A Two-Phase Local Search for the Biobjective Traveling Salesman Problem

Luis Paquete and Thomas Stützle

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

Abstract. This article proposes the Two-Phase Local Search for finding a good approximate set of non-dominated solutions. The two phases of this procedure are to (i) generate an initial solution by optimizing only one single objective, and then (ii) to start from this solution a search for non-dominated solutions exploiting a sequence of different formulations of the problem based on aggregations of the objectives. This second phase is a single chain, using the local optimum obtained in the previous formulation as a starting solution to solve the next formulation. Based on this basic idea, we propose some further improvements and report computational results on several instances of the biobjective TSP that show competitive results with state-of-the-art algorithms for this problem.

1 Introduction

Most successful, stochastic approaches to multiobjective combinatorial optimization problems apply a local search procedure embedded in some higher level search strategy to derive a good approximate set of efficient solutions. In order to guarantee that a dispersed set of efficient solutions is obtained, several techniques have been proposed; these can be classified into one of two main groups: Methods that aggregate the objective functions and dynamically modify the search direction during the search process or within several runs [3, 6, 7, 9, 10, 19, 23], and methods that are based on a Pareto-like dominance relation as an acceptance criterion in the search to distinguish between candidate solutions [24, 13]; this second type of methods avoid the aggregation of objectives. Recently, empirical evidence was gathered suggesting that methods belonging to the first group perform particularly well [10], the main reason being that local search algorithms can deal more easily with aggregated objective functions. However, most of the successful algorithms are rather complex search strategies.

In this article, the Two-Phase Local Search procedure (TPLS) is proposed to tackle biobjective combinatorial optimization problems. The first phase consists of finding a good solution to one single objective, using an effective single objective algorithm. This phase provides the starting solution for the second phase, in which a local search algorithm is applied to a sequence of different aggregations of the objectives, where each aggregation converts the biobjective problem into a single objective one. Important for the second phase is that successive aggregations *and* local searches are treated as a chain: an aggregation a_{i+1} modifies slightly the emphasis given to the different objectives when compared to aggregation a_i; the local search for aggregation a_{i+1} is started

Algorithm 1 Two-Phase Local Search

Require: F: vector objective function
W: aggregations of F
/*- - - - - - - - - - - - - phase one - - - - - - - - - - - - - - - - - - */
$s_0^* \leftarrow$ GenerateInitialSolution()
$A \leftarrow$ UpdateArchive(s)
/*- - - - - - - - - - - - - phase two - - - - - - - - - - - - - - - - - - */
for $i \leftarrow 1$ **to** $|W|$ **do**
  $f_i \leftarrow$ ModifyObjFunc(W_i, F, *history*)
  $s_i^* \leftarrow$ LocalSearch(s_{i-1}^*, f_i)
  $A \leftarrow$ UpdateArchive(s_i^*)
end for
$A^* \leftarrow$ FilterArchive(A)

from the local optimum solution s_i^* that was returned for aggregation a_i. The main motivation for such an approach is to exploit the effectiveness of local search algorithms for single objective problems.

To gain insight into the behaviour and performance of TPLS, we applied it to the biobjective Traveling Salesman Problem (TSP). The experimental with a very basic version of TPLS show a very promising performance and with a rather straightforward improvement it was even able to achieve state-of-the-art performance for the biobjective TSP. This is a very promising result, because the procedure is rather simple and appears to be quite robust.

The article is organized as follows. Section 2 introduces the TPLS procedure. Section 3 describes the multiobjective TSP and in Section 4 we present how we adapted TPLS to the biobjective TSP. Section 5 introduces some further improvements over the basic TPLS procedure and analyses their performance. Additional analyses on this procedure are described in Section 6. We conclude and indicate directions for future work in Section 7.

2 The Two-Phase Local Search Procedure

The underlying ideas of the TPLS procedure are (i) to exploit the very good performance of local search algorithms for single-objective problems, (ii) to solve a biobjective problem by chains of related aggregations into single-objective ones *and* chains of good solutions for aggregations which provide starting solutions for a search regarding a next aggregation and (iii) to have an easily understandable but at the same time robust, flexible, and efficient procedure.

Aggregating multiple objective functions into several single-objective problems by utility functions has shown to be an effective approach for multiobjective combinatorial optimization problems as reported in [6, 10, 23]. Several such aggregations were proposed including weighted Tchebycheff utility functions as well as weighted linear utility functions. For the following, we assume that the aggregation is based on a weighted linear utility function with normalized weight vectors. In this case we have that the utility of a solution is rated by

$$f(s) = \sum_{n=1}^{N} w_n f_n(s) \tag{1}$$

where N is the number of objectives, w_n is the weight assigned to the n-th objective, and s is a feasible solution. Additionally, we have $\sum_{n=1}^{N} w_n = 1$ due to the normalization.

From a high level perspective, the TPLS procedure can be described as follows (see Algorithm 1):

First phase: Generate an initial solution s considering only one of the objectives (procedure GenerateInitialSolution);

Second phase: Apply a local search algorithm (LocalSearch), using an aggregation of the objective functions (the aggregated objective function is represented as f_i) generated by ModifyObjFunc, until a solution s_i^* is found, which is added to the archive. The argument s_{i-1}^* of LocalSearch indicates that s_{i-1}^*, the local optimum found in iteration $i-1$ is used as a starting solution for the local search in iteration i. The process is repeated until all aggregations of the objective functions are explored.

The GenerateInitialSolution plays an important role in the procedure, since it generates a first approximation to the Pareto global optima set, and it is the starting solution for the second phase. LocalSearch is the local search algorithm embedded in the procedure for solving the single-objective formulations. In fact, LocalSearch and GenerateInitialSolution can be the same procedure, since an efficient algorithm for the first phase could also be well suited for tackling the second phase; in that case, GenerateInitialSolution could be a local search starting from a random initial solution. In the second phase, the starting solution of LocalSearch is always taken to be the solution returned in the previous iteration of the algorithm.

Let us explicitely remark that there is no need to restrict ourselves to true "local search" procedures. In fact, some single-objective versions of multiobjective problems are solvable in polynomial time by exact methods; this is the case, for example, for the multiobjective linear assignment problem and the multiobjective shortest path problem [18]. When faced with such a problem, it obviously may be preferable to use the polynomial exact algorithm instead of a true local search.

The modification of the aggregated objective function ModifyObjFunc can be carried out by changing the weights assigned to each objective.[1] The change could be random or gradual and the decision can be taken from a previous search space analysis of the problem: if good solutions are clustered in the search space, gradual changes should be preferable, since it has the advantage that the local optimum for aggregation a_{i+1} is close to the one for aggregation a_i and that the local search for a_{i+1} needs only few improvement steps to identify a very good solution to a_{i+1}. If good solutions are spread all over the search space, a randomized change of weights may be more useful.

[1] In case of different ranges of the objectives, a normalization of the objective function values, for example, by range equalization factors [21] is needed.

Each time a local optimum is found by LocalSearch, it is stored in the archive A of the set approximating the Pareto global optima set. Since the solution to the multiobjective problem is a set of all non-dominated objective vectors in A, the FilterArchive procedure is applied in a post-processing step; it deletes all dominated solutions and returns A^*. This last procedure can be excluded if it is applied every time the UpdateArchive procedure is called.

One obvious advantage of TPLS is its modularity, which enables us to focus on the solution methods embedded in GenerateInitialSolution and LocalSearch. Once the choice for these operators is taken, the only numeric parameter needed is the number of different aggregations of the objective functions. In a weighted sum approach, a high number of weight combinations should return a better approximation to the Pareto global optima set. However, some care must be taken, since increasing the number of weight combinations may not be enough to escape from local optima, resulting in a waste of computation time. A study of the trade-off between computation time and solution quality should be carried out according to the real application and together with the decision maker.

3 The Multiobjective TSP

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 $(c_{ij}^1, \ldots, c_{ij}^K)$, where each element c_{ij}^k corresponds to a certain measure like distance, cost, etc. between nodes i and j. For the following we assume that $c_{ij}^k = c_{ji}^k$ for all pairs of nodes i, j and objectives k, that is, we consider only symmetric problems. The multiobjective TSP is the problem of finding "minimal" Hamiltonian circuits of the graph, that is, a set of closed tours visiting each of the $n = |N|$ nodes of G exactly once; here "minimal" refers to the notion of Pareto optimality.

Usually there is not only one, but many Pareto global optimum solutions, which form the *Pareto global optima set*. This set contains all solutions that are not dominated by any other solution. The problem of finding the Pareto global optima set is $\mathcal{NP}$-hard [5] and, since for many problems determining exact solutions quickly becomes infeasible with increasing instance size, the goal typically shifts from identifying Pareto global optima solutions to obtaining a good approximation to this set. For this latter task, algorithms based on local search seem to be a suitable approach and already have shown to yield good performance [7, 10].

In this article, we apply TPLS to the biobjective case, i.e., $K = 2$. As benchmark instances we use combinations of single-objective TSP instances that are available at TSPLIB via `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/` with 100 cities (kroA100 and kroB100), 150 cities (kroA150 and kroB150) and 200 cities (kroA200 and kroB200) as defined in [7]. For convenience, were refer to them as instances kroAB100, kroAB150 and kroAB200, respectively. The first instance was also attacked in [2, 7, 10] and at least for the approach by Jaszkiewicz the solutions are publically available.

4 The Two-Phase Local Search for the Biobjective TSP

This section describes the details of how TPLS was adapted to the biobjective TSP.

4.1 First Phase

As argued before, it is likely to be best to generate the initial solution using a high performing algorithm for the single objective TSP. Research on the single-objective TSP has shown that Iterated Local Search (ILS) algorithms [14] are currently among the best available metaheuristics [11]. ILS is based on the observation that local search algorithms are easily trapped in a local optimum. Instead of restarting the local search from a new, randomly generated solution, it perturbs the current local optimum, moving it to a point beyond the neighborhood searched by the local search algorithm and, thus, allowing it to escape from local optima. An acceptance criterion decides from which local optimum solution the next perturbation is applied. We used an ILS algorithm that was extensively analysed in [22]; at a high level, it can be described as follows:

- It uses a *first improvement* local search based on a *3-exchange* neighborhood, where all tours are neighbors of some tour t that differ in at most three edges;
- The perturbation is a *double bridge* move [15] that cuts the current tour at four appropriately chosen edges into four sub-tours and reconnects these in a different order to yield a new starting tour for the local search;
- The acceptance criterion accepts only a tour if it is better than the best one found so far.

These steps are repeated for a given number of iterations. The local search applies two additional speed-up techniques, which are a fixed radius nearest neighbor within candidate lists of 40 nearest neighbors for each city [12] and *don't look bits* [1]. Some preliminary experiments showed that 50, 100 and 150 iterations for the number of perturbation steps was enough to obtain global optima solutions in short computation time (lower than 0.1 seconds) for kroAB100,kroAB150 and kroAB200, respectively. We refer to this procedure as 3.first.ils.

4.2 Second Phase

For the second step, various choices were tested for the LocalSearch. The reason was to study the trade-off between computation time and solution quality for using a powerful but time expensive local search or a less efficient but faster one. Specifically, for the biobjective TSP, the following algorithms were tested:

- 2.best, a *best improvement* local search in the *2-exchange* neighborhood (it tries to find the best solution in the neighborhood obtained by exchanging two edges in the current tour);
- 3.best, similar to 2.best but using the *3-exchange* neighborhood;
- 2.first, *first improvement* local search with *2-exchange* neighborhood;
- 3.first, similar to 2.first but using a *3-exchange* neighborhood ;

Table 1. Average and standard deviation of the R measure and CPU time on instances kroAB100, kroAB150, and kroAB200 for TPLS.

kroAB100		2.first	2.best	3.first	3.best	2.first.ils	2.best.ils	3.first.ils	3.best.ils
R Measure	Avg.	0.5186	0.9288	0.9316	0.9319	0.9290	0.9288	**0.9339**	0.9318
	Std.	0.0029	0.0009	0.0004	0.0002	0.0007	0.0006	0.0001	0.0002
CPU	Avg.	0.41	0.41	0.53	0.72	0.69	0.72	6.95	6.01
time (sec)	Std.	0.02	0.02	0.01	0.03	0.01	0.01	0.10	0.09

kroAB150		2.first	2.best	3.first	3.best	2.first.ils	2.best.ils	3.first.ils	3.best.ils
R Measure	Avg.	0.5511	0.9364	0.9394	0.9395	0.9365	0.9365	**0.9412**	0.9395
	Std.	0.0029	0.0005	0.0003	0.0002	0.0004	0.0005	0.0001	0.0002
CPU	Avg.	1.47	1.44	1.75	2.19	2.58	2.86	29.91	26.71
time (sec)	Std.	0.09	0.14	0.03	0.19	0.03	0.02	0.32	0.31

kroAB200		2.first	2.best	3.first	3.best	2.first.ils	2.best.ils	3.first.ils	3.best.ils
R Measure	Avg.	0.5710	0.9404	0.9426	0.9428	0.9403	0.9403	**0.9444**	0.9428
	Std.	0.0020	0.0004	0.0002	0.0001	0.0004	0.0004	0.0001	0.0001
CPU	Avg.	4.04	4.13	4.79	6.30	7.13	8.55	120.30	110.37
time (sec)	Std.	0.24	0.07	0.13	0.11	0.10	0.13	1.57	1.44

- 2.first.ils, similar to the 3.ils.first of phase one but using 2.first instead of the 3.first local search algorithm;
- 3.first.ils, the same as in phase one;
- 2.best.ils similar to 2.first.ils but using the 2.best local search;
- 3.best.ils similar to 2.best.ils but using the 3.best local search.

All variants use nearest neighbor lists of length 40.

For the second phase a weighted sum approach was considered. The experiments reported in [2, 16] showed a strong clustering of good solutions in these problems, which indicates that changes of the weights in ModifyObjFunc should be smooth. In particular, the weight of the first objective is decremented (*i.e.*, the single objective used in GenerateInitialSolutions) by $1/|W|$ and the weight of the second objective is incremented by the same amount, where $|W|$ is the number of aggregations tested. The latter was set to 100, 150 and 200 for kroAB100, kroAB150 and kroAB200, respectively. The nearest neighbor list was updated every time ModifyObjFunc was called. The code was written in C and 50 runs were performed on each instance, using a Dual Athlon with 1200 MHz and 512 MB of RAM running Suse Linux 7.3.

The expected value of the weighted Tchebycheff utility function measure (R measure) as proposed in [8] was computed for each of the runs.[2] This measure evaluates a non-dominated set by the expected value of the weighted Tchebycheff utility function

[2] We used the code available at `http://www-idss.cs.put.poznan.pl/~jaszkiewicz/mokp` and adapted it to compile it with gcc 2.95.3 under Linux.

Table 2. Average of the C measure for 3.first.ils compared to the remaining variants of the TPLS for instances kroAB100, kroAB150, kroAB200 (see text for details).

		2.first	2.best	3.first	3.best	2.first.ils	2.best.ils	3.best.ils
kroAB100								
3.first.ils	Covers	50%	87%	65%	59%	87 %	88%	59%
	Covered by	1%	1%	1%	2 %	1 %	1%	1%
kroAB150								
3.first.ils	Covers	50%	94%	79 %	75%	94%	94 %	71%
	Covered by	0%	1%	1%	1 %	1%	1%	1 %
kroAB200								
3.first.ils	Covers	62%	95%	82%	79 %	95%	96 %	80%
	Covered by	0%	0%	1%	1 %	0%	0 %	0%

over a set of normalized weight vectors. The parameters for analysing the result of the algorithm on instance kroAB100 were set following [10] and for instances kroAB150 and kroAB200 the global optima values available on the TSPLIB were used. As worst tour lenghts for the two last instances, the pairs (280000, 280000) and (370000, 370000), respectively, were used. Table 1 presents the results regarding the R measure obtained by the various variants tested. The results show that 3.first.ils gives the best performance in terms of this measure, though it takes large computation times. 3.best.ils and 3.best are almost equivalent, but the latter is much faster; both are closely followed regarding solution quality by 3.first. The results also suggest that the variants based on the 2-exchange neighborhood are inferior to the variants based on the 3-exchange neighborhood.

The results according to the Coverage measure (C measure) [24] are presented in Table 2. This measure compares pairs of non-dominated sets by calculating the fraction of each set that is covered by the other set. In this case, the outcomes of 3.first.ils were compared to each of the ones obtained by the remaining variants for LocalSearch. Therefore, the row *Covered by* indicates how much of the outcome of 3.first.ils is dominated by another variant and the row *Covers* says how much of the outcome of the variant is covered by 3.first.ils. The results are presented averaged over the pairwise comparison between all the runs of two variants. They clearly indicate a best performance of 3.first.ils according to this measure.

One possible drawback of TPLS could be that non-supported solutions, *i.e.*, solutions which are not optimal for any weighted aggregation of the objectives, are not obtained due to its *aggressive* search. However, the 3.first.ils obtained approximately an average of 12% non-supported points per run, which means that it still gets some points which are inside the convex hull of the non-dominated set. To verify how much of the remaining supported solutions were Pareto global optimaof solutions, we count how many were present in the set of supported Pareto global optimal solutions obtained by Borges and Hansen in [2] [3]. Approximately 55% of the supported solutions attained by the TPLS belong to the supported Pareto global optimal solutions. However, since the

[3] The set of supported Pareto global optima solutions is extracted from the supported Pareto global optima solution to the three-objective instance combination of kroA100, kroB100 and kroC100

Table 3. Average and standard deviation of the R measure, the C measure, and the CPU time on instances kroAB100, kroAB150, kroAB200 for the D-TPLS.

kroAB100		3.first	3.best	2.first.ils	3.first.ils
R Measure	Avg.	0.9327	0.9329	0.9303	**0.9345**
	Std.	0.0002	0.0001	0.0004	0.0001
CPU	Avg.	1.06	1.58	1.40	13.95
time (sec)	Std.	0.02	0.04	0.04	0.19
C Measure					
TPLS	Covered by	42%	58%	42%	56%
	Covers	17%	23%	21%	34%

kroAB150		3.first	3.best	2.first.ils	3.first.ils
R Measure	Avg.	0.9399	0.9399	0.9375	**0.9416**
	Std.	0.0002	0.0002	0.0003	0.0000
CPU	Avg.	3.47	4.80	5.02	59.92
time (sec)	Std.	0.06	0.10	0.14	0.35
C Measure					
TPLS	Covered by	38%	45%	47%	42%
	Covers	20%	27%	18%	22%

kroAB200		3.first	3.best	2.first.ils	3.first.ils
R Measure	Avg.	0.9432	0.9432	0.9412	**0.9445**
	Std.	0.0001	0.0001	0.0002	0.0000
CPU	Avg.	9.19	13.44	13.99	245.28
time (sec)	Std.	0.10	0.28	0.16	4.74
C Measure					
TPLS	Covered by	38%	44%	46%	33%
	Covers	18%	25%	17%	17%

data from Borges and Hansen is an incomplete set of Pareto gobal optimal solutions, one does not know if the remaining supported solutions of the TPLS are also Pareto global optimal solutions not found by Borges and Hansen.

5 Improvements on the Two-Phase Local Search

In this section, two further improvements on the TPLS are introduced: the Double Two-Phase Local Search, and the Pareto Double Two-Phase Local Search.

5.1 The Double Two-Phase Local Search

TPLS starts from a very good solution for only one of the objectives. Hence, one may expect a bias of the non-dominated points generated towards this first objective in the

Table 4. Average and standard deviation of the R measure, the C measure, and the CPU time, on instances kroAB100,kroAB150, kroAB200 for the PD-TPLS.

kroAB100		3.first	3.best	2.first.ils	3.first.ils
R Measure	Avg.	0.9335	0.9337	0.9314	**0.9351**
	Std.	0.0002	0.0001	0.0003	0.0000
C Measure					
D-TPLS	Covered by	50%	60%	54%	62%
	Covers	22%	21%	20%	13%
CPU	Avg.	1.26	1.72	1.59	14.14
time (sec)	Std.	0.02	0.01	0.03	0.24

kroAB150		3.first	3.best	2.first.ils	3.first.ils
R Measure	Avg.	0.9404	0.9405	0.9382	**0.9420**
	Std.	0.0002	0.0001	0.0003	0.0000
C Measure					
D-TPLS	Covered by	51%	59%	52%	51%
	Covers	25%	22%	25%	13%
CPU	Avg.	4.32	5.62	5.99	60.64
time (sec)	Std.	0.11	0.18	0.14	0.79

kroAB200		3.first	3.best	2.first.ils	3.first.ils
R Measure	Avg.	0.9436	0.9436	0.9416	**0.9450**
	Std.	0.0001	0.0001	0.0003	0.0000
C Measure					
D-TPLS	Covered by	49%	56%	46%	47%
	Covers	23%	19%	29%	12%
CPU	Avg.	11.87	16.08	17.11	248.45
time (sec)	Std.	0.44	0.44	0.57	6.90

second phase, resulting in a skewed set towards that objective in detriment of the other. This skewedness is exemplified by the fact that only 3.first.ils (but not the other local searches) was able, when starting the second phase from an optimal solution for the first objective, to obtain also the optimal solution of the second objective in most of the runs. One way of overcoming this skewedness effect is to apply the TPLS starting once from a solution for each single objective and then to filter the non-dominated solutions from the union of both sets of solutions. We call this procedure the *Double Two-Phase Local Search* (D-TPLS).

We run experiments with 2.first.ils, 3.first, 3.best and 3.first.ils for the LocalSearch using, as in Section 4, 50 trials per instance and the same number of aggregations. Table 3 presents the average and standard deviation of the R measure and computation time on the instances. In addition, the C measure [24] is presented given as the average of all pairwise comparisons between the TPLS and D-TPLS for each local search used in the second phase. Hence, the row *Covered by* indicates the relative frequency by which

Table 5. Comparison of the PD-TPLS (3.first.ils) with GSL in terms of average and standard deviation of R measure and computation time, and C measure values on the set of instances tackled in [10].

		kroAB100	kroAC100	kroAD100	kroAE100	kroBC100
R Measure						
PD-TPLS	Avg.	0.9351	0.9323	0.9344	0.9380	0.9361
	Std.	0.0000	0.0000	0.0000	0.0001	0.0000
GLS		0.9351	0.9321	0.9342	0.9379	0.9359
C Measure						
GLS	Covered by	48%	59%	61%	55%	53%
	Covers	29%	21%	20%	25%	20%
CPU	Avg.	14.14	13.72	13.69	13.70	14.65
time (sec)	Std.	0.24	0.10	0.10	0.10	0.30

		kroBD100	kroBE100	kroCD100	kroCE100	kroDE100
R Measure						
PD-TPLS	Avg.	0.9347	0.9335	0.9390	0.9352	0.9339
	Std.	0.0000	0.0001	0.0000	0.0000	0.0000
GLS		0.9344	0.9334	0.9389	0.9350	0.9338
C Measure						
GLS	Covered by	56%	54%	55%	58%	54%
	Covers	23%	20%	23%	23%	20%
CPU	Avg.	14.86	14.50	14.02	13.23	14.14
time (sec)	Std.	0.28	0.32	0.15	0.08	0.06

the outcome of TPLS is covered by D-TPLS and the row *Covers* gives the frequency by which the outcome of D-TPLS is covered by TPLS. The computational results indicate a general better performance of D-TPLS when compared to the same TPLS version, although at the cost of doubling the computation time. By analysing further the results of D-TPLS, approximately 25% of the solutions were considered as non-supported, and about 71% of the supported solutions were the same as in [2].

5.2 The Pareto Double Two-Phase Local Search

A disadvantage of the D-TPLS and TPLS is that in the local neighborhood of each solution they return (there is a maximum of $2|W|$ tours returned by D-PTLS and $|W|$ in the case of TPLS), there may be additional, non-dominated solutions, which are missed by the aggregation used. Therefore, intuitively D-TPLS can be further enhanced by searching for non-dominated solutions in the neighborhood of the solutions returned by LocalSearch. For this aim a *2-opt* local search that accepts solutions which are not dominated by the current local optimum in both objectives was applied.[4] This local search is applied after a local optimum is found on each aggregation. We call the so

[4] This algorithm was studied in [16]. An alternative to this approach would be to increase the number of aggregations; this is studied in Section 6.

enhanced D-TPLS the *Pareto Double Two-Phase Local Search* (PD-TPLS). A total of 50 runs of the PD-TPLS were performed using the same four variants of LocalSearch as in Section 5.1 and again analysed in terms of C and R measure.

The results are given in Table 4 and one can observe that, in fact, the PD-TPLS improves on D-TPLS and, hence, also on TPLS. Interestingly, the increase in computation time is rather small. Approximately 90% of the solutions were non-supported, and the same number of supported solutions as in D-TPLS were also found in [2].

5.3 Comparisons to State-of-the-art Algorithms

One important aspect of a new algorithmic proposal is to know how competitive it is compared to *state-of-the-art* algorithms. Apparently, the Genetic Local Search by Jaszkiewicz (GLS) was the best performing algorithm for the biobjective TSP [10]. GLS combines ideas from evolutionary algorithms (recombination, population of solutions), local search (definition of neighborhood) with modifications of the aggregation of the objective functions. The algorithm proceeds by first generating an initial population of solutions where each solution is optimized by a local search method using randomly generated weights to aggregate the objective functions as indicated in Equation 1. Then, at each iteration, a sub-set of the best solutions from the full population, according to a random weighted aggregation of the objective function, is extracted. From this subset, two solutions are randomly chosen and recombined. A local search is then applied to the new solution and added to the population if it is better then the worst solution in the full population. This procedure was able to outperform several multiobjective metaheuristics on a set of 50 and 100 city instances [10].

The performance of PD-TPLS and GLS were compared in all possible pairwise combinations of the instances kroA100, kroB100, kroC100, kroD100 and kroE100. The outcomes of GLS were taken from `http://www-idss.cs.put.poznan.pl/~jaszkiewicz/motsp`. The PD-TPLS ran using the 3.first.ils variants for the LocalSearch 50 times per instance. The results, which are given in Table 5, show the average data and standard deviation of the R measure on the outcomes of PD-TPLS and GLS for each instance. The average and standard deviation of computation times for PD-TPLS are also reported. In addition the C measure was also reported for each pair. In this case, the row *Covered by* presents the frequency by which the outcome of GLS is covered by PD-TPLS and the row *Covers* gives the frequency by which the outcome of PD-TPLS is covered by GLS. The results indicate that, according to the R measure, PD-TPLS performs slightly better than GLS; in addition, when considering the C measure, the advantage of PD-TPLS over GLS appears to be even stronger. Hence, we can conclude that our PD-TPLS approach definitely appears to be competitive, if not even slightly better, than current state-of- the-art algorithms. For further comparisons, the outcomes of PD-TPLS are available at `http://www.intellektik.informatik.tu-darmstadt.de/~lpaquete/TSP`.

6 Further Analysis

This section presents results of a further analysis on the influence of the number of aggregations and of the number of iterations in 3.first.ils on the solution quality (as judged by the R and C measures) and on the computation time.

6.1 Study on the Number of Aggregations

An interesting question is if one can get significant further improvements by increasing the number of aggregations, which should result in a potentially larger number of non-dominated solutions. To investigate this effect on algorithm performance, 50 runs were performed for each of 3.first, 3.best, 2.first.ils and 3.first.ils as LocalSearch variants on instance kroAB100 using $|W| \in \{500, 1000, 1500, 2000\}$.

Table 6 shows the average and standard deviation of the R measure for each variant, number of aggregations and instance. One can observe that, when compared to results obtained by 100 aggregations, given in Table 4, only minor or no improvement at all was found, but that increasing $|W|$ leads to significantly larger computation times.

Table 6. Average and standard deviation of R measure and computation time on instance kroAB100 considering 500, 1000, 1500 and 2000 aggregations for PD-TPLS.

3.first	500	1000	1500	2000
R Measure				
Avg.	0.9337	0.9338	0.9338	0.9338
Std.	0.0001	0.0002	0.0002	0.0002
CPU time				
Avg.	5.42	10.85	16.52	22.71
Std.	0.19	0.27	0.32	0.98

3.best	500	1000	1500	2000
R Measure				
Avg.	0.9338	0.9338	0.9338	0.9338
Std.	0.0001	0.0001	0.0001	0.0001
CPU time				
Avg.	5.96	11.45	17.39	23.55
Std.	0.11	0.27	0.66	0.60

2.first.ils	500	1000	1500	2000
R Measure				
Avg.	0.9316	0.9315	0.9316	0.9316
Std.	0.0004	0.0004	0.0004	0.0004
CPU time				
Avg.	7.17	14.42	21.77	26.10
Std.	0.20	0.54	0.91	1.29

3.first.ils	500	1000	1500	2000
R Measure				
Avg.	0.9352	0.9352	0.9352	0.9352
Std.	0.0000	0.0000	0.0000	0.0000
CPU time				
Avg.	69.53	139.30	209.01	279.88
Std.	0.46	0.64	0.58	0.96

6.2 Study of the Number of Search Steps of 3.first.ils on the LocalSearch

3.first.ils is used both in GenerateInitialSolution and LocalSearch, but so far no exploration of alternative parameter settings for the number of iterations of the ILS was carried out. Hence, it is interesting to observe how much the performance is affected by an increase of the number of iterations of 3.first.ils. A total of 50 runs were performed for each of the following parameters $\{st/5, st/2, st, 2st\}$, where st is the number of iterations used in the previous experiments on instances kroAB100, kroAB150 and kroAB200 (50, 100, and 150, respectively).

Table 7 shows the average and standard deviation of the R measure for each number of search steps and each instance. The C measure was also applied to compare between the algorithm using the original step value and the modified one. Therefore, the row *Covered by* presents how much of the outcome of the algorithm using the original number of steps is covered by the modified one, and the row *Covers* presents how much of

Table 7. Average and standard deviation of R measure, C measure and computation time on the instances kroAB100, kroAB150, kroAB200 considering fifth ($^{st}/_5$), half ($^{st}/_2$) and double ($2{\cdot}st$) of the steps used in GenerateInitialSolution (st) of the PD-TPLS.

	$^{st}/_5$	$^{st}/_2$	st	$2{\cdot}st$
kroAB100	10	25	50	100
R Measure				
Avg.	0.9350	0.9351	0.9351	0.9351
Std.	0.0001	0.0000	0.0000	0.0000
C Measure				
Covered by	40%	57%	-	69%
Covers	63%	64%	-	69%
CPU time				
Avg.	3.68	7.58	14.14	26.97
Std.	0.02	0.04	0.24	0.13

	$^{st}/_5$	$^{st}/_2$	st	$2{\cdot}st$
kroAB150	20	50	100	200
R Measure				
Avg.	0.9419	0.9420	0.9420	0.9420
Std.	0.0000	0.0000	0.0000	0.0000
C Measure				
Covered by	29%	47%	-	64%
Covers	67%	60%	-	57%
CPU time				
Avg.	15.05	32.12	60.64	116.87
Std.	0.07	0.29	0.79	0.50

	$^{st}/_5$	$^{st}/_2$	st	$2{\cdot}st$
kroAB200	30	75	150	300
R Measure				
Avg.	0.9449	0.9450	0.9450	0.9450
Std.	0.0000	0.0000	0.0000	0.0000
C Measure				
Covered by	21%	40%	-	61%
Covers	70%	58%	-	48%
CPU time				
Avg.	57.37	127.68	248.45	477.89
Std.	0.90	2.42	6.90	10.22

the outcome of the modified version is covered by the original. Also average and standard deviation of the computation time is reported. The results with the original step value are presented for reference.

The results indicate that, according to the measures used, almost no performance degradation is observable when using half of the iterations for the ILS, but that, as expected, the computation time could be roughly halved. Allowing still less iterations for ILS degrades somewhat more the solution quality, but computation times drop further. However, the performance is still much better than with other choices for LocalSearch (compare with Table 4). A further increase of the number of ILS iterations seems not to significantly improve performance, probably because the algorithm is already obtaining near-optimal solutions.

7 Conclusions and Further Work

A new two-phase local search algorithm was proposed for tackling biobjective combinatorial optimization problems. There are three main ideas behind this approach. Firstly, one exploits to the maximum possible the excellent performance of local search algorithms for single-objective problems. Secondly, one exploits a chain of aggregations of biobjective problems into single-objective ones *and* previously obtained good solutions to generate new, non-dominated solutions. Thirdly, we want to have a procedure that is

at the same time easy to understand and flexible enough to allow for enhancements that can convert it by minor modifications into higher performance algorithms.

As a first step, the impact of the local search algorithm in the second phase was performed. We found that more powerful local search algorithms results indeed also in better solutions, but at the cost of more computation time. We also proposed two enhancements of the TPLS, which finally lead to the PD-TPLS, which was shown to perform on par or even slightly better than the so far best known algorithm for the biobjective TSP.

The good results and low computation time obtained by this approach can be explained by the high level of clustering of good solutions found in these problems, as already observed in [2, 16]. We intend to extend this approach to other biobjective combinatorial optimization problems in which this clustering does not hold or study problems with different correlations between the cost matrices.

We also intend to extend this approach to more than two objectives. One can, indeed, see that a generalization of the current approach to n objectives can be easily formalized by performing $\frac{n(n-1)}{2}$ runs of PD-TPLS. Although the evident quadratic increase on the number of runs, one should also not expect to find practical applications with a large number of objectives. If this is the case, the Pareto global optima set could be of an infeasible size, and it should be preferable to perform a pre-defined number of runs of the ILS using several aggregations of the objectives with equally dispersed weights.

In addition, we would like to use other measures which allows for a sound statistical analysis of our results. To this aim we will adopt the attainment functions methodology [4] for experimental analysis, as already done in [17, 20].

Acknowledgements We would like to thank Michael Hansen and Pedro Borges for the providing the supported Pareto global optimal solutions, and Andrzej Jaszkiewicz for discussions about the R measure code. Finally, we also thank the suggestions given by the referees 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. 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.
4. V. G. da Fonseca, C. Fonseca, and A. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In E. Zitzler and et al., editors, *Evolutionary Multi-Criterion Optimization (EMO'2001)*, LNCS 1993, pages 213–225. Springer Verlag, 2001.

5. M. Ehrgott. Approximation algorithms for combinatorial multicriteria problems. *International Transactions in Operations Research*, 7:5–31, 2000.
6. X. Gandibleux, N. Mezdaoui, and A. Freville. A tabu search procedure to solve multiobjective combinatorial optimization problems. In R. Caballero et al., editor, *Advances in Multiple Objective and Goal Programming*, LNEMS, pages 291–300. Springer Verlag, 1997.
7. 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.
8. 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.
9. H. Ishibuchi and T. Murata. Multi-objective genetic local search algoritm. In T. Fukuda and T. Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Optimization*, pages 119–124, Nagoya , Japan, 1996. IEEE.
10. A. Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 1(137):50–71, 2002.
11. 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.
12. 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.
13. J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of CEC'99*, pages 98–105, 1999.
14. 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.
15. O. Martin, S.W. Otto, and E.W. Felten. Large-step markov chains for the traveling salesman problem. *INFORMS Journal on Computing*, 8(1):1–15, 1996.
16. L. Paquete, M. Chiarandini, and T. Stützle. A study of local optima in the biojective travelling salesman problem. Technical Report AIDA-02-07, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2002.
17. L. Paquete and C. Fonseca. A study of examination timetabling with multiobjective evolutionary algorithms. In *4th Metaheuristics International Conference (MIC 2001)*, pages 149–154, Porto, 2001.
18. P. Serafini. Some considerations about computational complexity for multiobjective combinatorial problems. In *Recent Advances and Historical Development of Vector Optimization*, LNEMS, pages 222–231. Springer-Verlag, 1986.
19. P. Serafini. Simulated annealing for multiple objective optimization problems. In *Multiple Criteria Decision Making*, LNEMS, pages 283–292. Springer-Verlag, 1994.
20. 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 *Proceeding of CEC'99*, pages 37–45, 1999.
21. R.E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1986.
22. T. Stützle and H. Hoos. Analyzing the run-time behaviour of iterated local search for the TSP. In *III Metaheuristic International Conference (MIC'99)*, pages 1–6, 1999.
23. D. Tuyttens, J. Teghem, P. Fortemps, and K. Van Nieuwenhuyze. Performance of the MOSA method for the bicriteria assignment problem. *Jornal of Heuristics*, 6:295–310, 2000.
24. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. on Evol. Comput.*, 4(3):257–271, 1999.