Université Libre de Bruxelles

*Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle*

On the Performance of Local Search for the Biobjective Traveling Salesman Problem

Luis PAQUETE and Thomas STÜTZLE

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2009-005

March 2009

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle*
UNIVERSITÉ LIBRE DE BRUXELLES
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2009-005

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsibility for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

On the Performance of Local Search for the Biobjective Traveling Salesman Problem

Luís Paquete¹ and Thomas Stützle²

¹ Luís Paquete, CISUC, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal, paquete@dei.uc.pt,

² Thomas Stützle, IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium, stuetzle@ulb.ac.be

Summary. In this chapter we investigate experimentally the performance of multi-objective local search approaches that are based on the component-wise acceptance criterion search model. This model gives a framework for many well-known evolutionary and local search algorithms. Using the biobjective traveling salesman problem as an example application, we analyse the impact of three important algorithmic components on the performance of a simple local search algorithm that follows this search model: initialization strategy, neighborhood structure and archive bounding. By following principles of experimental design, we study the effects of each component, both in terms of solution quality and computation time. The experimental analysis indicates the existence of several complex trade-offs between solution quality and run-time for many of the choices available for each component.

1 Introduction

The combination of local search principles with evolutionary techniques is among the most successful approaches for tackling multiobjective combinatorial optimization problems (MCOPs). One straightforward possibility of combining these paradigms is to base the acceptance criterion of a (perturbative) local search on the *component-wise ordering* of the corresponding objective function value vectors; in other words, neighboring solutions that are non-dominated with respect to the current solution are used to update an archive of solutions. Such a type of algorithm would be a direct adaptation of the *component-wise acceptance criterion* (CWAC) search model [27] into an iterative improvement algorithm [26]; the CWAC model itself was proposed to capture the common underlying principles among several well-known approaches in the literature, ranging from simple iterative improvement [2, 20, 24] to tabu search [3, 4] and evolutionary algorithms [16, 17, 18, 23, 33].

In this chapter, we present an experimental analysis of iterative improvement algorithms for MCOPs that follow the CWAC search model. Our analysis investigates the impact that some typical algorithmic components of CWAC algorithms have on the overall performance. In particular, we investigate the initialization strategy, the neighborhood structure, and the usage of archive bounding techniques. This study

is done using the biobjective traveling salesman problem (BTSP) as a paradigmatic example of an MCOP.

We use a sound experimental design for this analysis: algorithmic components are seen as *factors*, that is, as abstract characteristics of an algorithm that can affect response variables such as solution quality and computation time. The experimental results of the various factor combinations are carefully examined using a sound methodology for the assessment of the performance of algorithms for multiobjective problems, which is based on outperformance relations and attainment functions.

The careful experimental design and the advanced methods for the experimental analysis allow us to arrive at statistically sound conclusions on the importance of the algorithmic components and their mutual interdependencies. In fact, the experimental methodology we employ has already proved to give interesting insights into algorithm performance in our earlier research, which has focused on the analysis of algorithms that use different scalarizations of the multiple objectives [29, 30]. The main new contribution of this chapter is the analysis of local search algorithms following the CWAC search model and their behavior in dependence of the underlying algorithmic components.

The article is structured as follows. In Section 2, we introduce basic notions on MCOPs and the BTSP. Section 3 introduces the CWAC model and explains the particular components studied in our experiments. Next, in Section 4, we give an overview of the experimental design and the performance assessment methodology; we then also describe the experimental results obtained. We end with some concluding remarks in Section 5.

2 Multiobjective combinatorial optimization

We tackle MCOPs in terms of Pareto optimality. Solving such problems involves determining (all) solutions that are not dominated by any other solution. In MCOPs, each solution $s \in S$ has an associated objective function vector, which is given by $f : s \mapsto \mathbb{R}^Q$; here S is the set of all feasible solutions and Q is the number of objectives. For two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^Q$, the following order is defined: the *component-wise order*, $\mathbf{u} \leq \mathbf{v}$, indicates that $\mathbf{u} \neq \mathbf{v}$ and $u_i \leq v_i$, $i = 1, \dots, Q$. We say that (i) $f(s)$ *dominates* $f(s')$ if $f(s) \leq f(s')$; (ii) $f(s)$ and $f(s')$ are *non-dominated* if $f(s) \not\leq f(s')$ and $f(s') \not\leq f(s)$. The *weak component-wise order* $\mathbf{u} \leq \mathbf{v}$ says that $u_i \leq v_i$, $i = 1, \dots, Q$; based on this order we say that $f(s)$ *weakly-dominates* $f(s')$ if $f(s) \leq f(s')$. Occasionally, we use the same wording and notation for solutions if for their objective function vectors these relations hold.

If there is no $s' \in S$ with $f(s') \leq f(s)$, then $s \in S$ is said to be *Pareto optimal*. The *Pareto optimal set* is the set $S' \subseteq S$ in which *only* and *all* Pareto optimal solutions are contained. The *efficient set* is the image of the Pareto optimal set in the objective space.

In this study, we tackle the BTSP. It is defined by a set C of n nodes and a distance vector $d(c_i, c_j) \in \mathbb{N}^2$ for each pair of nodes $c_i, c_j \in C$; the goal is to find a permutation $\pi : [1..n] \rightarrow [1..n]$ of the node indices such that the length of the tour, that is,

$$f(\pi) = d(c_{\pi(n)}, c_{\pi(1)}) + \sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)})$$

Algorithm 1 Pareto Local Search.

```

1: Input:  $(\mathcal{N}, s)$ 
2:  $V_F = \{s\}$ ,  $V_T = \emptyset$ 
3: repeat
4:   choose  $s \in V_F$ 
5:    $V_T = V_T \cup \{s\}$ 
6:    $W = \emptyset$ 
7:   for all  $s' \in \mathcal{N}(s)$  do
8:     if  $(\nexists s'' \in (W \cup V_T \cup V_F) : f(s'') \leq f(s'))$  then
9:        $W = W \cup \{s'\}$ 
10:    end if
11:  end for
12:   $V_T = \{s' \mid s' \in V_T, \nexists s'' \in W : f(s'') \leq f(s')\}$ 
13:   $V_F = (V_F \setminus \{s\}) \cup W$ 
14:   $V_F = \{s' \mid s' \in V_F, \nexists s'' \in W : f(s'') \leq f(s')\}$ 
15: until  $V_F = \emptyset$ 
16: return  $V_T$ 

```

is “minimal” in terms of Pareto optimality. The BTSP is NP-hard [31] and it has been tackled in a number of research efforts [1, 2, 5, 7, 12, 15, 22].

3 The CWAC search model and components

The CWAC search model was recently proposed to stress the common features of algorithms for tackling MCOPs that base acceptance decisions of solutions on the component-wise ordering of the objective function value vectors [27]. A simple local search algorithm following the CWAC search model would accept neighboring solutions that are non-dominated with respect to the current solution and use these to update an archive of non-dominated solutions. Interestingly, variations of this basic idea have only recently been studied; examples include PAES [17], PLS [24], BLS [2], SEMO and FEMO [20].

Our experimental analysis will be based on PLS (short for Pareto Local Search), mainly because of its conceptual simplicity. The essential idea of PLS is to keep an archive of solutions, pick at each iteration a still unexamined solution s in the archive and explore its neighborhood $\mathcal{N}(s)$. All neighbors of s that are non-dominated with respect to s are used to update the archive. Updating the archive consists in adding new non-dominated solutions and removing all solutions of the archive that are dominated by any of the new solutions. Once the neighborhood of a solution is evaluated, the latter gets marked as visited. PLS continues until all solutions in the archive are marked as visited.

More formally, PLS can be described as follows (a pseudo-code is given in Algorithm 1 and follows [26].) PLS splits the solutions in the archive into two subsets V_T and V_F , where V_T is the subset of solutions in the archive for which their complete neighborhood was already examined (such solutions are marked as *visited*), and V_F contains the remaining solutions in the archive (solutions marked as *not visited*). Initially, PLS starts with a single solution s , which is marked as not visited (that

is, it is included in V_F); V_T is initialized as an empty set. At each repetition of the repeat-until loop (lines 3-15), PLS evaluates all neighbors of one solution $s \in V_F$. For each $s' \in \mathcal{N}(s)$, it checks whether there is another solution in $\mathcal{N}(s) \cup V_F \cup V_T$, which weakly dominates it (line 8); if this is not the case, s' is added to set W (line 9) and it will be chosen in a following iteration if it is still non-dominated with respect to $\mathcal{N}(s) \cup V_F \cup V_T$ or with respect to V_T (line 12). Note that PLS removes dominated solutions of the sets V_T and V_F at each iteration (in steps 12 and 14, respectively) and that these sets contain only solutions that are non-dominated with respect to any solution in $V_F \cup V_T$. This basic algorithm has some relevant properties: it does not cycle and naturally stops in a *Pareto local optimum set* [26].

We consider the following algorithmic components of PLS: neighborhood structure, initialization strategy, and archive bounding technique. We will discuss these components in the following sections.

Neighborhood

Typical neighborhoods for the traveling salesman problem (TSP), as well as for its multiobjective version, are 2- and 3-exchange. Two feasible solutions s and s' are 2-exchange (3-exchange) neighbors if they can be transformed into each other by replacing one pair (triple) of edges by another pair (triple) of edges. Note that the evaluation of a neighboring solution for the q th objective vector component can easily be done by adding and subtracting two distances in the 2-exchange case (three distances in the 3-exchange case). Note that in the 2-exchange case, the pair of edges added is uniquely determined; however, in the 3-exchange case four ways of reconnecting the resulting three tour segments are possible so that the three edges deleted and added are all distinct. In addition, when checking the 3-exchange moves, we also check for the corresponding 2-exchange moves.

Initialization strategy

We distinguish two strategies for generating the initial solution: one that starts from a randomly generated solution (**Rand**) and another that starts from a very high quality solution with respect to one objective (**2phase**). Figure 1 illustrates how these two search strategies *ideally* work. Roughly speaking, the **Rand** strategy moves *towards* the efficient set, whereas the **2phase** strategy moves in *parallel* to it. Their performance would obviously depend on how *close* are solutions under a given neighborhood structure [28]. In a similar way, the effectiveness of the **Rand** search strategy depends on the connectedness between randomly generated solutions and the Pareto optimal set.

The initial solution for the **2phase** strategy is generated by an Iterated Local Search (ILS) algorithm that was also used by Stützle and Hoos [32]. In particular, we generated the initial solution using an iterative improvement algorithm based on the 3-exchange neighborhood and applying 50 perturbations in the ILS algorithm; this was enough for obtaining a very good quality solution for the single-objective case. A version of this ILS algorithm is available on the web site www.sls-book.net.

Fig. 1. Illustration of the two strategies.

Archive bounding technique

In the worst case, if all feasible solutions of an MCOP are incomparable, PLS would take exponential time to terminate. Although such an extreme case may hardly be found in real-life situations, the number of solutions can still grow very large to make the algorithm become useless in practice. To avoid a too strong growth of the archive size, some *bounding* on the number of solutions is often used.

For PLS, we use the bounding technique proposed by Angel *et al.* [2] and Laumanns *et al.* [19]. It divides the objective space by a grid, where the intervals are defined according to a geometric sequence. Given a lower bound l_q and an ϵ_q , both associated to the q th objective, the intervals are defined as $[l_q, l_q(1+\epsilon_q)]$, $[l_q(1+\epsilon_q), l_q(1+\epsilon_q)^2]$, ..., $[l_q(1+\epsilon_q)^{k-1}, l_q(1+\epsilon_q)^k]$. In each of the resulting hypercubes, we only allow one solution to be maintained. Following the guidelines given by Angel *et al.* [2], the lower bound l is given by a heuristic procedure. Additionally, we use a same value of ϵ (we call it ϵ -value) for all objectives and we analyze the impact of different possible values for ϵ . For the q -th objective, the hypercube to which a solution s is assigned is defined by a vector $\mathbf{a}(s) = (a_1(s), \dots, a_Q(s))$, whose q -th component is defined as

$$a_q(s) = \left\lfloor \frac{\log\left(\frac{f_q(s)}{l_q}\right)}{\log(1 + \epsilon_q)} \right\rfloor.$$

In this bounding technique, at most one solution at a time may occupy a given hypercube. No other solution is allowed to *enter* into the hypercube unless it dominates the solution inside, in which case the latter is replaced. If a solution in a hypercube becomes dominated by another solution in a different hypercube, it is removed (directly following the rules of the PLS algorithm). Unfortunately, the maximum number of solutions that can be accepted to the archive is unknown and some tuning is needed in order to obtain a reasonable approximation to a desired

number of solutions. In general, the smaller the value of ϵ , the smaller will be the size of the hypercubes and, therefore, the larger the number of hypercubes. Hence, a smaller ϵ -value will lead to potentially more solutions in the archive and therefore also to higher computation times. On the contrary, the effect of large values of ϵ is that the hypercubes also will become larger; ultimately this may lead to severe restrictions of the search (possibly resulting in poor solution quality) but also to strongly reduced computation times. Unfortunately, it is *a priori* not clear how to choose ϵ to obtain a good trade-off between solution quality and computation time.

4 Experimental analysis

4.1 Biobjective TSP instances

Each distance matrix in a BTSP instance corresponds to one objective. We have generated the distance matrices using the TSP instance generator of the 8th DIMACS Implementation Challenge. In particular, we have generated three types of biobjective instances:

- *Random Uniform Euclidean (RUE)* instances, where each distance vector component corresponds to the Euclidean distance between two points in a plane. Each distance matrix was derived from a set of points that are uniformly distributed in a square with each side being in the range of $[0, 3163]$; distances have been rounded to the nearest integer.
- *Random distance matrix (RDM)* instances, where each distance vector component corresponds to an integer value generated randomly according to a uniform distribution in the range $[0, 4473]$.
- *Mixed* instances, where one objective corresponds to a distance matrix from an RUE instance, while the other corresponds to a distance matrix from an RDM instance.

The range of the distance values for RUE instances is chosen in analogy to the Krolak/Felts/Nelson instances available in TSPLIB (files with prefix `kro`), which are often tackled in the literature for the BTSP [2, 12, 15]. The range of distances for RDM instances was chosen such that it is similar to that of the RUE instances (note that $\lfloor \sqrt{2} \cdot 3163^2 + 0.5 \rfloor = 4473$).

In addition to the type of instance, we considered the three instance sizes 100, 300, and 500; we refer to these as *small*, *medium*, and *large*, respectively. We recall that many experimental results available in the literature consider only instances of size below 300. For each size and type of instance, three instances were generated, resulting in a total of 27 instances; all instances are available at `eden.dei.uc.pt/~paquete/tsp`.

4.2 Performance assessment methodology

We use a three step evaluation procedure for performance assessment to avoid the known drawbacks of unary performance indicators [34]. In a first step we use *better* relations; if these do not give a conclusive answer, we compute, in a second step, attainment functions and use statistical tests on the equality of attainment functions

[11]; in a third step we use visual means to detect large performance differences in the objective space between pairs of algorithms. For the sake of completeness, we summarize these three steps in what follows; more details of this three step methodology can be found in [21, 25].

Step 1: Better relations A basic outperformance relation among solution sets for MCOPs states that a set of objective vectors A is *better* than another set B if each point in B is either dominated by or equal to a point of A [13]. Hence, we count in the first step how often each outcome associated to one level of a component is *better* than the outcomes of another level of the same component. To reduce variability, these counts are done only on outcomes that were produced *within* the same levels of the other components. If this comparison allows to identify clear differences of performance, we have already an answer for the performance assessment; otherwise, we need to conclude that the outcomes are mostly *incomparable* and we proceed with the next two steps of the performance assessment.

Step 2: The *attainment function* gives the probability of attaining (dominating or being equal to) an *arbitrary point* in the objective space in one single run [8, 11]. Intuitively, this can be seen as a generalization of the distribution function of solution cost to the multiobjective case. The theoretical attainment function can be estimated by the outcomes of several runs of an algorithm, resulting in the *empirical attainment function* (EAF). In addition, one can also test statistical hypotheses on the equality of two or k , with $k > 2$, EAFs using as test statistic the maximum absolute distance between two or the k EAFs [6]; in the latter case, if the global hypothesis of equality is rejected, pairwise tests can be performed, where the p -values of the individual tests need to be corrected, for instance, by using Holm’s procedure [14]. In practice, since we do not have the exact sampling distribution, we use permutation tests based on restricted permutations [10].

Step 3: Location of differences In the final step, we visualize the largest differences in performance, if the null hypothesis on the equality of attainment functions is rejected. This is done by plotting the points in the objective space, where the differences between the EAFs are large. Here we plot points where the absolute differences are larger or equal to 20%. (Note that we run each algorithm on each instance five times and, hence, a difference of 20% is also the minimum difference that is observable.)

As an example, consider the plots in Figure 2 on page 9. On the upper plot are shown the points where the EAF of Algorithm 1 (using **2phase** strategy) becomes larger by at least 20% than that of Algorithm 2 (using **Rand** strategy); on the bottom plot is shown the opposite—this means in this case we have positive and negative differences between the algorithms, that is, for some regions the EAF of one algorithm is larger than that of the other one and for other regions the opposite is true. The observed differences are encoded in a grey-scale: the darker the points, the stronger are the differences. The lower line on each plot is a lower bound on the efficient set that is used simply as a visual reference when plotting the differences with respect to the EAFs³; the upper line gives the points that are attained in all runs of both algorithms.

³ We use a lower bound based on the solution of the 2-Matching Problem [29].

4.3 Experimental Setup

The strategies **Rand** and **2phase** were applied to all RUE and RDM instances; for mixed instances, we considered two variants of the **2phase** strategy: the **2phase_E** strategy, whose first phase starts at the RUE objective, and the **2phase_R** strategy, whose first phase starts at the RDM objective. We also tested the **2-exchange** and **3-exchange** neighborhoods, though the latter was not applied to instances of size 300 and 500 given the prohibitive computation times obtained in preliminary experiments. The ϵ -values for the bounding technique depended on the instance size. For instance sizes 100 and 300, we used $\{0.5/n, 1/n, 2/n\}$. However, some preliminary experiments for instances of size 500 indicated that these ϵ -values would result in an excessive amount of computation time: for example, it took 9898 seconds to terminate with $\epsilon = 2/n$ for an RUE instance of size 500 on an Intel Xeon 2.40 GHz CPU with 2 GB of RAM under Debian GNU/Linux. Additional experiments indicated that values of ϵ in $\{4/n, 5/n, 6/n\}$ would take affordable computation times. (In the following analysis, we remove the denominator n in order to simplify the reading.) In addition, we also used the algorithm without any bounding technique, which corresponds to the PLS algorithm (we shall denote this algorithm as $\epsilon = 0$). However, we only tested it on instances of size 100, since very high computation times were obtained for the larger ones. We performed five runs per configuration on a computer with two AMD Athlon(TM) 1.20 GHz CPUs, 512 MB of RAM running under Suse Linux 7.3.

4.4 Solution Quality

In this section, we report the results of the comparisons with respect to the better relation, the statistical tests, and the location of the differences with respect to the empirical attainment functions.

Initialization strategy

The results of better relations with respect to the strategy for generating the initial solution for instances of size 500 indicated a strong evidence of a better performance of the **2phase** strategy (between 75% and 68%). However, all results for smaller instances were inconclusive. Hence, for the outcomes on these instances, the second step of the experimental analysis was applied. The null hypothesis of equality of the EAFs was always rejected and the test statistic values in the original sample indicated the existence of large differences between the strategies. Figure 2 gives the location of differences between the different strategies on an RUE instance of size 300. The top plot indicates that a large difference is obtained towards the improvement of the first objective in favor of the **2phase** strategy (this observation holds for all sizes). The bottom plot shows the few differences that were detected in favor of the **Rand** strategy on the same instance; differences above 20% in favor of the **Rand** strategy were never found for instances of size 500.

Similar conclusions can be taken for RDM instances. However, differently from RUE instances, on none of the RDM instances differences above 20% were found in favor of the **Rand** strategy.

On mixed instances, each **2phase** variant performed better towards the objective where the corresponding second phase started, while the **Rand** strategy performed

Fig. 2. Location of differences between strategies for an RUE instance of size 300, between **2phase** and **Rand** in favor of the former (top) and in favor of the latter (bottom).

better on the center of the trade-off, except on instances of size 100, where no difference in favor of this search strategy was found. Figure 3 shows the location of differences on a mixed instance of size 500, between **2phase_E** and **2phase_R** in favor of the former (top plot); a similar result was observed to the differences in favor of

Fig. 3. Location of differences between strategies for a mixed instances of size 500 between 2phase_E and 2phase_R in favor of the former (top), and between 2phase_E and Rand in favor of the former (bottom).

the 2phase_R , which occur closer to the minimization of the RMD objective. The bottom plot shows the difference between 2phase_E and Rand in favor of the former. The location of the worst case in the bottom plot indicates that some algorithmic choices were performing very bad from a solution quality perspective. Looking closer

Fig. 4. Location of differences between neighborhoods in favor of the 3-exchange neighborhood for an RUE (top) and an RDM (bottom) instance of size 100.

to the outcomes, we could observe that the **Rand** strategy was the main cause for such result.

<i>size</i>	100				300			500		
<i>RUE</i>	0	0.5	1	2	0.5	1	2	4	5	6
0	-	0.0	17.0	80.3	-	-	-	-	-	-
(1)	0.0	-	2.7	75.0	-	8.0	79.3	-	62.8	63.3
(2)	0.0	0.0	-	53.0	0.0	-	74.0	0.0	-	28.0
(3)	0.0	0.0	0.0	-	0.0	0.0	-	0.0	2.0	-
<i>RDM</i>	0	0.5	1	2	0.5	1	2	4	5	6
0	-	0.0	0.0	43.0	-	-	-	-	-	-
(1)	0.0	-	0.0	33.0	-	1.0	18.0	-	58.7	72.0
(2)	0.0	0.0	-	6.0	0.0	-	11.3	0.0	-	29.3
(3)	0.0	0.0	0.0	-	0.0	0.0	-	0.0	8.7	-
<i>mixed</i>	0	0.5	1	2	0.5	1	2	4	5	6
0	-	0.0	7.0	67.0	-	-	-	-	-	-
(1)	0.0	-	1.0	53.7	-	1.0	47.5	-	62.7	73.3
(2)	0.0	0.0	-	21.3	0.0	-	31.1	0.0	-	24.7
(3)	0.0	0.0	0.0	-	0.0	0.0	-	1.0	12.0	-

Table 1. Results of better relation for different ϵ -values. See the text for more details on the actual ϵ -values used.

Neighborhood

The results of better relations showed that the **3-exchange** neighborhood was clearly better than **2-exchange**, mainly on RDM instances. Moreover, the null hypothesis of equality of EAFs was always rejected. Figure 4 gives the location of differences between the two neighborhoods with respect to their EAFs on an RUE and an RDM instance of size 100. The strengths of the differences confirm the results obtained with the other indicators, that is, the difference is stronger in RDM instances.

ϵ -values

Table 1 gives the results with respect to the better relation on the ϵ -values. In order to compress the table, we give different meanings to the numbers in the left part of the table, depending on the size of the instance: the numbers from (1) to (3) correspond to $\epsilon \in \{0.5, 1, 2\}$, for instances of sizes 100 and 300, and to $\epsilon \in \{4, 5, 6\}$ for instances of size 500, respectively. Only for instance size 100 we tested PLS with $\epsilon = 0$, that is, without archive bounding.

As expected, the performance decays as the ϵ -value increases. The null hypothesis was rejected, with the only exception being the comparison between $\epsilon = 0.5$ and 1 in the RDM instances of size 300. Figures 5 and 6 plot the differences found in RUE and RDM instances of size 100. Here, we see that the differences found in RUE instances are stronger than in RDM instances. In mixed instances we observed that the same stronger differences occur towards the improvement of the RUE objective. Similar results hold also for instances of size 300, that is, the differences are stronger for RUE instances, and towards the RUE objective in mixed instances. We also remark that the differences between $\epsilon = 0.5$ and 1 are stronger than those between the $\epsilon = 1$ and 2.

Fig. 5. Location of differences between the ϵ -values (ϵ equal 0.0 and 0.5 in favor of the former) for an RUE (top) and an RDM (bottom) instance of size 100.

Figures 7 and 8 show the location of differences for instances of size 500, where we can see that the chosen ϵ -values are too restrictive: for instance size 500, the lower bound is rather far from the points where differences in the objective space are found, especially when compared to other plots for instance sizes 100 and 300. Since the lower bound we use is about 14% below optimal, independent of instance

Fig. 6. Location of differences between the ϵ -values (ϵ_{equal1} and 2 in favor of the former) for an RUE (top) and an RDM (bottom) instance of size 100.

size, this fact indicates that for the large instances, the solution quality is rather far from the optimal solutions. This effect can fully be attributed to the usage of the archive bounding techniques.

Fig. 7. Location of differences between the ϵ -values (ϵ equal 4 and 5 in favor of the former) for an RUE (top) and an RDM (bottom) instances of size 500 .

Summary

In most instances, we noticed large differences in favor of the **2phase** strategy towards the improvement of the objective where the second phase started. This also means that the **Rand** strategy is not able to reach high quality solutions to each

Fig. 8. Location of differences between the ϵ -values (ϵ equal 5 and 6 in favor of the former) for an RUE (top) and an RDM (bottom) instance of size 500.

objective. The advantage of the **2phase** strategy over the **Rand** strategy is particularly noteworthy on the instances of size 500. Interestingly, in those instances, the **Rand** strategy performs much worse than the **2phase** strategy. In mixed instances, no advantage was noticed between using the **2phase_E** or **2phase_R** strategies. We observed very high computation times for large instances if no archive bounding

Group	ϵ -value	strategy	instance	size
1	{0.5, 1, 2}	{ Rand , 2phase }	{RUE, RDM}	{100, 300}
2	{0.5, 1, 2}	{ Rand , 2phase_E , 2phase_R }	{Mixed}	{100, 300}
3	{4, 5, 6}	{ Rand , 2phase }	{RUE, RDM}	{500}
4	{4, 5, 6}	{ Rand , 2phase_E , 2phase_R }	{Mixed}	{500}

Table 2. Groups of observations for the ANOVA analysis of the computation times.

technique is used at all or the ϵ -values are very small. In order to have results in a reasonable amount of time, the ϵ -values used for instances of size 500 had to be increased, but consequently, the solution quality worsened. Finally, the statistical tests indicated the existence of significant differences between the **3-exchange** and **2-exchange** neighborhoods for all instances; these differences were stronger on RDM instances.

4.5 Computation Time

Here we report an ANOVA analysis of the computation times of the different algorithms. We divided the observations into four groups, given that not all levels of the components and not all components were tested under the same experimental conditions. We also excluded the neighborhood component from this analysis. The reason is that, keeping all other components fixed, the algorithms using the **3-exchange** neighborhood take on average 67 times more computation time than those using the **2-exchange** neighborhood on the instances of size 100. Similarly, configurations without any archive bounding technique were only tested on instances of size 100 and are therefore removed from the analysis. Once these configurations are dropped, we are able to aggregate the results obtained for instances of size 100 and 300 since they use the same ϵ -values. Finally, we separate the results obtained in mixed instances from the RUE and RDM instances ones due to the different initialization strategies tested. Table 2 gives the details of the four groups that were considered.

For each group, we analyzed the computation times by ANOVA, where the factors are the components defined above with more than one level. The blocks were also defined above with respect to the instance features with more than one level. Note that only in group 1 we have crossed blocks (type and instance size). In order to detect possible interactions, we analysed the interaction plots for all groups with respect to each pair of components and between each component and block. We found many second-order interactions. Some of the most relevant ones are shown in Figure 9. We included these second-order interactions in the four ANOVA models and transformed the response variable in order to meet the usual ANOVA requirements.

The ANOVA results indicated that both, the ϵ -value and the search strategy have a significant effect in group 1 (RUE and RDM instances of size 100 and 300). For instance, we noticed that while a setting of $\epsilon = 0.5$ takes more time in RUE than in RDM instances, the opposite effect is found with higher ϵ -values (see Figure 9, top-left plot). In addition, the analysis showed that, in RUE instances, the **2phase** strategy is faster than the **Rand** strategy, whereas in RDM instances we observed the opposite (see Figure 9, top-right plot). In group 2 (mixed instances of size 100 and 300), we conclude that the **2phase_R** strategy is significantly slower than other

Fig. 9. Second-order interaction plots in group 1 for ϵ -value $\times$ type (top-left) and strategy $\times$ type (top-right); in group 2 for ϵ -value $\times$ strategy (middle-left) and size $\times$ strategy (middle-right); in group 3 and 4 for ϵ -value $\times$ strategy (bottom-left and bottom-right, respectively).

strategies as the ϵ -value decreases or the instance size increases (see both middle plots of Figure 9). A closer look to the outcomes obtained by these strategies indicated that this difference can be explained by the larger number of solutions that **2phase_R** returns. Note that the higher the number of solutions in the archive, the more time it takes to perform each iteration. Differently from the results above, the analysis showed that the interaction between strategy and ϵ -values is significant in groups 3 and 4 (instances of size 500), which is mainly due to the strong difference of the computation times taken by the **Rand** strategy with different ϵ -values (see bottom plots of Figure 9). We remind that the small amount of computation time is related to the early stopping of configurations using the **Rand** strategy in those instances, which resulted in very poor solution quality when compared to the other strategies.

We performed the Tukey HSD statistical test at 0.05 significant level to detect which pairs of combinations of algorithmic choices in each group resulted in significant differences. For group 1, the test indicated that all ϵ -values result in statistically significant differences in computation time. In group 2, the **2phase_R** strategy takes significantly more time than the other two variants. In group 3, only significant differences between the **Rand** and the **2phase** strategy were found for large ϵ -values. In group 4, we concluded that there are significant differences between the **Rand** strategy and the **2phase** variants with $\epsilon \geq 5$, and that **2phase_R** takes significantly more time than **2phase_E**.

4.6 Discussion

Our results indicate that the **2phase** strategy performs quite well on larger RUE and on all RDM instances tested, which may be related to the known results of *closeness* between TSP solutions under the neighborhoods tested [28]. However, we noticed a stronger dependency in terms of computation time between the strategy for generating initial solutions and the type of instance for small and medium size: the **2phase** strategy is faster than the **Rand** strategy for RUE instances, whereas the opposite is observed for RDM instances. For mixed instances, the **2phase_R** strategy takes more time than its **2phase_E** counterpart and the **Rand** strategy, and the difference grows with instance size.

We also observed that the basic PLS algorithm fails to terminate in a reasonable amount of time. For more than 300 nodes, its computation time becomes excessive which is due to the large number of solutions in the archive. This is the main reason for the use of the archive bounding technique. However, using this archive bounding technique also corresponds to a noticeable decrease of solution quality. Although more fine-tuned ϵ -values could probably be found, we think that finding a desired trade-off between computation time and solution quality is difficult.

There is also an interaction between initialization strategies and ϵ -values on the largest instances with respect to solution quality. The ϵ -value had to be increased strongly in order to have results in a reasonable amount of time. However, we noticed a strong decay of the performance of the **Rand** strategy when using the highest ϵ -values tested here. The results obtained by this strategy were not improving further than the starting solutions. This observation clearly indicates that the path from random solutions to near-optimal ones is not as well connected as among near-optimal solutions.

5 Conclusions

We presented an experimental analysis for understanding the effects various algorithm components have with respect to computation time and solution quality on different instance classes of the BTSP. This analysis allowed us to identify patterns of algorithm behavior and to establish dependence relations between certain choices for the algorithm components and instance features.

Further work is still needed. For example, while we studied the solution quality by looking to the main effects, it is possible to extend the current methodology to take into account interactions between components, as done for the computation time. Furthermore, the use of second-order attainment functions [9] would provide a more complete description of the outcomes.

Recent results showed that PLS can be quite successful for this problem when starting from high quality solutions obtained from scalarized versions of the multi-objective formulation [29, 22]. We believe that this hybrid approach is promising to other multiobjective combinatorial problems as well.

References

1. E. Angel, E. Bampis, and L. Gourvés. Approximating the Pareto curve with local search for the bicriteria TSP(1,2) problem. *Theoretical Computer Science*, 310:135–146, 2004.
2. E. Angel, E. Bampis, and L. Gourvés. A dynasearch neighborhood for the bi-criteria traveling salesman problem. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Computer Science*, pages 153–176. Springer Verlag, Berlin, Germany, 2004.
3. V. A. Armetano and J. E. Arroyo. An application of multi-objective tabu search algorithm to a bicriteria flowshop problem. *Journal of Heuristics*, 10(5):463–481, 2004.
4. A. Baykasoglu, S. Owen, and N. Gindy. A taboo search based approach to find the Pareto optimal set in multiobjective optimization. *Journal of Engineering Optimization*, 31:731–748, 1999.
5. P. Borges. CHESS – Changing Horizon Efficient Set Search: A simple principle for multiobjective optimization. *Journal of Heuristics*, 6(3):405–418, 2000.
6. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, 1980.
7. 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.
8. C. M. Fonseca and P. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwebel, editors, *Proceedings of PPSN-IV, Fourth International Conference on Parallel Problem Solving from Nature*, volume 1141 of *Lecture Notes in Computer Science*, pages 584–593. Springer Verlag, Berlin, Germany, 1996.

9. C. M. Fonseca, V. Grunert da Fonseca, and L. Paquete. Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In C. C. Coello, A. H. Aguirre, and E. Zitzler, editors, *Evolutionary Multi-criterion Optimization (EMO 2005)*, volume 3410 of *Lecture Notes in Computer Science*, pages 250–264. Springer Verlag, Berlin, Germany, 2005.
10. P. I. Good. *Permutation Tests: A practical guide to resampling methods for testing hypothesis*. Springer Series in Statistics. Springer Verlag, New York, USA, 2nd edition, 2000.
11. V. Grunert da Fonseca, C. M. Fonseca, and A. Hall. Inferential performance assessment of stochastic optimizers and the attainment function. In E. Zitzler, K. Deb, L. Thiele, C. C. Coello, and D. Corne, editors, *Evolutionary Multi-criterion Optimization (EMO 2001)*, volume 1993 of *Lecture Notes in Computer Science*, pages 213–225. Springer Verlag, Berlin, Germany, 2001.
12. M. P. Hansen. Use of substitute scalarizing functions to guide a local search base heuristics: The case of moTSP. *Journal of Heuristics*, 6:419–431, 2000.
13. M. P. Hansen and A. Jaskiewicz. 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.
14. J. Hsu. *Multiple Comparisons - Theory and Methods*. Chapman & Hall/CRC, Boca Raton, FL, 1996.
15. A. Jaskiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 1(137):50–71, 2002.
16. N. Jozefowicz, F. Semet, and E.-G. Talbi. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In J. M. Guervos, P. Adamis, H.-G. Beyer, J. Fernández-Villacañás, and H.-P. Schwefel, editors, *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature*, volume 2439 of *Lecture Notes in Computer Science*, pages 271–280. Springer Verlag, Berlin, Germany, 2002.
17. J. Knowles and D. Corne. The Pareto archived evolution strategy: A new base line algorithm for multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pages 98–105. IEEE Press, Piscataway, NJ, 1999.
18. J. Knowles and D. Corne. M-PAES: A memetic algorithm for multiobjective optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC'00)*, volume 1, pages 325–332. IEEE Press, Piscataway, NJ, 2000.
19. M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. On the convergence and diversity-preservation properties of multi-objective evolutionary algorithms. TIK-Report 108, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, May 2001.
20. M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In J. M. Guervos, P. Adamis, H.-G. Beyer, J. Fernández-Villacañás, and H.-P. Schwefel, editors, *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature*, volume 2439 of *Lecture Notes in Computer Science*, pages 44–53. Springer Verlag, Berlin, Germany, 2002.
21. M. López-Ibáñez, L. Paquete, and T. Stützle. Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, 5(1):111–137, 2006.

22. T. Lust and J. Teghem. Two phase stochastic local search algorithms for the biobjective traveling salesman problem. In E. Ridge, T. Stützle, M. Birattari, and H. H. Hoos, editors, *Proceedings of SLS-DS 2007, Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, pages 21–25, Brussels, Belgium, 2007.
23. H. Morita, X. Gandibleux, and N. Katoh. Experimental feedback on biobjective permutation scheduling problems solved with a population heuristic. *Foundations of Computing and Decision Sciences*, 26(1):23–50, 2001.
24. L. Paquete, M. Chiarandini, and T. Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 177–200. Springer Verlag, Berlin, Germany, 2004.
25. L. Paquete and C. M. Fonseca. A study of examination timetabling with multiobjective evolutionary algorithms. In *Proceedings of the Fourth Metaheuristics International Conference*, pages 149–154, Porto, 2001.
26. L. Paquete, T. Schiavinotto, and T. Stützle. On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, 156:83–98, 2007.
27. L. Paquete and T. Stützle. Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, Computer and Information Science Series, pages 29–1—29–15. Chapman & Hall/CRC, Boca Raton, FL, Boca Raton, FL, 2007.
28. L. Paquete and T. Stützle. Clusters of non-dominated solutions in multiobjective combinatorial optimization. In V. Barichard, M. Ehrgott, X. Gandibleux, and V. T'Kindt, editors, *Multiobjective Programming and Goal Programming: Theoretical Results and Practical Applications*, volume 618 of *Lecture Notes in Economics and Mathematical Systems*. Springer Verlag, Berlin, Germany, 2009. In press.
29. L. Paquete and T. Stützle. Design and analysis of stochastic local search algorithms for the multiobjective traveling salesman problem. *Computers & Operations Research*, 36(9):2610–2631, 2009.
30. L. Paquete, T. Stützle, and M. López-Ibáñez. Using experimental design to analyze stochastic local search algorithms for multiobjective problems. In K. F. Doerner, M. Gendreau, P. Greistörfer, W. J. Gutjahr, R. F. Hartl, and M. Reimann, editors, *Metaheuristics — Progress in Complex Systems Optimization*, volume 39 of *Operations Research/Computer Science Interface Series*, pages 325–344. Springer Verlag, New York, NY, 2007.
31. P. Serafini. Some considerations about computational complexity for multiobjective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems*, pages 222–231. Springer Verlag, Berlin, Germany, 1986.
32. T. Stützle and H. Hoos. Analyzing the run-time behaviour of iterated local search for the TSP. In P. Hansen and C. Ribeiro, editors, *Essays and Surveys on Metaheuristics*, pages 589–612. Kluwer Academic Publishers, Boston, MA, 2002.
33. E. G. Talbi. A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop. In E. Zitzler, K. Deb, L. Thiele, C. C. Coello,

- and D. Corne, editors, *Evolutionary Multi-criterion Optimization (EMO 2001)*, volume 1993 of *Lecture Notes in Computer Science*, pages 416–428. Springer Verlag, Berlin, Germany, 2001.
34. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.