# Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study

Luis Paquete, Marco Chiarandini and Thomas Stützle

Darmstadt University of Technology, CS Department, Intellectics Group
Alexanderstr. 10, 64283 Darmstadt, Germany
{lpaquete,machud,tom}@intellektik.informatik.tu-darmstadt.de

**Abstract.** In this article, we study Pareto local optimum sets for the biobjective Traveling Salesman Problem applying straightforward extensions of local search algorithms for the single objective case. The performance of the local search algorithms is illustrated by experimental results obtained for well known benchmark instances and comparisons to methods from literature. In fact, a 3-opt local search is able to compete with the best performing metaheuristics in terms of solution quality. Finally, we also present an empirical study of the features of the solutions found by 3-opt on a set of randomly generated instances. The results indicate the existence of several clusters of near-optimal solutions that are separated by only a few edges.

## 1 Introduction

Local search algorithms are at the core of the most successful metaheuristics for solving a wide variety of $\mathcal{NP}$-hard single objective combinatorial optimization problems such as the Traveling Salesman Problem (TSP) [22, 21]. Local search algorithms for single-objective problems can also be adapted in rather straightforward ways to multiobjective combinatorial problems, opening their use inside a metaheuristic [5, 11, 13, 20].

A first and simple idea for extending local search algorithms to multiobjective problems is to maintain the same search strategy of the single-objective local search. The main difference between the single and the multiobjective case then concerns the acceptance criterion of new solutions in the local search. Here, we adopt an acceptance criterion inspired by the notion of Pareto optimality: a solution in the neighborhood of the current one is accepted if it is not dominated by all non-dominated solutions found so far in the search. As a side effect, this multiobjective extension of local search allows us to define the notion of Pareto local optimum set analogous to the local optimum in single-objective case. We call the resulting local search algorithm Pareto Local Search (PLS). PLS allows to find Pareto local optimum sets, that is, a set of solutions that plays the same role as a local optimum in the single-objective case. Therefore, it has the large potential of serving as a reference for comparisons of multiobjective metaheuristics in the very same way as iterative improvement local search algorithms are used for the baseline evaluation of the usefulness of single-objective metaheuristics.

In this article, we tackle the biobjective TSP by PLS based on the well-known 2-opt, 2h-opt and 3-opt local search algorithms [1, 4, 9, 25]. We evaluate our PLS algorithms applying them to well-known benchmark instances and assess their performance by means of the attainment function methodology [6], the coverage measure ($C$ measure) [33] and the expected value of the weighted Tchebycheff scalarizing function ($R$ measure) [20]. In addition, we compare the solutions returned by PLS with those available from metaheuristic approaches to the biobjective TSP. Surprisingly, the results indicate a good performance of the 3-opt version in terms of solutions quality, which is comparable to that achieved by the Multiobjective Genetic Local Search (MOGLS), currently the best performing metaheuristic approach to the multiobjective TSP [20]. However, this high solution quality comes at the price of substantial computation times. Finally, we also present a study on the characteristics of the Pareto local optimum sets obtained by 3-opt. The results suggest that, once a reasonably good solution is found, most of the remaining non-dominated set is reachable by a small number of edge exchanges. This fact can even lead to the development of more sophisticated local search algorithms.

The article is structured as follows. In Section 2, we introduce the multiobjective TSP and in Section 3 we define Pareto local optimum set and present PLS. The methodology we use for assessing the performance of PLS is indicated in Section 4 and Section 5 contains experimental results on some benchmark instances. The insights gained from an analysis of the solution set are the topic of Section 6 and we conclude and indicate lines of future research in Section 7.

## 2 The Multiobjective TSP

In the multiobjective TSP, we are given a complete, weighted graph $G = (N, E, c)$ with $N$ being the set of nodes, $E$ being the set of edges fully connecting the nodes, and $c$ being a function that assigns to each edge $(i, j) \in E$ a vector $(c^1_{ij}, \ldots, c^K_{ij})$, where each element $c^k_{ij}$ corresponds to a certain measure like distance or cost between node $i$ and $j$. For the following we assume that $c^k_{ij} = c^k_{ji}$ for all pairs of nodes $i, j$ and every objective $k$, that is, we consider only symmetric problems. The multiobjective TSP is the problem of finding "minimal" Hamiltonian circuit(s) of the graph, that is, closed tour(s) visiting each of the $n = |N|$ nodes of $G$ exactly once; here "minimal" refers to the notion of Pareto optimality.

One widely used approach to find good quality solutions for the (single-objective) TSP is local search. In the most basic local search algorithm, one searches in the neighborhood of the current tour for an improved tour; if such a tour is found, it replaces the current one and the process is iterated until no improved tour can be found anymore. This is a so called iterative improvement algorithm. It critically depends on the notion of neighborhood. In this article, we consider three iterative improvement algorithms that differ only in the definition of the neighborhood. We considered the following iterative improvement algorithms.

**2-opt**: A 2-exchange deletes two edges and replaces them with the only possible pair of new edges that does not break the tour in two disjoint cycles. The 2-opt algorithm, given some starting tour, applies 2-exchanges as long as possible, and ends when no further improving 2-exchange can be applied. In our case, the neighborhood is searched in a randomly chosen order and the first 2-exchange move encountered that leads to a non-dominated solution is applied.

**2h-opt**: A 2h-exchange, in addition to 2-exchanges, considers moves of a single city from one position in the tour to another one. As suggested in Bentley [1], our implementation is a simple extension of the 2-opt algorithm. The search strategy employed in the 2h-opt algorithm is then analogous to the 2-opt case;

**3-opt**: A 3-exchange deletes three edges and replaces them by a different set of three edges. The search strategy applied in 3-opt is the same as the one we used for 2-opt.

There exist a number of speed-up techniques for the iterative improvement algorithms in the single-objective case [1, 21], however, for the multiobjective case we only use the $\Delta$-evaluation, that is, the quality of the new tour can be evaluated by computing the change in the objective function value incurred by the edge exchanges for each of the multiple objectives independently.

## 3 Pareto local optimum set and Pareto local search

Before defining the notion of Pareto local optimum set, we need to introduce the notion of dominance as follows (without loss of generality, we are assuming minimization problems).

**Definition 1. (Dominance)** *We say that a vector $v = (v^1, ..., v^K)$ dominates a vector $u = (u^1, ..., u^K)$ if and only if $v^k \leq u^k \; \forall \, k \in \{1, ..., K\} \wedge \exists \, k \in \{1, ..., K\} : v^k < u^k$. We denote this by $v \prec u$.*

Usually there is not only one single Pareto optimal solution, but several, which are elements of a *Pareto global optimum set*. This set contains all solutions that are not dominated by any other solution.[1] In analogy, we can define a *Pareto local optimum* and a *Pareto local optimum set* as follows.

**Definition 2. (Pareto local optimum and Pareto local optimum set)** *Let $p$ be a feasible solution, $\mathcal{N}$ be a neighborhood structure, and $f = (f^1, ..., f^K)$ be a vector objective function.*

- *We say that a solution $p$ is a* Pareto local optimum *with respect to $\mathcal{N}$ if and only if there is no $r$ in the neighborhood of $p$ such that $f(r) \prec f(p)$.*
- *We say that $P$ is a* Pareto local optimum set *with respect to $\mathcal{N}$ if and only if it contains only Pareto local optimum solutions with respect to the neighborhood considered.*

The problem of finding the Pareto global optimum set for the multiobjective TSP is $\mathcal{NP}$-hard [7]. Since obtaining exact solutions becomes extremely time consuming with increasing instance size, the main goal shifts from obtaining Pareto global optimal solutions to obtaining a good approximation to this set and metaheuristics seem

[1] Given two solutions $s$ and $r$, we say $s$ dominates $r$ if $f(s) \prec f(r)$, where $f = (f^1, ..., f^K)$ is a vector objective function.

to be a suitable approach for this task [14, 20]. In this paper, we study how good are approximations through Pareto local optimum sets, especially when compared to other metaheuristic approaches.

For this task, we have to extend the single-objective local search algorithms to the multiobjective case. A first decision is that we use the same notion of neighborhood as in the single-objective case. The main modifications are concerned with the acceptance criterion of the single-objective local search, because in the multiobjective case we need to take into account several objectives. In iterative improvement local search for the single-objective case a solution is usually accepted if it is better than the current one. For multiobjective problems where a set of non-dominated solutions has to be found, an extension of this acceptance criterion should take into account the concept of Pareto optimality.

For the sake of simplicity, a first approach for an acceptance criterion may be to accept a neighboring solution, if it is not dominated by the current solution. However, for a reasonable acceptance criterion and to avoid cycling, we also need to consider the non-dominated solutions that were previously produced during the local search. These are typically kept in an archive, and these solutions need to be considered by the acceptance criterion. Therefore, the final acceptance criterion we propose for the local search is the following: a new solution is accepted if it is not dominated by any solution in the archive. In fact, during this process, some solutions in the archive could become dominated by some of the recently introduced ones. Such solutions are eliminated from the archive. We call this general local search approach to multiobjective problems the *Pareto local search* (PLS).

The local search algorithm then works, from a high level perspective, as follows:

1. it starts from a randomly generated tour, which is added to the archive;
2. it picks a solution $s$ randomly from the archive and explores its neighborhood;
3. once a non-dominated solution $s'$ in the neighborhood of $s$ is found, $s'$ is added to the archive;
4. after the full neighborhood of $s$ is examined, $s$ is flagged as *visited* and we continue at step 2.

The local search procedure terminates if all the neighborhoods of all solutions in the archive were explored, *i.e.* every solution in the archive is flagged as *visited*. In this case, the archive is a Pareto local optimum set.

Algorithm 1 gives a pseudo-code for PLS. Some specific details were implemented to eliminate redundancy: $s'_t$ is only compared with the archive $A$ if it is not dominated by $s_t$ and PickArchive() picks only a solution from the archive $A$ which is not flagged as $visited$. The remaining procedures are self explanative: GenerateInitialSolution() generates the initial solution; UpdateArchive() adds the new non-dominated solution to the archive and removes the dominated ones. For tackling the biobjective TSP we adapted the PLS algorithm by extending the three 2-opt, 2h-opt, and 3-opt single-objective local search algorithms with the acceptance criterion of PLS.

A related approach to PLS can be found in [32]. There, the authors propose an hybrid evolutionary algorithm for the flow shop problem which is improved by post-optimizing the solution set with an additional local search algorithm analogous to PLS.

**Algorithm 1** Pseudo-code for Pareto Local search

```
$t \leftarrow 0$
$A \leftarrow \{\}$
$s_t \leftarrow$ GenerateInitialSolution()
$A \leftarrow$ UpdateArchive($s_t$)
repeat
  for all $s'_t \in$ Neighborhood($s_t$) do
    Evaluate($s'_t$)
    if $s'_t$ is not dominated by any $r \in A$ then
      $A \leftarrow$ UpdateArchive($s'_t$)
    end if
  end for
  $s_t \leftarrow visited$
  $t \leftarrow t + 1$
  $s_t \leftarrow$ PickArchive()
until all solutions in Archive are visited
```

Also SEMO [24] and PAES [23] have principles similar to PLS. However, PLS differs from these approaches in the sense that we do not consider any restrictions on the size of the archive and we take into account the complete examination of neighboring solutions. We stress here that we intend to match as much as possible the characteristics of usual local search algorithms for single-objective problems when extending these to multiobjective problems.

## 4 Performance assessment methodology

PLS takes random decisions in the GenerateInitialSolution() and PickArchive() procedures. Therefore, it is a stochastic search algorithm, which returns an outcome that is likely to be different for each run. As stressed by several researchers [17, 18], statistical methods are necessary to analyse the performance of such algorithms. A sound methodology for analysing sets of non-dominated vectors is the *attainment function* methodology [6] that allows to infer conclusions on the performance of stochastic search algorithms based on the outcomes of multiple runs.

The attainment function represents the probability that an arbitrary goal in the objective space is attained during a single run of the algorithm. This probability can be estimated using data collected from several optimization runs and normalizing the number of times each solution was attained at each run in a way analogous to estimating empirical distributions of one-dimensional random variables. This estimation process leads to the definition of the *empirical attainment function* (EAF) [6].[2] The comparison of the performance of different multiobjective stochastic search algorithms is then carried out by comparing the EAFs by means of hypothesis tests. This methodology has been used, for example, in [28, 31].

[2] The EAF is seen here as a distribution of the solution quality (set of non-dominated objective vectors obtained at each run) after running an algorithm for a specifi c time or, as it is in our case, until it terminates because of some 'natural" termination criterion.

When comparing the performance of two algorithms A and B, the null hypothesis is stated as follows

$H_0$: Given a specific instance, there is no difference in the performance between A and B in terms of the EAF;

and the alternative as

$H_1$: Given a specific instance, there is a difference in the performance between A and B in terms of the EAF.

Given independence between the runs of both algorithms, a suitable test statistic is the maximum absolute difference between the EAFs, which is analogous to the Kolmogorov-Smirnov test statistic [3]. The decision about acceptance or rejection of $H_0$ can be taken by comparing the test statistic obtained in the original samples with the critical value $C_v$, that is, the $(1 - \alpha)$-quantile of the distribution of the test statistic: $H_0$ is rejected if the former is greater or equal then $C_v$. Usually, $\alpha$ is between $0.01$ and $0.05$.

Since the distribution of the test statistic is unknown, a possible way of computing this bound is to use permutation tests [12], which can be described as follows:

1. label each outcome obtained by each algorithm;
2. compute the test statistic given the original labels;
3. permute the labels and recalculate the test statistic until all permutations considered;
4. take decision on acceptance or rejection of $H_0$ based on the distribution of the test statistic.

If the total number of possible permutations is too large, one can carry out a randomization test, by using only a limited number of random permutations in the above procedure [12].

If more than two algorithms are to be compared, the test statistic is formulated for a problem of $m$ independent samples. The maximum absolute difference among all combinations of pairs of EAFs can be used as analogous to the Birnbaum-Hall test [3]. In the case that a difference is found in terms of EAFs, a multiple comparison method needs to be used in order to detect which groups differ in terms of EAFs, controlling the maximum probability of two or more hypotheses tested in parallel and being rejected incorrectly [19]. Thus, we perform an all-pairwise comparison between all algorithms by applying the same procedure described above for comparison between two algorithms; the $\alpha$-level is then adjusted by the Holm sequential rejective method [16].

The result of this test allows to detect points which have a maximum absolute distance exceeding $C_v$ and to identify regions in the solution quality space, where the algorithms show large different performance. In addition, the sign of the differences can be used to indicate which algorithm performs better in which region.

We also use other measures as the $C$ measure [33] and the $R$ measure [20] to compare the performance of algorithms. The former compares pairs of non-dominated sets by calculating the fraction of each set that is covered by the other set, and the latter evaluates a non-dominated set by the expected value of the weighted Tchebycheff utility function over a set of normalized weight vectors.

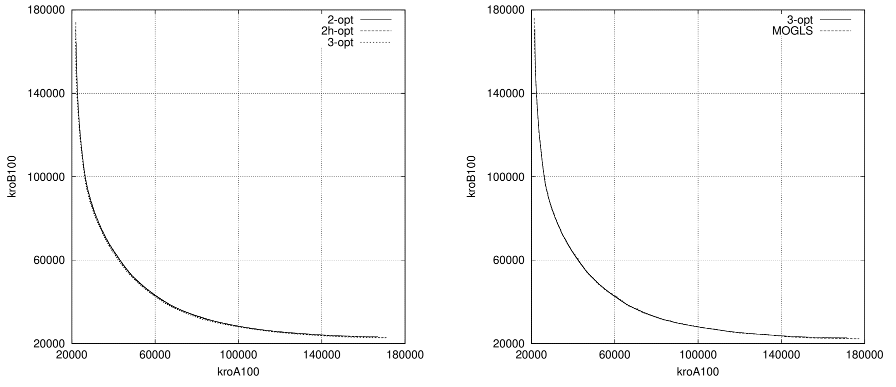


**Fig. 1.** Attainment surfaces at 50% of `2-opt`, `2h-opt` and `3-opt` (left) and `3-opt` and one outcome of MOGLS (right) in the instance kroAB100.

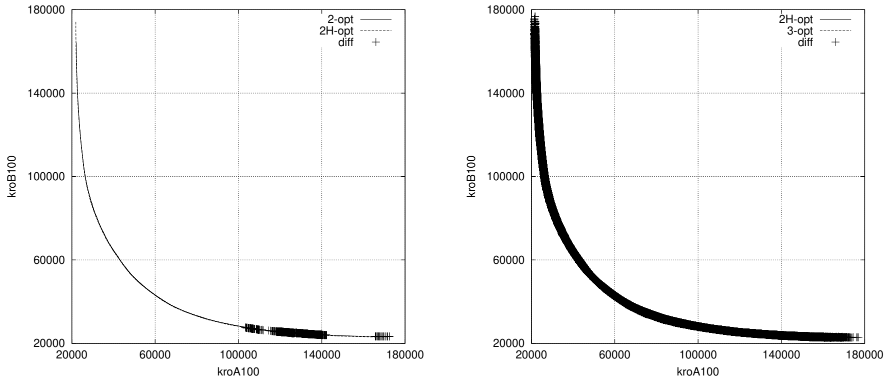
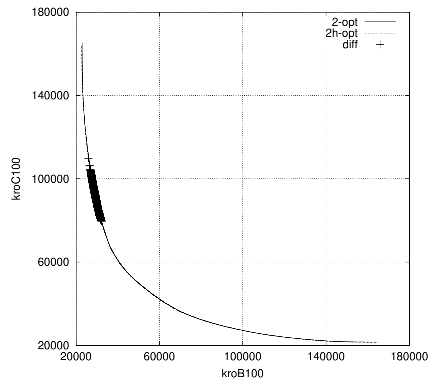


**Fig. 2.** Attainment surfaces at 50% of `2-opt` and `2h-opt` (left) and of `2h-opt` and `3-opt` (right) and the corresponding location of differences in terms of EAFs of more than $C_v$ for instance kroAB100. Locations of these differences are marked by $+$.

## 5 Experimental results

We run extensive experiments with the local search variants using benchmark instances that were derived from single-objective TSPs. In particular, we considered all six paired combinations of the 100-city instances kroA100, kroB100, kroC100 and kroD100 and the paired combination of the benchmark 150-city instances kroA150 and kroB150. In the biobjective case, each tour is then evaluated with respect to the distance matrix of each of the instances in the pair. These benchmark instances were introduced by Hansen [14] and were also used in [2, 20]. For convenience, we denote these instances as kroAB100, kroAC100, kroAD100, kroBC100, kroBD100, kroCD100 and kroAB150. Each algorithm was run 50 times on each of the 7 instances, until a Pareto local optimum set was found.[3]

[3] For further comparisons, the Pareto local optimum sets and the corresponding solutions used in this article are available at `http://www.intellektik.informatik.tu-darmstadt.de/~lpaquete/TSP`



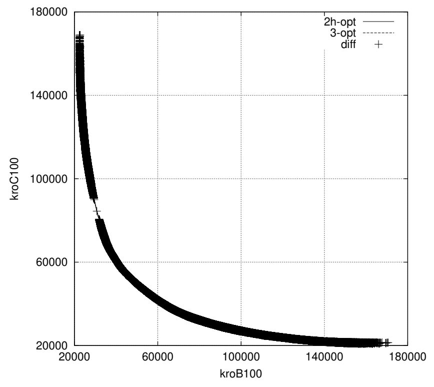
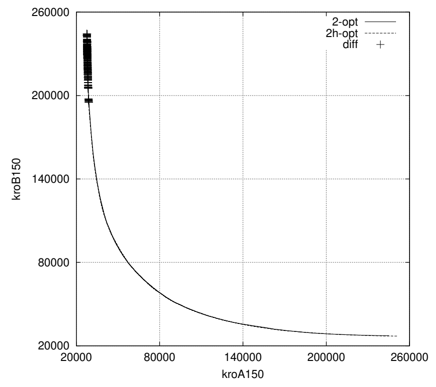


**Fig. 3.** Attainment surfaces at 50% of `2-opt` and `2h-opt` (left) and of `2h-opt` and `3-opt` (right) and the corresponding location of differences in terms of EAFs of more than $C_v$ for instance kroBC100. Locations of these differences are marked by $+$.



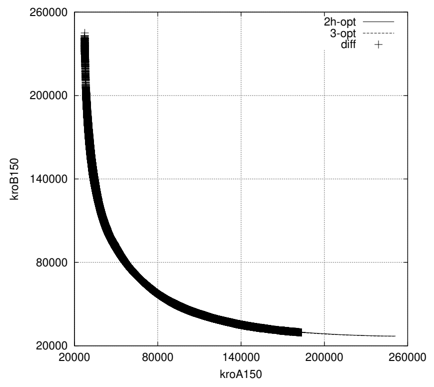


**Fig. 4.** Attainment surfaces at 50% of `2-opt` and `2h-opt` (left) and of `2h-opt` and `3-opt` (right) and the corresponding location of differences in terms of EAF of more than $C_v$ on the instance kroAB150. Locations of these differences are marked by $+$.

The code was written in C++ and tested on a Dual Athlon with 1200 MHz and 512 MB of RAM.

Considering an arbitrary instance $A$, we formulated a null hypothesis that there is no difference in performance between using `2-opt`, `2h-opt` or `3-opt` in terms of EAFs on instance $A$ *versus* the alternative that there is a difference between at least one of them. A significance value of 0.05 was defined *a priori* and a randomization test with 10000 random permutations, as described previously, was conducted for each instance, using the correction of the $\alpha$-level for multiple comparisons. According to these tests, statistical differences were found between the local search algorithms for all biobjective TSP instances. The sign of differences in terms of EAFs on the points with distance larger than $C_v$ indicate a better performance of `3-opt`, followed by `2h-opt` and `2-opt`. In fact, this ordering can be expected when taking into account the performance differences between these local search algorithms for the single-objective TSP [22,30].

Figure 1, left, plots the points whose probability of being attained by the three algorithm is $50\%$ on the instance kroAB100. These points are connected by a line corresponding to the $50\%$ *attainment surface* [10]. In other words, this plot describes the typical median outcome obtained if the algorithms are run several times. However, almost no difference is visible among the three algorithms.

In addition, Figure 1, right plots the $50\%$ attainment surfaces of `3-opt` and an outcome of MOGLS by Jaszkiewicz [20] that reached the highest value of the $R$ measure.[4] MOGLS works, from a high-level perspective as follows:

- it generates an initial population of solutions where each solution is optimized by a local search method based on a random weighted scalarization of the objective functions;
- at each iteration, it chooses randomly two solutions from a temporary smaller population, extracted from the real population according to a random weighted scalarization of the objective function, and recombines them;
- it optimizes the new solution by a local search procedure based on a weighted scalarization of the objective functions and adds it to the population if it is better than the worst solution in the population according to the scalarization considered.

The stopping criteria is based on a pre-defined number of iterations. MOGLS is currently one of the best performing metaheuristics for multiobjective problems. In fact, for the biobjective TSP, MOGLS was able to outperform several multiobjective metaheuristics on the same 100-city instances. However, despite this high performance of MOGLS, in Figure 1 (right) almost no difference among the two algorithms is visible, indicating the good performance of PLS.

Figures 2 to 4 plot, for the instances kroAB100, kroBC100, and kroAB150, respectively, the $50\%$ attainment surfaces with additional points where differences in terms of EAFs are larger than the critical value $C_v$ were found. In each figure, the left plot gives the comparison between `2-opt` and `2h-opt`, while the right plot gives the comparison between `2h-opt` and `3-opt`. In general, the following observations can be made. The highest differences between the outcomes of `2-opt` and `2h-opt` are found on all instances in a relatively small portion of the objective space covered by the two algorithms. On two instances (kroAB100 and kroAB150) these differences appear rather close to the "tail", while on the instance kroBC100 only in some portion of the middle. This clearly indicates that there are some unreachable regions for the `2-opt` algorithm that can be attained by `2h-opt`.

When comparing the `2h-opt` to `3-opt`, the differences larger than $C_v$ are located over most of the objective space. This result indicates that `2h-opt` cannot attain most of the objective space attained by the `3-opt`, meaning a clear better performance of the latter.[5]

[4] The results are available in fi le ND_kroab100_4.txt from `http://www-idss.cs.put.poznan.pl/~jaszkiewicz/motsp/`.

[5] When comparing `2-opt` to `3-opt`, the differences were still a bit larger and the locations of the differences cover almost the whole objective space. Hence, the general conclusions are the same as for the comparison between `2h-opt` and `3-opt`.

**Table 1.** Comparison between `2-opt`, `2h-opt` and `3-opt` in terms of $R$ and $C$ measure on the instances kroAB100, kroAC100, kroAD100, kroBC100, kroBD100 and kroCD100.

| kroAB100 | 2-opt | 2h-opt | 3-opt |
|---|---|---|---|
| ***R* Measure** | | | |
| Min | .9332 | .9334 | .9348 |
| Median | .9340 | .9342 | .9350 |
| Mean | .9339 | .9341 | .9350 |
| Max | .9345 | .9346 | .9352 |
| ***C* Measure** | | | |
| 2-opt | - | 29% | 4% |
| 2h-opt | 50% | - | 8% |
| 3-opt | 81% | 74% | - |

| kroAC100 | 2-opt | 2h-opt | 3-opt |
|---|---|---|---|
| ***R* Measure** | | | |
| Min | .9302 | .9309 | .9320 |
| Median | .9313 | .9315 | .9323 |
| Mean | .9312 | .9314 | .9323 |
| Max | .9317 | .9319 | .9324 |
| ***C* Measure** | | | |
| 2-opt | - | 30% | 5% |
| 2h-opt | 52% | - | 9% |
| 3-opt | 82% | 75% | - |

| kroAD100 | 2-opt | 2h-opt | 3-opt |
|---|---|---|---|
| ***R* Measure** | | | |
| Min | .9329 | .9330 | .9342 |
| Median | .9335 | .9336 | .9344 |
| Mean | .9334 | .9336 | .9344 |
| Max | .9339 | .9339 | .9345 |
| ***C* Measure** | | | |
| 2-opt | - | 30% | 4% |
| 2h-opt | 50% | - | 6% |
| 3-opt | 83% | 78% | - |

| kroBC100 | 2-opt | 2h-opt | 3-opt |
|---|---|---|---|
| ***R* Measure** | | | |
| Min | .9345 | .9348 | .9357 |
| Median | .9350 | .9353 | .9360 |
| Mean | .9351 | .9353 | .9360 |
| Max | .9356 | .9357 | .9361 |
| ***C* Measure** | | | |
| 2-opt | - | 29% | 6% |
| 2h-opt | 53% | - | 12% |
| 3-opt | 82% | 72% | - |

| kroBD100 | 2-opt | 2h-opt | 3-opt |
|---|---|---|---|
| ***R* Measure** | | | |
| Min | .9326 | .9331 | .9342 |
| Median | .9336 | .9337 | .9344 |
| Mean | .9335 | .9337 | .9344 |
| Max | .9340 | .9341 | .9346 |
| ***C* Measure** | | | |
| 2-opt | - | 33% | 7% |
| 2h-opt | 48% | - | 11% |
| 3-opt | 78% | 72% | - |

| kroCD100 | 2-opt | 2h-opt | 3-opt |
|---|---|---|---|
| ***R* Measure** | | | |
| Min | .9371 | .9375 | .9384 |
| Median | .9378 | .9380 | .9388 |
| Mean | .9378 | .9380 | .9388 |
| Max | .9383 | .9385 | .9391 |
| ***C* Measure** | | | |
| 2-opt | - | 30% | 8% |
| 2h-opt | 50% | - | 12% |
| 3-opt | 77% | 69% | - |

| kroAB150 | 2-opt | 2h-opt | 3-opt |
|---|---|---|---|
| ***R* Measure** | | | |
| Min | .9400 | .9403 | .9415 |
| Median | .9407 | .9409 | .9417 |
| Mean | .9407 | .9408 | .9417 |
| Max | .9412 | .9412 | .9418 |
| ***C* Measure** | | | |
| 2-opt | - | 33% | 7% |
| 2h-opt | 48% | - | 11% |
| 3-opt | 78% | 72% | - |

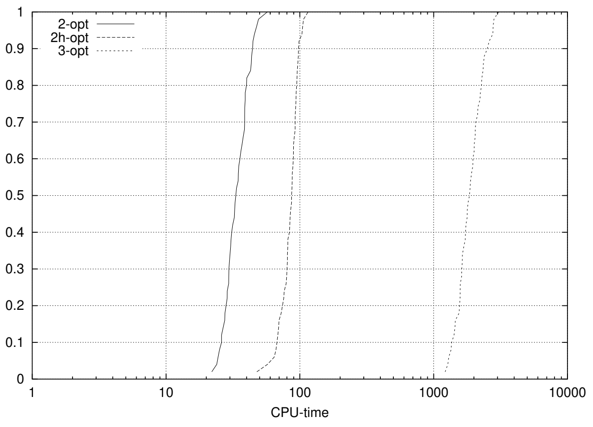


**Fig. 5.** Distribution of CPU-time in seconds of `2-opt`, `2h-opt` and `3-opt` to obtain the Pareto local optimum set on the instance kroAB100.

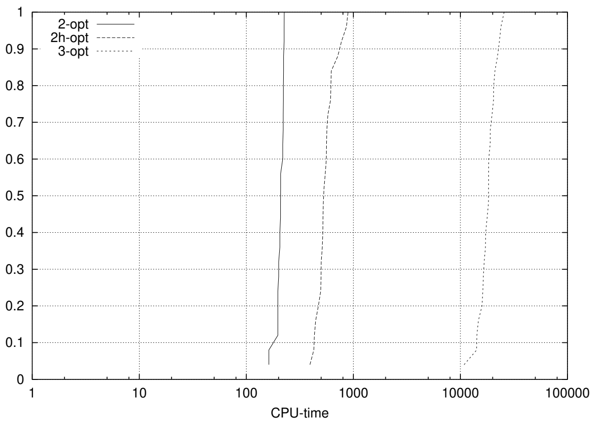


**Fig. 6.** Distribution of CPU-time in seconds of `2-opt`, `2h-opt` and `3-opt` to obtain the Pareto local optimum set on the instance kroAB150.

In addition, we computed the $R$ measure [15] for all algorithms on all instances based on the the code by Jaszkiewicz [20].[6] The metric parameters were defined according to [20], and only in the case of the 150-city instance we used 280,000 as the worst attainable objective function value for both objectives. Table 1 presents the minimum, median, mean and maximum values of $R$ measure. It also gives the $C$ measure for each pair of algorithms and for each instance; each table entry is averaged over all pairwise comparisons between pairs of runs of two algorithms. The entries indicate how

[6] The code was downloaded from `http://www-idss.cs.put.poznan.pl/~jaszkiewicz/mokp` and adapted according to [20].

**Table 2.** Comparison of the `3-opt` PLS algorithm with MOGLS in terms of $R$ and $C$ measure on the instances kroAB100, kroAC100, kroAD100, kroBC100, kroBD100 and kroCD100.

| | kroAB100 | kroAC100 | kroAD100 | kroBC100 | kroBD100 | kroCD100 |
|---|---|---|---|---|---|---|
| **$R$ Measure** | | | | | | |
| `2-opt` (median) | .9340 | .9313 | .9335 | .9350 | .9336 | .9378 |
| `2h-opt` (median) | .9342 | .9315 | .9336 | .9353 | .9337 | .9380 |
| `3-opt` (median) | .9350 | **.9323** | **.9344** | **.9360** | **.9344** | .9388 |
| MOGLS | **.9351** | .9321 | .9342 | .9359 | **.9344** | **.9389** |
| **$C$ Measure** | | | | | | |
| `3-opt` covers MOGLS | **40%** | **47%** | **59%** | **36%** | **33%** | **39%** |
| MOGLS covers `3-opt` | 13% | 13% | 12% | 16% | 15% | 15% |

much the outcome of an algorithm in the row covers the outcome of an algorithm in the column.

The results shown in Table 1 clearly indicate the superior performance of `3-opt`: The minimum $R$ value for `3-opt` is always larger than the maximum values for `2-opt` and `2h-opt`; the only exception is instance kroCD100, where the maximum value for `2h-opt` is slightly larger than the minimum value for `3-opt`. The difference of performance between the `2-opt` and `2h-opt` is much smaller, however, still indicating a better performance for `2h-opt` over `2-opt` on some of the instances. The comparisons based on the $C$ measure lead to the same conclusions.

Figures 5 and 6 show the distribution of the computation time of `2-opt`, `2h-opt` and `3-opt` for finding a Pareto local optimum set on the instances kroAB100 and kroAB150. `2-opt` takes roughly half the time of `2h-opt` and only around $1\%$ of the time necessary to run `3-opt`. Obviously, these differences are due to the size of the neighborhood searched by the respective algorithms. In fact, `3-opt` is the slowest to get a Pareto local optimum set, but gives a higher quality of solution because of the larger neighborhood searched. Clearly, `2-opt` is the fastest but also gives the worst quality solutions.

From our point of view, PLS can be extremely useful as a reference for metaheuristic approaches to multiobjective combinatorial optimization problems. This use is especially important, because only few results on multiobjective combinatorial problems are available from the literature. Additional advantages of PLS are that it is a very simple algorithm and that it has a "natural" termination condition and, therefore, the results are more reproducible.

The question to be posed, when using PLS for a baseline comparison is "*Can algorithm A outperform the Pareto local optimum set in terms of solution quality and run-time?*". This question should be posed for any metaheuristic approach. We posed the same question on the results obtained by MOGLS in [20] which is one of the best performing algorithms for the biobjective TSP. Figure 1 already presented some comparisons between the $50\%$ EAFs of `2-opt`, `2h-opt` and `3-opt` with the best available outcome of MOGLS regarding the $R$ measure. Figure 1 showed that no clear visual differences are found. Since also insufficient results of MOGLS were available publically, the attainment function methodology could not be used. Instead, we reported the comparison in terms of the median of the $R$ measure and the $C$ measure in Table 2.

As a result, we observe that `2-opt` and `2h-opt` produce Pareto local optimum sets inferior to the outcome of MOGLS. However, the Pareto local optima sets of `3-opt` are comparable to MOGLS, and seems to be even better on half of the instances. The exceptions are on instances kroAB100 and kroCD100. The $C$ measure was computed only between `3-opt` and MOGLS. The results show that `3-opt` is able to cover more solutions of MOGLS than vice versa. Despite these unfavorable results for MOGLS in terms of solution quality, the computation time reported here for `3-opt` appears to be much higher than [20] (MOGLS took 88.3 seconds averaged on all instances on a Pentium with a 350 MHz processor).

## 6 Characterization of the Pareto local optimum sets

Despite its long computation time, `3-opt` seems to obtain a Pareto local optimum set which is comparable to high performance algorithms such as MOGLS. This fact allows us to suppose that there is a path in the search space which guides the local search to near-optimal solutions by means of `3-opt` exchanges and that these solutions are somehow near each other. In fact, Borges and Hansen [2] showed empirically a concentration of both, supported optimal and near-optimal solutions, in the search space of several multiobjective TSP instances with 3 objectives. One interesting question to explore is how the solutions in the Pareto local optimum set obtained by `3-opt` are distributed in the search space, hoping that an answer to this question can give us further insights to solve this kind of problems even more efficiently.

The Pareto local optimum set $H$ can be represented as a complete, undirected and weighted graph, where each node corresponds to a solution and each weight corresponds to the distance between each pair of solutions. In the TSP case, we use the number of different edges as the distance between two tours $i$ and $j$ which is proportional to the number of edge-exchanges one has to apply to $i$ to transform it into $j$. We are particularly interested in identifying clusters of solutions, where all solutions of a cluster are reachable by using a $d$-opt algorithm as it was used here, *i.e.*, the minimum distance of a solution to at least one solution of the cluster is at most $d$. These clusters define a subgraph $H' \subseteq H$ which contains only edges with weights below or equal to a fixed value $d$ and the nodes connected by these edges. A cluster of distance $d$ is a maximally connected component of $H'$.

In our analysis, we enumerate all clusters with $d = \{2, 3, 4\}$ for the Pareto local optimum sets from the experiments with `3-opt` for the 100-city instances. Figure 7 gives the box-plots for the total number of solutions (plot a), the number of clusters with $d = 2$ (plot b), the number of clusters with $d = 3$ (plot c), and the number of clusters with $d = 4$ (plot d) for all the instances tested. The number of clusters with $d = 2$ is approximately 10% of the total number of solutions. With $d = 3$, the number of clusters drops down to one fifth of the number of clusters with $d = 2$. Finally, the number of clusters is further halved with $d = 4$. The percentage of the total number of solutions inside of clusters with $d = 2$ was approximately 90% and it reached almost

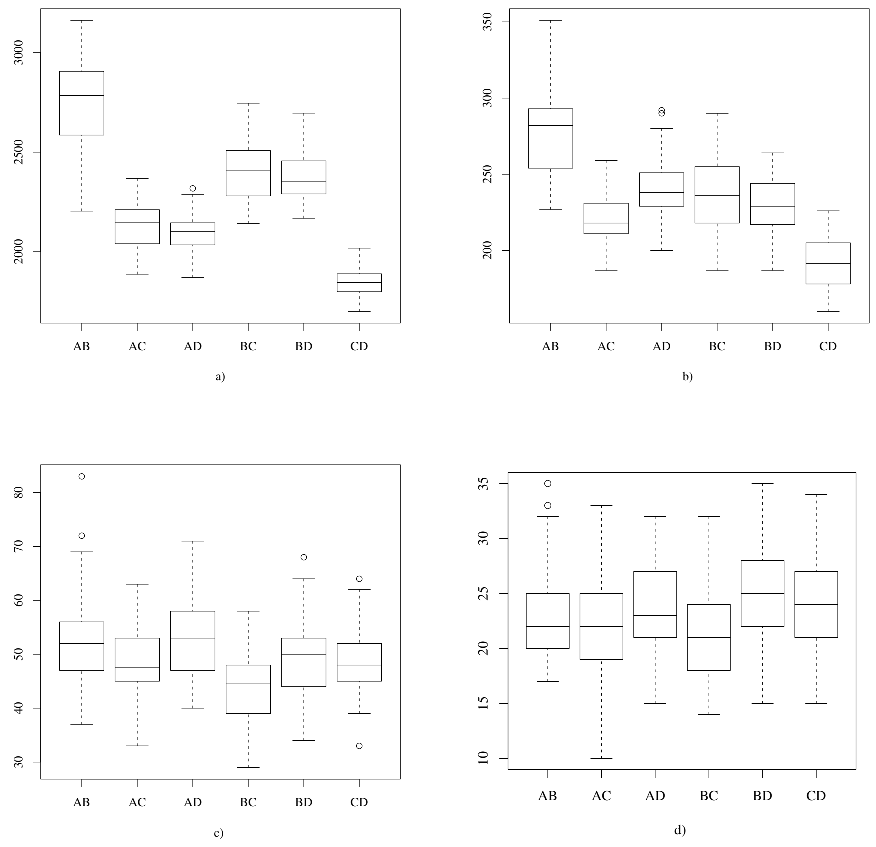


**Fig. 7.** Boxplots for the total number of solutions (a), number of clusters with $d = 2$ (b), number of clusters with $d = 3$ (c), number of clusters with $d = 4$ (d) for the instances kroAB100 (AB), kroAC100 (AC), kroAD100 (AD), kroBC100 (BC), kroBD100 (BD) and kroCD100 (CD) using the `3-opt` algorithm.

100% with $d = 4$. These results clearly indicate that near-optimal solutions are strongly clustered in the search space.

In a second analysis, we examined how these results are affected by the instance size. Thus, we applied `3-opt` to 50 pairs of randomly generated instances with Euclidean distances with 30 to 100 cities, using a step size of 10 cities.[7] We ran `3-opt` once for each instance until a Pareto local optimum set was found and then carried out a regression analysis to study the effect of some of its features, such as the total number

[7] The code for generating random instances was downloaded from the site of the 8th DIMACS Implementation Challenge for the TSP at `http://www.research.att.com/~dsj/chtsp/`

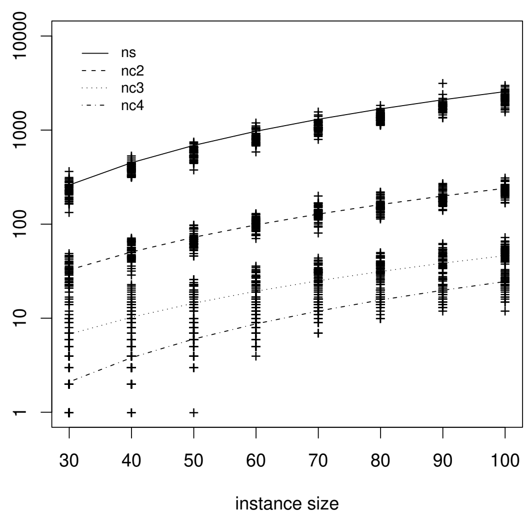


**Fig. 8.** Number of solutions $(ns)$, number of clusters with $d = 2$ $(nc2)$, $d = 3$ $(nc3)$ and $d = 4$ $(nc4)$ (in logarithmic scale) against size of the instance and corresponding least-square regression fi ts.

of solutions and number of clusters and its growth with instance size. In all the regression models, the constant variance was scattered symmetrically and vertically around 0 and the residuals followed a normal distribution. This means that the regression lines provide a fairly good description of the data. We also remark here that more than $90\%$ of the solutions in the Pareto local optimum sets were present in the clusters, as in the previous experiment.

Figure 8 gives the number of solutions $(ns)$, the number of clusters with $d = 2$ $(nc2)$, $d = 3$ $(nc3)$ and $d = 4$ $(nc4)$ against instance size $(n)$ and the least-square regression fit. For the number of solutions against instance size we performed a least-square regression fit by applying a logarithmic transformation to both variables. Translating back to the linear scale we observe that the number of solutions grows as $n^{1.9}e^{-0.9}$, that is, almost quadratically. Therefore, with increasing instance size one cannot expect to obtain the complete Pareto local optimum sets by means of a `3-opt` in reasonable time, which also explains the strong growth in computation time of the three algorithms from 100 cities to 150 cities (see Figures 5 and 6). This can be also expected, because of the intractability results for these problems [8].

We also fitted the number of clusters against instance size. For the three different cluster sizes $d = \{2, 3, 4\}$ we observed growth rates of the number of clusters as $(0.14n + 1.51)^2$, $(0.06n + 0.8)^2$ and $(0.05n - 0.05)^2$, respectively. Although the number of clusters for all the values of $d$ increase quadratically, the increase rate is very low. Thus, we would not be surprised if the relationship between the number of clusters

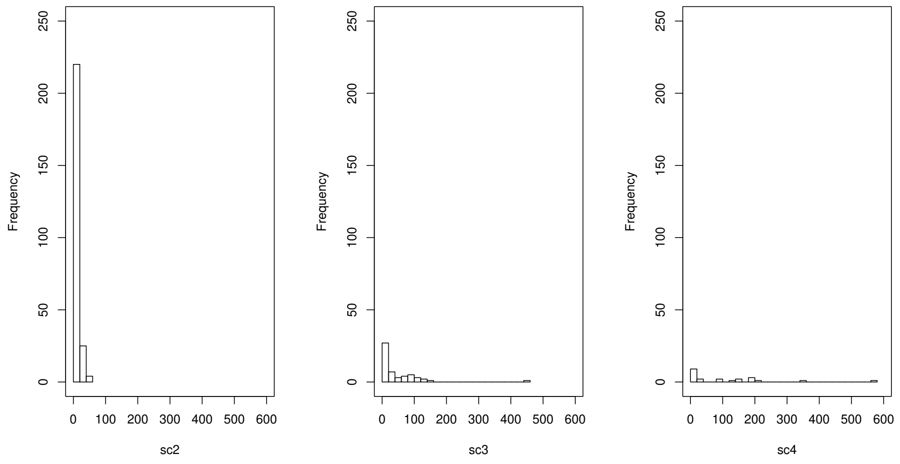


**Fig. 9.** Histogram of the number of solutions in the clusters with $d = 2$ $(sc2)$ (left), with $d = 3$ $(sc3)$ (middle) and with $d = 4$ $(sc4)$ (right).

and instance size for larger instances could be expressed by a linear model. However, a more detailed analysis would require to consider larger instances, involving very large computation times.

Figure 9 shows histograms of the number of solutions in the clusters with $d = \{2, 3, 4\}$ of a Pareto local optimum set for a 100-city instance solved by `3-opt`. The plots indicate the existence of a large number of small clusters with less than 75 solutions each for $d = 2$, while for $d = \{3, 4\}$ there are some clusters with more than 200 solutions. This is clearly an effect of the increasing $d$, although the difference of the distributions when going from $d = 2$ to $d = 3$ appears to be much larger than when going from $d = 3$ to $d = 4$. This could mean that most of the small clusters with $d = 2$ are near each other, since they are merged in larger clusters with $d = \{3, 4\}$. We also computed how large should be $d$ $(d_{max})$ such that all complete Pareto local optimum set forms only one cluster. We observed growth rates of $d_{max}$ as $(0.06n + 2.5)^2$. This means that there are some few solutions which are not easy reachable by a small value of $d$.

These empirical results corroborate that, besides the fact that already the multiobjective TSP is $\mathcal{NP}$-hard, it also presents the well-known problem of a strongly increasing number of solutions, adding an additional factor to the difficulty of solving. This fact especially becomes a problem to a local search algorithm like PLS dealing with very large instances, because it has to maintain a large set of solutions in the archive. To alleviate this problem, one may introduce more complex strategies to reduce computation time. One obvious alternative for PLS could be to restrict the number of solutions in the archive. Another possibility is to use a different acceptance criterion, possibly by including some distance measure in the acceptance criterion, which only accepts a reduced set of solutions. In any case, these additional strategies need to ensure that a good distribution of the solutions in the objective space can be obtained. However, they all have the drawback of adding more complexity to PLS and it becomes not clear what is the main contribution of the local search.

How can these insights be exploited to design more powerful local search algorithms for the biobjective TSP? What general conclusions can we take from the insights of our analysis? In general, the results presented here show that for the biobjective TSP we must expect that most of the Pareto optimal or near-optimal solutions are near each other. These results are further backed up by those of Borges and Hansen [2]. These observations give also an explanation why such simple local search algorithms as the ones presented here are able to give such high solution quality. In particular, given one single solution of a cluster, PLS has the potential to find all other solutions in the cluster.

The main disadvantage of PLS, especially the `3-opt` version, was the high computation time. To reduce computation time, one possibility is to use the following procedure:

1. solve the weighted scalarization of the objectives in order to hopefully obtain seed points for a cluster;
2. scan the neighborhood of the solution found using the acceptance criterion of PLS to obtain solutions inside the cluster;
3. jump to another cluster by changing slightly the weights of the scalarization of the objectives.

In fact, there are effective algorithms available, such as iterated local search [22, 27], for solving the single-objective problem resulting from the weighted scalarization. For example, optimal solutions for the single-objective instance kroA100 are typically found in less than 0.1 seconds. In fact, results already available for such an approach [29] suggest that comparable solution quality to `3-opt` can be found in a computation time that is several orders of magnitude smaller than that of `3-opt`.

## 7 Conclusions and further work

We presented an investigation of Pareto local search algorithm for the biobjective TSP. PLS maintains a high level of simplicity, because (i) no ideal points need to be computed, (ii) no aggregation of objectives is required, and (iii) no parameter settings are necessary for the algorithm. Despite this simplicity, experimental results show that especially the `3-opt` version of PLS returns very good solutions in terms of solution quality, comparable to those of the currently best performing metaheuristics for the biobjective TSP. However, a drawback of PLS is that the computation times are much higher than that of state-of-the-art metaheuristics.

One of the main uses of PLS is certainly that it can serve as a baseline benchmark algorithm for metaheuristic approaches to multiobjective combinatorial optimization problems. In this case, the interesting question to be answered is whether the metaheuristic is able to outperform PLS from a solution quality and from a computation time point of view, *i.e.*, whether it *dominates* PLS. A further conclusion of our research is that the ranking of the three PLS algorithms we tested here is the same as for the single-objective case. `2-opt` is the fastest but returns the worst solution quality, while `3-opt` takes the longest computation time but returns the best quality solutions. It would be intersting to use well-known speed up techniques as nearest neighbors list [21] and *don't look bits*[1] in order to reduce computation time of the latter. in addition, one could also adapt variable-depth search algorithms like the Lin-Kernighan heuristic [26] to the multiobjective TSP, since these algorithms are known to perform even better than `3-opt` for the single-objective case. However, the computation time of such an algorithm may become infeasibly large.

An analysis of the Pareto local optimum sets returned by `3-opt` for randomly generated biobjective TSP instances showed empirical evidence that (i) the number of solutions grows quadratically with instance size, (ii) the solutions are strongly clustered and (iii) the number of clusters of solutions with a specific, maximum inter-tour distance, grows quadratically with instance size, but with a very small rate. On one side, the observation (i) suggests that computation times of PLS for large instances may become extremely large. Hence, one conclusion is that for large instances, strategies that strongly limit the number of solutions returned, while keeping at the same time a good spread of solutions, are required. The observations that solutions are strongly clustered and the small growth rate of the number of clusters with instance size indicate that these solutions are not very much spread in the search space. This suggests that local search methods in which jumps between clusters and the in depth exploration of clusters are intertwined, are promising candidates for better performing local search algorithms.

Our study can be extended in several ways. One first possibility is to consider more objectives in PLS. However, we believe that it will incur further increase of computation times due to the expected size of the Pareto local optimum set. Therefore, it should be interesting to study the rate of acceptance of solutions to the archive of the PLS according to the number of objectives. The analysis of the set of solutions can also be extended to other types of instances like clustered TSP instances (here clustering refers to the clustering of city locations in the instance) or instances with different degrees of correlation for the cost matrixes.

**Acknowledgments** We would like to thank Carlos Fonseca and Viviane Grunert da Fonseca for valuable discussions on the topic of this research. We also thank Andrej Jaszkiewicz and Joshua Knowles for discussions on the evaluation methods, and Irina Dumistrescu and Tommaso Schiavinotto for the comments on a draft version of this paper. This work was supported by the Metaheuristics Network, a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

## References

1. J.L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

2. P. C. Borges and P. H. Hansen. A study of global convexity for a multiple objective travelling salesman problem. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 129–150. Kluwer, 2000.
3. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, 1980.
4. G.A. Croes. A method for solving traveling salesman problems. *Operations Research*, 6:791–812, 1958.
5. P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing - a metaheuristic technique for multiple objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.
6. V. G. da Fonseca, C. Fonseca, and A. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization (EMO 2001)*, Lecture Notes in Computer Science 1993, pages 213–225. Springer Verlag, 2001.
7. M. Ehrgott. Approximation algorithms for combinatorial multicriteria problems. *International Transactions in Operations Research*, 7:5–31, 2000.
8. V.A. Emelichev and V.A. Perepelitsa. On the cardinality of the set of alternatives in discrete many-criterion problems. *Discrete Mathematics and Applications*, 2(5):461–471, 1992.
9. M.M. Flood. The travelling salesman problem. *Operations Research*, 4:61–75, 1956.
10. C. Fonseca and P. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In W. Ebeling, I. Rechenberg, H.-P. Schwefel, and H.-M. Voigt, editors, *Proceedings of the 4th Conference on Parallel Problem Solving from Nature (PPSN IV)*, Lecture Notes in Computer Science 1141, pages 584–593. Springer Verlag, 1996.
11. X. Gandibleux and A. Freville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: the two objectives case. *Journal of Heuristics*, 6:361–383, 2000.
12. P. Good. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer-Verlag, Berlin, Germany, 1994.
13. M.P. Hansen. Tabu search for multiobjective optimization: MOTS. In *Proceedings of MCDM'97*, Cape Town, South Africa, January 1997.
14. M.P. Hansen. Use of subsitute scalarizing functions to guide a local search base heuristics: the case of moTSP. *Journal of Heuristics*, 6:419–431, 2000.
15. M.P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby , Denmark, 1998.
16. S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.
17. J.N. Hooker. Needed: an empirical science of algorithms. *Operations Research*, 42:201–212, 1994.
18. H.H. Hoos and T. Stützle. Evaluating Las Vegas algorithms — pitfalls and remedies. In G. F. Cooper and S. Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 238–245. Morgan Kaufmann, San Francisco,USA, 1998.
19. J. Hsu. *Multiple Comparisons - Theory and Methods*. Chapman & Hall/CR, 1996.
20. A. Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 1(137):50–71, 2002.
21. D. S. Johnson and L. A. McGeoch. The travelling salesman problem: A case study in local optimization. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, Chichester, UK, 1997.
22. D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer Academic Publishers, 2002.

23. J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation (CEC 99)*, pages 98–105, 1999.
24. M. Laumanns, L. Thiele, E. Zitzler, E. Welsi, and K. Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In J.J. Merelo Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, editors, *Proceedings of the 7th Conference on Parallel Problem Solving from Nature (PPSN VI)*, Lecture Notes in Computer Science 2439, pages 44–53. Springer Verlag, 2002.
25. S. Lin. Computer solutions for the traveling salesman problem. *Bell Systems Technology Journal*, 44:2245–2269, 1965.
26. S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.
27. H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, 2002.
28. L. Paquete and C. Fonseca. A study of examination timetabling with multiobjective evolutionary algorithms. In *Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, pages 149–154, Porto, 2001.
29. L. Paquete and T. Stützle. A two-phase local search for the biobjective traveling salesman problem. In C. M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-criterion Optimization (EMO 2003)*, Lecture Notes in Coputer Science 2632. Springer Verlag, 2003. 479–493.
30. G. Reinelt. *The Traveling Salesman Problem: Computational Solutions for TSP applications*. Lectures Notes in Computer Science 840. Springer Verlag, 1994.
31. K. Shaw, C. Fonseca, A. Nortcliffe, M. Thompson, J. Love, and P. Fleming. Assessing the performance of multiobjetive genetic algorithms for optimization of a batch process scheduling problem. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation (CEC 99)*, pages 37–45, 1999.
32. E. Talbi, M. Rahoual, M. Mabed, and C. Dhaenens. A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization (EMO 2001)*, Lecture Notes in Computer Science 1993, pages 416–428. Springer Verlag, 2001.
33. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 4(3):257–271, 1999.