell 3 is hatched because it represents a subspace that is dominated by the solution found in the first iteration. We find a new efficient solution in cell 0. The grid is partitioned and the cells are renumbered. In iteration 3, we again start the search with cell 0. Cell 4 is hatched because it is dominated by the solution found in iteration 2. Cells 5 and 7 are not dominated because they might contain solutions with a lower f_1 value. The search in cell 0 is without result, i.e., $opt(f, \epsilon, \epsilon')$ returns null. So, cell 0 is marked as searched and we continue with cell 1. Here, we find a new solution, partition the grid, and mark cell 6 as searched.

It is essential to search the cells in ascending order of indices starting with cell 0: Let us assume that in iteration 2 there is a solution s^2 with objective vector (100, 2, 12) in cell 2 and a solution s^1 with objective vector (100, 2, 10) in cell 0. If we started with cell 2, we would partition the grid based on s^2 even though s^1 dominates s^2.

The pseudo-code of the algorithm is given in Algorithm 1. The procedure is initialized with an empty set of efficient solutions E, an empty set of already searched subspaces F, and the initial coordinates of the grid e. Variable m is an iteration counter that indicates the index of the cell that is to be searched and variable c is the current number of cells in the grid. Initially, the grid contains one cell that is defined by the lower and upper bounds on f_2 and f_3, respectively. In each iteration of the outer loop (lines 3–24), the algorithm provides one new solution. The inner loop (lines 4–18) examines each cell for new efficient solutions; in each iteration, the subspaces are explored in increasing number of cell index starting with index 0. The current subspace is provided by function $getConstraints(m, e,$

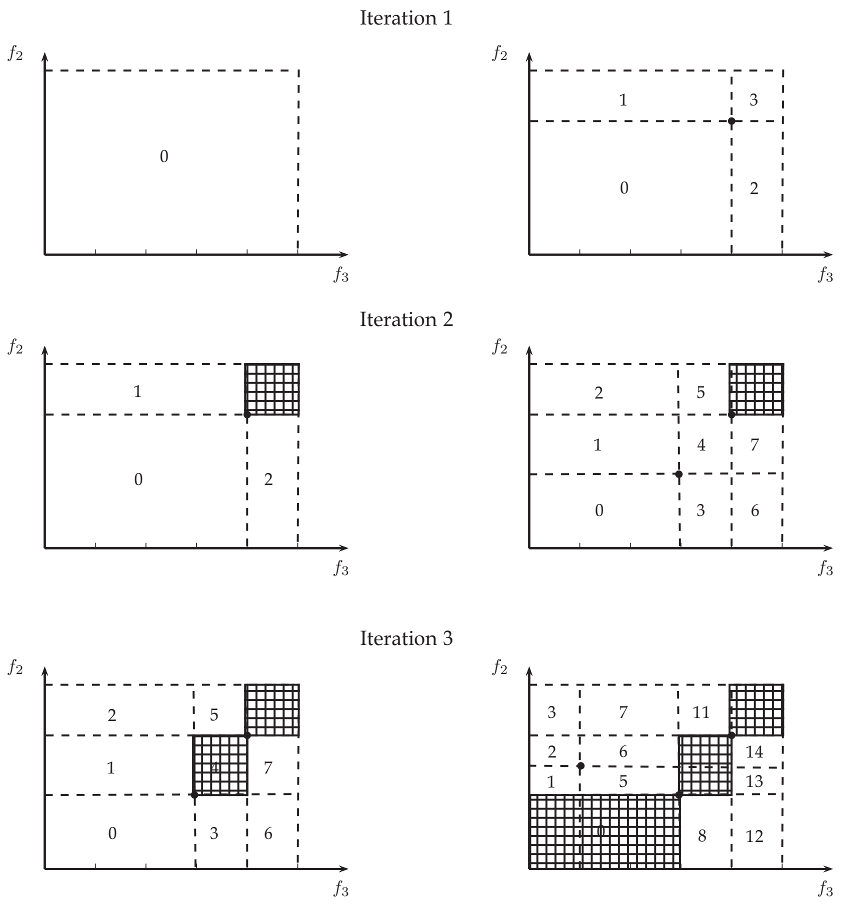


Fig. 1. f_2–f_3 projection of the objective space for three iterations of Algorithm 1, respectively.

$|E|$). The function takes the current number of iterations, the coordinates of the grid, and the cardinality of the set of efficient solutions as arguments and translates the current cell (i.e., iteration m) into ϵ-constraints (line 8) (the implementation of this function is given in Laumanns et al. (2006)). If the current search space has not yet been examined (line 9), we apply $opt(f, \epsilon, \epsilon')$. Depending on the outcome, we distinguish two paths: First, $opt(f, \epsilon, \epsilon')$ provides a solution that is dominated by a solution in E or there is no feasible solution in the respective cell. Then, we mark the current subspace defined by ϵ and ϵ' as searched (line 12) and move on to the next cell (line 17). Second, $opt(f, \epsilon, \epsilon')$ provides a new efficient solution. In this case, we leave the inner loop (line 14), add the solution to E (line 19), and mark the subspace that is dominated by the new solution $([f(s), \epsilon'])$ as searched (line 20). In line 21, we call the function *updateConstraints*($f(s)$, e) that divides the objective space into smaller subspaces. This is achieved by inserting the new objective values into matrix e and updating the coordinates of the grid (for details on this function see Laumanns et al. (2006)). The number of cells grows with each solution found; c is updated in line 22. The algorithm generates one solution for each point in P, i.e., $|E| = |P|$. This is guaranteed by the definition of the subspaces (inequalities (20)) and the lexicographic optimization within each subspace. Finally, we reset the iteration counter to zero in order to start the search in the inner loop with cell 0. The algorithm stops when all cells have been examined (lines 5–7); the output is a set of efficient solutions E.

3.1.2. Single objective optimizer

The computation time of our exact approaches is mainly affected by the lexicographic optimizer $opt(f, \epsilon, \epsilon')$. Therefore, we put emphasis on strengthening the model in a branch-and-cut fashion. In the following, we describe and evaluate the efficiency of five types of valid inequalities: Capacity cuts and subtour cuts are general inequalities for routing problems (see, e.g., Laporte & Nobert, 1987; Naddef & Rinaldi, 2001; Toth & Vigo, 2001). Symmetry breaking constraints have been investigated in Coelho and Laporte (2013b) and Fischetti, Salazar González, and Toth (1995); we present a modification of these constraints that is valid for the MOGenConVRP. Finally, two time consistency related inequalities are introduced.

Capacity cuts (CC) and subtour cuts (ST-|S|). The model defined by inequalities (5)–(18) is complete. Nevertheless, adding a limited number of rounded capacity constraints (inequalities (23)) and generalized subtour elimination constraints (inequalities (24)) can speed up the optimization process.

$$\sum_{i\notin S}\sum_{j\in S} x_{ijkd} \geq \left\lceil \frac{\sum_{i\in S} q_{id}}{Q} \right\rceil \quad \forall\, S \subseteq N_d, |S| \geq 2, d \in D, k \in K \tag{23}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ijkd} \leq |S| - 1 \quad \forall\, S \subseteq N_d, |S| \geq 2, d \in D, k \in K \tag{24}$$

Symmetry breaking (SB). Drivers are homogeneous in the MOGenConVRP; therefore, the number of feasible driver-customer assignments grows exponentially with the number of customers. If $z_{max} \leq 1$, there are v^n feasible assignments; v is the number of available drivers and n is the number of customers. Motivated by Coelho and Laporte (2013b) and Fischetti et al. (1995), we use symmetry breaking inequalities that reduce the number of feasible assignments significantly:

$$z_{ik} \leq \sum_{j<i} z_{j,k-1} \quad \forall\, k \in K\backslash\{0\}, i \in N^f \tag{25}$$

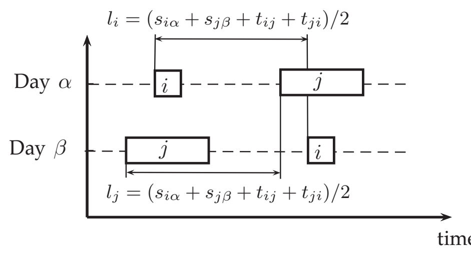

$l_i = (s_{i\alpha} + s_{j\beta} + t_{ij} + t_{ji})/2$
Day α
i
j
Day β
j
i
$l_j = (s_{i\alpha} + s_{j\beta} + t_{ij} + t_{ji})/2$
time

Fig. 2. Time consistency-related inequalities (TC1).

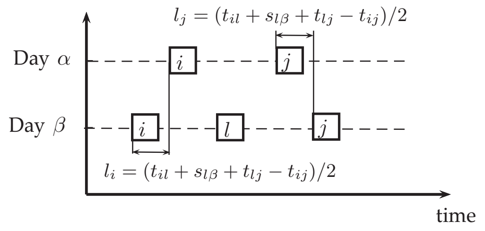

$l_j = (t_{il} + s_{l\beta} + t_{lj} - t_{ij})/2$
Day α
i
j
Day β
i
l
j
$l_i = (t_{il} + s_{l\beta} + t_{lj} - t_{ij})/2$
time

Fig. 3. Time consistency-related inequalities (TC2).

$$z_{ik} \le \sum_{d\in D} y_{ikd} \quad \forall\, k \in K, i \in N^f \tag{26}$$

Inequalities (26) restrict the z_{ik} variables; inequalities (25) allow a customer–driver assignment (i, k) only if driver $k-1$ visits a customer with a smaller index than i.

Inequalities (25) are invalid if $z_{max} \ge 2$. In this case, the number of feasible driver–customer assignments is $\prod_{i=0}^{z_{max}-1} (\nu - i)^n$ because each customer can be assigned to different drivers on different days. A slight modification of inequalities (25) is valid regardless of the number of different drivers per customer:

$$z_{ik} \le \sum_{j\le i} z_{j,k-1} \quad \forall\, k \in K\backslash\{0\}, i \in N^f \tag{27}$$

Proposition. Inequalities (27) are valid for the MOGenConVRP.

Proof: Let $\mathcal{P}(N)$ be the power set of all customers. $N_k \in \mathcal{P}(N)$ is the set of customers assigned to driver $k \in K$ and i_k is the customer with smallest index in set N_k. By sorting the i_k's in non-descending order, we achieve either $i_{k-1} < i_k$ or $i_{k-1} = i_k \;\forall\, k \in K\backslash\{0\}$. $\square$

Time consistency-related inequalities 1 (TC1). For a given bound on l_{max} (that is imposed by the ϵ-constraint method), L, we can add inequalities to the model that exclude solutions with poor arrival time consistency. Time consistency-related inequalities TC1 (28) state that if customer i is visited before customer j on the same route on one day, then it is infeasible to assign customer j before customer i on the same route on another day if the arrival time consistency constraint would be violated. The reasoning behind these inequalities is illustrated in Fig. 2. The arrival time difference at customer i (l_i) is $s_{j\beta} + t_{ji}$ minus the departure time from the depot $a_{0\alpha}$[1]; the arrival time difference at customer j (l_j) is $s_{i\alpha} + t_{ij} + a_{0\alpha}$. In an optimal solution l_i is equal to l_j; otherwise, the maximum arrival time difference could be reduced by leveling the arrival time differences at the two customers. So, $s_{j\beta} + t_{ji} - a_{0\alpha} = s_{i\alpha} + t_{ij} + a_{0\alpha}$. This leads to $a_{0\alpha} = (s_{i\beta} - s_{j\alpha} - t_{ij} + t_{ji})/2$ and, therefore, to $l_i = l_j = (s_{i\alpha} + s_{j\beta} + t_{ij} + t_{ji})/2$.

$$x_{ij\delta\alpha} + x_{ji\gamma\beta} \le 1 \quad \forall\, (i, j) \in A_d^0 : (s_{i\alpha} + s_{j\beta} + t_{ij} + t_{ji})/2 > L, \alpha, \beta \in D, \delta, \gamma \in K \tag{28}$$

Time consistency-related inequalities 2 (TC2). Time consistency-related inequalities TC2 (29) prevent the assignment of a customer l between two customers i and j if i and j are scheduled one after the other on another day and if the assignment of l would violate the constraint on the maximum arrival time difference. Fig. 3 illustrates the reasoning: The arrival time difference at customer i is $l_i = a_{0\alpha}$ the arrival time difference at customer j is $l_j = t_{il} + t_{lj} + s_{l\beta} - t_{ij} - a_{0\alpha}$. Again, we set $l_i = l_j$, i.e., $2a_{0\alpha} = t_{il} + t_{lj} + s_{l\beta} - t_{ij}$. Finally, we obtain $l_i = l_j = (t_{il} + t_{lj} + s_{l\beta} - t_{ij})/2$.

$$x_{ij\delta\alpha} + x_{il\gamma\beta} + x_{lj\gamma\beta} \le 2 \quad \forall\, (i, j), (i, l), (l, j) \in A_d^0 : (t_{il} + s_{l\beta} + t_{lj} - t_{ij})/2 > L, \alpha, \beta \in D, \delta, \gamma \in K \tag{29}$$

[1] We consider only two routes on two different days α and β. Therefore, we can assume that the vehicle departure time is fixed on day β and flexible on day α.

Algorithm 2 .

```
Require: set of efficient solutions $E$
1: for all unprocessed solutions in $E$ do
2:     $s = getUnprocessedSolution(E)$
3:     for all $j \in \{1, 2, 3\}$ do
4:         $applyLNS(s, f_j, E)$
5:     end for
6:     $setProcessed(s)$
7: end for
8: return $E$
```

3.2. Heuristic approach

Our heuristic approach for approximating the Pareto-front is based on the MDLS framework for general multi-objective problems (Tricoire, 2012). For a given solution, MDLS applies different local search strategies in order to identify new efficient solutions. Each objective value in the objective vector is improved by a specialized search strategy. As mentioned above, for the MOGenConVRP, we combine the MDLS framework with the LNS algorithm proposed in Kovacs et al. (2014a). The resulting algorithm is referred to as MDLNS.

Algorithm 2 gives the outline of the MDLNS. Starting with an initial set of efficient solutions E, we iteratively select an unprocessed solution (line 2) and perform a local search for each objective function (lines 3–5). New efficient solutions are added to E; processed solutions are marked in line 6. The algorithm stops as soon as all solutions have been processed.

Set E is initialized by using the construction heuristics proposed in Kovacs et al. (2014a): two solutions are generated by a travel-time oriented heuristic; one by ignoring driver consistency and one by restricting driver consistency to one driver per customer. One solution is generated by a time-consistency oriented heuristic. Among the three solutions, we keep only efficient ones. Function *applyLNS*(s, f_j, E), applies the LNS algorithm for a given number of iterations on solution s with the goal of improving objective function f_j. The LNS was devised for the GenConVRP; it integrates arrival time consistency into the objective function and achieves driver consistency by restricting the feasible driver-to-customer assignments. In the MOGenConVRP, for each $j \in \{1, 2, 3\}$, the objective function f_j is a weighted average of the total travel time and the maximum arrival time difference; the number of different drivers per customer is bounded because this objective is not suitable as an evaluation function. The f_1 and f_3 values are weighted by α_j and $(1 - \alpha_j)$, respectively:

$$f_j = \alpha_j f_1 + (1 - \alpha_j) f_3. \tag{30}$$

We set $\alpha_1 = \frac{1}{1+\frac{\delta}{UB_3}}$ when we optimize f_1 and $\alpha_3 = \frac{1}{1+\frac{UB_1}{\delta}}$ when we optimize f_3; parameter δ is set to 10^{-3} and UB_i is an upper bound for the respective objective function. By using α_1 and α_3, we perform a lexicographic optimization, respectively. In both cases, f_2 is bounded by $f_2(s) + 1$ in order to exclude solutions with large f_2 values. Parameter α_2 is set to $(\alpha_1 + \alpha_3)/2$ when optimizing f_2 (α_1 and α_3 are calculated as defined above); additionally, f_2 is bounded by $f_2(s) - 1$.

Table 1
Data structure for managing the set of efficient solutions.

f_2	(f_1, f_3) Pairs			
	1	2	3	4
1	(189,80)	(190,71)	(191,44)	(192,40)
2	(178,77)	(179,73)	(181,70)	
3	(179,55)	(180,52)	(181,49)	(196,19)

Each solution found during the search is checked whether or not it is dominated by another solution in E; if not, it is added to E. Solutions that are dominated by the new solution are removed from E. This approach is time consuming; yet, we aim for finding the best possible approximation of the Pareto-front. Adding solutions to E is performed efficiently by using the data structure illustrated in Table 1. For each f_2 value, we maintain a sorted list that contains all solutions with the respective f_2 value. Sorting solutions in ascending order of f_1 will automatically sort the solutions in descending order of f_3. This results from the definition of an efficient solution: a solution can be improved in one objective only by deteriorating another. In the worst case, the computation time for finding the potential insertion position of a new solution s increases logarithmically with the number of solutions in E with the same f_2 value. Solutions to the left of s and solutions with a smaller f_2 value than s have a chance of dominating s; solutions to the right of s and solutions with a larger f_2 value than s might be dominated by s.

The data structure relies on the property that $f_2 \in \mathbb{N}$ and on the assumption that the domain of f_2 is small. A similar approach for managing E is proposed in Tricoire (2012) for the bi-objective case; a general approach for maintaining efficient solutions for an arbitrary number of objectives is presented in Habenicht (1983) and further examined in Mostaghim and Teich (2005) and Sun and Steuer (1996).

The computation time of the MDLNS can be reduced by applying *applyLNS*(s, f, E) only to a subset of E, by checking only selected solutions for dominance (e.g., best found solutions or current incumbent solutions) and ignoring other solutions that are generated during the search, and by reducing the number of LNS iterations.

4. Computational experiments

Solutions for different test instances are obtained by applying the described algorithms. In this section, we examine the results and analyze the trade-off between routing cost and service consistency. All algorithms are implemented in C++ and run on Intel Xeon X5550 computers with 2.67 GHz. Mixed integer linear programs (i.e., calls to $opt(f, \epsilon, \epsilon'))$ are solved by IBM's CPLEX 12.5. Experiments are run on a single CPU except stated otherwise. Approximate solution sets are obtained by merging the solutions of three independent MDLNS runs, i.e., we perform three independent runs and aggregate the results to a single approximation set that contains only efficient solutions. The reported computation times are average computation times for a single run. The LNS algorithm embedded in the MDLNS performs robustly with regard to solution quality; for a sample of 10 runs, the average variation coefficient (standard deviation/mean) is 0.0077 (Kovacs et al., 2014a). The stochastic variations are further mitigated in the multi-objective problem: MDLNS explores the solution space extensively and we check for each generated solution whether or not it is efficient. The parameters of the LNS are set as proposed by Kovacs et al. (2014a).

In the following, we introduce the data sets and examine the performance of the proposed solvers. In Section 4.3, we quantify the conflict between the objective functions. Additionally, we investigate the effect of lexicographically optimizing driver consistency first and travel time second on the arrival time consistency. This approach is similar to assigning customers to districts and serving each district with a separate driver on minimal cost vehicle routes. In Section 4.4, we summarize our findings.

4.1. Data sets

The test instances for the MOGenConVRP are taken from the GenConVRP (Kovacs et al., 2014a). The instances are extensions of the benchmark data set for the ConVRP by Groër et al. (2009). ConVRP instances are turned into GenConVRP instances by associating customers with AM/PM time windows. The new data sets are named $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$ ($C = \{C_{0.5}, C_{0.7}, C_{0.9}\}$), with respect to the service frequencies that are set to 50 percent, 70 percent, and 90 percent, respectively. Service frequency is a measure for the variation in the demand of the customers. A service frequency of 100 percent means that each customer is visited each day. In this case, we could achieve perfect service consistency by executing the same routing plan on each day of the planning horizon. With decreasing service frequency, there is more variation in the demand. Each data set consists of 12 instances with 50–199 customers and a planning horizon of 5 days. Additionally, we consider data set C_{small} and C_{smallE} with 10 and 5 instances, respectively. Set C_{small} contains instances with 10–12 customers that are visited over a planning horizon of 3 days. In set C_{smallE}, we extend the instances with 10 customers to a planning horizon of 5 days. Data set C_{small} and C_{smallE} are derived from the small ConVRP instances presented in Groër et al. (2009).

4.2. Results for small instances

Experiments on data set C_{small} are mainly performed for benchmarking tests. In this section, we examine the results to evaluate the efficiency of the single objective optimizer. Furthermore, we experiment with the 3D method and evaluate the performance (i.e., solution quality and computation time) of the MDLNS.

4.2.1. Evaluating the single objective optimizer

In Section 3.1.2, we proposed several valid inequalities for the single-objective model. The efficiency of these inequalities is tested by optimizing only the total travel time (f_1) of the instances in C_{small} and by bounding the maximum number of different drivers per customer (f_2) and the maximum arrival time difference (f_3). Each instance is run with three different driver consistency configurations ($z_{max} \leq Z, Z \in \{1, 2, 3\}$) and four different arrival time consistency configurations ($l_{max} \leq l \min_{(i,j)\in A^0}\{t_{ij}\}$, $l \in \{0, 1, 2, 3\}$). We have a separate set of bounds on l_{max} for each instance, but for a specific instance the bounds are the same for all solver configurations. The total number of test instances is 120, i.e., 10 instances times (3×4) configurations.

In Table 2, we present the results for different combinations of valid inequalities. The first column gives the configuration of the optimizer. Configuration "Plain" refers to the model specified by inequalities (5)–(18). The remaining rows refer to the plain model extended by the respective inequalities: *SB* are symmetry breaking inequalities, *ST*2 and *ST*3 are subtour elimination inequalities with $|S| = 2$ and $|S| = 3$, respectively, *TC*1 and *TC*2 are time consistency-related inequalities, *CC* are capacity cuts, and "All" means that all inequalities are added to the model. Capacity cuts are added only if there are not more than 100 combinations of choosing $|S|$ customers out of N_d (i.e., $\binom{N_d}{|S|} \leq 100$) and only for subsets S that require at least two vehicles (i.e., $\frac{\sum_{i\in S} q_{id}}{Q} > 1$). The second column gives the number of instances that were solved to proven optimality within 24 hours per instance. The remaining columns show the best, average, and worst speed up that is obtained compared to the plain model. The speed up values are averaged over the 83 instances that were solved by each configuration.

Each configuration is helpful in increasing the number of instances solved. However, the computation time might increase by

Table 2
Efficiency of valid inequalities for the single-objective problem on 120 test instances in C_{small}. The configurations of the optimizer are sorted in ascending order of the number of instances solved within 24 hours.

Configuration	Speed up (percent)			
	Solved	Best	Average	Worst
Plain	95	–	–	–
TC2 + SB	96	99.86	−587.74	−12753.53
ST2	96	99.81	−34.75	−3430.45
TC1	97	99.69	−63.55	−2971.62
ST3	97	99.09	14.46	−808.43
TC2 + SB + ST3	98	99.87	−448.31	−12425.50
CC	99	99.38	−3.95	−2283.39
All	101	99.96	−515.24	−14582.65
TC2 + ST3	103	98.97	−129.24	−3164.06
SB	103	99.89	−120.06	−3845.46
TC2	104	98.15	−231.78	−5785.63
SB + ST3	104	99.90	18.71	−793.68

12753 percent (from 82.75 seconds to 10636.6 seconds) in the worst case (see configuration TC2 + SB). Even though configuration TC2 + SB performs poorly, configurations TC2 and SB achieve good results separately. This indicates that combining different inequalities might deteriorate the performance significantly. The most efficient configuration is SB + ST3: we can solve 104 out of 120 instances with an average speed up of more than 18 percent; the average computation time for solving the 104 instances is 56 minutes. This configuration is used in the lexicographic optimizer $opt(f, \epsilon, \epsilon')$.

4.2.2. Experiments with the 3D method

In the MOGenConVRP, we assume that the number of vehicles is not restrictive. This assumption is necessary for achieving the highest consistency level for any problem instance: in the extreme case, we could assign each customer to one driver exclusively that visits the customer at exactly the same time of the day. However, in most solutions only a small fraction of the fleet is active; several customers are consolidated on each route in order to reduce travel cost. We define a reasonable fleet size by generating a solution with $l_{max} = 0$, $z_{max} = 1$, and a total travel time that is not worse than 1.2 times the optimal travel time. The fleet size is set to the number of vehicles used in the obtained solution. The computation time of $opt(f, \epsilon, \epsilon')$ is significantly reduced by this preprocessing approach; however, it might prevent us from finding optimal solutions.

In Table 3, we demonstrate results of experiments with the 3D method on data set C_{small}. In the first column, we list the instances: in "convrp_*x*_test_*y*.vrp" *x* is a wildcard for the number of customers and *y* is a wildcard for the instance ID. The second column gives the computation time in minutes for the preprocessing phase, $CPU_{pp}(min)$. The third column shows the resulting decrease in the fleet size $|K|$. The fourth column is the number of points on the Pareto-front $|P|$. For the 3D method, we report the number of calls to the lexicographic optimizer $opt(f, \epsilon, \epsilon')$ and the total computation time in minutes, $CPU(min)$, for generating the entire Pareto-front.

On average, the preprocessing phase described above reduces the fleet size by 73.5 percent in 1 minute. The size of the model is significantly influenced by the number of vehicles; yet, our approach might generate solutions that are not optimal with regard to travel cost. The lexicographic optimizer is called 24.7 times (the average number of points on the Pareto-front is 20). The average computation time is more than four hours. Compared to the other instances, the computation time for solving convrp_12_test_3.vrp is significantly longer when the bound on the number of different drivers is loose. In this instance, several customers are arranged almost on a ray originating from the depot presumably causing a high level of symmetry.

4.2.3. Evaluating the performance of multi-objective heuristics

The performance of multi-objective heuristics is defined by the quality of the generated set of solutions and the required computation time. With a single objective, the quality of an optimizer is proportional to the achieved objective value: the lower the objective value (in minimization problems), the better the algorithm. However, in multi-objective optimization, we need to evaluate sets of objective vectors. Zitzler, Deb, and Thiele (2000) list three features of high quality multi-objective optimizers: the distance between P_{approx} and P is small, the points in P_{approx} are distributed evenly, and the range of each objective value (i.e., for each objective function, the difference between the best and worst objective value in P_{approx}, respectively) is large. Unary quality measurements assign each approximation set a single value that reflects a certain quality feature (Zitzler, Thiele, Laumanns, Fonseca, & da Fonseca, 2003). Typically, heuristics are evaluated by several quality measurements. Binary quality measurements assign a value to each pair of approximation sets and are used for comparing heuristics by pairs. Papers that examine quantitative approaches for evaluating multi-objective optimizers are presented, e.g., by Sarker and Coello Coello (2002), Van Veldhuizen and Lamont (2000), Knowles and Corne (2002), and Zitzler et al. (2003).

We evaluate the quality of the MDLNS by applying five unary quality measurements. Each measurement has drawbacks that prevent us from using it as the only quality measurement. A meaningful evaluation of the algorithm is possible only if all measurements are considered at the same time.

Number of points on the Pareto-front. Providing more efficient solutions means giving the decision maker more choices (Schott, 1995; Van Veldhuizen, 1999). The cardinality of the Pareto-front ($|P_{approx}|$) gives the number of alternative solutions generated by the algorithm. However, this measurement ignores the quality of the approximation: a single solution might dominate the entire approximation set.

Table 3
Experiments with 3D method on data set C_{small}.

Instances				3D method	
Instances	$CPU_{pp}(min)$	Imp_K(percent)	$\|P\|$	Calls $opt()$	$CPU(min)$
convrp_10_test_1.vrp	0.01	70	16	19	5.41
convrp_10_test_2.vrp	0.09	80	20	24	8.39
convrp_10_test_3.vrp	0.10	70	10	11	21.76
convrp_10_test_4.vrp	0.23	70	10	13	54.32
convrp_10_test_5.vrp	0.02	70	17	22	21.48
convrp_12_test_1.vrp	5.90	75	21	26	260.82
convrp_12_test_2.vrp	0.07	75	25	32	178.11
convrp_12_test_3.vrp	3.69	75	34	45	1266.44
convrp_12_test_4.vrp	0.46	75	26	31	679.51
convrp_12_test_5.vrp	0.02	75	21	24	48.25
Average	1.06	73.50	20.00	24.70	254.45

Table 4
Quality evaluation of the MDLNS.

	Optimal	3×10^4 iterations					4×10^4 iterations					5×10^4 iterations				
	$\|P\|$	$\|P_{approx}\|$	ER	GD	I_ϵ	$GapHV$	$\|P_{approx}\|$	ER	GD	I_ϵ	$GapHV$	$\|P_{approx}\|$	ER	GD	I_ϵ	$GapHV$
Average C_{small}	20.0	21.3	0.253	0.127	1.025	0.449	21.1	0.241	0.128	1.025	0.477	21.2	0.257	0.120	1.022	0.424
Average C_{smallE}	36.2	39.2	0.351	0.207	1.089	0.720	41.0	0.398	0.235	1.079	0.796	38.6	0.393	0.167	1.092	0.795

Additionally, decision makers might be overwhelmed by an (unnecessarily) large number of solutions.

Error ratio. The error ratio (ER) (Van Veldhuizen, 1999) gives the share of points in P_{approx} that are not in P:

$$ER = \frac{\sum_{p \in P_{approx}} e_p}{|P_{approx}|}. \tag{31}$$

For each point $p \in P_{approx}$, e_p is 0 if $p \in P$ and 1, otherwise. Lower ER values indicate better approximations; yet, the value may be misleading. Consider, for example, two approximation sets: one with a single solution that is $\in P$ and one that contains many alternative solutions $\in P$ and one solution $\notin P$. The first set has an ER value of zero and would, therefore, be preferred to the second set with $ER > 0$.

Generational distance. The generational distance (GD) gives the distance between P_{approx} and P (Rudolph, 1998; Van Veldhuizen & Lamont, 1998, 1999):

$$GD = \frac{\sqrt{\sum_{p \in P_{approx}} d_p^2}}{|P_{approx}|}; \tag{32}$$

d_p is the Euclidean distance from a point p to the closest point $q \in P$ ($d_p = min_{q \in P}\sqrt{(p-q)(p-q)}$). A generational distance of zero indicates that all points in P_{approx} are also in P; the larger GD, the larger the distance between the approximation set and the optimal Pareto-front. Similar to the error ratio, GD may favor algorithms that generate one solution $\in P$ over algorithms that find many solutions $\in P$ but also solutions $\notin P$. Additionally, the results will be biased if the objective values are of different orders of magnitude.

Unary ϵ-indicator. The unary ϵ-indicator (I_ϵ) defines by how much an approximation set is worse than the optimal set with regard to all objective functions (Zitzler et al., 2003):

$$I_\epsilon = \max_{q \in P} \min_{p \in P_{approx}} \max_{j \in \{1,2,3\}} \frac{p_j}{q_j}. \tag{33}$$

For each point $p \in P_{approx}$ and each point $q \in P$, we have $p_j \leq I_\epsilon q_j \forall j = \{1, 2, 3\}$; $I_\epsilon = 1$ indicates that all points in P_{approx} are also in P. The ϵ-indicator is a worst case quality measurement; so, the quality evaluation might be based on a single poor solution in P_{approx}.

Hypervolume. Hypervolume (HV) is the size of the bounded objective space that is dominated by a set of solutions P:

$$HV = \left\{ \bigcup_p c_p | p \in P \right\}. \tag{34}$$

Each point $p \in P$ covers a space c_p where c_p is the volume of the cuboid that is defined by p and a reference point r ($r_j \geq p_j \; \forall \; p \in P, j = \{1, 2, 3\}$). An example of a hypervolume in the two-dimensional objective space is given in Fig. 4: p and q are elements of P, r is a reference point, and c_p and c_q are areas dominated by p and q, respectively. The union of c_p and c_q is the hyperarea. The HV measurement is sensitive to the choice of the reference point as it may affect the quality

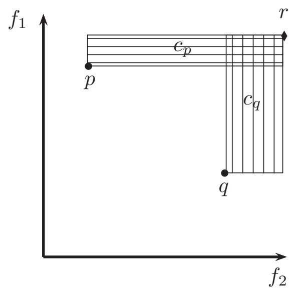


Fig. 4. Example for the hypervolume in the two-dimensional objective space. .

evaluation (Knowles & Corne, 2002). For easier comparability, we report the gap between the HV of the optimal set P (HV_P) and the HV value of the approximation set P_{approx} ($HV_{P_{approx}}$):

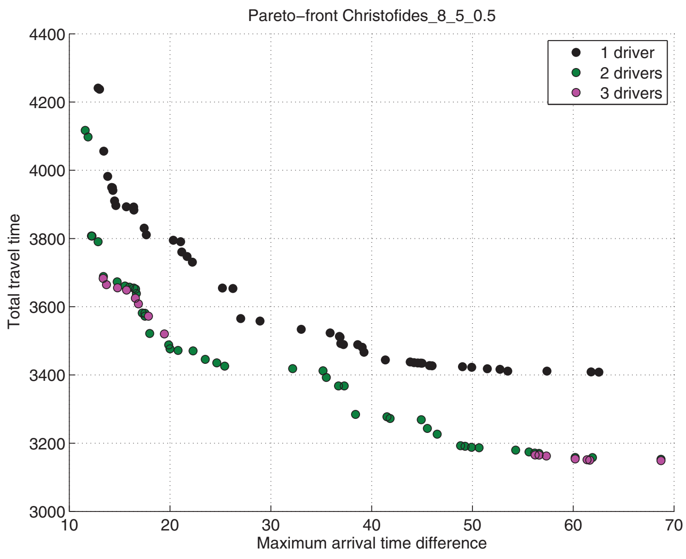
$$GapHV = \frac{100(HV_P - HV_{P_{approx}})}{HV_P} \tag{35}$$

Computing HV in a three-dimensional objective space is non-trivial. We apply version 1.3 of the recursive, dimension-sweep algorithm for computing the hypervolume proposed and implemented by Fonseca, Paquete, and López-Ibáñez (2006) and Fonseca, López-Ibáñez, Paquete, and Guerreiro (2010).

4.2.4. Evaluating the performance of the MDLNS

In Table 4, we evaluate the quality of the MDLNS on data sets C_{small} and C_{smallE}. Solution sets are aggregated over three runs and results are averaged over all instances. The second column gives the number of points in P. The remaining columns show the five unary quality indicators for the MDLNS with 30, 40, and 50 thousand iterations per LNS run. The number of iterations seems to have a low influence on the solution quality. Starting from different solutions, the MDLNS explores the solution space by optimizing several objective functions. Therefore, the algorithm performs robustly regardless of the chosen parameters. Running the LNS with 20 thousand iterations does not provide the required performance; we have (ER, GD, I_ϵ) = (0.411, 0.204, 1.096), on average, on data set C_{smallE}. In both data sets, each MDLNS configuration provides at least as many points on average as there are points on the Pareto-front. In C_{small}, on average, 24.1 percent to 25.7 percent of the solutions are not element of P; in C_{smallE} the average ER value is between 35.1 percent and 39.8 percent. The average generational distance is between 0.120 and 0.128 in C_{small} and between 0.167 and 0.235 in C_{smallE}. The average unary ϵ-indicator shows that the approximation set is between 2.2 percent and 2.5 percent worse than the optimal set in C_{small} and less than 9.2 percent worse in C_{smallE}. The MDLNS provides approximation sets that, on average, have only 0.424–0.477 percent smaller hypervolumes than P ($GapHV$) in C_{small} and between 0.720 percent and 0.796 percent smaller hypervolumes in C_{smallE}. For each instance, the reference point for computing the hypervolume is (r_1, r_2, r_3) where r_1 is the longest total travel time, r_2 is the maximum number of drivers per customer, and r_3 is the maximum arrival time difference among all solutions in the examined approximation sets.

The approximation sets generated by a single run with 40 thousand iterations are, on average, 0.64 percent worse in terms



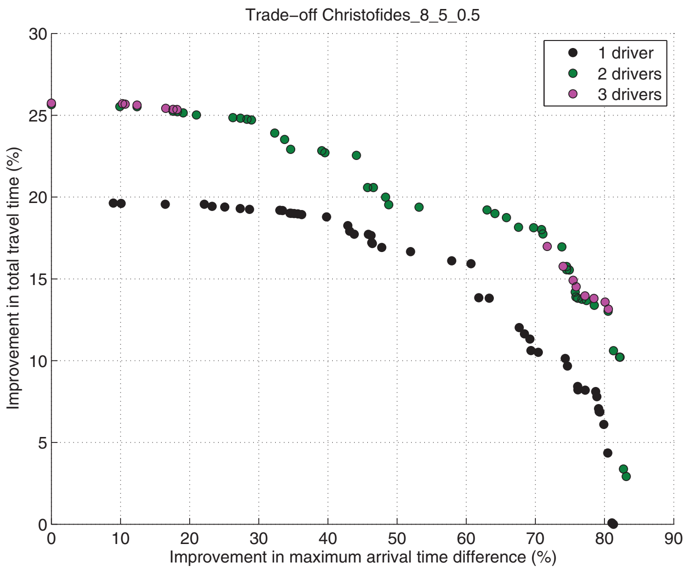


Fig. 5. Example of a Pareto-front and the associated improvement front of instance “Christofides_8_5_0.5”. The instance ID is 8, the planning horizon is 5 days, and the service frequency is 50 percent.

Table 5
Runtime evaluation of the MDLNS on data set C_{small}. The reported average computation time is given in minutes per run.

	3D method	MDLNS		
		30k	40k	50k
convrp_10_test_1	5.41	0.61	0.86	1.09
convrp_10_test_2	8.39	1.07	1.42	1.65
convrp_10_test_3	21.76	0.59	0.80	0.98
convrp_10_test_4	54.32	0.65	0.82	1.03
convrp_10_test_5	21.48	1.13	1.47	1.75
convrp_12_test_1	260.82	1.50	2.11	2.62
convrp_12_test_2	178.11	2.29	3.29	3.30
convrp_12_test_3	1266.44	2.00	2.77	3.29
convrp_12_test_4	679.51	1.82	2.26	2.70
convrp_12_test_5	48.25	1.08	1.42	1.69
Average	254.45	1.27	1.72	2.01

Table 6
Runtime evaluation of the MDLNS on data set C_{smallE}. The 3D method is run on eight threads; the reported time is the wall-clock time in minutes per run.

	3D method	MDLNS		
		30k	40k	50k
convrpNew_10_1_5	836.47	3.56	5.63	5.76
convrpNew_10_2_5	3953.57	3.21	3.99	6.73
convrpNew_10_3_5	2434.82	2.44	3.05	3.62
convrpNew_10_4_5	1540.03	2.80	4.32	5.34
convrpNew_10_5_5	427.33	5.17	5.18	6.86
Average	1838.44	3.43	4.43	5.66

of the ϵ-indicator in C_{small} and 1.34 percent worse in C_{smallE}; the average *GapHV* increases from 0.477 percent to 0.562 percent in C_{small} and from 0.796 percent to 1.134 percent in C_{smallE}. The results indicate that we cannot generate better solutions by increasing the number of iterations. However, aggregating approximation sets from independent runs can improve the solution quality. We conclude that the MDLNS approach is appropriate for performing a meaningful trade-off analysis.

The computation time of the MDLNS is examined in Table 5 for data set C_{small} and in Table 6 for data set C_{smallE}. As a reference point, we report the computation time of the 3D method in the second column; the remaining columns show the average computation time for the MDLNS with 30, 40, and 50 thousand iterations per LNS run, respectively. On data set C_{small} (Table 5), the MDLNS requires not more than 2.01 minutes on average for a single run. This result is noticeable when compared to the 3D method that requires more than 4 hours. For solving the instances in C_{smallE} (Table 6), we run the 3D method in parallel on eight CPUs; the reported time is the wall-clock time. MDLNS runs on a single CPU. The 3D method requires more than 30 hours while MDLNS provides results between 3.34 and 5.66 minutes on average per run. This result indicates that applying the proposed exact solution approaches in practice is unrealistic.

4.3. *Results for large instances (data sets* $C_{0.5}$, $C_{0.7}$, *and* $C_{0.9}$*)*

The trade-off analysis between total travel time, driver consistency, and arrival time consistency is performed on data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$. The results are obtained by applying the MDLNS with 40 thousand iterations per LNS run. Solution sets are aggregated over three independent runs and dominated solutions are removed. An example of the output of the algorithm is shown at the left of Fig. 5. The horizontal axis of the diagram shows the maximum arrival time difference and the vertical axis shows the total travel time. The maximum number of drivers per customer is distinguished by colors; each f_2 value is associated with a different color.

Each point on the Pareto-front P represents a solution in the set of efficient solutions E, i.e., for each point $p = (p_1, p_2, p_3)$, there exists a solution $s \in E$ such that $p = f(s)$. Points p with a lower p_3 value than 0.01 $\max_{p \in P}\{p_3\}$ are excluded from the approximation set. In instance Christofides_4_5_0.7 with a planning horizon of 5 days and a service frequency of 70 percent, for example, the travel time increases by 332 percent in the solution with $f_3 = 0$ compared to the solution with the second best f_3 value. Solutions with very small f_3 values and very large f_1 values would distort the analysis and are, therefore, treated as outliers. We think this approach would also be justified from a managerial perspective, because aiming for perfect arrival time consistency is unreasonable.

The magnitude of the objective values varies instance-by-instance. In order to make general statements, we transform the Pareto-front P into an improvement front T that shows the percentage improvement in one objective as a function of the percentage

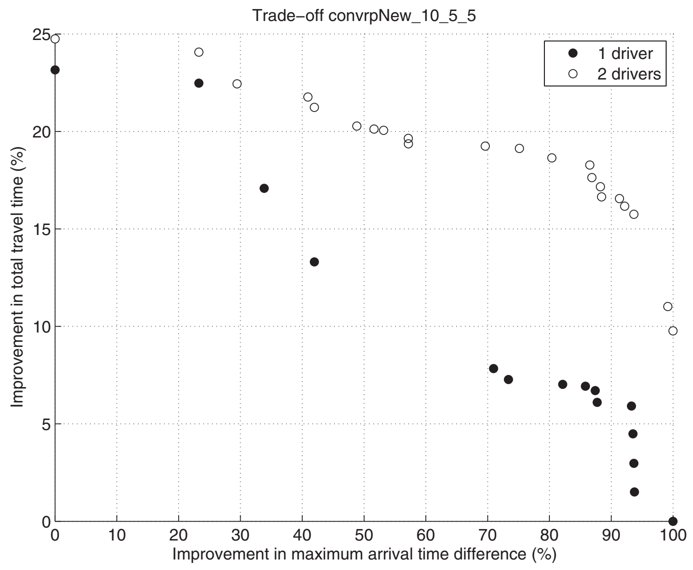


Fig. 6. Partly convex relationship between travel time and arrival time consistency at the left and stepwise relationship at the right.

improvement in another objective:

$$T = \left\{ (a, b, c) : a = \frac{p_1^{\mathcal{N}} - p_1}{p_1^{\mathcal{N}}}, b = p_2, c = \frac{p_3^{\mathcal{N}} - p_3}{p_3^{\mathcal{N}}} \ \forall \ p \in P \right\}. \quad (36)$$

Each point on the Pareto-front is transformed into a triple (a, b, c): a and c are the improvements in f_1 and f_3, respectively, compared to the Nadir-point $(p_1^{\mathcal{N}}, p_3^{\mathcal{N}})$ ($p_1^{\mathcal{N}} = \max_{p\in P}\{p_1\}$ and $p_3^{\mathcal{N}} = \max_{p\in P}\{p_3\}$); b is the number of drivers per customer in the respective solution. The right figure in Fig. 5 shows the improvement front associated with the Pareto-front at the left. By putting the absolute values into perspective, we can aggregate several results in order to conduct a meaningful analysis.

4.3.1. Cost of arrival time consistency

The correlation between total travel time and maximum arrival time difference is unclear when examining single results. For example, in the figure on the left of Fig. 6, we see a partly convex relationship, i.e., we have to give up much travel time in order to improve arrival time consistency a little. In the figure on the right, the maximum arrival time difference seems to improve gradually with increasing travel time. We examine the trade-off that is to be expected by aggregating the results of several instances as follows: The hull of the f_1–f_3 projection of the improvement front is defined by a set of line segments that are parallel to the f_3-axis and bound the improvement in f_1 from above. Let $T^{f_1-f_3}$ denote the subset of T that contains only points that are non-dominated in the f_1–f_3 plane. The points in $T^{f_1-f_3}$ are sorted in ascending order of the improvements in f_3; the line segment between each pair of neighboring solutions (p^i, p^{i+1}) is defined by the improvement in f_1 of point p^{i+1}. By quantizing the improvement in f_3 and averaging the hull over several instances, we can aggregate the results of several instances and observe the correlation between the maximum arrival time difference and the total travel time that is to be expected. An example of a quantized hull, illustrated by horizontal dotted lines, is given at the left of Fig. 7; the average improvement curve for data sets with different service frequency is given at the right.

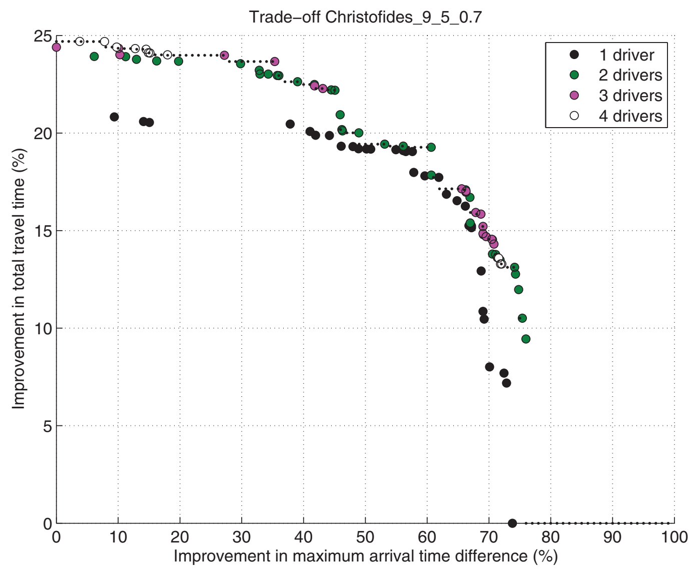
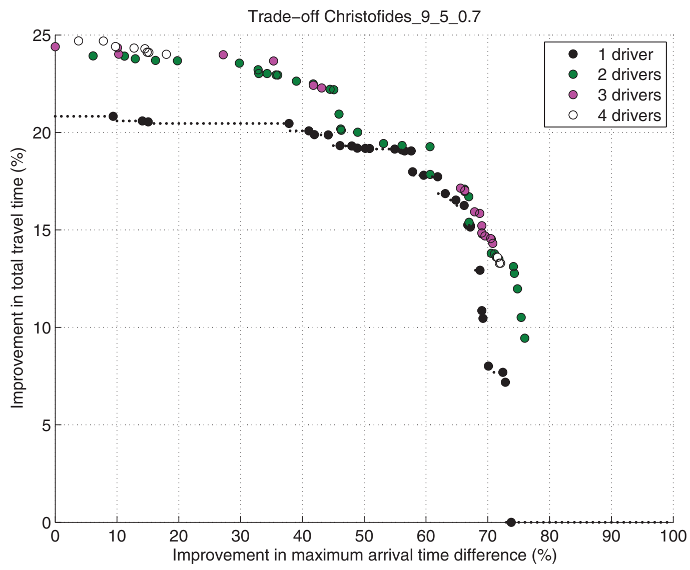


Fig. 7. Hull of instance Christofides_9_5_0.7 (indicated by horizontal dotted lines) when driver consistency is ignored (i.e., in the relaxed scenario) at the left and the average improvement curves for different service frequencies at the right.



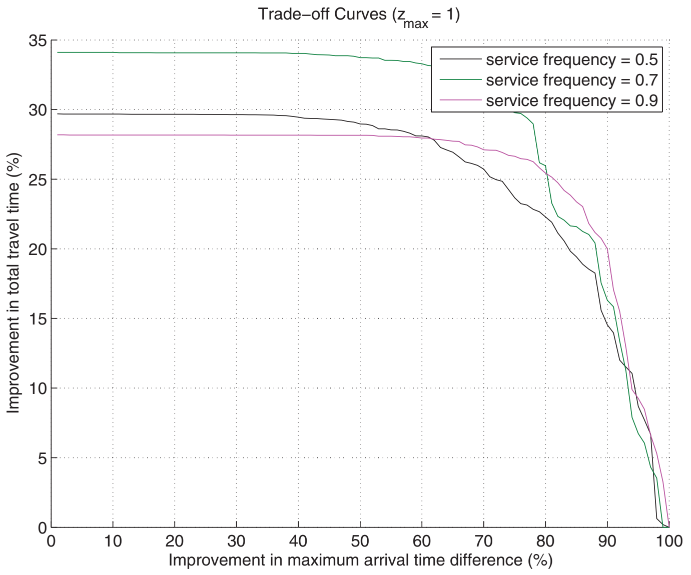


Fig. 8. Hull of instance Christofides_9_5_0.7 (indicated by horizontal dotted lines) when driver consistency is fixed to one driver per customer (i.e., in the strict scenario) at the left and the average improvement curves for different service frequencies at the right.

Table 7
Cost of arrival time consistency in percent when driver consistency is ignored for data sets with 50 percent, 70 percent, and 90 percent service frequency (i.e., data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$).

Imp l_{max} (percent)	Increase in TT (percent)			
	$C_{0.5}$	$C_{0.7}$	$C_{0.9}$	Average
10	0.09	0.08	0.14	0.10
30	0.53	0.55	0.39	0.49
50	1.86	1.77	1.11	1.58
70	5.09	3.99	2.46	3.84
90	17.92	21.42	11.86	17.07

Table 8
Cost of arrival time consistency in percent when driver consistency is fixed to one driver per customer for data sets with 50 percent, 70 percent, and 90 percent service frequency (i.e., data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$).

Imp l_{max} (percent)	Increase in TT (percent)			
	$C_{0.5}$	$C_{0.7}$	$C_{0.9}$	Average
10	0.02	0.00	0.01	0.01
30	0.06	0.04	0.02	0.04
50	0.72	0.38	0.03	0.38
70	3.97	2.24	1.07	2.43
90	15.73	18.27	10.42	14.81

By ignoring driver consistency, we obtain a relaxed improvement curve between cost and arrival time consistency. For examining a strict scenario, we use the same principle as above but the hull is defined only for points in $\{T|f_2 = 1\}$. An example of this hull, again illustrated by horizontal dotted lines, is given at the right of Fig. 8; the left figure shows the average improvement curves for different service frequencies when $f_2 = 1$. The findings are summarized in Tables 7 and 8. The tables show by how much the total travel time increases compared to the solutions with minimal travel time when the maximum arrival time difference is decreased in the relaxed scenario and in the strict scenario, respectively. The first column is the improvement in l_{max}; columns 2–4 are the percentage increase in the total travel time (TT) for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively. The last column gives the average increase in the travel time. The results are similar in both tables: we can improve arrival time consistency by 50 percent at the cost of increasing travel time by less than 1.58 percent, on average. Decreasing l_{max} by 70 percent is achieved at 3.84 percent higher cost when driver consistency is ignored and at 2.43 percent higher cost when driver consistency is optimal, i.e., there is one driver per customer. Decreasing l_{max} further significantly increases travel time: the travel time increases by 17.07 percent in the relaxed scenario and by 14.81 percent in the strict scenario.

4.3.2. Cost of driver consistency

We can use the hull of the improvement front for examining the cost of driver consistency. In Table 9, we compare the relaxed scenario to the strict scenario. For each data set and for different levels of arrival time consistency, we give the average increase in travel time in the strict scenario ($z_{max} = 1$) and in the relaxed scenario ($z_{max} \leq \infty$), respectively, compared to the solution with minimal travel time (i.e., the solution that ignores arrival time and driver consistency). Column Diff gives the difference between the strict scenario and the relaxed scenario. The comparison shows the cost of perfect driver consistency for different levels of arrival time consistency.

The larger the maximum arrival time difference, the higher the cost of driver consistency: the average increase in travel time caused by servicing each customer by a single driver is 4.61 percent, 3.88 percent, and 2.08 percent for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively, when l_{max} is improved by 10 percent. At the highest level of arrival time consistency (improving l_{max} by 90 percent), the increase in travel time decreases to 2.49 percent, 0.81 percent, and 0.77 percent in $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively. This result suggests that arrival time consistency and driver consistency can be optimized simultaneously.

The cost of driver consistency is further examined by defining a variant of the unary ϵ-indicator (see Section 4.2.3):

$$I_{\epsilon}^{xQ} = \underset{q \in P^{Z+1}}{\text{x-quantile}} \min_{p \in P^{Z}} \max_{i \in \{1,3\}} \frac{p_i}{q_i}. \tag{37}$$

Sets P^Z and P^{Z+1} are subsets of the Pareto-front that contain only points with $p_2 = Z$ and $p_2 = Z + 1$, respectively. The modified ϵ-indicator gives the value for which x percent of the solutions get less than I_{ϵ}^{xQ} times worse if z_{max} is reduced from $Z + 1$ to Z.

Figs. 9 , 10, and 11 show the average I_{ϵ}^{xQ} values for $x = 25$ percent, 50 percent, and 75 percent, respectively. In each figure, we show by how much the solutions deteriorate for different Z values and for different service frequencies (i.e., for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$).

Table 9
Cost of arrival time consistency in percent when driver consistency is fixed to one driver per customer compared to solutions in which driver consistency is ignored; $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$ are data sets with 50 percent, 70 percent, and 90 percent service frequency.

Imp l_{max} (percent)	Increase in *TT* (percent)								
	$C_{0.5}$			$C_{0.7}$			$C_{0.9}$		
	$z_{max} = 1$	$z_{max} \leq \infty$	Diff	$z_{max} = 1$	$z_{max} \leq \infty$	Diff	$z_{max} = 1$	$z_{max} \leq \infty$	Diff
10	4.70	0.09	4.61	3.96	0.08	3.88	2.22	0.14	2.08
30	4.74	0.53	4.20	3.99	0.55	3.44	2.23	0.39	1.84
50	5.40	1.86	3.55	4.34	1.77	2.57	2.24	1.11	1.13
70	8.65	5.09	3.56	6.20	3.99	2.21	3.28	2.46	0.83
90	20.41	17.92	2.49	22.22	21.42	0.81	12.63	11.86	0.77

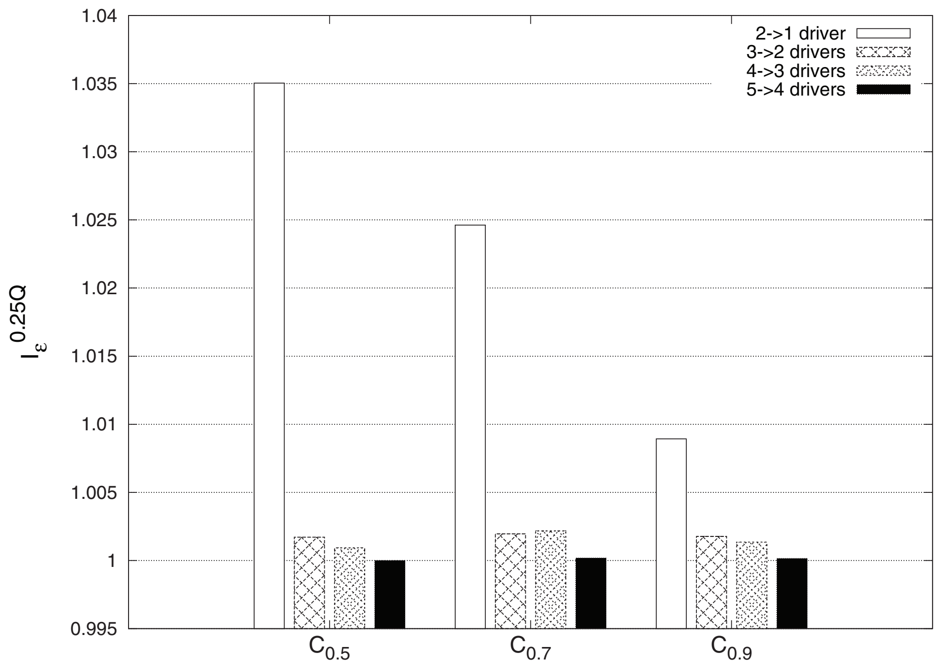


Fig. 9. First quartile of the average increase in cost for improving driver consistency. A value of $I_\epsilon^{0.25Q} = 1$ indicates that the solutions are not deteriorating when the maximum number of different drivers per customer is reduced. Results are given for different levels of driver consistency and different service frequencies.

The pattern in all figures is similar: the solutions deteriorate most if the number of drivers per customer is reduced from two drivers to one driver. The difference between solutions is minor when $Z > 1$; in this case, the solutions do not deteriorate by more than 0.9 percent. The effect of driver consistency decreases with increasing service frequency, i.e., decreasing fluctuations in the demand. A quarter of the solutions with $z_{max} = 1$ in data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$ deteriorates by less than 3.5 percent, 2.5 percent, and 0.9 percent, respectively, compared to solutions with $z_{max} = 2$ (Fig. 9); half of the solutions deteriorate by less than 5.1 percent, 3.7 percent, and 1.5 percent in data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively (Fig. 10). In three quarters of the solutions, the cost of achieving perfect driver consistency increases by less than 6.7 percent, 5 percent, and 2.3 percent in in data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively (Fig. 11).

4.3.3. Effect of strict driver consistency on arrival time consistency

In this section, we examine the effect of enforcing perfect driver consistency on arrival time consistency when the focus is on travel cost. Many companies, e.g., in the small package shipping industry, perform a districting strategy in order to reduce the operational complexity. Here, the service territory is partitioned into smaller areas called districts and each driver is assigned to one district. Districts are designed in a tactical phase and routes are planned in an operational phase. Typically, the focus in both phases is on travel cost.

Visiting each customer with only one driver is equivalent to assigning customers to a district that is served by the same driver each day. Bounding the number of different drivers by one results in disjoint districts but, in contrast to districting approaches, the districts might be non-convex and dispersed over a large area. Each customer experiences perfect driver consistency but the effect of districting on arrival time consistency is unclear. Our intuition is that it is easier to achieve arrival time consistency for each district separately than for the entire service territory. With districting there is less room for reducing travel cost by visiting customers outside a district and, therefore, daily routes become more similar. An example is illustrated in the left figure of Fig. 12. By restricting the maximum number of different drivers to one, we implicitly improve arrival time consistency by 81 percent compared to the solution without districting when travel cost is the primary objective. However, in the figure on the right, we see the opposite: with districting, we have a 32 percent worse arrival time consistency (and longer travel time), i.e., improving arrival time consistency and reducing the number of different drivers per customer are conflicting objectives.

In Table 10, we present aggregated results that show the effect of strict driver consistency on arrival time consistency. We report the number of instances in which the arrival time difference is larger with districting (i.e., with emphasis on cost and $z_{max} = 1$) than without (# Worst l_{max}) and the average improvement in l_{max}. (Each l_{max} value

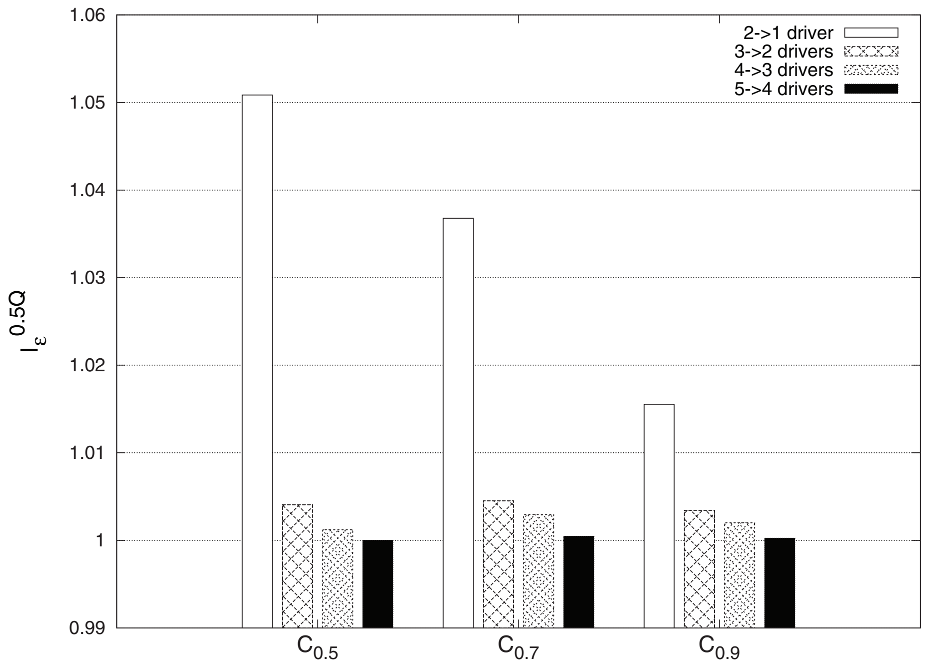


Fig. 10. Median of the average increase in cost for improving driver consistency. A value of $I_\epsilon^{0.5Q} = 1$ indicates that the solutions are not deteriorating when the maximum number of different drivers per customer is reduced. Results are given for different levels of driver consistency and different service frequencies.

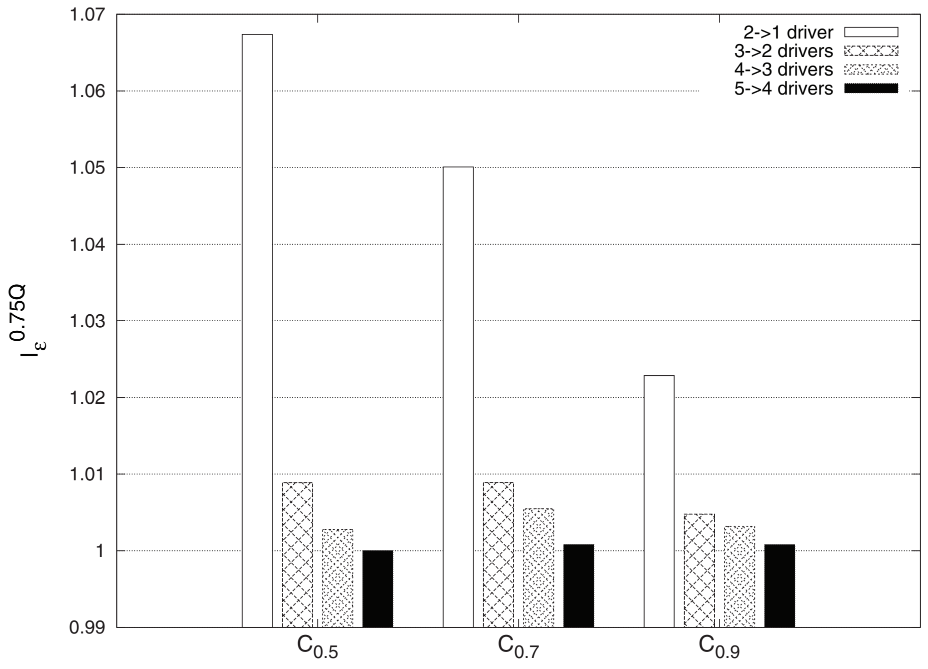


Fig. 11. Third quartile of the average increase in cost for improving driver consistency. A value of $I_\epsilon^{0.75Q} = 1$ indicates that the solutions are not deteriorating when the maximum number of different drivers per customer is reduced. Results are given for different levels of driver consistency and different service frequencies.

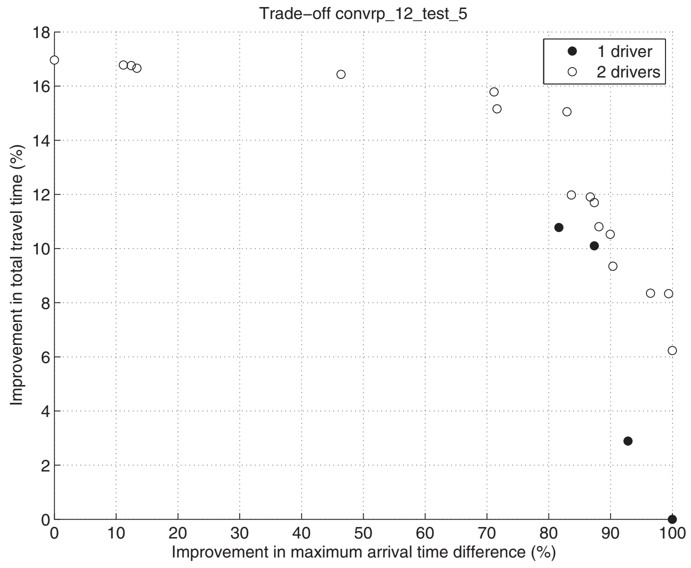
Table 10
Effect of strict driver consistency on arrival time consistency when travel cost is the primary objective. # Worst l_{max} is the number of instances in which l_{max} is larger with districting than without, Average Imp. l_{max} is the average improvement in l_{max} compared to the solution with minimal travel time and without districting, and I'_ϵ gives the factor by how much solutions deteriorate because of districting.

	C_{small}	C_{smallE}	$C_{0.5}$	$C_{0.7}$	$C_{0.9}$
# Worst l_{max}	6/10	0/5	1/12	0/12	0/12
Average Imp. l_{max} (percent)	13.34	26.02	37.54	35.33	54.25
I'_ϵ	1.145	1.048	1.052	1.049	1.014

refers to the solution in P_{approx} with minimal total travel time.) Additionally, we use another variant of the unary ϵ-indicator to measure by how much the districting solution, denoted by p, is worse compared to solutions in which each customer may be visited with an arbitrary number of different drivers (denoted by $P^{Z\leq\infty}$):

$$I'_\epsilon = \min_{q\in P^{Z\leq\infty}} \max_{i\in\{1,3\}} \frac{p_i}{q_i}. \tag{38}$$

The effect of districting depends primarily on the number of days in the planning horizon. The planning horizon involves 3 days in data set C_{small} and 5 days in the remaining data sets (i.e., C_{smallE}, $C_{0.5}$, $C_{0.7}$,



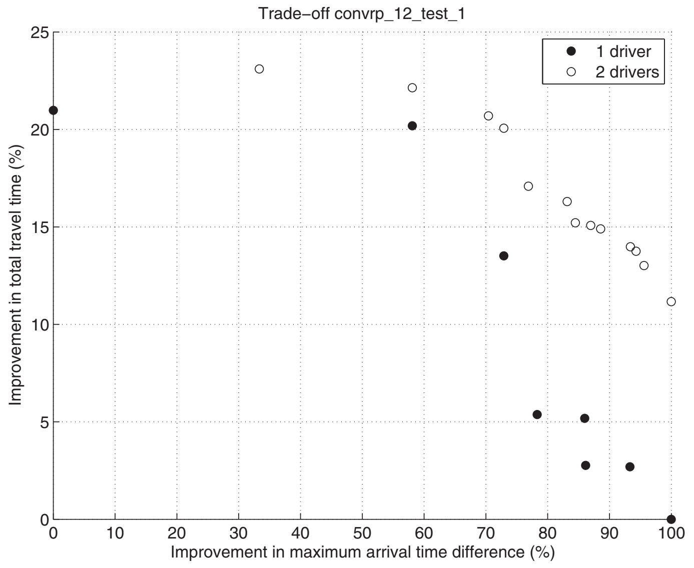


Fig. 12. Two examples of the effect of districting on arrival time consistency when travel cost is the primary objective: districting improves arrival time consistency implicitly in the left figure. In the right figure, arrival time consistency is worse with districting than without.

and $C_{0.9}$). In C_{small}, the arrival time difference increases in 6 out of 10 instances; nevertheless, l_{max} decreases by 13.34 percent on average; in terms of the I'_ϵ indicator, the closest efficient solution without districting is 1.145 times better, on average, than the solution with districting. In data set C_{smallE}, l_{max} improves in each of the five instances due to strict driver consistency; the average improvement is 26 percent. The solutions are, on average, 4.8 percent worse compared to solutions without districting, i.e., $I'_\epsilon = 1.048$. A positive impact of districting on arrival time consistency is observed in the large instances. The arrival time difference increases in only one out of 36 instances and the average improvement is between 35.33 percent and 54.25 percent. The cost of districting, i.e., I'_ϵ, decreases with larger service frequency from 1.052 in data set $C_{0.5}$ to 1.014 in data sets $C_{0.9}$.

We remark that different strategies for partitioning the service territory might influence the effect of districting on arrival time consistency. For example, the results might be significantly different when customers are assigned to drivers not only with regard to cost but also with regard to workload balancing.

4.4. Discussion of the results

Based on different unary quality indicators, we can summarize that the proposed MDLNS is an eligible algorithm for solving the MO-GenConVRP. Compared to the 3D method, the heuristic provides good approximation sets in reasonable amount of time. MDLNS is specialized in finding the best possible approximation of the optimal Pareto-front: we check each solution generated during the search process whether or not it is dominated and we apply the LNS for a large number of iterations. The average computation time on the large instances (data set C) is 23.2 hours (the average number of elements in P is 88.5). Nevertheless, the MDLNS is applicable in the field subject to slight modifications, e.g., by checking only selected solutions for efficiency and by reducing the number of LNS and MDLNS iterations, respectively.

The trade-off between travel time, driver consistency, and arrival time difference depends significantly on the fluctuations in the demand: the lower the service frequency, the more costly it is to achieve service consistency. Adequate levels of arrival time consistency can be provided with modest increase in travel time. Compared to the minimal cost routing plan, the maximum arrival time difference can be reduced by 50 percent at the cost of 1.58 percent longer travel time, on average; improving arrival time consistency by 70 percent costs 3.84 percent. Feillet et al. (2014) report 5.9 percent higher routing cost when arrival time consistency is considered; the increase in Groër et al. (2009) is between 6.6 percent and 15 percent (the results in Groër et al. (2009) also include the cost of perfect driver consistency.) The vehicle departure time from the depot is fixed in Feillet et al. (2014) and Groër et al. (2009); with this restriction, a 60 percent tighter constraint on l_{max} increases cost by up to 186.15 percent if departure times from the depot are fixed (Kovacs et al. (2014c)). High levels of time consistency can be provided with small increases in travel cost when departure times are flexible (Kovacs et al., 2014a, 2014c). Improving arrival time consistency by 90 percent can still be achieved at 17.07 percent higher travel cost, on average. However, perfect arrival time consistency is not manageable at a reasonable cost.

In many cases, the level of arrival time consistency affects the cost of driver consistency: Visiting each customer with the same driver when arrival time consistency is improved by 10 percent increases travel time between 2.08 percent and 4.61 percent compared to the solution with minimal travel time. Achieving perfect driver consistency when l_{max} is decreased by 90 percent increases travel time between 0.77 percent and 2.49 percent. Perfect driver consistency is significantly more expensive than visiting each customer with two different drivers: the routing cost increases by up to 5.1 percent in 50 percent of the solutions; visiting each customer by two drivers instead of three drivers increases cost by less than 0.9 percent. Similar results have been presented in the literature: Feillet et al. (2014) report cost savings of up to 7.5 percent if driver consistency is ignored. Enforcing perfect driver consistency increases cost by up to 9.85 percent in the inventory routing problem (Coelho et al., 2012). In the driver assignment vehicle routing problem, the increase in cost decreases from 12 percent to 2.9 percent if a perfect driver consistency is enforced only for 75 percent of the customers (Spliet, 2013). Francis et al. (2007) and Smilowitz et al. (2013) report that improving driver consistency increases routing cost by up to 6.1 percent and 5.2 percent, respectively. Allowing more than one driver per customer reduces the routing cost by up to 6.5 percent in the GenConVRP (Kovacs et al., 2014a).

A strict driver consistency, as it is enforced, e.g., in districting applications, improves arrival time consistency in most cases as a side benefit. The average maximum arrival time difference decreases between 35.33 percent and 54.25 percent as a consequence of

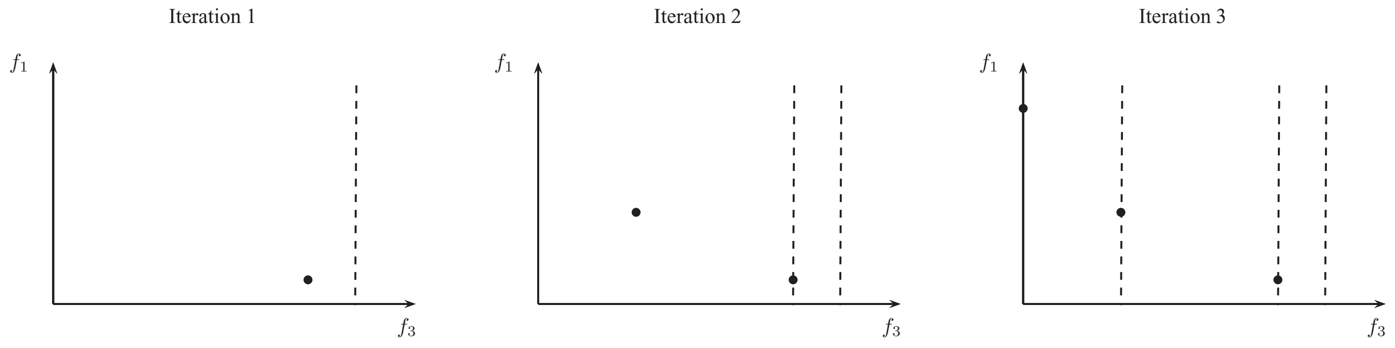


Fig. A1. Objective space for three iterations of the inner loop (lines 5–10) of Algorithm 3, i.e., the bi-objective problem for an arbitrary bound on f_2.

allowing only one driver per customer when travel cost is the primary objective.

5. Conclusion

Multi-objective optimization is a suitable approach for investigating the trade-off between conflicting objectives. In this paper, we introduced the multi-objective generalized consistent vehicle routing problem; the objective functions minimize the total travel cost and improve driver consistency (i.e., the maximum number of different drivers a customer encounters) and arrival time consistency (i.e., the maximum difference between the earliest and the latest arrival time). We proposed two exact solution approaches and one heuristic. The exact approaches are based on the ϵ-constraint framework. The heuristic is a combination of the large neighborhood search for the generalized consistent vehicle routing problem and the multi directional local search framework; we refer to our heuristic as multi directional large neighborhood search (MDLNS). Solving the MOGenConVRP is challenging; therefore, we apply the exact solution approaches only to small problem instances. The quality of the MDLNS is evaluated by comparing the set of approximate solutions to the set of efficient solutions. Based on five multi-objective quality indicators, we showed that MDLNS provides good solution sets in reasonable amount of time. The trade-off analysis is performed on problem instances of realistic size.

An important managerial implication is that a 70 percent better arrival time consistency is achieved by increasing travel time by not more than 3.84 percent, on average. Interestingly, the average cost of visiting each customer with a single driver depends on the level of arrival time consistency: the larger the maximum arrival time difference, the higher the average cost of driver consistency, i.e., arrival time consistency and driver consistency tend to be non-conflicting.

Our results also show that

- arrival time consistency improves as a side effect of visiting each customer with a single driver;
- perfect driver consistency is significantly more expensive than allowing two different drivers per customer; visiting each customer with the same driver increases travel time by up to 5.1 percent in half of the trade-off solutions;
- the actual cost of providing consistent customer service depends on the level at which demand is fluctuating: the stronger the fluctuation, the higher the cost of providing service consistency.

Future research should involve the validation of our results in real-world applications. Another challenge is developing a decision support system that guides decision makers in selecting an appropriate trade-off solution.

Acknowledgments

The authors would like to thank the two anonymous referees for constructive comments and suggestions. The second author is supported by the Austrian Science Fund (FWF): T514-N13. The computational results presented have partly been achieved using the Vienna Scientific Cluster (VSC). This support is gratefully acknowledged.

Appendix A. Two dimensional adaptive ϵ-constraint method

The two dimensional (2D) adaptive ϵ-constraint method is based on the observation that the number of different drivers per customer (f_2) is, typically, low (in many cases less than $|D|$). Therefore, we can enumerate the bi-objective problem with objectives f_1 and f_3 for all possible bounds on f_2. For any f_2 value, the bi-objective problem is solved by repeatedly executing $opt(f, \epsilon, \epsilon')$ with modified ϵ-constraints.

Fig. A1 illustrates the general idea of the algorithm for a given bound on f_2. The dashed lines denote ϵ-constraints on f_3 and the dots represent the objective vector of solutions found by $opt(f, \epsilon, \epsilon')$. The left figure shows the initial bound on f_3 and the first solution with minimal f_1 value. In the second iteration, the bound on f_3 (i.e., ϵ'_3) is set to the f_3 value of the solution found previously and the constrained problem is solved. The figure on the right shows a solution with $f_3 = 0$. The inner loop stops because f_3 is optimal. The algorithm proceeds by tightening the bound on f_2 and solving the bi-objective problem again.

The outline of the 2D method is given in Algorithm 3. The procedure starts with an empty set of efficient solutions E. In lines 1 and 2, we initialize the ϵ-constraints on f_2 and f_3. In each iteration of the outer loop (line 3–12), the bound on f_2 is tightened by at least one

Algorithm 3 .

```
Require: $E = \{\}$
1:  $\epsilon = (1, 0)$                                   ▷ $\epsilon = (\epsilon_2, \epsilon_3)$
2:  $\epsilon' = (\infty, \infty)$                        ▷ $\epsilon' = (\epsilon'_2, \epsilon'_3)$
3:  while $\epsilon'_2 > \epsilon_2$ do
4:      $k = 0$
5:      while $\epsilon'_3 > \epsilon_3$ do
6:          $s = opt(f, \epsilon, \epsilon')$
7:          $E = E \cup \{s\}$
8:          $\epsilon'_3 = f_3(s)$
9:          $k = max\{k, f_2(s)\}$
10:     end while
11:     $\epsilon' = (k, \infty)$    ▷ bound $f_2$ by $k$ (i.e., $\epsilon'_2 = k$) and remove bound
        on $f_3$ (i.e., $\epsilon'_3 = \infty$)
12: end while
13: return $E$
```

Table A.11
Comparison of 2D method and 3D method on data set C_{small}.

Instances	2D method		3D method		Imp_{CPU}(percent)
	Calls *opt*()	CPU(min)	Calls *opt*()	*CPU*(*min*)	
convrp_10_test_1.vrp	17	5.63	19	5.41	3.87
convrp_10_test_2.vrp	20	7.86	24	8.39	−6.72
convrp_10_test_3.vrp	10	19.64	11	21.76	−10.80
convrp_10_test_4.vrp	11	51.83	13	54.32	−4.81
convrp_10_test_5.vrp	20	19.75	22	21.48	−8.76
convrp_12_test_1.vrp	21	224.73	26	260.82	−16.06
convrp_12_test_2.vrp	25	265.35	32	178.11	32.88
convrp_12_test_3.vrp	34	1900.5	45	1266.44	33.36
convrp_12_test_4.vrp	26	953.66	31	679.51	28.75
convrp_12_test_5.vrp	21	50.43	24	48.25	4.33
Average	20.5	349.94	24.7	254.45	5.60

(line 11); variable k memorizes the maximum number of drivers per customer among all solutions for the respective bi-objective problem. For each upper bound on f_2, the inner loop (lines 5–10) provides efficient solutions for the bi-objective problem by repeatedly tightening the bound on f_3 (line 8). The algorithm stops as soon as the bi-objective problem is solved for the strictest driver consistency ($f_2 \leq 1$). The output is a set of solutions that contains at least one solution for each point on the Pareto-front.

In Table A.11, we compare the 2D method to the 3D method on data set C_{small}. For both solution approaches, we report the number of calls to the lexicographic optimizer $opt(f, \epsilon, \epsilon')$ and the total computation time in minutes, $CPU(min)$, for generating the entire Pareto-front. The last column shows the improvement in the computation time of the 3D method over the 2D method.

In the 2D method, the lexicographic optimizer is called only 20.5 times on average (the average number of points on the Pareto-front is 20). The 3D method requires 24.7 calls; nevertheless, the 3D method is 5.6 percent faster on average. The subproblems are smaller in the 3D method and, therefore, each call is executed more quickly. The advantage of the 3D method seems to increase with the number of customers: the 3D method is 5.44 percent slower on average than the 2D method with 10 customers (first five instances) but 16.65 percent faster with 12 customers (last five instances).

References

Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies, 24*(1), 270–287. doi:10.1016/j.trc.2012.03.007.

Coelho, L. C., & Laporte, G. (2013a). A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research, 51*(23–24), 7156–7169. doi:10.1080/00207543.2012.757668.

Coelho, L. C., & Laporte, G. (2013b). The exact solution of several classes of inventory-routing problems. *Computers & Operations Research, 40*(2), 558–565. doi:10.1016/j.cor.2012.08.012.

Czyżak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing – a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis, 7*(1), 34–47.

Doerner, K., Gutjahr, W., Hartl, R., Strauss, C., & Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research, 131*(1–4), 79–99. doi:10.1023/B:ANOR.0000039513.99038.c6.

Ehrgott, M., & Gandibleux, X. (2002). Multiobjective combinatorial optimization theory, methodology, and applications. In M. Ehrgott, & X. Gandibleux (Eds.), *Multiple criteria optimization: State of the art annotated bibliographic surveys*. In *International Series in Operations Research & Management Science: 52* (pp. 369–444). USA: Springer. ISBN 978-1-4020-7128-7. doi:10.1007/0-306-48107-3_8.

Feillet, D., Garaix, T., Lehuédé, F., Péton, O., & Quadri, D. (2014). A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks, 63*(3), 211–224. doi:10.1002/net.21538.

Fischetti, M., Salazar González, J. J., & Toth, P. (1995). Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem. In *Proceedings of the 3rd meeting of the EURO working group on transportation* (pp. 169–173).

Fonseca, C. M. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation, 3*(1), 1–16.

Fonseca, C. M., López-Ibáñez, M., Paquete, L., & Guerreiro, A. P. (2010). Computation of the hypervolume indicator. http://iridia.ulb.ac.be/~manuel/hypervolume. Retrieved: 8 May 2014.

Fonseca, C. M., Paquete, L., & López-Ibáñez, M. (2006). An improved dimension-sweep algorithm for the hypervolume indicator. In *IEEE congress on evolutionary computation, 2006* (pp. 1157–1163). doi:10.1109/CEC.2006.1688440.

Francis, P., Smilowitz, K., & Tzur, M. (2007). Flexibility and complexity in periodic distribution problems. *Naval Research Logistics, 54*(2), 136–150. doi:10.1002/nav.20195.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management, 11*(4), 630–643.

Habenicht, W. (1983). Quad trees, a data structure for discrete vector optimization problems. In P. Hansen (Ed.), *Essays and surveys on multiple criteria decision making*. In *Lecture Notes in Economics and Mathematical Systems: 209* (pp. 136–145). Berlin, Heidelberg: Springer. ISBN 978-3-540-11991-3. doi:10.1007/978-3-642-46473-7_12.

Haimes, Y., Lasdon, L. S., & Wismer, D. A. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics, SMC-1*(3), 296–297. doi:10.1109/TSMC.1971.4308298.

Jozefowiez, N., Laporte, G., & Semet, F. (2012). A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multilabel traveling salesman problem. *INFORMS Journal on Computing, 24*(4), 554–564. doi:10.1287/ijoc.1110.0476.arXiv: http://pubsonline.informs.org/doi/pdf/10.1287/ijoc.1110.0476

Jozefowiez, N., Semet, F., & Talbi, E.-G. (2002). Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, & J.-L. Fernández-Villacañas (Eds.), *Parallel problem solving from nature PPSN VII*. In *Lecture Notes in Computer Science: 2439* (pp. 271–280). Berlin, Heidelberg: Springer. ISBN 978-3-540-44139-7. doi:10.1007/3-540-45712-7_26.

Knowles, J., & Corne, D. (2002). On metrics for comparing nondominated sets. In *Proceedings of the 2002 congress on evolutionary computation: 1* (pp. 711–716). doi:10.1109/CEC.2002.1007013.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety, 91*(9), 992–1007. http://dx.doi.org/10.1016/j.ress.2005.11.018.

Kovacs, A. A., Golden, B. L., Hartl, R. F., & Parragh, S. N. (2014a). The generalized consistent vehicle routing problem. *Transportation Science* Published online. doi:10.1287/trsc.2014.0529.

Kovacs, A. A., Golden, B. L., Hartl, R. F., & Parragh, S. N. (2014b). Vehicle routing problems in which consistency considerations are important: A survey. *Networks, 64*(3), 192–213. doi:10.1002/net.21565.

Kovacs, A. A., Parragh, S. N., & Hartl, R. F. (2014c). A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks, 63*(1), 60–81. doi:10.1002/net.21522.

Laporte, G., & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics, 31*, 147–184.

Laumanns, M., Thiele, L., & Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research, 169*(3), 932–942. http://dx.doi.org/10.1016/j.ejor.2004.08.029.

Mostaghim, S., & Teich, J. (2005). Quad-trees: A data structure for storing Pareto sets in multiobjective evolutionary algorithms with elitism. In A. Abraham, L. Jain, & R. Goldberg (Eds.), *Evolutionary multiobjective optimization*. In *Advanced Information and Knowledge Processing* (pp. 81–104). London: Springer. ISBN 978-1-85233-787-2. doi:10.1007/1-84628-137-7_5.

Naddef, D., & Rinaldi, G. (2001). Branch-and-cut algorithms for the capacitated VRP. In P. Toth, & D. Vigo (Eds.), *The vehicle routing problem* (pp. 1–26). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. ISBN 0-89871-498-2.

Paquete, L., Chiarandini, M., & Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In X. Gandibleux, M. Sevaux, K. Sörensen, & V. T'Kindt (Eds.), *Metaheuristics for multiobjective optimisation*. In *Lecture Notes in Economics and Mathematical Systems: 535* (pp. 177–199). Berlin, Heidelberg: Springer. ISBN 978-3-540-20637-8. doi:10.1007/978-3-642-17144-4_7.

Parragh, S. N., Doerner, K. F., Hartl, R. F., & Gandibleux, X. (2009). A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks, 54*(4), 227–242. doi:10.1002/net.20335.

Parragh, S. N., & Tricoire, F. (2014). Branch-and-bound for bi-objective integer programming. *Technical Report*. Austria: Department of Business Administration, University of Vienna.

Przybylski, A., Gandibleux, X., & Ehrgott, M. (2008). Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research, 185*(2), 509–533. http://dx.doi.org/10.1016/j.ejor.2006.12.054.

Rudolph, G. (1998). On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In *The 1998 IEEE international conference on evolutionary computation proceedings* (pp. 511–516). doi:10.1109/ICEC.1998.700081.

Sarker, R., & Coello Coello, C. A. (2002). Assessment methodologies for multiobjective evolutionary algorithms. In *Evolutionary optimization*. In *International Series in Operations Research & Management Science: 48* (pp. 177–195). USA: Springer. ISBN 978-0-7923-7654-5. doi:10.1007/0-306-48041-7_7.

Schott, J. R. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis. Cambridge, USA: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.

Smilowitz, K., Nowak, M., & Jiang, T. (2013). Workforce management in periodic delivery operations. *Transportation Science, 47*(2), 214–230. doi:10.1287/trsc.1120.0407.

Spliet, R. (2013). *Vehicle routing with uncertain demand*. Erasmus University Rotterdam Ph.D. thesis.

Srinivasan, V., & Thompson, G. L. (1976). Algorithms for minimizing total cost, bottleneck time and bottleneck shipment in transportation problems. *Naval Research Logistics Quarterly, 23*(4), 567–595. doi:10.1002/nav.3800230402.

Sun, M., & Steuer, R. E. (1996). Quad-trees and linear lists for identifying non-dominated criterion vectors. *INFORMS Journal on Computing, 8*(4), 367–375. doi:10.1287/ijoc.8.4.367.

Toth, P., & Vigo, D. (2001). An overview of vehicle routing problems. In P. Toth, & D. Vigo (Eds.), *The vehicle routing problem* (pp. 1–26). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. ISBN 0-89871-498-2.

Tricoire, F. (2012). Multi-directional local search. *Computers & Operations Research, 39*(12), 3089–3101. http://dx.doi.org/10.1016/j.cor.2012.03.010.

Van Veldhuizen, D. A. (1999). *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. Ohio, USA: Air Force Institute of Technology, Wright-Patterson Air Force Base Ph.D. thesis.

Van Veldhuizen, D. A., & Lamont, G. B. (1998). Evolutionary computation and convergence to a Pareto front. In J. R. Koza (Ed.), *Late breaking papers at the genetic programming 1998 conference* (pp. 221–228). Madison, USA: University of Wisconsin.

Van Veldhuizen, D. A., & Lamont, G. B. (1999). Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM symposium on applied computing, New York, NY, USA* (pp. 351–357). ISBN 1-58113-086-4. doi:10.1145/298151.298382.

Van Veldhuizen, D. A., & Lamont, G. (2000). On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 congress on evolutionary computation: 1* (pp. 204–211). doi:10.1109/CEC.2000.870296.

Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., & Gandibleux, X. (2013). Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research, 40*(1), 498–509. http://dx.doi.org/10.1016/j.cor.2012.08.003.

Vise, M., Teghem, J., Pirlot, M., & Ulungu, E. (1998). Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization, 12*(2), 139–155. doi:10.1023/A:1008258310679.

Wong, R. (2008). Vehicle routing for small package delivery and pickup services. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The vehicle routing problem: Latest advances and new challenges*. In *Operations Research/Computer Science Interfaces Series: 43* (pp. 475–485). New York: Springer.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation, 8*(2), 173–195.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms a comparative case study. In A. Eiben, T. Bck, M. Schoenauer, & H.-P. Schwefel (Eds.), *Parallel problem solving from nature PPSN V*. In *Lecture Notes in Computer Science: 1498* (pp. 292–301). Berlin, Heidelberg: Springer. ISBN 978-3-540-65078-2. doi:10.1007/BFb0056872.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., & da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation, 7*(2), 117–132. doi:10.1109/TEVC.2003.810758.