Ann Oper Res (2017) 254:111–130
DOI 10.1007/s10479-017-2415-5

ORIGINAL PAPER

Proper balance between search towards and along Pareto front: biobjective TSP case study

Andrzej Jaszkiewicz[1] · **Thibaut Lust**[2]

Published online: 21 February 2017

Abstract In this paper we propose simple yet efficient version of the two-phase Pareto local search (2PPLS) for solving the biobjective traveling salesman problem (bTSP). In the first phase the powerful Lin–Kernighan heuristic is used to generate some high quality solutions being very close to the Pareto front. Then Pareto local search is used to generate more potentially Pareto efficient solutions along the Pareto front. Instead of previously used method of Aneja and Nair we use uniformly distributed weight vectors in the first phase. We show experimentally that properly balancing the computational effort in the first and second phase we can obtain results better than previous versions of 2PPLS for bTSP and at least comparable to the state-of-the art results of more complex MOMAD method. Furthermore, we propose a simple extension of 2PPLS where some additional solutions are generated by Lin–Kernighan heuristic during the run of PLS. In this way we obtain a method that is more robust with respect to the number of initial solutions generated in the first phase.

Keywords Multiobjective optimization · Pareto local search · Traveling salesman problem

1 Introduction

In order to generate good approximations of the Pareto fronts of hard multiobjective optimization problems, each multiobjective metaheuristic has to achieve two different goals. The solutions generated by such method should both approach the Pareto front and cover all areas along this front. In many methods a single mechanism is expected to assure both goals.

✉ Thibaut Lust
thibaut.lust@lip6.fr

Andrzej Jaszkiewicz
andrzej.jaszkiewicz@put.poznan.pl

[1] Faculty of Computing, Institute of Computing Science, Poznan University of Technology, ul. Piotrowo 2, 60-965 Poznan, Poland

[2] CNRS, LIP6, UMR 7606, Sorbonne Universités, UPMC Universités Paris 06, 75005 Paris, France

For example, many evolutionary methods use single mechanism of Pareto ranking to both approach and search along the Pareto front (Deb et al. 2000). However, it is quite natural to expect that different techniques may be the best for achieving the two different goals and that the importance of these goals changes at different stages of the optimization process. At the initial iterations of the algorithm when the current population of solutions is far from the Pareto front the main goal is to move towards this front. In later iterations when the population is composed of solutions very close to the Pareto front the main goal is to search for new solutions along this front.

In fact, combination of two different mechanisms for search towards and along Pareto front was used in a number of successful methods even if it was not stated explicitly. Lust and Teghem (2010) proposed the two-phase Pareto local search (2PPLS) method that combines Pareto local search (Angel et al. 2004; Paquete and Stutzle 2006; Paquete et al. 2007) with efficient problem-specific local search heuristics, e.g. Lin and Kernighan (1973) for traveling salesman problem (TSP; Laporte and Osman 1995). The local search heuristic is used to optimize scalarizing functions with various weights. In this way, a starting population of solutions being very close to the Pareto front is generated. Then, the Pareto local search (PLS) is used to search for new solutions along Pareto front. Pareto local search works with a population of potentially efficient solutions, i.e. solutions that are not dominated by any other solution generated so far. For each of these solutions its neighborhood is searched for all new potentially efficient solutions. New potentially efficient solutions enter the current population while dominated solutions are removed. Standalone PLS starting from random solutions is very inefficient since it spends a lot of time generating large numbers of solutions being still very far from the Pareto front. 2PPLS, however, it is at present one of the best methods for multiobjective knapsack (Lust and Teghem 2012), bTSP (Lust and Teghem 2010; Lust and Jaszkiewicz 2010) and set covering problem (Lust and Tuyttens 2014) due to combination of the two search mechanisms.

The concept of combining mechanisms for search towards and along Pareto front was also underlying the idea of the Pareto memetic algorithm with path relinking proposed by Jaszkiewicz and Zielniewicz (2009) for bTSP. The algorithm uses special multiobjective path relinking evolutionary operator generating multiple potentially Pareto efficient solutions along a path linking two recombined solutions, while typical crossover operators generate a single solution only. It was observed that although this operator decreases the performance of the optimization of a single scalarizing function (i.e. deteriorates the search capability towards the Pareto front), the overall performance is improved due to a better search along Pareto front.

The idea of combination of search towards and along Pareto front was stated explicitly by Lara et al. (2010) in hill climber with sidestep (HCS)—a new local search procedure that automatically switches its modes of operation. If the current solution is far from the Pareto front HCS tries to improve it on all objectives to approach the Pareto front. If the current solution is already close to the Pareto front the method automatically switches to the search along this front. The method has been applied to continuous problems but the general idea could be used in case of combinatorial optimization as well.

In this paper we propose a new version of 2PPLS for solving the bTSP using the powerful Lin–Kernighan heuristic. Instead of the method of Aneja and Nair (1979) used in Lust and Teghem (2010) and Lust and Jaszkiewicz (2010) we use uniformly distributed weight vectors in the first phase. The method of Aneja and Nair is an exact method for the generation of all supported non-dominated points, i.e. points in the objective space that correspond to optima of some weighed linear scalarizing functions. The method of Aneja and Nair assumes the use of an exact single objective optimizer. It recursively searches for new supported non-

dominated points in gaps between two neighbor points. If no new point is found between two points, this gap is not searched any more. However, if a heuristic, even a very good one, is used instead of the exact optimizer, the method ceases to be exact and may be stopped prematurely.

To avoid this drawback and to be able to generate more solutions in the first phase than the method of Aneja and Nair we use a set of uniformly distributed weight vectors. Each weight vector defines a weighted linear scalarizing function, i.e. weighed sum of the objectives. Since the problem of optimizing the weighed sum corresponds to the standard single objective TSP the Lin–Kernighan heuristic could be used to solve it. Furthermore, an interesting observation is that suboptimal solutions generated by Lin–Kernighan heuristic may correspond to some non-supported non-dominated points that could not be generated by an exact optimizer.

The number of uniform weight vectors becomes a new parameter of 2PPLS. We show experimentally that properly balancing the computational effort in the first and second phase with constant total running time we can obtain results better than previous versions of 2PPLS for bTSP and at least comparable to state-of-the art results of more complex MOMAD method (Liangjun et al. 2014).

Furthermore, in order to reduce the sensitivity with respect to the number of uniform weight vectors, we propose a simple extension of 2PPLS where some additional solutions are generated by Lin–Kernighan heuristic during the run of PLS. The additional solutions are obtained using random weight vectors. The method obtained in this way does not improve the best results but is much more robust with respect to the number of initial solutions generated in the first phase. This improved behavior could be explained by the fact that this method makes a more smooth shift from the search towards and along Pareto front, since it still generates some new solutions being very close to the Pareto front during the run of PLS.

The paper is organized in the following way. In the next section, basic definitions are presented. The new methods and state-of-the-art MOMAD method used in the experiments are described in the third section. Then the design of the computational experiment is presented in fourth section. In the fifth section, the results of the experiments are presented and discussed. Finally, we present conclusions and perspectives for further research.

2 Basic definitions

2.1 Multiobjective combinatorial optimization

A general multiobjective combinatorial optimization problem is defined as follows:

$$\begin{array}{ll} \underset{x}{\text{“minimize”}} & y(x) = Cx \\ \text{subject to} & x : Ax \leq b \\ & x \in \{0,1\}^n \end{array}$$

$$\begin{array}{rcl} x \in \{0,1\}^n & \longrightarrow & n \text{ variables}, i = 1, \ldots, n \\ C \in \mathbb{N}^{p \times n} & \longrightarrow & p \text{ objective functions}, k = 1, \ldots, p \\ A \in \mathbb{N}^{m \times n} \text{ and } b \in \mathbb{N}^{m \times 1} & \longrightarrow & m \text{ constraints}, j = 1, \ldots, m \end{array}$$

A combinatorial structure is associated to this problem, which can be path, tree, flow, tour, etc.

We denote by $\mathcal{X}$ the feasible set in the decision space, defined by $\mathcal{X} = \{x \in \{0,1\}^n : Ax \leq b\}$. The image of the feasible set in the objective space is called $\mathcal{Y}$ and is defined by $\mathcal{Y} = y(\mathcal{X}) = \{Cx : x \in \mathcal{X}\} \subset \mathbb{N}^p \subset \mathbb{R}^p$. Due to the contradictory features of the objectives,

there does not exist a feasible solution simultaneously minimizing each objective (that is why the word minimize is placed between quotation marks) but a set of feasible solutions called *efficient solutions*. We present below some definitions that characterize these efficient solutions.

We first define the dominance relation of Pareto:

Definition 1 Dominance relation of Pareto: we say that a vector $u = (u_1, \ldots, u_p)$ *dominates* a vector $v = (v_1, \ldots, v_p)$ if, and only if, $u_k \leq v_k \ \forall k \in \{1, \ldots, p\} \wedge \exists k \in \{1, \ldots, p\} : u_k < v_k$. We denote this relation by $u \prec v$.

We can now define an efficient solution, a non-dominated point, the efficient set and the Pareto front.

Definition 2 Efficient solution: a feasible solution $x^* \in \mathcal{X}$ is called *efficient* if there does not exist any other feasible solution $x \in \mathcal{X}$ such as $y(x) \prec y(x^*)$.

Definition 3 Non-dominated point: the image $y(x^*)$ in objective space of an efficient solution x^* is called a non-dominated point.

Definition 4 Efficient set: the efficient set denoted by $\mathcal{X}_E$ contains all the efficient solutions.

Definition 5 Pareto front: the image of the efficient set in $\mathcal{Y}$ is called the Pareto front (or non-dominated frontier/set), and is denoted by $\mathcal{Y}_N$.

We can distinguish two types of efficient solutions: supported efficient solutions and non-supported efficient solutions (Ehrgott 2005).

Definition 6 Supported efficient solutions: supported efficient solutions are optimal solutions of a weighted sum single-objective problem

$$\text{minimize} \left\{ \sum_{k=1}^{p} \lambda_k \, y_k(x) : x \in X \right\}$$

for some weight vector $\Lambda > 0$, that is with all positive components ($\lambda_k > 0, \forall \ k \in \{1, \ldots, p\}$). The image in the objective space of the supported efficient solutions, called supported non-dominated points, are located on the "lowerleft boundary" of the convex hull of $\mathcal{Y}$ (conv $\mathcal{Y}$), that is they are non-dominated points of (conv $\mathcal{Y}$)$+\mathbb{R}^p_+$. We can obtain all supported solutions by varying the weight set Λ and by solving the corresponding weighted sum single-objective problems.

Definition 7 Non-supported efficient solutions: non-supported efficient solutions are efficient solutions that are not optimal solutions of any weighted sum single-objective problem with $\Lambda > 0$. Non-supported non-dominated points are located in the interior of (conv $\mathcal{Y}$)$+\mathbb{R}^p_+$.

In this work, if two potentially efficient solutions $x_1, x_2 \in \mathcal{X}$ are equivalent, that is if $y(x_1) = y(x_2)$, only one of these solutions will be retained (therefore to each potentially non-dominated point will correspond only one potentially efficient solution).

We also define weighted Chebycheff scalarizing functions:

Definition 8 Weighted Chebycheff scalarizing function is defined in the following way:

$$s_\infty(x, y^0, \Lambda) = \max_{k=1,\ldots,p} \{\lambda_k \, (y_k(x) - y_k^0)\}$$

where y^0 is a reference point, $\Lambda = [\lambda_1, \ldots, \lambda_p]$ is a weight vector such that $\lambda_k \geq 0 \ \forall k$.

Each weighted Chebycheff scalarizing function has at least one global optimum (minimum) belonging to the set of efficient solutions. For each efficient solution x there exists a weighted Chebycheff scalarizing function s_∞ such that x is a global optimum (minimum) of s_∞ [see Steuer (1986), ch. 14.8].

2.2 The biobjective traveling salesman problem

Given a set $\{v_1, v_2, \ldots, v_N\}$ of cities and two costs $c_1(v_i, v_j)$ and $c_2(v_i, v_j)$ between each pair of distinct cities $\{v_i, v_j\}$ (with $i \neq j$), the biobjective traveling salesman problem (bTSP) consists of finding a solution, that is an order π of the cities, so as to minimize the following costs ($k = 1, 2$):

$$\text{“minimize”} y_k(\pi) = \sum_{i=1}^{N-1} c_k(v_{\pi(i)}, v_{\pi(i+1)}) + c_k(v_{\pi(N)}, v_{\pi(1)})$$

Hence, two values are associated to a tour realized by a traveling salesman, who has to visit each city exactly once and to return to the starting city. We are interested here only in the symmetric biobjective traveling salesman problem (bTSP), that is $c_k(v_i, v_j) = c_k(v_j, v_i)$ for $1 \leq i, j \leq N$. In this paper, we will use different types of biobjective instances of size going from 100 to 1000. Four different types of instances are used in the computational experiments:

- *Euclidean instances* The costs between the edges correspond to the Euclidean distance between two points in a plane. Such instances are denoted by Kro…and Eucl…
- *Random instances* The costs between the edges are randomly generated from a uniform distribution.
- *Mixed instances* The first cost corresponds to the Euclidean distance between two points in a plane and the second cost is randomly generated from a uniform distribution.
- *Clustered instances* The points are randomly clustered in a plane, and the costs between the edges correspond to the Euclidean distance.

These data have been used by different authors to experiment multiobjective methods. They are available on http://www-desir.lip6.fr/~lustt/.

3 Methods

We first present PLS that is used in all methods studied in this paper. We then present two different versions of 2PPLS, called U2PPLS and R2PPLS. We also present an extension of U2PPLS, called U2PPLS+R, and the MOMAD method (all the methods studied in this paper will be compared to MOMAD).

3.1 Pareto local search (PLS)

We present in this section the algorithm of PLS that will be used in the different 2PPLS methods presented hereafter. The potentially efficient set obtained will be called $\widehat{\mathcal{X}}_E$. The general algorithm of our version of PLS works as follows (see Algorithm 1). Two parameters are needed: an initial population P composed of potentially efficient solutions and a maximum running time T.

The approximation $\widehat{\mathcal{X}}_E$ is first updated with the population P. Then, neighbors p' of the solutions $p \in P$ are generated. If the image of a neighbor p' is not dominated and not equal

to the current solution p, we update the approximation $\widehat{\mathcal{X}}_E$ with p'. The procedure to update $\widehat{\mathcal{X}}_E$ simply consists in adding p' to $\widehat{\mathcal{X}}_E$ if it is non-dominated with respect to any solution in $\widehat{\mathcal{X}}_E$ and to remove all solutions of $\widehat{\mathcal{X}}_E$ that could be found dominated by p'. This procedure returns true if the new solution has been added and false otherwise. If the new solution p' has been added to $\widehat{\mathcal{X}}_E$, we add it to an auxiliary population P_a for further exploration. Once the neighborhood of all the solutions p of P have been explored, we copy P_a into P and we start again the process with P, until P is empty (which happens when a Pareto local optimum has been obtained) or the maximum running time allowed has been reached.

Algorithm 1 `PLS`

Parameter $\downarrow$: Maximum running time T
Parameter $\updownarrow$: An initial population P composed of potentially efficient solutions

```
--| Initialization of X̂_E with P
X̂_E ← P
--| Initialization of an auxiliary population P_a
P_a ← ∅
while Total running time is lower that T and P ≠ ∅ do
  --| Generation of all the neighbors p' of all the solutions p ∈ P
  for all p ∈ P do
    for all p' ∈ N(p) do
      if y(p) ⊀ y(p') and y(p) ≠ y(p') then
        if Update(X̂_E ↕,p' ↓) then
          P_a ← P_a + p'
  --| P is composed of the new non-dominated solutions
  P ← P_a
  --| Re-initialization of P_a
  P_a ← ∅
```

where $\mathcal{N}(p)$ denotes the neighborhood of p and `Update()` updates the set of potentially efficient solutions $\widehat{\mathcal{X}}_E$.

3.2 U2PPLS

The pseudo-code of `U2PPLS` is given by Algorithm 2.

1. *Phase 1* This phase focuses on the search towards the Pareto front. The goal is to find a number of heuristic solutions being very close to the Pareto front and well dispersed over this front. These solutions are generated by solving weighted sum single-objective problems obtained by applying a linear aggregation of the objectives. In U2PPLS method we modify the first phase of the original 2PPLS. Instead of using the method of Aneja and Nair (1979) we use a predefined set of uniformly distributed weight vectors, i.e. vectors with the following values $(0, 1), (1/(L-1), 1-1/(L-1)), (2/(L-1), 1-2/(L-1)), \ldots, (1, 0)$ where L is the number of weight vectors. Each single-objective problem is solved with one of the best heuristics for the single-objective TSP: the Lin–Kernighan heuristic (Lin and Kernighan 1973). We use the chained Lin–Kernighan version of Applegate (2003)[1], as used in 2PPLS and MOMAD.
2. *Phase 2* This phase focuses on the search along the Pareto front. The goal is to find some additional approximately efficient solutions located along Pareto front. In this phase,

[1] The source code of the this heuristic is available on http://www.math.uwaterloo.ca/tsp/concorde.

Algorithm 2 U2PPLS

Parameter ↓: The number of weight vectors L
Parameter ↓: Maximum running time T
Parameter ↑: An approximation $\widehat{\mathcal{X}}_E$ of the efficient set

--| **Start of Phase 1**
for $(i = 0; i < L; i++)$ **do**
  $\lambda_1 \leftarrow 1 - (i/(L-1))$
  $\lambda_2 \leftarrow 1 - \lambda_1$
  LKTSP(Λ, x)
  Update($\widehat{\mathcal{X}}_E$ ↕,x ↓)

--| **Start of Phase 2**
PLS(T,$\widehat{\mathcal{X}}_E$)

where LKTSP() denotes Lin–Kernighan heuristic

PLS is used, with the well-known 2-opt neighborhood. However, as generating all the 2-opt moves for each solution of the population will be very time-consuming ($\frac{N(N-3)}{2}$ neighbors to generate for each solution), we only consider a small proportion of all the possible 2-opt moves. Otherwise, large instances could not be solved in a reasonable time (Lust and Jaszkiewicz 2010). As done in Lust and Jaszkiewicz (2010), candidate lists for the 2-opt moves are used: the candidate lists are created on the basis of the edges used by the solutions found during Phase 1 of 2PPLS. More precisely, we explore the set of candidate edges of Phase 1 (that is the edges used by at least one solution generated in Phase 1), and for each candidate edge $\{v_i, v_j\}$, we add the city v_j to the candidate list of the city v_i.

As it was mentioned above the method of Aneja and Nair ceases to be exact when a heuristic method is used to optimize weighted sum single-objective problems instead of an exact solver. In this case the method of Aneja and Nair may stop prematurely and generate insufficient number of initial solutions. By using a predefined number of weight vectors we get the control over the number of generated solutions and thus on the computational effort in the first phase.

3.3 R2PPLS

A potential disadvantage of U2PPLS is that the number of runs of the problem-specific local search heuristics in Phase 1 has to be defined a priori. In other words it is not possible to stop the first phase earlier, that is before the local search has been run the predefined number of times. It is also not possible to continue the first phase performing more runs of local search. Thus, we test also another approach were the weight vectors used in Phase 1 are generated randomly. To generate a random weight vector we draw the first weight from range $< 0, 1 >$ with uniform distribution, and the second weight is set such that the sum of weights is equal to one. In R2PPLS it is possible to define some dynamic stopping conditions of the first phase. For example, Phase 1 could be stopped when the observed probability of generating a new potentially efficient solution falls below some threshold. On the other hand, the use of random weight vectors may result in a set of solutions less uniformly covering the Pareto front, and thus being worse starting point to Phase 2. The pseudo-code of R2PPLS is given by Algorithm 3.

Algorithm 3 R2PPLS

Parameter ↓: The number L of weight vectors
Parameter ↓: Maximum running time T
Parameter ↑: An approximation $\widehat{\mathcal{X}}_E$ of the efficient set

--| **Start of Phase 1**
for $(i = 0; i < L; i++)$ **do**
  $\lambda_1 \leftarrow$ random number drawn from $< 0, 1 >$ with uniform probability
  $\lambda_2 \leftarrow 1 - \lambda_1$
  LKTSP(Λ, x)
  Update($\widehat{\mathcal{X}}_E \updownarrow, x \downarrow$)

--| **Start of Phase 2**
PLS($T, \widehat{\mathcal{X}}_E$)

3.4 U2PPLS+R

As it was mentioned above, in order to reduce the sensitivity with respect to the number of uniform weight vectors, we propose a simple extension of U2PPLS where some additional solutions are generated by Lin–Kernighan heuristic during the run of PLS. After each main iteration of PLS a number of additional heuristic solutions is generated. The additional solutions are obtained using random weight vectors. If the number of uniform weight vectors used in Phase 1 was set too low, the additional solutions may constitute and additional seed for PLS, and thus improve the final result of PLS. The pseudo-code of U2PPLS+R is given by Algorithm 4.

3.5 MOMAD

We compare the proposed methods with MOMAD algorithm (Liangjun et al. 2014) that is, to our knowledge, state-of-the-art algorithm for the biobjective TSP. Alike U2PPLS, MOMAD starts with generating a number of initial solutions using uniformly distributed weight vectors and a high quality problem-specific single objective heuristic to optimize linear scalarizing functions defined by these weight vectors. Then PLS algorithm with bounded number of generations is applied. For each of the uniformly distributed weight vectors MOMAD stores a single solution being very good but not necessarily the best for linear scalarizing function defined by this weight vector. All such solutions form population P_L. Periodically, after some iterations of PLS, each solution in P_L is first perturbed, and then local search is applied to obtain a new solution. After the perturbation phase population P_L is updated by a procedure that takes into account both quality of the solutions and diversity of P_L. Then PLS is applied again and the phases of perturbation and PLS are repeated several times. For the details of this algorithm see Liangjun et al. (2014).

In case of bTSP, the authors of MOMAD use Lin–Kernighan heuristic (same version as in this paper) in the first phase, a double bridge move as perturbation and 2-opt local search.

4 Computational experiments

4.1 Quality indicators

Several indicators have been introduced in the literature to measure the quality of an approximation [see Zitzler et al. (2002) and Hansen and Jaszkiewicz (1998) for instance].

Algorithm 4 U2PPLS+R

Parameter ↓: The number L of weight vectors
Parameter ↓: The number RD of additional heuristic solutions generated after each main iteration of PLS
Parameter ↓: Maximum running time T
Parameter ↑: An approximation $\widehat{\mathcal{X}}_E$ of the efficient set

--| **Start of Phase 1** (same Phase 1 as U2PPLS)

--| **Start of Phase 2**
--| Initialization of a population P with $\widehat{\mathcal{X}}_E$
$P \leftarrow \widehat{\mathcal{X}}_E$
--| Initialization of an auxiliary population P_a
$P_a \leftarrow \emptyset$
while Total running time is lower that T and $P \neq \emptyset$ **do**
  --| Generation of all the neighbors p' of all the solutions $p \in P$
  for all $p \in P$ **do**
    for all $p' \in \mathcal{N}(p)$ **do**
      if $y(p) \not\prec y(p')$ and $y(p) \neq y(p')$ **then**
        if Update($\widehat{\mathcal{X}}_E \updownarrow, p' \downarrow$) **then**
          $P_a \leftarrow P_a + p'$
  --| P is composed of the new non-dominated solutions
  $P \leftarrow P_a$
  --| Reinitialization of P_a
  $P_a \leftarrow \emptyset$
  --| Generation of additional heuristic solutions
  for $(i = 0; i < RD; i++)$ **do**
    $\lambda_1 \leftarrow$ random number drawn from $< 0, 1 >$ with uniform probability
    $\lambda_2 \leftarrow 1 - \lambda_1$
    LKTSP(Λ, x)
    Update($\widehat{\mathcal{X}}_E \updownarrow, x \downarrow$)
    $P_a \leftarrow P_a + x$

Let $\widehat{\mathcal{Y}}_N$ denote an approximation of $\mathcal{Y}_N$ generated by a multiobjective metaheuristic composed of mutually non-dominated points. In this paper we use the following quality indicators:

- The hypervolume $\mathcal{H}$ (to be maximized; Zitzler 1999): the volume of the dominated space defined by $\widehat{\mathcal{Y}}_N$, limited by a reference point.
- The R measure (normalized between 0 and 1, to be maximized; Jaszkiewicz 2002, 2004): evaluation of $\widehat{\mathcal{Y}}_N$ by the expected value of the weighted Chebycheff utility function over a set of normalized weight vectors. In order to estimate the expected value we use an average over a set of uniform weight vectors (Jaszkiewicz 2002):

$$R(\widehat{\mathcal{Y}}_N, \mathcal{L}) = \frac{\sum_{\Lambda \in \mathcal{L}} \min_{y \in \widehat{\mathcal{Y}}_N} s_\infty(y, y^0, \Lambda)}{|\mathcal{L}|}$$

  where $\mathcal{L}$ is a set of uniformly distributed weight vectors.
- The average distance D_1 and maximum distance D_2 (to be minimized; Czyzak and Jaszkiewicz 1998; Ulungu et al. 1999) between the points of a reference set and the points of $\widehat{\mathcal{Y}}_N$, by using the Euclidean distance. Ideally, the reference set is $\mathcal{Y}_N$ itself, but generally it is not available; otherwise, it can be the non-dominated points existing among the union of various sets $\widehat{\mathcal{Y}}_N$ generated by several methods, or a lower bound of $\mathcal{Y}_N$ Ehrgott and Gandibleux (2007). The quality indicators are defined in the following way:

$$D_1\left(\widehat{\mathcal{Y}}_N, \mathcal{RS}\right) = \frac{\sum_{r\in\mathcal{RS}} \min_{y\in\widehat{\mathcal{Y}}_N} d(r, y)}{|\mathcal{RS}|}$$
$$D_2\left(\widehat{\mathcal{Y}}_N, \mathcal{RS}\right) = \max_{r\in\mathcal{RS}} \min_{y\in\widehat{\mathcal{Y}}_N} d(r, y)$$

where $\mathcal{RS}$ is a reference set, and $d(.,.)$ denotes Euclidean distance. Please note that D_1 is also known as inverted generational distance (Coello and Cortés 2005).

4.2 Experiment design

All the algorithms tested in this work have been run on a Intel Core i5-450M CPU, at 2.4 GHz. We present the average of the indicators over 10 executions. The reported running times correspond to the wall clock time. The running times were limited to be approximately equal to the running times of MOMAD reported in Liangjun et al. (2014) (the computer used for the results of MOMAD has the same processor than the computer used to generate our results). Please note, that the total running time is the parameter of the algorithms 1–3 (U2PPLS, R2PPLS and U2PPLS+R). Therefore, if we increase the number of weight vectors L, the running time of Phase 1 is increased and the running time of Phase 2 is automatically decreased. In other words, by changing L we can modify the balance of computational effort dedicated to Phases 1 and 2. Thus all methods except of 2PPLS use approximately the same CPU time. The exception of 2PPLS is caused by the fact that in this method running time is not a free parameter and instead the method uses built-in stopping rules.

For U2PPLS+R, the parameter RD has been experimentally fixed to 25.

To compute the distances D_1 and D_2 (see Sect. 4.1), reference sets based on the notion of ideal set (Lust and Teghem 2010) have been generated for all the instances experimented. The ideal set is defined as the best potential Pareto front that can be produced from the extreme supported non-dominated points. This is a lower bound of the Pareto front (Ehrgott and Gandibleux 2007). For generating the extreme supported non-dominated points, we have used the method of Aneja and Nair coupled with an exact TSP solver (Concorde, available in http://www.math.uwaterloo.ca/tsp/concorde/). However, for the instances of more than 200 cities, the exact method was too time consuming. Thus, for these instances, we have generated the extreme supported non-dominated points of the biobjective minimum spanning tree problem (bMST) associated to the same data than the bTSP. The ideal set is then produced on the basis of the extreme supported non-dominated points of the bMST. As the biobjective minimum spanning tree problem is a relaxation of the bTSP, all feasible solutions of the bTSP remain dominated or equivalent to the solutions of the ideal set of the bMST.

For the computation of the R and $\mathcal{H}$ indicators, the reference points are determined according to the reference sets (for $\mathcal{H}$, we use the Nadir point of the reference set multiplied by 1.1 and for R, we use the ideal point of the reference set). For the R indicator, the number of weight sets used is equal to 501 for all instances. This indicator has been normalized between 0 and 1, such that it would obtain 1 for ideal point and 0 for Nadir point.

5 Results

The main results of the computational experiments are presented in Figs. 1, 2, 3 and 4. The figures show average values of D_1 quality measure obtained by U2PPLS, R2PPLS and U2PPLS+R methods for different number of initial solutions denoted by L and for different

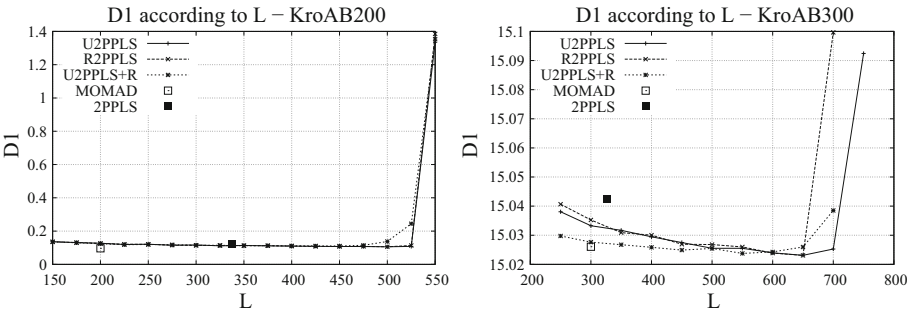

Fig. 1 KroAB200–KroAB300

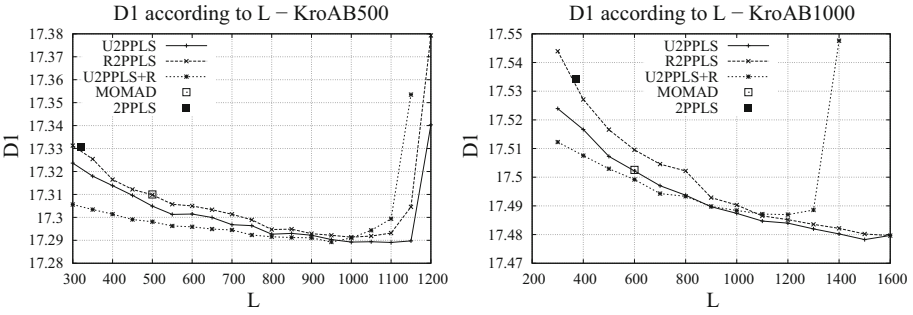

Fig. 2 KroAB500–KroAB1000

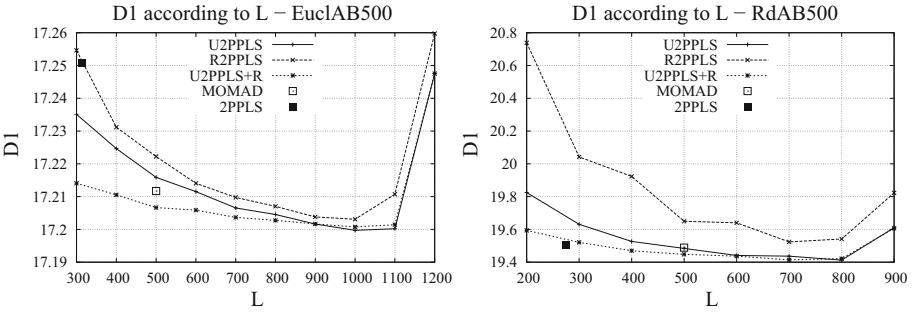

Fig. 3 EuclAB500–RdAB500

types of instances. Because of limited space we do not include graphs for other quality indicators but the results are very similar for all indicators.[2] We also show the values of D_1 obtained by the original version of 2PPLS in Lust and Jaszkiewicz (2010) and values obtained by MOMAD method in Liangjun et al. (2014). Points corresponding to the two latter methods are located on x-axis at positions corresponding to the number of initial solutions used in these experiments. Let us remind that the running times of U2PPLS, R2PPLS and U2PPLS+R were limited to be approximately equal to the running times of MOMAD reported in Liangjun

[2] Additional figures for other instances and the R indicator can be found on http://www-desir.lip6.fr/~lustt/Research.html#ProperBalance.

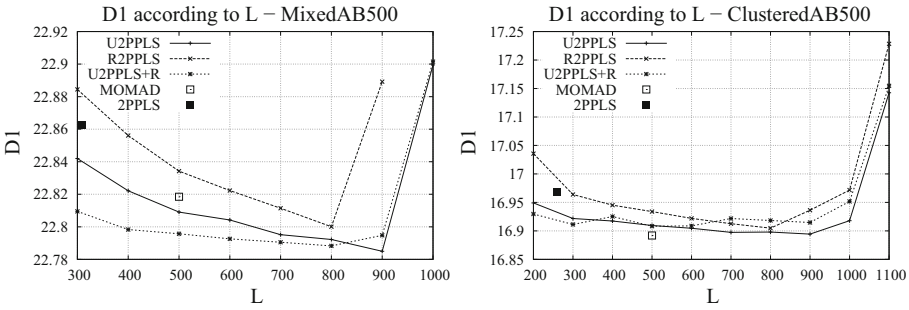

Fig. 4 MixedAB500–ClusteredAB500

et al. (2014). In other words if we increased the number of weight vectors L, the running time of Phase 2 was automatically decreased. The main observations are:

- In all cases U2PPLS was able to generate better results than original 2PPLS if the number of initial solutions L was set to larger values than the number of initial solutions generated by Aneja and Nair in the original 2PPLS. The difference becomes larger for bigger instances.
- U2PPLS gives the best results for some specific values of parameter L (the number of initial solutions) and deteriorates if either lower or greater values of L are used. In other words, the best results are obtained if the optimum balance of the computational effort dedicated to Phase 1 and Phase 2 is assured.
- In many cases U2PPLS is able to generate similar or better results than MOMAD. However, for clustered instances MOMAD remains better than U2PPLS. The main difference between MOMAD and U2PPLS is the perturbation step used in the former algorithm. This indicates that for this type of instances Lin–Kernighan heuristic is insufficient to provide high quality initial solutions for PLS, and that the use of perturbation may help in generating some additional solutions hardly achievable with Lin–Kernighan heuristic standalone.
- R2PPLS performs worse than U2PPLS.
- U2PPLS+R is for many values of L better that U2PPLS but the best values achieved are similar to that of U2PPLS. The picture is again different for clustered instances on which UPPLS+R performs relatively poorly.

The above results support the statement that the original version of 2PPLS did not exploit fully the power of Lin–Kernighan heuristic stopping the first phase too early. To study it in more details in Figs. 5, 6 and 7 we show the number of efficient solutions obtained in Phase 1 for a given L, i.e. the number of weight vectors, and thus the number of Lin–Kernighan runs. We show these results for both uniformly distributed and random weight vectors. One can see that new efficient solutions are still generated even for relatively large numbers of L, which confirms that stopping Phase 1 too early may deteriorate final results. In addition, the use of random weight vectors in general results in a smaller number of efficient solutions, which explains worse final results after the second phase.

The average values of all indicators ($\mathcal{H}$, R, D_1 and D_2) obtained by 2PPLS, U2PPLS, U2PPLS+R and MOMAD are given in Tables 1 and 2. For U2PPLS and U2PPLS+R we give the result for only one value of L, the value for which the methods gave best results. These values of L are reported in brackets. For each instance and each indicator, the best values are marked in bold.

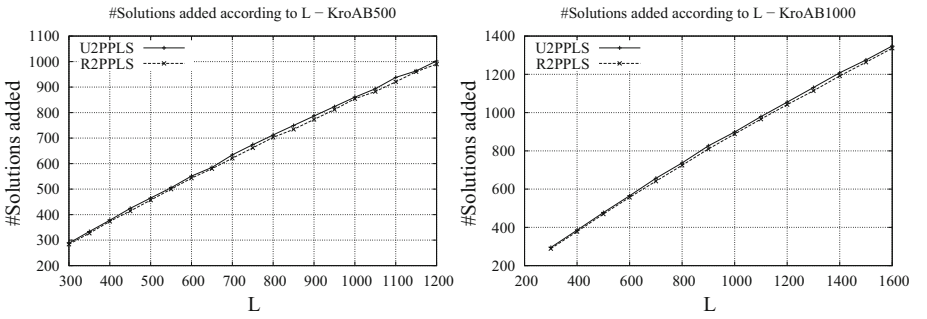

Fig. 5 KroAB500–KroAB1000

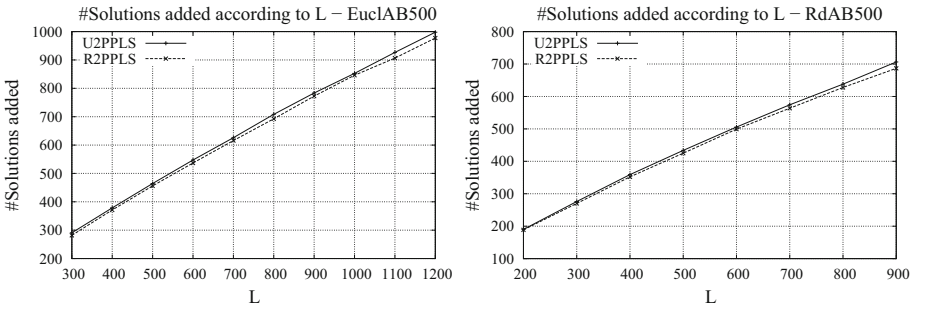

Fig. 6 EuclAB500–RdAB500

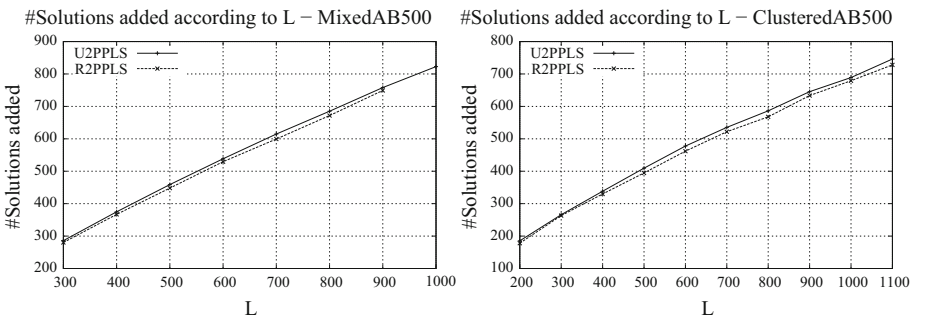

Fig. 7 MixedAB500–ClusteredAB500

We see that the best results are never achieved by 2PPLS and there is no clear winner among other methods. $\mathcal{H}$, R and D_1 usually give very similar ranking, while U2PPLS+R performs particularly well for D_2, i.e. maximum distance from the reference set. It indicates that U2PPLS+R gives particularly uniform representation of the reference set even if not the best on average.

We have also indicated in these tables the number of full iterations of PLS performed. We see that, for most of the instances, the number of iterations of U2PPLS is less than the number of iterations of 2PPLS, which means that, for a fixed running time, it is better to spend more times to generate a good initial population than applying PLS until the end and reaching a Pareto local optimum.

Table 1 Comparison between methods

Instance	Methods	$\mathcal{H}(10^8)$	R	D_1	D_2	#ItPLS	$\|PE\|$	Time(s)
KroAB200	2PPLS	1071.793	0.938409	0.123	5.103	15.9	6216.6	79.51
	U2PPLS(500)	1071.817	0.938423	0.106	4.410	13.0	6301.6	91.29
	U2PPLS+R(450)	1071.811	0.938420	0.110	4.486	5.0	5786.1	97.78
	MOMAD	**1071.840**	**0.938428**	**0.097**	**3.615**	–	7236.0	94.00
KroAB300	2PPLS	2409.263	0.942089	15.043	18.695	17.6	14131.3	152.09
	U2PPLS(650)	**2409.334**	**0.942101**	**15.023**	18.675	13.9	14418.2	210.90
	U2PPLS+R(550)	2409.309	0.942100	15.024	18.680	5.6	14101.3	218.61
	MOMAD	2409.299	**0.942101**	15.026	**18.668**	–	15273.7	213.00
KroAB500	2PPLS	6819.152	0.946481	17.331	23.973	23.4	32086.8	466.49
	U2PPLS(1000)	**6819.478**	**0.946499**	**17.289**	22.467	18.7	32848.0	813.06
	U2PPLS+R(950)	6819.432	0.946498	**17.289**	**22.463**	5.9	29850.2	852.71
	MOMAD	6819.250	0.946490	17.310	22.502	–	33005.85	823.00
KroAB750	2PPLS	15960.567	0.951929	17.173	28.859	29.3	59450.0	1419.66
	U2PPLS(1300)	**15961.192**	**0.951949**	**17.116**	29.800	5.9	52065.8	1849.01
	U2PPLS+R(1300)	15961.008	0.951946	17.121	26.200	4.0	40580.5	1918.51
	MOMAD	15960.834	0.951944	17.132	**24.402**	–	62026.6	1849.00
KroAB1000	2PPLS	27922.649	0.953752	17.534	23.036	39.0	97521.2	3908.79
	U2PPLS(1500)	**27923.471**	**0.953768**	**17.478**	**22.855**	5.0	75192.2	3303.03
	U2PPLS+R(1200)	27923.091	0.953765	17.487	22.896	4.3	63921.7	3553.75
	MOMAD	27922.790	0.953762	17.503	22.810	–	99480.3	3303.00

Table 2 Comparison between methods

Instance	Methods	$\mathcal{H}(10^8)$	R	D_1	D_2	#ItPLS	$\|PE\|$	Time(s)
EuclAB300	2PPLS	2309.662	0.941224	16.869	21.591	21.3	14022.6	141.04
	U2PPLS(850)	**2309.793**	**0.941247**	**16.827**	21.469	11.7	14203.90	274.54
	U2PPLS+R(800)	2309.772	0.941245	16.828	**21.464**	4.0	12208.80	280.48
	MOMAD	2309.754	0.941242	16.841	21.511	–	15095.9	275.00
EuclAB500	2PPLS	7165.381	0.948574	17.251	21.501	26.0	32955.60	424.15
	U2PPLS(1000)	**7165.719**	**0.948594**	**17.200**	21.352	5.2	30791.00	711.00
	U2PPLS+R(1000)	7165.660	0.948593	17.201	**21.337**	4.0	26053.4	734.35
	MOMAD	7165.629	0.948591	17.212	21.397	–	34412.2	711.00
RdAB300	2PPLS	4804.461	0.966649	20.912	39.333	12.7	1768.7	193.530
	U2PPLS(450)	**4804.642**	**0.966720**	**20.862**	35.961	13.2	1816.10	266.88
	U2PPLS+R(350)	4804.486	0.966714	20.885	37.505	5.0	1632.70	276.12
	MOMAD	4804.375	0.966704	20.905	**30.432**	–	2064.05	274.00
RdAB500	2PPLS	13987.876	0.973473	19.508	54.215	15.6	3115.70	500.07
	U2PPLS(800)	**13988.597**	**0.973541**	**19.413**	53.937	14.3	3119.40	1147.74
	U2PPLS+R(700)	13988.545	**0.973541**	19.414	**41.348**	7.9	2949.30	1235.11
	MOMAD	13987.812	0.973531	19.489	74.474	–	3259.95	1211.00
MixedAB300	2PPLS	3410.411	0.956133	20.135	28.437	17.6	5203.3	155.15
	U2PPLS(650)	**3410.662**	**0.956169**	**20.082**	28.295	15.9	5291.70	294.247
	U2PPLS+R(500)	3410.625	0.956166	**20.082**	**28.285**	8.0	5039.20	315.61
	MOMAD	3410.567	0.956168	20.098	28.304	–	6144.40	309.00
MixedAB500	2PPLS	10440.300	0.960325	22.863	31.782	22.2	13096.6	490.79
	U2PPLS(900)	**10440.926**	**0.960355**	**22.785**	31.610	17.6	13034.20	1091.59
	U2PPLS+R(800)	1044.083	0.960353	22.788	**31.556**	6.0	12139.6	1132.78
	MOMAD	10440.628	0.960346	22.818	31.608	–	13284.15	1104.00

Table 2 continued

Instance	Methods	$\mathcal{H}(10^8)$	R	D_1	D_2	#ItPLS	$\|PE\|$	Time(s)
ClusteredAB300	2PPLS	2565.281	0.956260	17.768	27.160	42.3	15318.0	197.81
	U2PPLS(500)	2565.943	0.956304	17.681	22.572	44.8	15786.9	277.934
	U2PPLS+R(500)	2565.589	0.956300	17.695	**21.815**	8.2	14027.60	352.39
	MOMAD	**2566.348**	**0.956309**	**17.672**	28.425	–	16637.30	347.00
ClusteredAB500	2PPLS	8516.523	0.962101	16.969	27.803	66.3	40118.10	612.74
	U2PPLS(900)	8517.361	**0.962131**	16.894	23.105	8.1	38302.40	849.00
	U2PPLS+R(500)	8517.155	0.962125	16.908	24.091	12.6	38001.20	866.79
	MOMAD	**8517.475**	**0.962131**	**16.892**	**23.095**	–	43293.15	849.00

Table 3 Results of the Mann–Whitney test for D_1

	2PPLS vs U2PPLS	2PPLS vs U2PPLS+R	2PPLS vs MOMAD	U2PPLS vs U2PPLS+R	U2PPLS vs MOMAD	U2PPLS+R vs MOMAD
KroAB200	⊖	⊖	⊖	⊕	⊖	⊖
KroAB300	⊖	⊖	⊖	=	⊕	⊕
KroAB500	⊖	⊖	⊖	=	⊕	⊕
KroAB750	⊖	⊖	⊖	⊕	⊕	⊕
KroAB1000	⊖	⊖	⊖	⊕	⊕	⊕
EuclAB300	⊖	⊖	⊖	=	⊕	⊕
EuclAB500	⊖	⊖	⊖	=	⊕	⊕
RdAB300	⊖	⊖	=	⊕	⊕	⊕
RdAB500	⊖	⊖	⊖	=	⊕	⊕
MixedAB300	⊖	⊖	⊖	=	⊕	⊕
MixedAB500	⊖	⊖	⊖	⊕	⊕	⊕
ClusteredAB300	⊖	⊖	⊖	⊕	⊖	⊖
ClusteredAB500	⊖	⊖	⊖	⊕	=	⊖

To take into account the variations in the results of the algorithms, we have tested if the observed differences are statistically significant. As we do not know the distributions of the indicators, we carried out the non-parametric statistical test of Mann–Whitney (Ferguson 1967).

The results of the comparison with D_1 and R indicators are given in Tables 3 and 4 respectively. For each pair of methods we test the following hypothesis: "the two samples come from identical populations" for the D_1 or R indicator on a given instance. When the hypothesis is satisfied, the result "=" is indicated (no differences between the indicators of the algorithms). When the hypothesis is not satisfied, the sign "⊕" indicates that the first method is better and "⊖" vice versa. The level of risk of the test has been fixed to 5%. As in Tables 1 and 2 for U2PPLS, U2PPLS+R we use the value of L for which the methods gave best results.

One can see that 2PPLS is significantly worse than the other methods in all cases. For the other methods the results are inconclusive with slight advantage of U2PPLS. For example, U2PPLS is better than MOMAD in 18 cases, indifferent in 4 cases, and worse in 4 cases.

These results are only for a single value of L for which a given method gave the best results. However, we could have obtained the same conclusions with other values of L. Indeed, in Table 5, we show that the values of L for which U2PPLS or U2PPLS+R can obtain better or equal results than MOMAD, for the R and D_1 indicators (according to the Mann–Whitney test) are quite large: for example, for the KroAB500 instance, using a value of L between 550 and 1150 for U2PPLS, or between 300 and 1150 for U2PPLS+R, allows to obtain better results than MOMAD. On the other hand there are few instances for which the results of MOMAD are always better.

Furthermore, although Liangjun et al. (2014) do not give much details about how the parameters of MOMAD were set, we can expect that they were based on at least some preliminary experiments to assure good performance. In particular the value of L used in MOMAD depends on the size of the instance. Moreover the authors present additional experiments for a selected instance showing that the value of L has a strong influence on the obtained results.

Table 4 Results of the Mann–Whitney test for R

	2PPLS vs U2PPLS	2PPLS vs U2PPLS+R	2PPLS vs MOMAD	U2PPLS vs U2PPLS+R	U2PPLS vs MOMAD	U2PPLS+R vs MOMAD
KroAB200	⊖	⊖	⊖	⊕	⊖	⊖
KroAB300	⊖	⊖	⊖	⊕	=	⊖
KroAB500	⊖	⊖	⊖	⊕	⊕	⊕
KroAB750	⊖	⊖	⊖	⊕	⊕	⊕
KroAB1000	⊖	⊖	⊖	⊕	⊕	⊕
EuclAB300	⊖	⊖	⊖	⊕	⊕	⊕
EuclAB500	⊖	⊖	⊖	⊕	⊕	⊕
RdAB300	⊖	⊖	⊖	⊕	⊕	⊕
RdAB500	⊖	⊖	⊖	=	⊕	⊕
MixedAB300	⊖	⊖	⊖	⊕	=	=
MixedAB500	⊖	⊖	⊖	⊕	⊕	⊕
ClusteredAB300	⊖	⊖	⊖	⊕	⊖	⊖
ClusteredAB500	⊖	⊖	⊖	⊕	=	⊖

Table 5 Intervals values of L for which U2PPLS and U2PPLS+R are better or equal than MOMAD for the D_1 and R indicators

	U2PPLS vs MOMAD		U2PPLS+R vs MOMAD	
	D_1	R	D_1	R
KroAB200	∅	∅	∅	∅
KroAB300	[650]	[550]	[550]	∅
KroAB500	[550, 1150]	[550, 1150]	[300, 1100]	[300, 1100]
KroAB750	[800, 1600]	[800, 1600]	[800, 1400]	[800, 1400]
KroAB1000	[700, 1600]	[700, 1600]	[700, 1300]	[700, 1300]
EuclAB300	[450, 850]	[450, 850]	[200, 850]	[400, 850]
EuclAB500	[700, 1100]	[700, 1100]	[400, 1100]	[600, 1000]
RdAB300	[350, 450]	[350, 450]	[275, 400]	[250, 425]
RdAB500	[600, 800]	[600, 800]	[400, 800]	[400, 800]
MixedAB300	[500, 650]	[500, 650]	[350, 600]	[500, 600]
MixedAB500	[500, 900]	[500, 900]	[300, 900]	[400, 900]
ClusteredAB300	∅	∅	∅	∅
ClusteredAB500	[900]	∅	[500]	∅

6 Conclusions and perspectives

We have presented two new versions of two-phase Pareto local search for the biobjective traveling salesman problem—U2PPLS and U2PPLS+R. The methods are relatively simple (which we believe is an advantage), yet they improve previous results for bTSP obtained with the original version 2PPLS and are at least comparable to state-of-the-art results generated by MOMAD method. The new methods are motivated by the idea of finding a proper balance between the effort of search towards and along Pareto front. In fact, we have shown that

the original version of 2PPLS did not exploit fully the power of Lin–Kernighan heuristic in the first phase, and that by increasing the number of Lin–Kernighan runs even at the cost of running time dedicated to PLS better results can be obtained.

Although the computational experiment has been performed for bTSP only, we expect that the general idea is applicable also for other problems where very good heuristics exists for optimization of weighted sum of objectives. Thus tests on other multiobjective problems is natural direction for further research.

In the proposed methods the number of weight vectors L is a free parameter that has to be appropriately set. Please note, that it is a typical situation in metaheuristics that usually have some parameters that have to be tuned to a particular case. In order to decrease sensitivity with respect to this parameter we have proposed U2PPLS+R method which performs relatively well for a wide range of L and achieves the best results similar to U2PPLS. An interesting direction for further research is to propose a systematic method to adapt the parameter L for a given instance e.g. in the spirit of Battiti et al. (2008).

Another interesting direction is to consider more smooth shift from the search towards and along Pareto front. In U2PPLS the shift from the search towards Pareto front by single objective heuristics to the search along Pareto front by PLS is immediate. U2PPLS+R makes more smooth shift, since it still generates some new solutions by the single objective heuristic during the run of PLS. Another possibility would be to use some additional method with some intermediate properties from the point of view search towards and along Pareto front, between initial phase and PLS.

Acknowledgements The research of Andrzej Jaszkiewicz was funded by the the Polish National Science Center, Grant No. UMO-2013/11/B/ST6/01075.

References

Aneja, Y., & Nair, K. (1979). Bicriteria transportation problem. *Management Science*, *25*, 73–78.

Angel, E., Bampis, E., & Gourvès, L. (2004). A dynasearch neighborhood for the bicriteria traveling salesman problem. In X. Gandibleux, M. Sevaux, K. Sörensen, & V. T'kindt (Eds.), *Metaheuristics for multiobjective optimisation. Lecture notes in economics and mathematical systems* (Vol. 535, pp. 153–176). Berlin: Springer.

Applegate, D. (2003). Chained Lin–Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, *15*, 82–92.

Battiti, R., Brunato, M., & Mascia, F. (2008). *Reactive search and intelligent optimization, operations research/Computer science interfaces*. Berlin: Springer. (ISBN 978-0-387-096 23-0).

Coello, C. A. C., & Cortés, N. C. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, *6*, 163–190.

Czyzak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing—A metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, *7*, 34–47.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the parallel problem solving from nature VI conference. Lecture notes in computer science no. 1917* (pp. 849–858). Paris, France: Springer.

Ehrgott, M. (2005). *Multicriteria optimization* (2nd ed.). Berlin: Springer.

Ehrgott, M., & Gandibleux, X. (2007). Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, *34*, 2674–2694.

Ferguson, T. (1967). *Mathematical statistics, a decision theoretic approach*. New York: Academic Press.

Hansen, M. P., & Jaszkiewicz, A. (1998). *Evaluating the quality of approximations to the non-dominated set*. Lyngby: IMM, Department of Mathematical Modelling, Technical University of Denmark.

Jaszkiewicz, A. (2002). On the performance of multiple-objective genetic local search on the 0/1 knapsack problem—A comparative experiment. *IEEE Transactions on Evolutionary Computation*, *6*, 402–412.

Jaszkiewicz, A. (2004). A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the Pareto memetic algorithm. *Annals of Operations Research*, *131*, 135–158.

Jaszkiewicz, A., & Zielniewicz, P. (2009). Pareto memetic algorithm with path-relinking for biobjective traveling salesman problem. *European Journal of Operational Research*, *193*, 885–890.

Laporte, G., & Osman, I. (1995). Routing problems: A bibliography. *Annals of Operations Research*, *61*, 227–262.

Lara, A., Sanchez, G., Coello, C. C., & Schutze, O. (2010). HCS: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *14*, 112–132.

Liangjun, K., Qingfu, Z., & Battiti, R. (2014). Hybridization of decomposition and local search for multiobjective optimization. *IEEE Transactions on Cybernetics*, *44*, 1808–1820.

Lin, S., & Kernighan, B. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, *21*, 498–516.

Lust, T., & Jaszkiewicz, A. (2010). Speed-up techniques for solving large-scale biobjective TSP. *Computers & Operations Research*, *37*, 521–533.

Lust, T., & Teghem, J. (2010). Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, *16*, 475–510.

Lust, T., & Teghem, J. (2012). The multiobjective multidimensional knapsack problem: A survey and a new approach. *International Transactions in Operational Research*, *19*, 495–520.

Lust, T., & Tuyttens, D. (2014). Variable and large neighborhood search to solve the multiobjective set covering problem. *Journal of Heuristics*, *20*, 165–188.

Paquete, L., Schiavinotto, T., & Stützle, T. (2007). On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, *156*, 83–97.

Paquete, L., & Stutzle, T. (2006). A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *European Journal of Operational Research*, *169*, 943–959.

Steuer, R. (1986). *Multiple criteria optimization: Theory, computation and applications*. New York: Wiley.

Ulungu, E., Teghem, J., Fortemps, P., & Tuyttens, D. (1999). MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, *8*, 221–236.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Ph.D. thesis, Zurich: Swiss Federal Institute of Technology (ETH).

Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C., & Grunert da Fonseca, V. (2002). Why quality assessment of multiobjective optimizers is difficult. In W. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, & N. Jonoska (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)* (pp. 666–673). San Francisco: Morgan Kaufmann Publishers.