Evaluation of a Multi-Objective EA on Benchmark Instances for Dynamic Routing of a Vehicle

Stephan Meisel
Dept. of Information Systems
University of Münster, Germany
stephan.meisel@uni-muenster.de

Christian Grimme
Dept. of Information Systems
University of Münster, Germany
christian.grimme@uni-muenster.de

Jakob Bossek
Dept. of Information Systems
University of Münster, Germany
bossek@uni-muenster.de

Martin Wölck
Dept. of Information Systems
University of Münster, Germany
martin.woelck@uni-muenster.de

Günter Rudolph
Dept. of Computer Science
TU Dortmund Univ., Germany
guenter.rudolph@tu-dortmund.de

Heike Trautmann
Dept. of Information Systems
University of Münster, Germany
trautmann@uni-muenster.de

ABSTRACT

We evaluate the performance of a multi-objective evolutionary algorithm on a class of dynamic routing problems with a single vehicle. In particular we focus on relating algorithmic performance to the most prominent characteristics of problem instances. The routing problem considers two types of customers: mandatory customers must be visited whereas optional customers do not necessarily have to be visited. Moreover, mandatory customers are known prior to the start of the tour whereas optional customers request for service at later points in time with the vehicle already being on its way. The multi-objective optimization problem then results as maximizing the number of visited customers while simultaneously minimizing total travel time. As an a-posteriori evaluation tool, the evolutionary algorithm aims at approximating the related Pareto set for specifically designed benchmarking instances differing in terms of number of customers, geographical layout, fraction of mandatory customers, and request times of optional customers. Conceptional and experimental comparisons to online heuristic procedures are provided.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization

Keywords

Transportation; Metaheuristics; Online algorithms; Multi-objective optimization; Combinatorial optimization

GECCO '15, July 11 - 15, 2015, Madrid, Spain

ISBN 978-1-4503-3472-3/15/07…$15.00

DOI: http://dx.doi.org/10.1145/2739480.2754705

1. INTRODUCTION

Many real-world vehicle routing problems are of dynamic nature [20]. As one of the most prominent examples of dynamic routing of a vehicle, we consider the case where customer requests are not known in advance but are revealed in the course of time while the vehicle already is on its way. In practice both decisions about accepting dynamic customer requests for service and decisions about routing the vehicle are made in real-time. To this end decision makers typically rely on decision rules that they apply each time a new request occurs (see, e.g., [16]). However, real-time vehicle routing necessarily suffers from the fact that decisions are made without information about all customer requests being known. As a consequence, optimal decisions will rarely occur, and the problem of a-priori selection of the best decision rule is hard.

Yet one way of gradually gaining insights into the performance of decision rules consists of assessing applied rules in retrospect. In this paper we provide two fundamental means that enable such an assessment, provided that the decision maker pursues the two competing objectives of maximization of served requests and of minimization of travel distance.

On the one hand, we apply a multi-objective evolutionary algorithm for approximating the Pareto-front of a-posteriori solutions. The set of Pareto efficient a-posteriori solutions of a given problem instance may serve as a point of reference for solutions resulting from application of any real-time decision rule. On the other hand, we take into account the fact that algorithmic performance often strongly depends on the specific problem instance at hand. Therefore we carefully design and generate a set of benchmarking problem instances that systematically cover realizations of the most relevant problem characteristics.

The two main contributions of the present work are the new benchmarking instances that we propose as well as our analysis of the behavior of the evolutionary algorithm with respect to the different instances of the benchmarking set.

The following Section 2 provides a summary of related work, before Section 3 mathematically defines the class of routing problems we consider. Section 4 illustrates the approach we propose for generating a set of benchmarking in-

stances of the class of routing problems. Section 5 introduces the multi-objective evolutionary algorithm. Finally, Section 6 provides computational results that illustrate the performance of the evolutionary algorithm with respect to the set of benchmarking instances. Section 7 concludes the paper.

2. RELATED WORK

Most of the existing works in the literature only consider the single objective of maximizing the number of customers visited within a given amount of time. The single vehicle problem with optional customers is frequently addressed as "orienteering problem" (e.g. [10]). A survey on different kinds of orienteering problems is provided in [21]. Although it has been recognized that orienteering problems should be formulated as bi-objective optimization problems [15], it can be concluded [7] that subsequent work (e.g. [5]) avoided dealing with true bi-objective formulations and aimed at minimization of the sum of total distance traveled and penalties for unvisited customers.

Publications that deal with a bi-objective formulation of the orienteering problem are rare. In [2] the ϵ-constraint approach is used to approximate the Pareto frontier by solving a series of single-objective problems where one of the two objectives is transformed into a constraint with varying bound ϵ. The solution of a series of single-objective problems is also necessary for the approximation schemes proposed in [8] which are proven to provide a Pareto-ϵ-approximation of the efficient set of solutions.

In contrast to the two preceding works the approach proposed in [13] explicitly solves the bi-objective formulation of the orienteering problem. Their method determines a high quality approximation of the efficient Pareto-frontier by two main steps. First an NSGA-II based multi-objective evolutionary algorithm is applied for generating a set of initial solutions. These solutions serve as starting points for an ejection chain process with two sets of neighborhood moves. A computational comparison with an iterated ϵ-constraint implementation of a state-of-the-art meta-heuristic for the single-objective orienteering problem shows that the method has advantages as the problem size increases.

To our knowledge, the approach of [13] so far is the only one that solves the bi-objective version of an orienteering problem without transformations to (a series of) single-criterion problems. Therefore it may be considered as most related to our approach, however time windows are not considered. Our problem scenario differs from traditional orienteering problems with time windows [14] as follows: the lower bound of a customer's time window does not represent the earliest point in time at which the vehicle may arrive at this customer's location; instead, the lower bound of a customer's time window represents the earliest point in time at which the vehicle may leave the preceding customer location. As a consequence, a comparison to previous work is hardly possible and new benchmark problems have to be developed. Thus, the few existing benchmark problems for orienteering problems with time windows [21] will not be included here as those would have to be adapted for our purposes and moreover lack a systematic generation procedure.

3. PROBLEM CLASS

Each problem instance of the benchmarking set proposed in the following Section 4 belongs to the class of vehicle routing problems defined in this section. An instance comprises one single vehicle and a set $C = \{1, 2, \ldots, N\}$ of locations, where location $i = 1$ is the start depot and location $i = N$ is the end depot of the vehicle. Travel distances d_{ij} between any pair (i, j) of locations are both known and deterministic, and one unit of travel distance corresponds to one unit of time. At time $t = 0$, the vehicle is located at $i = 1$ and at the end of the tour the vehicle must reach $i = N$.

The set $C \setminus \{1, N\}$ of customers consists of both the subset C^m of mandatory customers and the subset C^o of optional customers, where $C^m \cap C^o = \emptyset$. Mandatory customers request for service before time $t = 0$ and must necessarily be visited once by the vehicle. Each optional customer i issues one service request at a particular point in time $t_i > 0$ and must immediately be either accepted for service or rejected. Note that rejecting a customer typically saves travel time, and that in practice such a customer will become a mandatory customer of the following day.

Maximization of customer satisfaction clearly suggests accepting all optional customers. On the other hand, however, satisfying all requests typically implies high costs in terms of a long overall travel distance. In order to make an appropriate trade-off between maximization of customer satisfaction and minimization of travel costs both of these objectives are considered simultaneously.

The resulting bicriteria problem to be solved for determination of ideal a-posteriori solutions may be described by means of the following Equations 1a-1h:

$$\min \Big((|C| - \sum_{i \in C} x_i^c), (\sum_{(i,j) \in E} d_{ij} x_{ij}^r) \Big) \tag{1a}$$

$$s.t. \quad x_i^c = 1 \qquad \forall i \in C \setminus \{C^o\}, \tag{1b}$$

$$\sum_{(i,j) \in \phi^+(i)} x_{ij}^r = x_i^c \qquad \forall i \in C \setminus \{N\}, \tag{1c}$$

$$\sum_{(k,i) \in \phi^-(i)} x_{ki}^r = x_i^c \qquad \forall i \in C \setminus \{1\}, \tag{1d}$$

$$\sum_{(k,i) \in \phi^-(i)} (y_{ki} + d_{ki} x_{ki}^r) \leq \sum_{(i,j) \in \phi^+(i)} y_{ij} \qquad \forall i \in C \setminus \{1\}, \tag{1e}$$

$$\sum_{(i,j) \in \phi^+(i)} (y_{ij} - t_i x_{ij}^r) \geq \sum_{(k,i) \in \phi^-(i)} d_{ki} x_{ki}^r \qquad \forall i \in C, \tag{1f}$$

$$y_{ij} \in \mathbb{N}_0 \qquad \forall (i,j) \in E, \tag{1g}$$

$$x_{ij}^r, x_i^c \in \{0, 1\} \qquad \forall (i,j) \in E, \forall i \in C \tag{1h}$$

Three types of decisions are involved. For each location $i \in C$, decision x_i^c indicates whether or not the request of customer i has been accepted or not. Decisions x_{ij}^r determine whether or not the road link connecting locations i and j is part of the vehicles ideal route. Note that we assume that all locations of C are connected with each other, and that distances between any pair i, j of locations are symmetric. The set of all road links connecting any pair (i, j) is referred to as E. Set $\phi^+(i)$ (set $\phi^-(i)$) contains all road links that are going out of (or into) location i. Each of the variables y_{ij} equals zero if link (i, j) is not part of the solution, i.e., $y_{ij} = 0$ if $x_{ij}^r = 0$. In case of $x_{ij}^r = 1$, variable y_{ij} denotes the point in time at which the vehicle arrives at location i.

Our goal is minimization of both the number of unserved customers and the total distance traveled by the vehicle (Equation 1a). Equation 1b ensures that both depots as well as all mandatory customers $i \in C^m$ will be visited be-

fore returning to N. Equations 1c and 1d ensure that the vehicle leaves $i = 0$, reaches $i = N$ and visits each customer with an accepted request exactly once.

Equation 1e makes sure that the vehicle never arrives at location i before it's predecessor k (according to the route) has been visited and the distance d_{ki} has been traveled. As a consequence we have $y_{ki} \leq y_{ij}$ with j representing the successor of i. Additionally, Equation 1f ensures that the vehicle is never allowed to leave the preceding location k for moving on to i before i has actually issued a service request. Note that the claim of Equation 1f reflects the conditions present in a real-time setting.

Against the background of the vehicle routing literature, the proposed model represents a bicriteria orienteering problem. In the following Section we provide an approach to generating a set of benchmarking problem instances of the class of vehicle routing problems defined by Equations 1a-1h.

4. BENCHMARKING SET GENERATION

Our experimental study is based on a specifically designed set of 600 problem instances. The generated instances differ in the number of customers $n \in \{50, 100, 200, 400\}$ (including both start and end depot), the number of clusters $n_c \in \{1, 2, 3, 5, 10\}$ and the fraction of optional customers $f \in \{0.25, 0.5, 0.75\}$, representing a variety of typical scenarios for the VRP.

We started by generating a single geographical layout for each combination of the number of customers and the number of clusters. Single cluster instances consist of randomly placed customers in the euclidean plane, starting with a bounding box of $[0, 100]^2$ for $n = 50$ and doubling the bounding box sides on doubling the number of customers. The generation of instances with two or more clusters follows a more sophisticated procedure: 1) We generate a latin hypercube design (LHS) of size n_c in two dimensions to get reasonable cluster centers. The space-filling property of LHS designs ensures, that the cluster centers are highly distributed in the available space and avoids overlapping of clusters. 2) We then sample random points from a multivariate Gaussian distribution using the corresponding cluster center coordinates as the mean vector and the euclidean distance to the nearest cluster center as the variance.

Figure 1: Greedy point matching (left) and optimal point matching (right). The three longest distances between assigned points are highlighted with bold arrows.

Moreover, we adopt the concept of *morphing* instances first introduced by [17] and later improved in [18], using a convex combination of point coordinates of two instances to generate instances in between the original ones [18]. However, we further improved the underlying point matching algorithm, which determines points of both instances, which should be combined. Recognizing that point matching on instances of the same size is equivalent to an assignment problem in bipartite graphs, we substituted the greedy approach used in [18] with a procedure, which makes use of a linear programming solver and finds the optimal point matching regarding to euclidean distance. The superiority of our point matching approach is depicted in Figure 1 by way of example.

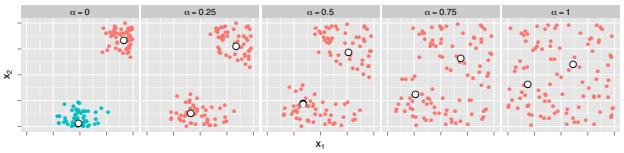

Figure 2: Example: Morphing of a clustered instance with two clusters and a random instance of equal size with two depots in each case for different morphing coefficients α.

An example of our morphing approach is visualized in Figure 2, morphing an instance with two clusters with a random instance. We observe a smooth and natural transition of one instance into another for increasing morphing coefficient α ($\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$), with $\alpha = 0$ and $\alpha = 1$ representing the original instances.

We used the enhanced morphing algorithm – handling depots and customers separately – to generate instances in between the random instance and the instance with five clusters for each cluster size and corresponding instance size, resulting in 15 further geographical layouts. An overwiew of all generated geographies is given in Figure 3.

As a next step, we assigned request times to a fraction of f ($f \in \{0.25, 0.5, 0.75\}$) customers randomly for each of our 40 geographies (5 random plus 20 clustered plus 15 morphed), increasing the number of optional customers stepwise. The request times were sampled from an Exponential distribution with the rate parameter set accordingly. We started with a maximal arrival limit of 400 time units for instances with 50 customers and doubled the arrival limit for instances next in size each time. This process was repeated five times with different seeds for the random number generator to get different realizations of the underlying Poisson process.

We wrapped up the implementation of instance generation and enhanced morphing in the R package *netgen* [3].

5. MULTIOBJECTIVE APPROACH

In order to experimentally evaluate the considered routing problem instances regarding their complexity for multiobjective problem solving approaches as well as for evaluating the principal capabilities of multi-objective evolutionary algorithms to tackle this problem, we apply NSGA-II [4] as widely accepted algorithmic framework for our study. Starting with an initial population of size μ, the algorithm performs an evolutionary loop consisting of generating μ offspring solutions by variation, ranking the union set of parents and offspring regarding non-domination, and finally creating a new parental population by adding better ranked solutions until it contains μ or more individuals. If the new

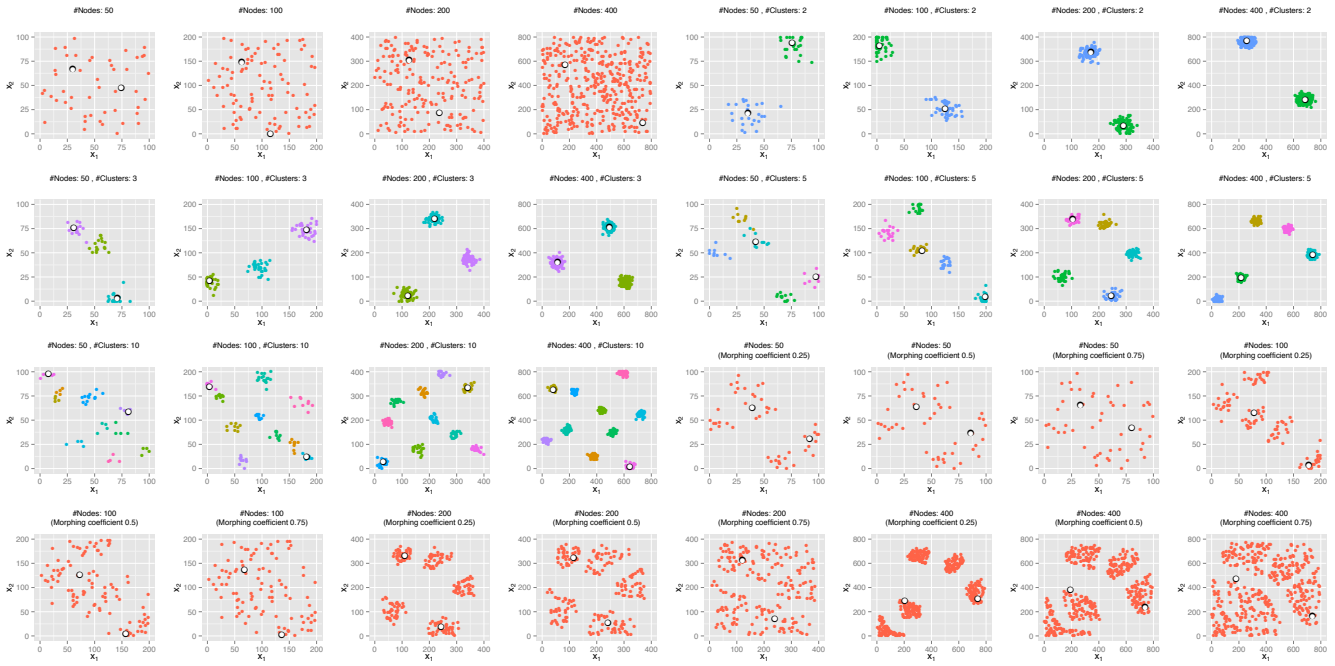

Figure 3: Overwiew of generated geographies.

population size is larger than μ, solutions with worst rank and smallest crowding distance [4] are removed. In the following we briefly explain the instantiation of the algorithmic framework in terms of solution encoding, variation operators, and integration of local search.

A solution to the vehicle routing problem as defined in Section 3 has several components: a subset of customers from C must be selected, while simultaneously an optimal tour for visiting those customers has to be determined. Thus, the encoding of a solution candidate consists of a permutation string and a binary string both of length $N-2$. The permutation string encodes the sequence of visits to all customers, while the binary string $B = (b_2, \ldots, b_{N-1}) \in \{0,1\}^{N-2}$ indicates for each customer $i \in C \setminus \{1, N\}$ whether it is visited ($b_i = 1$) or not ($b_i = 0$). This encoding disregards the start ($i = 1$) and end ($i = N$) depots, which are always part of the tour. We additionally introduce a probability encoding for each position in the binary string as strategy parameters to distinguish between the subsets C^m and C^o of C. Herein, we assign a flip probability $p_i \in [0,1]$ for each element in the binary string. If $p_i = 0$, the value of b_i never changes throughout the variation process. This way we denote customers from C^m as obligatory. A probability $0 < p_i \leq 1$ indicates a certain variability of value b_i, denoting dynamic customers from C^o.

With this encoding scheme a tour can be constructed from the permutation string by only considering the active customers. Fig. 4 shows two exemplary solutions for a problem with six customers (start and end depots are excluded from the encoding) and different activation states. In the left scenario only customers 3 and 4 are active resulting in a tour that only visits those customers. In the right scenario customers 3, 4, and 5 are active. The tours result from the permutations, with sequences $4 \to 3$ (left) and $5 \to 3 \to 4$ (right).

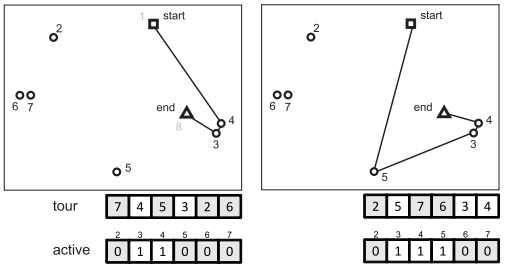

Figure 4: Exemplary encoding of two (Pareto-optimal) solutions for a routing problem instance with six customers, two of those active in the left and three active in the right case.

5.1 Variation operators

For the proposed encoding consisting of two chromosomes we combined common variation operators to specialized mutation and crossover building blocks.

Swap/Flip-Mutation: The permutation chromosome is varied by repeated pairwise swapping of randomly selected entries. This is controlled by parameter $\sigma_p \in \mathbb{N}$ which determines the maximum number of swaps performed. The actual number of swaps is drawn uniformly random from $\{0, \ldots, \sigma_p\} \subset \mathbb{N}$. Simultaneously, the binary string undergoes a flip mutation, where each position $b_i \in \{0,1\}$ in the string is flipped with probability $p_i \in [0,1]$ taken from the strategy parameters.

PMX/One-Point-Crossover: The crossover of two solutions is performed with Partially Mapped Crossover (PMX) for the permutation string and with One-Point-Crossover for the binary string. In PMX offspring is generated by interchanging a sub-sequence of the parental solutions and repair-

ing violations to the permutation [19]. One-Point-Crossover recombines offspring by splitting up the binary string at a random position and alternately connecting the parts.

Note, during mutation no repair of infeasible solutions is needed. While swap mutation only changes the order of visits to customers, i.e. the tour, flip mutation influences tour length by activating and deactivating customers. During evaluation, if an active customer's order is not yet available due to a later request time, the vehicle remains waiting at the previous customer and the waiting time is added to tour length. Thus, inefficient tours containing long waiting times are punished during selection and removed on the long run.

5.2 Integration of Local Search

In addition to variation operators we integrate local search—especially for optimizing tour length—into NSGA-II. Instead of starting with a random initial solution set we applied 2-OPT local search to the initial population. Additionally, 2-OPT is repeatedly activated after variation. 2-OPT local search optimizes tour length by removing edge crossings from a tour. It is based on the insight, that in euclidean graphs planar tours are more cost-efficient than non-planar. In our implementation we apply the steepest descent variant of 2-OPT, which performs the most effective interchange of edges per step.

6. EXPERIMENTS

In this section we briefly describe the evaluation settings before showing and discussing experimental results.

6.1 Experimental Setup

We applied NSGA-II with the described variation operators and 2-OPT local search. However, the parameter settings for population size, variation operators, and number of function evaluations depends on the considered instance size (i.e. number of customers contained). As problem instance sizes systematically double (see Section 4), we applied the same approach to a basic set of good parameters.[1] In Table 1 we show generally formulated parameter settings and denote the instance sizes with $K = 0$ when comprising 50 customers up to $K = 3$ when comprising 400 customers. Experiments were repeated 10 times with independent random initialization. The chosen amount of basic function evaluations resembles approximately 120s running time of the algorithm implementation[2] on a Intel i7-3667U CPU machine at 2.5GHz with 8GB RAM. This makes the running time comparable with the maximum time allowed for the applied real-time decision rules executed in CPLEX.

Table 1: General algorithm parameters' scheme.

Parameter	Setting
Population size	$\mu = 100 \cdot 2^K$
Swap number	$\sigma_p = 2 \cdot 2^K$
Flip probability	$\frac{1}{N-2}$
Function evaluations	$\nu = 6,500,000 \cdot 2^K$
Initial local search	$0.1 \cdot \nu$
Internal local search	$0.01 \cdot \nu$, only 65 times per run

[1]This setting yielded best behavior for several considered test instances of comprising 50 customers.
[2]Implemented in the jMetal framework [6].

For comparison we consider two real-time decision rules, both of which are similar to rules that have been discussed in the literature (see, e.g., [16]). Each rule makes decisions about acceptance of new customer requests as well as about revision of the routing plan at each point in time a new customer request occurs. An acceptance decision is made by maximizing the number of newly accepted customers subject to a predefined overall time horizon. Decisions about plan revisions are then made by solving an open travelling salesman problem with all customers that have already been accepted but not visited yet. Note that this approach essentially pursues the single objective of maximizing the number of served customers at each point in time, with the second objective being represented in terms of a constraint that limits the total amount of travel time available.

The fact that the overall time horizon is predefined allows for the decision rules to strategically include waiting times into the route, where the idea is that allocating a certain amount of waiting time at specific customer locations will eventually reduce the overall distance travelled. Decisions about waiting time allocation are made if the vehicle has just arrived at a customer location, or if the vehicle currently waits and a new request appears. In particular, we consider the following two decision rules:

Distributed Waiting (DW): Waiting decisions are made such that the total amount of currently available waiting time is distributed equally among all customer locations of the current planned route. The amount of time to be spent at customer locations is recalculated as soon as a new customer request is accepted.

Drive First (DF): The vehicle only waits at its current customer location if both waiting time is available and the planned route only contains the end depot.

We apply each rule to each of the problem instances featuring either 50 or 100 locations. In order to approximate the set of Pareto efficient solutions with a given pair of instance and decision rule, we vary the predefined total amount of travel time and solve the instance several times. For each instance we start with a time horizon that merely allows for serving all customers C^m, i.e., that leads to rejection of all optional customers. We then increase the time horizon with a step size of ten time units until the time horizon is sufficiently long for also serving all customers C^o. Both the orienteering problem and the travelling salesman problem occuring at each decision time are solved optimally by a branch-and-cut procedure.

6.2 Indicators

For evaluation of experimental results we applied some standard as well as some specifically designed indicators, which are briefly described in the following paragraphs.

Hypervolume.

For determining the algorithms' performance on the evaluated instances, we applied the hypervolume indicator [22] which measures the normalized hypervolume enclosed by a non-dominated solution front and a given reference point. The reference point is determined as nadir point for each problem instance considering all NSGA-II and CPLEX results. Additionally, the utopia point is determined from the same set of solutions fronts. Both utopia and nadir points are used for normalizing the hypervolume.

Empirical Attainment Function.

The Empirical Attainment Function [9] allows the statistical interpretation of stochastic multi-objective algorithms by mapping the results of repeated algorithm runs to so called k%-attainment surfaces. Each k%-surface partitions the objective space in two areas: the area which is dominated by k% solution sets over all runs and the area, which is not dominated. Consequently, the best solutions obtained over all runs form the 0%-surface, while the worst solutions from all runs form the 100%-surface. In our evaluation we always also provide the 50%-surface as median.

Coverage.

For the defined vehicle routing problem the number of elements in the Pareto-front (PF) is determined by the number of optional customers: $|PF| = |C^o| + 1$. Given a non-dominated solution set $\mathcal{X}$ as result of applying algorithm A, this value can be used to define the coverage of the solution front as ratio

$$cov_A = \frac{|\mathcal{X}|}{|C^o| + 1}.$$

Obviously, this indicator does not provide information on the convergence behavior of algorithm A but indicates to what extent tour solutions for all (optional) customer configurations are found.

Determinism.

In order to make some statement on the algorithmic challenge posed by a problem instance topology, we define a measure which considers problem characteristics as well as solution quality: with T_A denoting the longest tour (travel time) in the non-dominated solution set after application of algorithm A,

$$Det_A = 1 - \frac{\max\{t_i |\ i \in C^o\}}{T_A}$$

is called the *determinism* of the instance under A. This measure relates the latest service request over all optional customers (instance property) to the longest tour in the non-dominated solution set. Clearly, the longest tour (largest travel time) will always exceed the latest service request. Thus determinism ranges in $0 < Det_A \leq 1$. In this range, determinism can be seen as a kind of approximation factor for the solution element with only mandatory customers. The lower Det_A is, the closer to the determined tour's end the last service request is dispatched.

A very specific variant of this indicator can be defined as purely instance related: instead of considering the longest tour of an algorithm solution set, we compute the optimal tour length T_{OPT} for all (mandatory and optional) customers in C without respecting service requests. We term this indicator

$$Det_{OPT} = 1 - \frac{\max\{t_i |\ i \in C^o\}}{T_{OPT}}.$$

This indicator specifically shows for $0 < Det_{OPT} \leq 1$ that the optimal travel time T_{OPT} exceeds the last service request. For $Det_{OPT} \leq 0$ the last service request appears after the lower bound for travel time.

In order to compute T_{OPT} as shortest paths from start to end depots, the Concorde solver [1] for the (symmetric) Travelling Salesperson Problem (STSP) was used. As mentioned, we assumed, that there are no dynamic requests and thus all the customers are mandatory. Since the problem of finding an optimal hamiltonian path differs slightly from the TSP, we applied two transformations to the distance matrix. First, we replaced the two depots with a dummy depot. The outgoing edges of this dummy node contain the distances of the start depot and the edge weights of the ingoing edges correspond to the edge weights of the end depot. However, the distance matrix is not symmetric anymore and hence the generated TSP problem is asymmetric (ATSP). Since Concorde is specially designed for the symmetric TSP we used a reformulation af the ATSP as a STSP by doubling the number of customers [12], i.e., adding another dummy customer for each customer. The resulting STSP was solved to optimality with Concorde. The tours for the ATSP were obtained by removing the dummy customers from the optimal STSP tour and and finally the optimal hamiltonian path was extracted by "cutting" the tour at the dummy depot [11]. Figure 5 shows some exemplary paths for some of our benchmark instances.

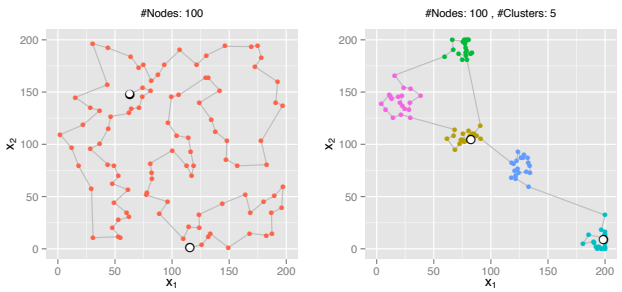

Figure 5: Examples for optimal hamiltonian pathes from start to end depot on different geographies of our benchmarking set.

6.3 Results

The indicators introduced in Section 6.2 are used to analyze problem characteristics in relation to EA behavior. Indicator values can be compared over all instances as they were designed in a normalized way such that the scaling is independent from the individual instance characteristics. At first, the relationship of both determinism indicators is investigated in Figure 6. A clear structure is visible when labeling the points regarding the underlying instance size. Of course the Det_{EA} indicator always exceeds Det_{OPT} as

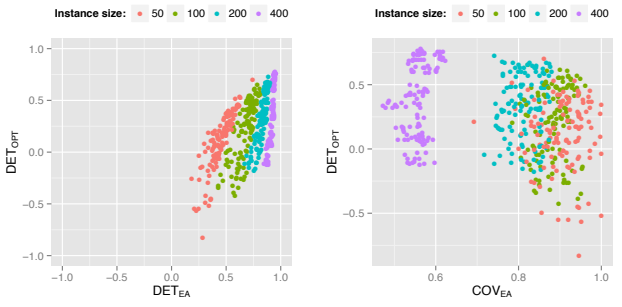

Figure 6: Det_{EA} (left) resp. cov_{EA}(right) vs. Det_{OPT} for all considered instances. Color labels indicate the instance size.

the maximum tour length of the evolutionary algorithm is always longer than the optimal solution for all nodes disregarding the request times. The extent of the difference increases with the instance size as larger problems are more difficult to solve for the EA. The same effect is visible in case Det_{OPT} is related to cov_{EA}(see right part of 6). Clearly, for higher instance sizes the fraction of approximated Pareto optimal solutions on the front decreases. Contrarily, Det_{OPT} is hardly influenced by the instance size which reflects that our request time settings were chosen in a reasonable way.

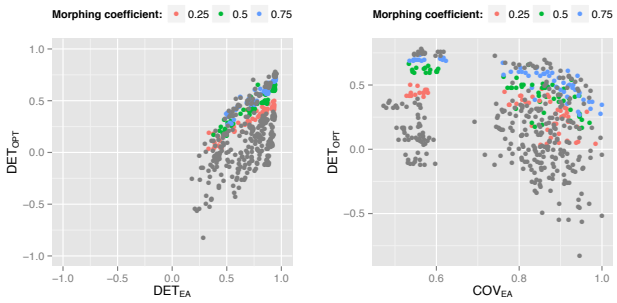

Figure 7: Det_{EA} (left) resp. cov_{EA}(right) vs. Det_{OPT} for all considered instances. Color labels indicate the morphing coefficient.

The topology of the instances influences the problem and EA characteristics. More specifically, Det_{OPT} simultaneously increases with the morphing coefficient (Figure 7, left). Thus, the optimal open TSP solution is higher for more randomly structured instances. The number of clusters has an impact on Det_{OPT} (Figure 8, left) such that random instances lead to higher open TSP solutions. Smallest respective values result from the smallest number of clusters which is intuitive as short paths are present within the clusters. In Figure 7, Det_{OPT} shows a smooth transition between clustered and random instances. Contrarily, cov_{EA} is not influenced by the morphing coefficient (Figure 7, right) while Det_{EA} has a slight tendency that the problem becomes less deterministic with decreasing morphing coefficient, i.e., for stronger clustered instances.

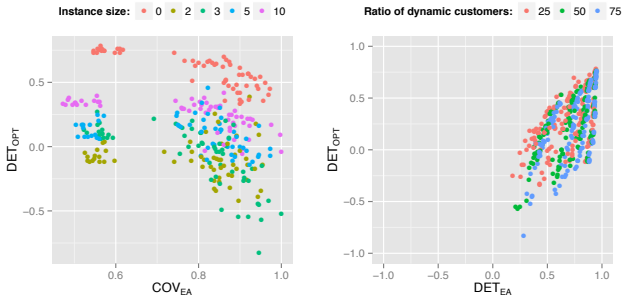

Figure 8: cov_{EA}(left) resp. Det_{EA} (right) vs. Det_{OPT}. Color labels indicate the number of clusters resp. the fraction of dynamic customers. In the left figure the morphed instances are neglected as the number of clusters is not assigned here.

Of course, the fraction of dynamic customers is only of interest for investigating the EA behaviour as the open TSP solution does not take into account the request times. From Figure 8 (right) we can conclude that within a specific instance size (see Fig. 6 for relating the labels) a higher fraction of dynamic customers increases the EA determinism. Due to the increasing tour uncertainty coming along with a higher fraction of dynamic customers, tour lengths tend to be higher than for a larger number of mandatory customers.

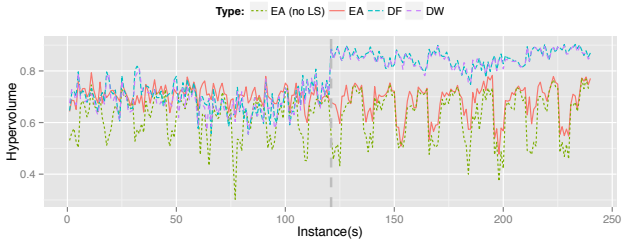

Figure 9: Dominated Hypervolume of the EA (with and w/o local search) and DF / DW solutions for instances of size 50 (left part) and 100 (right part). For the evolutionary algorithm, the average of all ten runs is shown.

In order to assess the EA performance on individual instances the Dominated Hypervolume indicator (HV) is used. Though the EA performance increases with higher HV values, the indicator level is only roughly comparable across instances due to the nature of the underlying normalization (see Section 6.2) but has to be evaluated in comparison to other approaches. Here, we relate it to the single-objective real-time (SORT) decision rules briefly discussed in Section 6.1. As the computational requirements of the latter approaches heavily increases with the instance size as well – due to a desired fine discretization of the Pareto front – with the number of optional customers, the former solutions are only available for instance sizes 50 and 100. Figure 9 shows the corresponding HV values. For the smallest instance size of 50, the EA and the SORT rules are comparable, while the EA often even provides better results than SORT. For the subsequent instance size of 100, the SORT solutions outperform the EA which gives the impression that performance difference increases with instance size. However, the SORT rules have to be conducted for each desired point on the Pareto front separately which eventually leads to enormous computation times and are thus not reasonably applicable for bigger instance sizes. Contrarily, the EA approximates the complete Pareto front in a single run and might perform better with higher budget of function evaluations and tuned parameter settings.

This potential can also be observed Figure 9 which shows the benefit of applying local search in the EA compared to applying a pure EA without local search mechanisms. Additionally, Figure 10 exemplarily shows the difference in approximation quality of EA and SORT solutions. In instances of size 50, SORT solutions range in the attainment area of the EA, while in 100 customer instances the optimization of (long) tours with many customers often causes the major difference in HV performance. Specifically designed and adjusted operators and superior local search mechanisms could help to overcome this issue. However, the focus is not on algorithm tuning at this stage but rather understanding EA behavior related to instance characteristics.

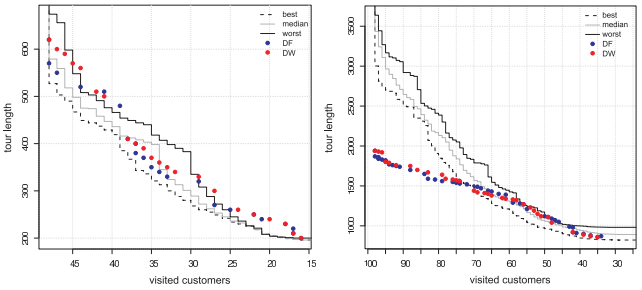

Figure 10: Exemplary attainment plots of EA results and DF / DW solutions for a 50 customers 2 cluster instance with 75% optional customers (left) and a 100 customers random instance with 75 % optional customers (right).

7. CONCLUSIONS

In this paper, we addressed a class of vehicle routing problems with two competing objectives. In order to establish the ground for following analysis, we carefully designed and created a set of benchmark instances for this problem class. Characteristics of these instances are experimentally investigated by applying a well-established multi-objective evolutionary approach. First results show different aspects of the benchmark problems' topologies like number of customers, dynamism, and number of clusters to be gradually influential onto algorithm performance. Furthermore, a closer look into the comparison of EA results and ILP-based methods highlights the potential of EAs to provide good solution sets in a single run for small problem instances. The comparison also reveals the need for application of alternative (evolutionary) multi-objective approaches and for integration of more sophisticated local search mechanisms. Thus, this paper may be considered as fundamental work proposing a valid set of benchmarking instances, and allowing for insights into algorithmic challenges posed by the considered problem class. Starting from this, more rigorous analysis of the benchmarking set as well as algorithm tuning and development are promising future directions of research.

8. REFERENCES

[1] D. Applegate, W. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS J. on Computing*, 14(2):132–143, 2002.

[2] J.-F. Berube, M. Gendreau, and J.-Y. Potvin. An exact ε-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *Eur. J. Oper. Res.*, 194(1):39–50, 2009.

[3] J. Bossek. *netgen: Network Generator for Combinatorial Graph Problems*, 2015. R package v1.0.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[5] M. Dell'Amico, F. Maffioli, and P. Vaerbrand. On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research*, 2(3):297–308, 1995.

[6] J. Durillo and A. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[7] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005.

[8] C. Filippi and E. Stevanato. Approximation schemes for bi-objective combinatorial optimization and their application to the tsp with profits. *Computers & Operations Research*, 40(10):2418–2428, 2013.

[9] C. Fonseca and P. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In H.-M. Voigt et al., editors, *Parallel Problem Solving from Nature, PPSN IV*, pages 584–593. Springer Berlin Heidelberg, 1996.

[10] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Nav. Res. Log.*, 34(3):307–318, 1987.

[11] M. Hahsler and K. Hornik. Tsp – Infrastructure for the traveling salesperson problem. *Journal of Statistical Software*, 23(2):1–21, December 2007.

[12] R. Jonker and T. Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Oper. Res. Lett.*, 2(4):161–163, November 1983.

[13] N. Jozefowiez, F. Glover, and M. Laguna. Multi-objective meta-heuristics for the traveling salesman problem with profits. *Journal of Math. Modelling and Algorithms*, 7(2):177–195, 2008.

[14] M. G. Kantor and M. B. Rosenwein. The orienteering problem with time windows. *The Journal of the Operational Research Society*, 43(6):629–635, 1992.

[15] C. P. Keller and M. Goodchild. The multiobjective vending problem: A generalization of the traveling salesman problem. *Environ. Planning B: Planning Design*, 15(2):447–460, 1988.

[16] S. Meisel. *Anticipatory Optimization for Dynamic Decision Making*, Springer New York, 2011.

[17] O. Mersmann, B. Bischl, J. Bossek, H. Trautmann, M. Wagner, and F. Neumann. Local search and the traveling salesman problem: A feature-based characterization of problem hardness. In Y. Hamadi and M. Schoenauer, editors, *Proc. LION 6*, pages 115–129. Springer, 2012.

[18] O. Mersmann, B. Bischl, H. Trautmann, M. Wagner, and F. Neumann. A novel feature-based approach to characterize algorithm performance for the traveling salesman problem. *Annals of Mathematics and Artificial Intelligence*, 69:151–182, 2012.

[19] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, 3rd ed., 1999.

[20] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.*, 225(1):1–11, 2013.

[21] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *Eur. J. Oper. Res.*, 209(1):1–10, 2011.

[22] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.