Local Search Effects in Bi-Objective Orienteering

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

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

Stephan Meisel
Dept. of Information Systems
University of Münster, Germany
stephan.meisel@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 analyze the effects of including local search techniques into a multi-objective evolutionary algorithm for solving a bi-objective orienteering problem with a single vehicle while the two conflicting objectives are minimization of travel time and maximization of the number of visited customer locations. Experiments are based on a large set of specifically designed problem instances with different characteristics and it is shown that local search techniques focusing on one of the objectives only, improve the performance of the evolutionary algorithm in terms of both objectives. The analysis also shows that local search techniques are capable of sending locally optimal solutions to foremost fronts of the multi-objective optimization process, and that these solutions then become the leading factors of the evolutionary process.

CCS CONCEPTS

• **Applied computing → Multi-criterion optimization and decision-making**; **Transportation**;

KEYWORDS

Transportation, Metaheuristics, Multi-objective optimization, Combinatorial optimization, Orienteering

ACM Reference Format:
Jakob Bossek, Christian Grimme, Stephan Meisel, Günter Rudolph, and Heike Trautmann. 2018. Local Search Effects in Bi-Objective Orienteering. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205548

1 INTRODUCTION

Local search techniques have been successfully applied to a large variety of vehicle routing problems (VRP; see, e.g., [6], [9]). However, most of these works focus on classic, single-objective problem formulations which mostly are not sufficient for matching the requirements of real-world applications. Transportation applications frequently require consideration of additional properties in terms of multiple objectives, optional customer locations, and time windows.

In this work, we analyze the effect of local search techniques for a class of routing problems that includes all of these properties. In particular, we analyze the effect of integrating local search techniques into a multi-objective evolutionary algorithm for routing a vehicle, where the objectives are minimization of travel time and maximization of the number of visited locations. The set of locations is divided into the two disjunct subsets of mandatory and optional locations. We assume that each location corresponds to one customer. Each mandatory customer can be visited at an arbitrary point in time within the planning horizon, whereas each optional customer cannot be visited earlier than a certain point in time within the planning horizon.

Single vehicle routing problems with optional customers are often referred to as "orienteering problems" in the literature, see, e.g., [7], or [22] for a survey of problem variants. A bi-objective model of an orienteering problem (without time windows) was first formulated and solved heuristically in [13]. Since then the need for multi-objective handling of orienteering problems has been emphasized repeatedly (see, e.g., [4]). However, works that treat proper multi-objective formulations of orienteering problems are still rare.

Both [1] and [5] approximate the Pareto-frontier of a bi-objective orienteering problem without time windows by solving a series of single-objective optimization problems. The approach of [12] solves a similar problem variant without transformation to single-objective formulations. Instead, a multi-objective evolutionary algorithm (MOEA) is used to generate a set of initial solutions, which are then improved by an ejection chain process. In contrast to the previously discussed works, [8] proposes an approach for the orienteering problem with time windows. However, the considered MOEA does not make use of local search. The empirical analysis of this MOEA presented in [16] indicates that the algorithm's performance may depend significantly on individual features of the problem instance, and that the performance could be improved by integration of local search methods. Considering this background, the present works makes the following three main contributions:

- We integrate a number of alternative local search methods into our multi-objective evolutionary concept proposed in

GECCO '18, July 15–19, 2018, Kyoto, Japan
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5618-3/18/07...$15.00
https://doi.org/10.1145/3205455.3205548

[8], and we compare the resulting performances of the alternative approaches.

- We analyze a self-designed set of problem instances with respect to a large variety of instance features. The analysis illustrates the structural diversity of the instances, i.e., it shows that the instances represent a well-suited test-bed for comparing algorithms without overfitting to individual instance features.
- We analyze the general effects of local search on the multi-objective optimization process. We show that local search improves the process by boosting offspring solutions to foremost fronts, which leads to an increased selection pressure.

The remainder of the paper is structured as follows. In Section 2 we define the considered class of routing problems. In Section 3 we introduce our approach of generating problem instances as well as a large variety of instance features, and we present an instance feature analysis for illustrating the structural diversity of the instance set. In Section 4 we describe the multi-objective evolutionary algorithm that we combine with different local search methods in Section 5. Section 6 describes our experimental setup as well as the computational results of our analysis. Section 7 concludes the paper.

2 PROBLEM DESCRIPTION

Each problem instance consists of a set $C = \{1, 2, \ldots, N\}$ of locations, where location 1 is the start depot and location N is the end depot of the vehicle. Travel distances between any pair i, j of locations are denoted as d_{ij}. For the sake of notational convenience, we assume that one distance unit corresponds to one unit of time. All locations of C are mutually connected, and distances between any pair i, j of locations are symmetric. We refer to the set of all connections of pairs i, j as E. At time $t = 0$, the vehicle is located at location 1, and at the end of the tour it must be located at location N. The set $C \setminus \{1, N\}$ of customer locations consists of the subset C^m of mandatory locations and of the subset C^o of optional locations, with $C^m \cap C^o = \emptyset$. Each optional customer location i has an open time window with a lower bound $t_i > 0$ and must be accepted for service or rejected. For each location $i \in C$, the binary decision variable x_i^c indicates whether customer location i is visited or not. Decision variable x_{ij}^r indicates whether or not the road link connecting locations i and j is part of the vehicle's route. In order to make an appropriate trade-off between maximization of customer satisfaction (accepting all customers) and minimization of travel costs (typically rejecting all optional customers) both of these objectives are considered. The corresponding objective function can be formulated as

$$\min_{x^c, x^r} \left((|C| - \sum_{i \in C} x_i^c), (\sum_{(i,j) \in E} d_{ij} x_{ij}^r) \right).$$

We refer to [16] for an explicit statement of the full mathematical decision model imposed by the described problem.

3 PROBLEM INSTANCES

Vehicle routing algorithms are often evaluated using the well-established Solomon instances benchmark [21], which comprises (with respect to the model introduced in Section 2) a variety of instances with 50 and 100 customers in geographies with uniformly distributed or clustered locations. However, only eight instances of this benchmark set have both mandatory and optional customers, as required by the model of Section 2.

In order to evaluate our algorithmic approach on an adequate set of diverse and representative instances (avoiding overfitting to instance features), we generate a larger benchmark set, taking into account Solomon's basic types of clustered, uniformly distributed and mixed customer location geographies. We define a systematic generation process for problem instances providing (1) a wider range of instance sizes in terms of both numbers of locations and geographic scale, (2) variability in customer density and the number of customer clusters, (3) smoother and systematic morphing of clustered and uniform geographies, (4) variability in start and end depot positions, and (5) a variety of different ratios of mandatory and optional customers.

After presenting the instance generation approach in Section 3.1, we define specific VRP-features as well as high-level geography features in Section 3.2, both of which are used in Section 3.3 for a detailed analysis of the characteristics of our generated problems.

3.1 Instance Generation

We conduct our experiments on a specifically designed set of 360 problem instances. These instances differ in the number of customers $N \in \{50, 100, 200\}$ (including both start and end depot), the number $n_c \in \{1, 2, 3, 5, 10\}$ of clusters of customer locations, and the fraction $f \in \{0.25, 0.5, 0.75\}$ of optional customers, representing a variety of typical real-world applications.

As a starting point we generate a single geographical layout for each combination of number of customers and number of clusters. Randomly uniform instances (termed single-cluster instances) are generated by randomly placing locations 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 clustered instances was introduced in [8] and works as follows: 1) a latin hypercube design (LHS) of size n_c forms the cluster centers $c_1, \ldots, c_{n_c}$ (the space-filling property of LHS design ensures well-spread cluster locations). 2) Each cluster is crowded with $\lfloor N/n_c \rfloor$ random points from a multivariate Gaussian distribution $\mathcal{N}(c_i, \text{diag}(d_i, d_i))$. The cluster center is used as the mean and the distance $d_i = \min_{i \neq j} d(c_i, c_j)$ to the nearest cluster center as the standard deviation in both dimensions to avoid overlapping of clusters.

Moreover, we consider instances in-between clustered and uniform by adopting the concept of *morphing* (see [16–18]). In a nutshell, given two instances of same size, a minimal weighted point matching is computed first between the point coordinates of the source instances. This results in a one-to-one assignment of points. In a subsequent step the point coordinates of the morphed instance are generated by convex combination $p = \alpha p_1 + (1 - \alpha) p_2$ of the paired points p_1, p_2 from the source instances. Here, the *morphing coefficient* $\alpha \in [0, 1]$ regulates the similarity between the morphed instance and its parents respectively (see Figure 1 for a smooth transition effect between a uniform and a clustered instance with $\alpha \in \{0, 0.5, 1\}$).

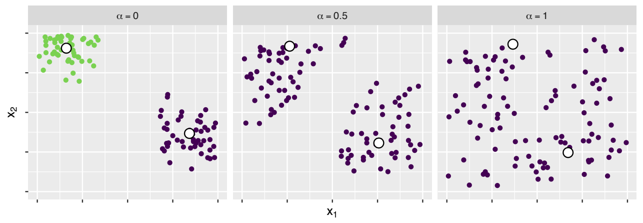

Figure 1: Morphing of a clustered instance with two clusters and a random instance of equal size with two depots (white circles) in each case for different morphing coefficients α.

We use morphing to generate instances that are in-between a random instance and an instance with five clusters for each cluster size and corresponding instance size, resulting in 9 further geographical layouts. The point matching of depots and customers is done separately within this procedure to ascertain, that there is a smooth morphing between the coordinates of the depots and customers. Otherwise it would be very likely to morph a depot with a non-depot point.

In summary we come up with 24 geographies (3 random, 12 clustered, 9 morphed), see Figure 2 for examples. In the final step of the generation process, we assign lower time window bounds to a fraction of $f \in \{0.25, 0.5, 0.75\}$ customers randomly for each of our geographies increasing the number of optional customers stepwise. The lower bounds are sampled from an exponential distribution with the rate parameter set accordingly. We start with a lower bound limit of 400 time units for instances with 50 customers, and double the limit for instances next in size each time. This process is repeated five times with different seeds for the random number generator to obtain different realizations of the underlying Poisson process. The implementation of the instance generation is available in the R package *netgen* [2].

3.2 Instance Features

Each instance that falls into the problem class of Section 2 is fully characterized in terms of the coordinates of the locations in C, the distances between any pair of locations, the partitioning of C into C^m and C^o, as well as the lower time window bounds of optional customers. We derive the following instance features from these elementary characteristics:

- average distance and standard deviation of distances between all pairs (i, j) of locations $i, j \in C$, all pairs of mandatory locations $i, j \in C^m$ and all pairs of optional locations $i, j \in C^o$
- the ratio $|C^m|/|C^o|$ of the number of mandatory and optional locations
- the ratio $|C^o|/|C^m \cup C^o|$ of the number of optional and all locations
- average and standard deviation of distances between all pairs $(1, i)$ or (i, N) with $i \in C^m$, $i \in C^o$ respectively
- distance between start depot and end depot d_{1N}
- coordinates of the start depot (x_1, y_1) and end depot (x_N, y_N)

3.3 Instance Feature Analysis

First, we compare the generated instances ("*netgen* instances") with instances introduced by [21] ("Solomon instances"). Second, we perform a feature analysis on the netgen instances in order to gain insights into the structural diversity of this newly generated instance set.

The Solomon instances are frequently used in the orienteering literature [see, e.g., 15, Chapter 8]. However, these instances are restricted to only eight geographies. Furthermore, 50% of the instances have dynamic (optional) customers exclusively, i.e., all customers have lower time window bounds $t_i > 0$ and none of the customers can be visited at a completely arbitrary point in time. In contrast, the 360 considered netgen instances are based on 24 different geographies (see Figure 2 for examples), contain both customers with $t_i = 0$ and customers with $t_i > 0$ as well as a much more diverse system of clustering.

We investigate the distributions of VRP features for both netgen and Solomon instances. In particular, we identify the features, for which a significance test with the alternative of netgen instances having a larger variance rejected the null hypothesis of equal variance. This holds true for 27 out of 77 features. For the remaining features, where the statement holds for the opposite hypothesis, however, differences in variance are very small and only marginally significant. A wider range of features indicates, that the instances are more diverse and thus more types of real-world instances are be covered. Moreover, we can see from Figure 3 that the VRP features are very informative in that a huge fraction of the latter does not correlate with most of the remaining ones. Thus, each of these features presents additional and almost unique information.

4 THE MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM

The evolutionary algorithm developed here, follows the reproduction/selection pattern of NSGA-II [3], which is a widely accepted multi-objective evolutionary algorithm. It uses a population of size μ and performs an evolutionary loop for generating μ offspring solutions by variation. Selection is performed by ranking the union set of parents and offspring regarding non-domination, and creating the next generation population by adding better ranked solutions until it contains μ or more individuals. In case the new population size exceeds μ, solutions with worst rank and smallest crowding distance are removed. For our considered VRP problem, encoding and variation operators are adapted. The adaptations are described in the following.

4.1 Encoding

A solution to the vehicle routing problem defined in Section 2 consists of two main components: a subset of visited customers and an optimal tour for visiting those customers. In a solution candidate, selected (i.e., visited) customers are expressed via a binary string, while the sequence of visits is encoded as a permutation string. Both strings are of length $N - 2$, i.e., start and end depots are not explicitly encoded as they are not subject to change during the evolution. Formally, 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

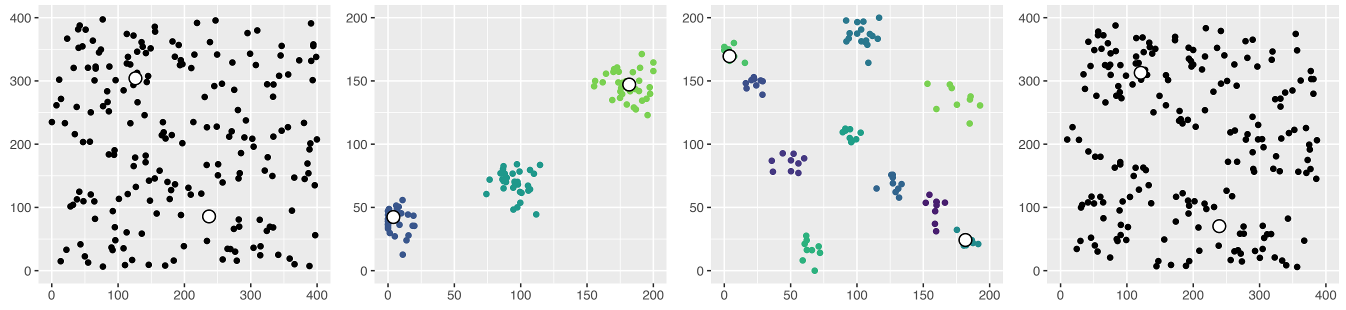

Figure 2: Examplary geographies of the generated problem instances.

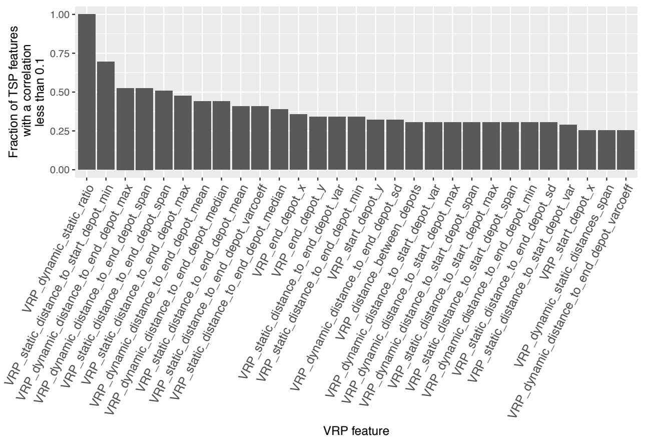

Figure 3: For each VRP feature f the fraction of TSP-features g with $|cor(f, g)| \leq 0.1$. VRP features with fraction ≤ 0.25 are omitted.

is visited ($b_i = 1$) or not ($b_i = 0$). A schematic example for two scenarios encoded in the suggested way is shown in Figure 4.

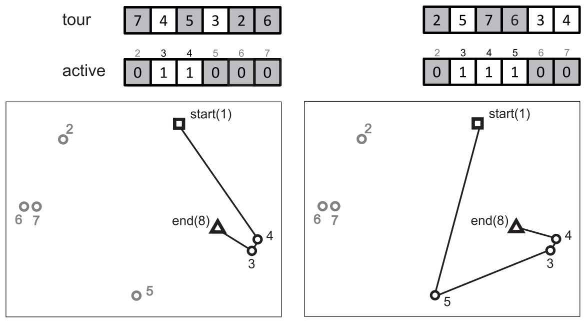

Figure 4: Two encoding examples. Deactivated customers are shown in grey and determined by a 0 in the activation chromosome. Note that they remain part of the permutation string of the tour, however they are not considered during tour planning and evaluation.

For practical reasons and to ensure constant genome length over all individuals, 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 cannot change throughout variation leading to a fixed activation or deactivation of a customer. This way we ensure that customers from C^m always stay obligatory. A probability $0 < p_i \leq 1$ denotes dynamic customers from C^o[1].

Service request times t_j for a customer $j \in C$ are directly integrated into cost computation for traveling an edge (i, j) during evaluation. If a customer's service request is not yet available (later request time), the vehicle remains waiting at the previous customer and waiting time is added to tour length. This way, all solutions are feasible (although sometimes inefficient).

4.2 Variation

The combined representation of solutions by two chromosomes demands for special variation operators, which we constructed as combinations of standard mutation and crossover operations respectively.

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 at random from $\{0, \ldots, \sigma_p\} \subset \mathbb{N}$. Simultaneously, the binary string undergoes a flip mutation, where each position $b_i \in \{0, 1\}$ is flipped with probability $p_i \in [0, 1]$, where p_i is encoded in 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 repairing 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.

The influence of Swap/Flip-mutation is twofold: while swapping inside the permutation only changes the order of visits to customers,

[1]Note that this is not necessary when individuals of varying length are handled adequately. However, the disposition of customers inside the chromosome may contribute as temporarily inactive building blocks to other solutions in the evolutionary process. Especially in the light of multi-objective optimization where different activation scenarios are considered at once, related tour building blocks can be beneficial.

flip mutation influences tour length by activating or deactivating customers. Contrary, the crossover operator is expected to recombine fragments of tours as well as "sub-scenarios" of customers (activated and deactivated) into new – hopefully efficient – settings.

5 INTEGRATION OF LOCAL SEARCH

In order to improve the performance of NSGA-II for the bi-objective orienteering problem defined in Section 2, we transform it to a memetic approach by integrating local search mechanisms. In the following Section 5.1 we detail the injection point of local search into the NSGA-II algorithmic architecture. In Section 5.2 we briefly describe the local search strategies applied.

5.1 Injection Point for Local Search

For the considered VRP, we identify the optimization of the vehicle's tour length as the more difficult goal of optimization and restrict local search integration to integrating different TSP solvers. The selection of an adequate set of optional customers is left to the unbiased mutation operator. We assume, that mutation and diversity preserving measures of NSGA-II ensure a good subset selection.

The second objective – vehicle's tour length – is addressed by considering TSP as a relaxed problem without service request times. Note, that a tour length resulting from TSP optimization is only a lower bound of the tour length of the VRP. Respecting service requests during evaluation can eventually lead to longer tours than in the original solution before applying local search. Thus, TSP local search is injected into the NSGA-II architecture after offspring generation and before selection of the next generation ensuring the removal of (at least) substantially deteriorated individuals. Moreover, local search is only applied to offspring individuals and parents at five distinct points in time (0%, 25%, 50%, 75%, and 100% of total generations).

5.2 Local Search Methods

Here, we apply three heuristic approaches to solve the TSP problem inside our MOP:

The basic *2-opt heuristics*, which repeatedly tries to remove edge crossings from a tour is based on the insight, that in Euclidean graphs planar tours are more cost-efficient than non-planar.

LKH-2 [10] (denoted as LKH in the following) is a very efficient implementation of the Lin-Kerninghan algorithm extended to k-opt sub-moves while such moves alter a tour by replacing k edges such that a tour of shorter length is obtained. More specifically, 5-exchange moves are combined with a specific nearest neighbour heuristic and a measure resulting from sensitivity analysis of minimum spanning trees helps to identify excellent candidate sets of reasonable size. LKH proved to be the state of the art solver in inexact TSP solving since its introduction in 2000.

The *edge assembly crossover* (EAX) genetic algorithm [20] combines edges from two parent tours with a small number of new, short edges based on a sophisticated tabu-search strategy focusing on very high-quality parent solutions. The initial population is formed by means of the 2-opt heuristic and diversity of candidate solutions is ensured by relying on an entropy-based mechanism. The authors show that the EAX is competitive to LKH on commonly studied Euclidean TSP instances and even outperforms LKH on specific instances [20]. A systematic comparison of both algorithms can be found in [14].

Note, that the original implementations of LKH and EAX only support closed TSP instances. The VRP, however, consists of open tours connecting start and end depots. Therefore problem instances are transformed on-the-fly to be handled by the local strategies: Start and end depots are replaced by introducing a virtual node with outgoing edge weights of the start depot and incoming edge weights of the end depot yielding a closed but asymmetric graph. The asymmetric graph is accepted by the LKH implementation. For EAX a second transformation of the asymmetric closed instance into a symmetric closed instance is necessary. This is achieved by duplicating nodes and setting infinite small distances between twin nodes. In this construction one node is responsible for all incoming and the other node for the outgoing distances. All other distances are set to infinity [11].

6 COMPUTATIONAL EXPERIMENTS

6.1 Experimental Setup

NSGA-II is contrasted to memetic NSGA-II variants, i.e., combined with 2-opt, LKH and EAX local search based on 10 independent runs each. Local search terminated by internal termination criteria of the respective approaches. Table 1 lists general parameter settings while instance sizes are set to $50 \cdot 2^K, K = 1, 2, 3$.

Table 1: General algorithm parameters' scheme.

Population size	$\mu = 100 \cdot 2^K$
Swap number	$\sigma_p = 2 \cdot 2^K$
Flip probability	$\frac{1}{N-2}$
Function evaluations	$\nu = 6,500,000 \cdot 2^K$
Internal local search	only at 0%, 25%, 50%, 75%, and 100% of total Generations

However, for a detailed local search analysis we used $\nu = 65,000 \cdot 2^K$ on instance sizes of 50 and 100 customers.

6.2 Analysis of Local Search Effects

The different algorithm variants are compared in terms of obtained Dominated Hypervolume (HV) [23]. HV is computed for each problem instance using the overall nadir point plus $(1, 1)^T$ as reference point. For this purpose algorithm results are combined for all variants in order to obtain the borders of the required bounding box. Subsequently, the HV values are normalized per instance for visualization on a common scale, see Figure5.

It becomes obvious that the memetic NSGA-II variant integrating TSP local search strategies significantly (Wilcoxon rank test used to reject equality) improves the original algorithm performance (EA_noLS). This effect even increases with problem dimensionality, i.e., with instance size. Moreover, the superiority of state-of-the-art inexact TSP solvers w.r.t. simple local search heuristics such as 2-opt is reflected. Whereas no relevant distinction can be made between LKH and EAX, the improvement gained by a hybrid with 2-opt is significantly smaller for all instance sizes. Moreover, the

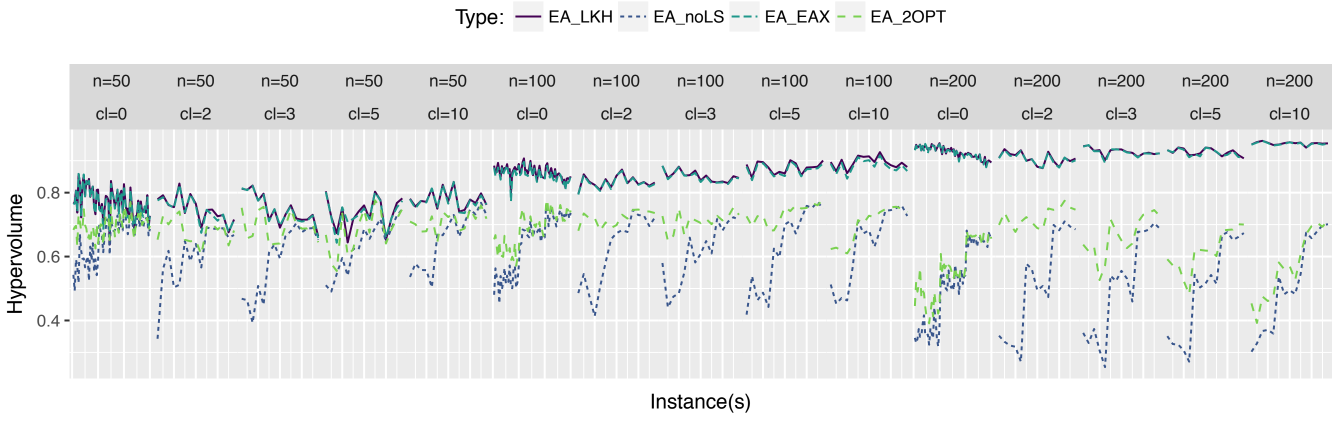

Figure 5: Algorithm results in terms of obtained mean hypervolume. The EA was either applied without local search ("noLS") or as a memetic variant combined with different local searches. Instance sizes *n* are marked together with the number of clusters *cl*. Morphed instances are assigned $cl = 0$.

advantage of 2-opt integration decreases with increasing number of clusters while the opposite is true for the sophisticated strategies LKH and EAX.

Note that the HV values cannot be compared across instances due to different maximum numbers of points on the Pareto front as well as varying shapes of the latter. The increasing HV trend along with the instance size, however, can simply be explained by the fact that with increasing number of nodes, independent from the ratio of dynamic customers, the number of points on the Pareto front gets higher in general.

In addition to the overall gain in hypervolume performance, we investigate the observed effects of local search application in detail. Therefore, we recorded the population development of parent and offspring sub-populations during the whole evolutionary process in separate experiments with 65,000 allowed function evaluations per run. Figure 6 shows the development of the Pareto front at specific generations. Here, we depict the generations for which local search has been activated, and the respective generation before for which EA has performed a maximum of generations since the last local search. For comparison reasons we also show the Pareto front development without application of local search. At generations with activated EAX local search, we observe a strong performance gain regarding tour length. Especially in the beginning of the evolutionary process, local search pushes solutions towards shorter tour lengths while keeping the customer setting[2]. During intermediate application of only MOEA variation operators, however, the population diversifies and explores additional customer configurations. Subsequently, these solutions are again pushed towards shorter tours by the next local search application. This observation is especially interesting, as it suggests that the integration of single-objective local search operators can still largely contribute to performance of the multi-objective solution.

Derived from the above discussion based on qualitative observations, we quantitatively investigate the influence of local search on the increase in algorithmic performance in more detail. Specifically, we aim to gain insight into two further aspects of the memetic approach: (1) the rationale of the chosen injection point for local search and (2) how sparsely applied local search influences population development in the course of a complete algorithm run. The discussion is based on two hypotheses:

Hypothesis 1: *The performance increase by integrating local search only for TSP suggests that offspring individuals are strongly boosted to lower (better) fronts.*

In order to check this hypothesis, we specifically recorded the front positions individuals had during non-dominated sorting and selection during our additional experiments. Figure 7 shows the distribution of front positions for all individuals and runs regarding the first three fronts. As representatives, we show 2-opt and EAX as differently performing local search strategies and pure NSGA-II for comparison.

We observe, that the application of local search to offspring individuals leads to a significant larger amount of selected offspring in fronts 1 to 3 at four points in time (25%, 50%, 75% and 100%) than the application of standard variation both in between local search and for pure NSGA-II. This holds for state-of-the-art high performing as well as for simple and only fair performing local search strategies, although modern approaches like EAX contribute far stronger to convergence, astonishingly also at late generations. This leads to two conclusions: First, although TSP local search strategies do not address the VRP tours (including service requests) directly, the found TSP solutions are often advantageous also for the vehicle routes. I.e., also inexact expert knowledge integration can contribute to solution quality. Second, even boosting solution quality at only few points in time during evolution contributes to faster convergence. This aspect leads to the second hypothesis.

Hypothesis 2: *Boosted offspring individuals originating from local search serve as "forerunners" and increase selection pressure for the intermediate evolutionary process.*

For finding empirical evidence supporting this hypotheses, we deeply investigate the recorded data on all individuals throughout

[2]As the local search only influences the permutation encoding part of a solution's chromosome, customer configurations do not change.

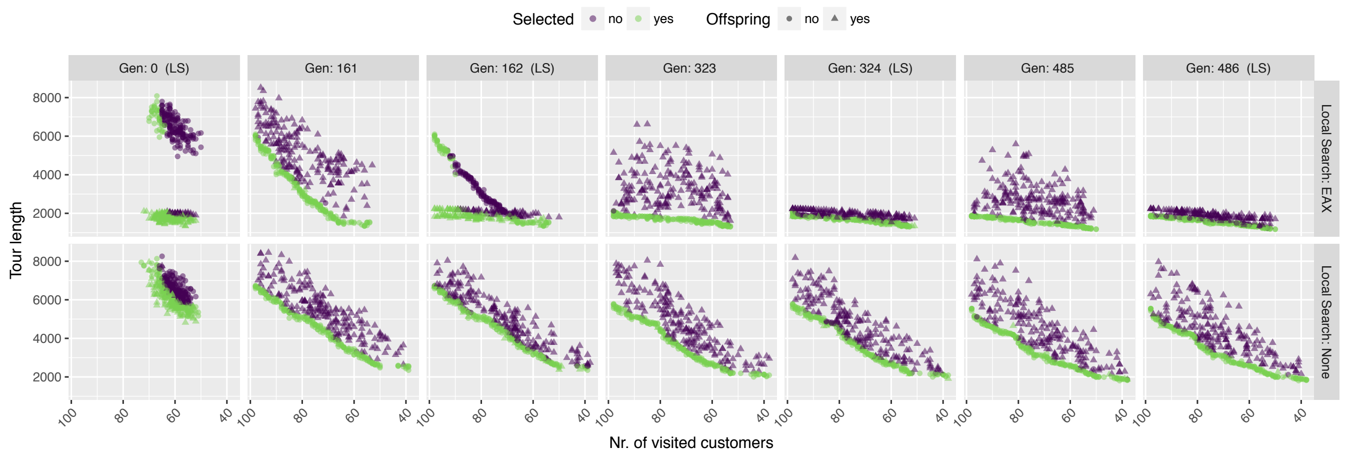

Figure 6: Exemplary evolution of the Pareto fronts for EAX local search and intermediate generations as well as no local search are shown. Parental individuals are denoted by filled circles, offspring by filled triangles, while surviving solutions are marked in green, disregarded solutions in red color.

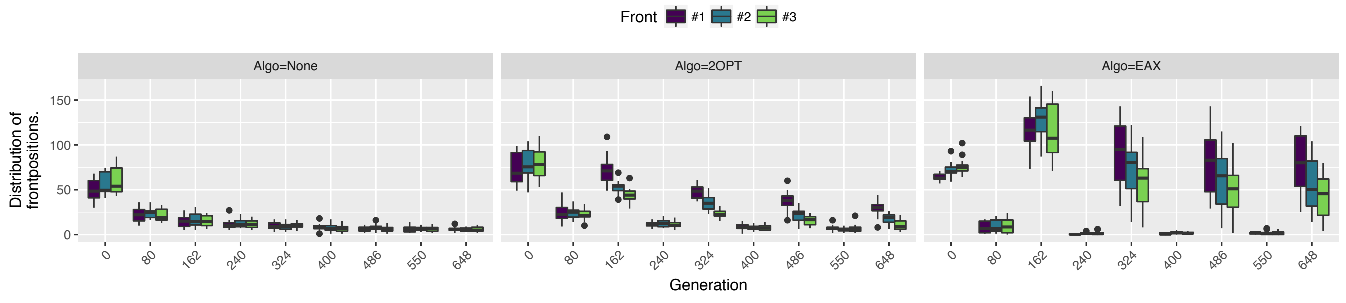

Figure 7: Distribution of front positions for selected offspring. Here only specific generations without activated local search for two strategies and pure NSGA-II are shown.

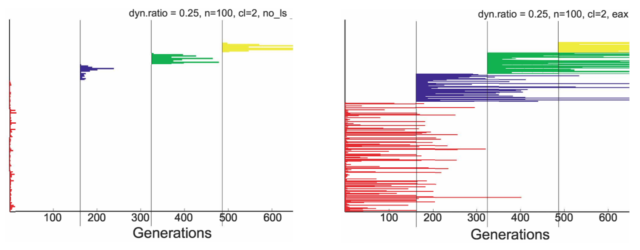

Figure 8: Left: Detailed depiction of the life-line of each individual generated at four selected generations (1, 163, 325, and 487) for NSGA-II without local search. Right: Detailed depiction of the life-line of each individual generated during local search which is only applied in the above generations. All other offspring is generated by standard variation.

the evolutionary process and specifically concentrate on the lifespan of offspring individuals which survived selection after creation. For each individual the remaining lifespan until algorithm termination is determined. Figure 8 shows lifespans of all selected offspring individuals from generations 1, 163, 325, and 487. EAX local search was performed for Figure 8 (right) whereas no local search was performed in Figure 8 (left). Lifespan was investigated at the same points in time in both cases. This exemplary behavior shown here is transferable to all runs.

The results show a distinct longer lifespan for individuals resulting from a local search operation than for solutions generated by standard variation. Some individuals even survive more than 3/4 of the evolutionary process. On the one hand, this impressively demonstrates the power of the applied local search approach in generating good solutions even for VRP tour length. It suggests, that few local searches suffice to keep good genetic information in the population. On the other hand, individuals with a long lifespan become almost constant members of the population occupying places permanently. This leads to higher selection pressure for individuals striving for membership in the population during the evolutionary process. Finally, a longer lifespan theoretically enables individuals to spread genetic information more often than individuals which outlast only few generations.

7 CONCLUSIONS

We address a class of bi-objective vehicle routing problems and propose a memetic multi-objective algorithm based on NSGA-II into which we integrated state-of-the-art (but in the context of the problem inexact) local search strategies for optimizing the vehicle's tour. In numerical experiments, we demonstrate the good performance of the memetic strategies and analyze the effects of local search in detail. Thereby, we find evidence that our local search integration (1) outperforms the original NSGA-II, (2) is capable of sending locally optimal individuals to foremost fronts of the NSGA-II selection process and (3) significantly extending the lifespan of these individuals making them leading factors in the evolutionary process. Interestingly, focusing on only one of the objectives within local search combined with the EA characteristics substantially improves both objectives along the algorithm run, even given the very small number of local searches used.

Moreover, we present a set of problem-tailored features characterizing problem instances of the class of so-called orienteering problems. Together with presenting a set of systematically generated instances with different levels of clustering a well-defined benchmark set for small to moderate instance sizes together with its characteristics is provided. These instances are shown to be more comprehensive and flexible than the commonly used Solomon instances.

This work offers many perspectives for future directions of research: The in-depth analysis of the local search influence can be extended in order to identify beneficial heredity patterns induced by local search. This may lead to sophisticated local search integration guidelines or even to inspiration for variation operator design. Moreover, we will compare the results to linear programming techniques generating points on the tradeoff surface in several parallel or sequential runs and relate all respective algorithm performances to instance characteristics. The long-term goal is to design an efficient multi-objective memetic algorithm which dynamically optimizes the orienteering problem, i.e., without the full knowledge of request times.

REFERENCES

[1] J.-F. Berube, M. Gendreau, and J.-Y. Potvin. 2009. An exact [epsilon]-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research* 194, 1 (2009), 39–50.

[2] Jakob Bossek. 2015. *netgen: Network Generator for Combinatorial Graph Problems.* https://github.com/jakobbossek/netgen R package version 1.0.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[4] Dominique Feillet, Pierre Dejax, and Michel Gendreau. 2005. Traveling Salesman Problems with Profits. *Transportation Science* 39, 2 (2005), 188–205.

[5] Carlo Filippi and Elisa Stevanato. 2013. Approximation schemes for bi-objective combinatorial optimization and their application to the TSP with profits. *Computers & Operations Research* 40, 10 (2013), 2418–2428.

[6] Birger Funke, Tore Grünert, and Stefan Irnich. 2005. Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration. *Journal of Heuristics* 11, 4 (2005), 267–306.

[7] Bruce L. Golden, Larry Levy, and Rakesh Vohra. 1987. The orienteering problem. *Naval Research Logistics* 34, 3 (1987), 307–318.

[8] Christian Grimme, Stephan Meisel, Heike Trautmann, Günter Rudolph, and Martin Wölck. 2015. Multi-Objective Analysis of Approaches to Dynamic Routing of a Vehicle. In *ECIS 2015 Completed Research Papers. Paper 62.* AIS Electronic Library. https://doi.org/10.18151/7217333

[9] Chris Groër, Bruce Golden, and Edward Wasil. 2010. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation* 2, 2 (2010), 79–101.

[10] Keld Helsgaun. 2009. General k-opt submoves for the Lin-Kernighan TSP heuristic. *Mathematical Programming Computation* 1, 2-3 (2009), 119–163.

[11] Roy Jonker and Ton Volgenant. 1983. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters* 2, 4 (1983), 161 – 163. https://doi.org/10.1016/0167-6377(83)90048-2

[12] Nicolas Jozefowiez, Fred Glover, and Manuel Laguna. 2008. Multi-objective Meta-heuristics for the Traveling Salesman Problem with profits. *Journal of Mathematical Modelling and Algorithms* 7, 2 (2008), 177–195.

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

[14] L Kotthoff, P Kerschke, H Hoos, and H Trautmann. 2015. Improving the State of the Art in Inexact TSP Solving using Per-Instance Algorithm Selection. In *LION 9*, C. Dhaenens et al. (Eds.). Cham, 202–217. https://doi.org/10.1007/978-3-319-19084-6_18

[15] Stephan Meisel. 2011. *Anticipatory Optimization for Dynamic Decision Making.* Operations Research/Computer Science Interfaces Series, Vol. 51. Springer New York.

[16] Stephan Meisel, Christian Grimme, Jakob Bossek, Martin Wölck, Günter Rudolph, and Heike Trautmann. 2015. Evaluation of a Multi-Objective EA on Benchmark Instances for Dynamic Routing of a Vehicle. In *GECCO '15.* ACM, New York, NY, USA, 425–432. https://doi.org/10.1145/2739480.2754705

[17] Olaf Mersmann, Bernd Bischl, Jakob Bossek, Heike Trautmann, Markus Wagner, and Frank Neumann. 2012. Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness. In *LION 6, Paris (LNCS)*, Y Hamadi and M Schoenauer (Eds.), Vol. 7219. Springer, 115–129.

[18] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, and Frank Neumann. 2012. A Novel Feature-Based Approach to Characterize Algorithm Performance for the Traveling Salesman Problem. *Annals of Mathematics and Artificial Intelligence* 69 (2012), 151–182.

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

[20] Yuichi Nagata and Shigenobu Kobayashi. 2013. A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem. *INFORMS Journal on Computing* 25, 2 (2013), 346–363.

[21] M Solomon. 1987. Algorithms for the vehicle routing and scheduling problem with time windows. *Operations Research* 35, 2 (1987), 254–265.

[22] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209, 1 (2011), 1–10.

[23] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7, 2 (2003), 117–132.