J Heuristics
DOI 10.1007/s10732-009-9103-9

Two-phase Pareto local search for the biobjective traveling salesman problem

Thibaut Lust · Jacques Teghem

Received: 20 November 2007 / Revised: 26 December 2008 / Accepted: 5 February 2009

Abstract In this work, we present a method, called Two-Phase Pareto Local Search, to find a good approximation of the efficient set of the biobjective traveling salesman problem. In the first phase of the method, an initial population composed of a good approximation of the extreme supported efficient solutions is generated. We use as second phase a Pareto Local Search method applied to each solution of the initial population. We show that using the combination of these two techniques: good initial population generation plus Pareto Local Search gives better results than state-of-the-art algorithms. Two other points are introduced: the notion of ideal set and a simple way to produce near-efficient solutions of multiobjective problems, by using an efficient single-objective solver with a data perturbation technique.

Keywords Combinatorial multiobjective optimization · Biobjective traveling salesman problem · Two-phase method · Supported efficient solutions · Pareto local search · Data perturbation technique · Performance assessment of multiobjective metaheuristics

1 Introduction

The Pareto Local Search (PLS) method (Angel et al. 2004; Basseur 2006; Paquete et al. 2004) is one of the simplest method for resolving combinatorial multiobjective optimization problems. This method is the generalization in the multiobjective case of a basic metaheuristic: the hill-climbing method.

T. Lust thanks the "Fonds National de la Recherche Scientifique" for a research fellow grant (Aspirant FNRS).

T. Lust (✉) · J. Teghem
Faculté Polytechnique de Mons, Laboratory of Mathematics and Operational Research, Mons, Belgium
e-mail: thibaut.lust@fpms.ac.be

Published online: 21 February 2009

Paquete et al. (2004) have exploited this method but only with a theoretical point of view. First, to provide reference sets for the assessment of results obtained with more complex algorithms and secondly, to study the connectedness between the efficient solutions (Ehrgott and Klamroth 1997), by identifying cluster of potentially efficient solutions (Paquete et al. 2004; Paquete and Stützle 2006).

However, we show in this paper that this method can be very efficient if the method starts from a population of good quality, in the place of using only one random solution as starting solution (as done by Paquete et al. 2004). The initial population is composed of potentially extreme supported efficient solutions, obtained with the method of Aneja and Nair (1979). By starting the PLS method from this population, we show two interesting results: the convergence time of the PLS method is reduced and the results obtained are better than with the original version of Paquete et al.

At final, we obtain a simple method, called Two-Phase Pareto Local Search (2PPLS), with no numerical parameters and with a natural stop criterion. We show that this method gives better results for several indicators than state-of-the-art algorithms, quite complicated methods requiring sometimes many parameters.

The paper is organized as follows: in the next section, we present general definitions used in the rest of the paper. In the third section, we define the biobjective traveling salesman problem. The Sect. 4 is dedicated to the presentation of the PLS method. After that we present our contribution, that is the Two-Phase Pareto Local Search method, where we focus ourselves on the presentation of the first phase, given that we use the PLS method as second phase. The following sections are devoted to the application of the 2PPLS method to the biobjective traveling salesman problem. The different quality indicators used are presented at Sect. 6, as well as the concept of ideal set. The different results are presented at Sect. 7. We finally present at Sect. 8 a simple way to improve the quality of the results given by the 2PPLS method by using a single-objective solver with a data perturbation technique.

2 Multiobjective combinatorial optimization

A multiobjective combinatorial optimization problems is defined as follows:

$$\begin{aligned} \text{“}\min_{x}\text{”} \quad & z(x) = Cx \\ \text{subject to} \quad & Ax = b \\ & x \in \{0,1\}^n \end{aligned}$$

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

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

We denote by X the feasible set in the decision space, defined by $X = \{x \in \{0,1\}^n : Ax = b\}$. The feasible set in objective space is called Z and is defined by $Z = z(X) =$

$\{Cx : x \in X\} \subset \mathbb{N}^p \subset \mathbb{R}^p$. Due to the contradictory features of the objectives, it does not exist a feasible solution simultaneously minimizing each objective but a set of feasible solutions called *efficient solutions*. We present below some definitions that characterize these efficient solutions.

We first define four different dominance relations:

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 $u \prec v$.

Definition 2 (Strict dominance relation of Pareto) We say that a vector $u = (u_1, \ldots, u_p)$ *strictly dominates* a vector $v = (v_1, \ldots, v_p)$ if, and only if, $u_k < v_k \ \forall k \in \{1, \ldots, p\}$. We denote this relation by $u < v$.

Definition 3 (Weak dominance relation of Pareto) We say that a vector $u = (u_1, \ldots, u_p)$ *weakly dominates* a vector $v = (v_1, \ldots, v_p)$ if, and only if, $u_k \leq v_k \ \forall k \in \{1, \ldots, p\}$. We denote this relation by $u \leq v$.

We have:

$$u < v \Rightarrow u \prec v \Rightarrow u \leq v.$$

Definition 4 (ϵ-dominance) We say that a vector $u = (u_1, \ldots, u_p)$ *ϵ-dominates* a vector $v = (v_1, \ldots, v_p)$ if, and only if, $u_k \leq \epsilon \cdot v_k \ \forall k \in \{1, \ldots, p\}$, with $\epsilon \in \mathbb{R}^+$. We denote this relation by $u \leq_\epsilon v$.

For this last definition, if it exists at least one value for ϵ in $(0, 1)$ such that $u \leq_\epsilon v$, then $u \prec v$. If it is not the case, $v \leq u$.

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

Definition 5 (Efficient solution) A feasible solution $x^* \in X$ is called *efficient* if there does not exist any other feasible solution $x \in X$ such that $z(x) \prec z(x^*)$.

Definition 6 (Non-dominated point) The image $z(x^*)$ in objective space of an efficient solution x^* is called a non-dominated point.

Definition 7 (Efficient set) The efficient set denoted by X_E contains all the efficient solutions.

Definition 8 (Pareto front) The image of the efficient set in Z is called the Pareto front (or non-dominated frontier), and is denoted by Z_N.

We also add the definition of a weakly efficient solution:

Definition 9 (Weakly efficient solution) A feasible solution $x^* \in X$ is called *weakly efficient* if there does not exist any other feasible solution $x \in X$ such that $z(x) < z(x^*)$.

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

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

$$\min \left\{ \sum_{k=1}^{p} \lambda_k z_k(x) : x \in X \right\}$$

  for some vector $\lambda > 0$, that is each component is positive ($\lambda_k > 0, \forall\ k \in \{1, \ldots, p\}$). The image in objective space of the supported efficient solutions, called supported non-dominated points, are located on the "lowerleft boundary" of the convex hull of Z (conv Z), that is they are non-dominated points of (conv Z) $+ \mathbb{R}^p_+$. We can obtain all supported solutions by varying the weight set λ and by solving the corresponding weighted sum single-objective problems. Supported efficient solutions are denoted by X_{SE} and supported non-dominated points by Z_{SN}.
- 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 Z) $+ \mathbb{R}^p_+$. Non-supported efficient solutions are denoted by X_{NE} and non-supported non-dominated points by Z_{NN}.

We can also make a distinction between supported efficient solutions and define extreme supported efficient solutions and non-extreme supported efficient solutions (Ehrgott 2005).

- Extreme supported efficient solutions: the objective vectors $z(x)$ of these supported efficient solutions (called extreme supported non-dominated points) are located on the vertex set of (conv Z), that is there are extreme points of (conv Z) $+ \mathbb{R}^p_+$. We call the extreme supported efficient solutions X_{SE1} and the image $z(X_{SE1})$ of X_{SE1} in objective space is called Z_{SN1}.
- Non-extreme supported efficient solutions: the objective vectors $z(x)$ of these supported efficient solutions (called non-extreme supported non-dominated points) are not located on the vertex set of (conv Z) and located in the relative interior of the faces of (conv Z) $+ \mathbb{R}^p_+$. This set is denoted by X_{SE2} and the image $z(X_{SE2})$ of X_{SE2} in objective space by Z_{SN2}.

It is also important to introduce the following classification of the set X_E of efficient solutions (Hansen 1979).

Definition 10 (Equivalent solutions) Two solutions $x_1, x_2 \in X_E$ are equivalent if $z(x_1) = z(x_2)$.

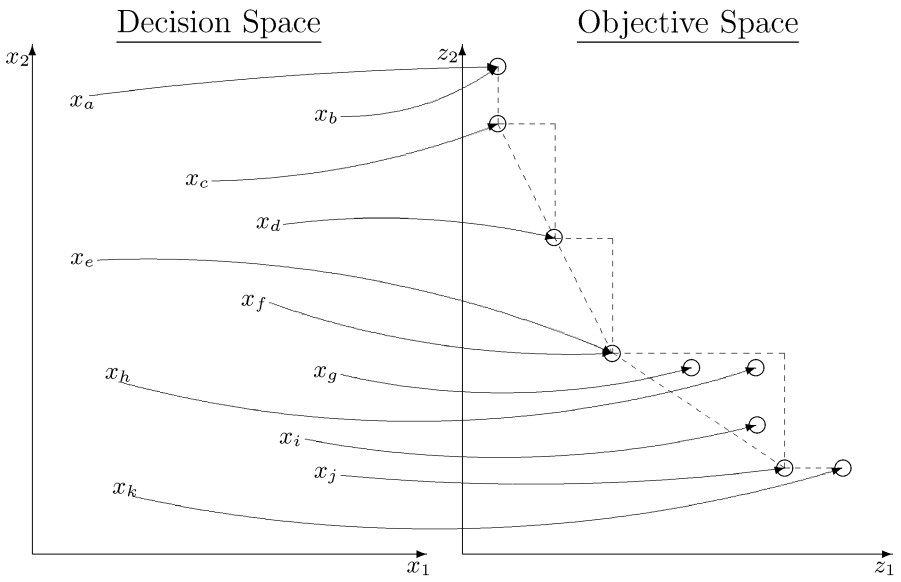

Fig. 1 Representation in decision space (with an arbitrary dimension of two) and objective space of different types of efficient solutions. From this figure: set of weakly efficient solutions $= \{x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j, x_k\}$; $X_E = \{x_c, x_d, x_e, x_f, x_g, x_i, x_j\}$; $X_{SE} = \{x_c, x_d, x_e, x_f, x_j\}$; $X_{SE1} = \{x_c, x_e, x_f, x_j\}$; $X_{SE2} = \{x_d\}$; $X_{NE} = \{x_g, x_i\}$; a minimal complete set $= \{x_c, x_d, x_e, x_g, x_i, x_j\}$

Definition 11 (Complete set) A complete set X_{Ec} is a subset of X_E such that each $x \in X \backslash X_{Ec}$ is weakly dominated by at least one $x \in X_{Ec}$, that is either dominated by or equivalent to at least one $x \in X_{Ec}$. In other words, for each non-dominated point $z \in Z_N$ there exists at least one $x \in X_{Ec}$ with $z(x) = z$.

Definition 12 (Minimal complete set) A minimal complete set X_{Em} is a complete set without equivalent solutions. Any complete set contains at least a minimal complete set.

We have represented at Fig. 1 an example showing the different types of efficient solutions in the decision and objective space, as well as an example of a minimal complete set.

It is important to say that, in this work, we only try to find an approximation of a minimal complete set: no equivalent solution generated will be thus retained.

We add two final definitions, of the ideal point and the nadir point (Ehrgott 2005).

Definition 13 (Ideal point) We call z^I the ideal point, which has for coordinates in the objective space, the best values for each objective:

$$z_k^I = \min_{x \in X} z_k(x) = \min_{z \in Z_N} z_k, \quad k = 1, \ldots, p.$$

Definition 14 (Nadir point) We call z^N the nadir point which has for coordinates in the objective space, the worse values for each objective, for the solutions belonging to the efficient set:

$$z_k^N = \max_{x \in X_E} z_k(x) = \max_{z \in Z_N} z_k, \quad k = 1, \ldots, p.$$

3 The biobjective traveling salesman problem

Given a set $\{v_1, v_2, \ldots, v_N\}$ of cities and two costs $c_{i,j}^1$ and $c_{i,j}^2$ 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 ($k = 1, 2$):

$$\text{“min”}\ z_k(\pi) = \sum_{i=1}^{N-1} c_{\pi(i),\pi(i+1)}^k + c_{\pi(N),\pi(1)}^k.$$

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 $c_{i,j}^k = c_{j,i}^k$ for $1 \leq i, j \leq N$ and $k = (1, 2)$.

4 Pareto local search

The Pareto Local Search (PLS) method is a purely local search algorithm used and developed by different authors (Angel et al. 2004; Basseur 2006; Paquete et al. 2004). The method, based on the notion of *Pareto local optimum set*, does not require any objectives aggregation nor any numerical parameters. In PLS, the neighborhood of every solution of a population is explored, and if the neighbor is not weakly dominated by a solution of the list of potentially efficient solutions, the neighbor is added to the population and to the list of potentially efficient solutions. The method stops when it is no more possible to find new non-weakly dominated neighbors starting from a solution of the population.

We first present some definitions. Afterwards, we present two different versions of the PLS method.

4.1 Definitions

First, we define a *Pareto local optimum*, which is a generalization, in the multiobjective case, of the concept of local optimum.

Definition 15 (Pareto local optimum; Paquete et al. 2004) Let $p \in X$ and $\mathcal{N}$ a neighborhood of p. We say that a solution p is a *Pareto local optimum* with respect to (w.r.t.) $\mathcal{N}$ if, and only if, there is no solution $p' \in \mathcal{N}(p)$ such that $z(p') \prec z(p)$.

As in multiobjective optimization, we are looking for a set of solutions, we define the notion of *Pareto local optimum set*.

Definition 16 (Pareto local optimum set; Paquete et al. 2004) We say that P is a *Pareto local optimum set* w.r.t. $\mathcal{N}$ if, and only if, P contains only Pareto local optimum solutions w.r.t. $\mathcal{N}$.

But, as this set may contain dominated solutions, we define the notion of *non-dominated Pareto local optimum set*.

Definition 17 (Non-dominated Pareto local optimum set; Paquete 2005) We say that P is a *non-dominated Pareto local optimum set* w.r.t. $\mathcal{N}$ if, and only if, it is a Pareto Local optimum set w.r.t. $\mathcal{N}$ and it is a non-dominated set, that is:

$$\forall\, p \in P,\ \nexists\, p' \in P \,|\, z(p') \prec z(p).$$

Also, as a non-dominated Pareto local optimum set w.r.t. $\mathcal{N}$ can still be improved by adding solutions $p' \in \mathcal{N}(p)$ such that $z(p) \not\prec z(p')$, we define the notion of *critical non-dominated Pareto local optimum set*.

Definition 18 (Critical non-dominated Pareto local optimum set) We say than P is a *critical non-dominated Pareto local optimum set* w.r.t. $\mathcal{N}$ if, and only if, it is a non-dominated Pareto local optimum set and

$$\forall\, p \in P, \forall\, p' \in (\mathcal{N}(p) \backslash P), \exists\, p'' \in P \,|\, z(p'') \prec z(p').$$

Finally, as realized in Definition 12, we define the notion of *minimal critical non-dominated Pareto local optimum set*.

Definition 19 (Minimal critical non-dominated Pareto local optimum set) We say that P is a *minimal critical non-dominated Pareto local optimum set* w.r.t. $\mathcal{N}$ if, and only if, it is a critical non-dominated Pareto local optimum set w.r.t. $\mathcal{N}$ without equivalent solutions.

Note that a critical non-dominated Pareto local optimum set and a minimal critical non-dominated Pareto local optimum set are both considered as *maximal Pareto local optimum set* by Paquete et al. (2007) and defined by:

$$\forall\, p \in P, \forall\, p' \in \mathcal{N}(p), \exists\, p'' \in P \,|\, z(p'') \preceq z(p').$$

Obviously, the efficient set X_E is a critical non-dominated Pareto local optimum set w.r.t. any neighborhood $\mathcal{N}$, because if it was not the case, the set X_E could be improved by adding a non-dominated neighbor.

4.2 Different versions of PLS

We present here the two different versions of the `PLS` method experimented in this work. Both versions need an initial population; that can be composed of a single

Procedure 1 `AddSolution`

Parameters ↕: A list $\widehat{X_E}$ of potentially efficient solutions
Parameters ↓: a solution p, its evaluation $z(p)$
Parameters ↑: *Added* (optional)

Added ← *true*
for all $x \in \widehat{X_E}$ **do**
  if $z(x) \preceq z(p)$ **then**
    Added ← *false*
    Break - -| We leave the for loop
  if $z(p) \prec z(x)$ **then**
    $\widehat{X_E} \leftarrow \widehat{X_E} \setminus \{x\}$
if *Added* = *true* **then**
  $\widehat{X_E} \leftarrow \widehat{X_E} \cup \{p\}$

solution or of a list of potentially efficient solutions. The method returns a minimal critical non-dominated Pareto local optimum set.

We first give the pseudo-code of the Procedure 1, called `AddSolution`, which is used by each version of the `PLS` method. In the different procedures or algorithms, the symbols ↓, ↑ and ↕ specify respectively the transmission mode IN, OUT and IN OUT of a parameter to a procedure or an algorithm. The symbol - -| marks the beginning of a comment line.

The Procedure 1 simply consists of updating an approximation $\widehat{X_E}$ of the efficient set, when a new solution p is added to $\widehat{X_E}$. This procedure has four parameters, the set $\widehat{X_E}$ to actualize, the new solution p, its evaluation $z(p)$ and an optional boolean variable called *Added* that returns *True* if the new solution has been added.

PLS method of Angel et al. Angel et al. (2004) have proposed a `PLS` method, that they call `BLS` for Bicriteria Local Search. Even though, we call this version `PLS1`. This version has also been used by Basseur (2006) as local search in a memetic algorithm.

The pseudo-code of the `PLS1` method is given by the Algorithm 1.

The method starts with a population P composed of potentially efficient solutions given by the initial population P_0, which is an input parameter of the method. Then, all the neighbors p' of each solution p of P are generated. If a neighbor p' is not weakly dominated by the current solution p, we try to add the solution p' to the approximation $\widehat{X_E}$ of the efficient set, which is updated with the Procedure 1. If the solution p' has been added to $\widehat{X_E}$, the boolean variable *Added* is true and we add the solution p' to an auxiliary population P_a, which is also updated with the Procedure 1. Once all the neighbors of each solution of P have been generated, the algorithm starts again, with P equal to P_a, until $P = \emptyset$. The auxiliary population is used such that the neighborhood of each solution of the population P is explored, even if some solutions of P become dominated by a new solution. Thus, sometimes, in this version, neighbors are generated from a dominated solution.

Algorithm 1 `PLS1`

Parameters $\downarrow$: An initial population P_0, the cost matrices C_1 and C_2 of the bTSP
Parameters $\uparrow$: An approximation $\widehat{X_E}$ of the efficient set

--| Initialization of $\widehat{X_E}$ and a population P with the initial population P_0
$\widehat{X_E} \leftarrow P_0$
$P \leftarrow P_0$
--| Initialization of an auxiliary population P_a
$P_a \leftarrow \emptyset$
while $P \neq \emptyset$ **do**
  --| Generation of all the neighbors p' of each solution $p \in P$
  for all $p \in P$ **do**
    for all $p' \in \mathcal{N}(p)$ **do**
      if $z(p) \nleq z(p')$ **then**
        `AddSolution`($\widehat{X_E} \updownarrow$, $p' \downarrow$, $z(p') \downarrow$, $Added \uparrow$)
        if $Added = true$ **then**
          `AddSolution`($P_a \updownarrow$, $p' \downarrow$, $z(p') \downarrow$)
  --| P is composed of the new potentially efficient solutions
  $P \leftarrow P_a$
  --| Reinitialization of P_a
  $P_a \leftarrow \emptyset$

Algorithm 2 `PLS2`

Parameters $\downarrow$: An initial population P_0, the cost matrices C_1 and C_2 of the bTSP
Parameters $\uparrow$: An approximation $\widehat{X_E}$ of the efficient set

--| Initialization of $\widehat{X_E}$ and a population P with the initial population P_0
$\widehat{X_E} \leftarrow P_0$
$P \leftarrow P_0$
while $P \neq \emptyset$ **do**
  --| Generation of all the neighbors p' of each solution $p \in P$
  for all $p \in P$ **do**
    $Deleted \leftarrow false$
    for all $p' \in \mathcal{N}(p)$ **do**
      if $z(p) \nleq z(p')$ **then**
        `AddSolution`($\widehat{X_E} \updownarrow$, $p' \downarrow$, $z(p') \downarrow$, $Added \uparrow$)
        if $Added = true$ **then**
          `AddSolution`($P \updownarrow$, $p' \downarrow$, $z(p') \downarrow$)
      if $z(p') \prec z(p)$ **then**
        $Deleted \leftarrow true$
  --| We remove p from P if p has not already been removed
  if $Deleted = false$ **then**
    $P \leftarrow P \setminus \{p\}$

PLS method of Paquete et al. In the PLS method of Paquete et al. (2004), called PLS2, the population P is immediately updated after the addition of a neighbor. Therefore, contrary to PLS1, the neighbors are never generated from a dominated solution. The main difference in the pseudo-code of the PLS2 method given by the Algorithm 2, compared to that of the PLS1 method, is that no auxiliary population is used.

We just have to employ a boolean variable *Deleted* to check if the current solution p has been eliminated of the population P following the addition of a neighbor p' of p. If it is not the case, the solution p has to be removed from P as the neighborhood of p has already been explored.

Therefore, the results given by PLS2 depend on the order according to which the solutions of P are examined, which was not the case with PLS1.

5 Two-phase Pareto local search

The spirit of the two phases of the Two-Phase Pareto Local Search method is similar to that of the exact two-phase method developed by Ulungu and Teghem (1995), but here, approximation methods are used in both phases. The two phases of the method are as follows:

1. Phase 1: Find a good approximation of the supported efficient solutions. These solutions can be obtained by resolution of weighted sum single-objective problems obtained by applying a linear aggregation of the objectives. We limit ourselves to find a good approximation of a minimal complete set of extreme supported efficient solutions, called $\widehat{X_{SE1_m}}$.
2. Phase 2: Find a good approximation of the non-supported efficient solutions located between the supported efficient solutions. In this phase, we apply the PLS method.

It should be noted that the Two-Phase Pareto Local Search method is close to the method of Hamacher and Ruhe (1994), developed in 1994, to find a well distributed approximation of the efficient solutions of the multiobjective spanning tree problem. However, two differences can be noticed: they use the exact set of extreme efficient solutions in phase 1 and a stop criterion related to the distance between two consecutive solutions is applied in phase 2. This method has only been applied to the multiobjective spanning tree problem.

5.1 First phase: generation of the initial population $\widehat{X_{SE1_m}}$

We employ the method of Aneja and Nair (1979), initially proposed for the resolution of a bicriteria transportation problem. This method consists in generating all the weight sets which make it possible to obtain a minimal complete set of extreme supported efficient solutions of a biobjective problem (non-extreme supported efficient solutions and equivalent solutions can however be generated). For each weight set generated, a linear aggregation of the objectives is carried out and the single-objective problem obtained is solved by an exact method. In this work, as the use

of an exact method to solve the single-objective problems is very time-consuming, we have heuristically adapted the method of Aneja and Nair. As a result, the solutions obtained are not necessarily efficient, nor supported efficient but constitute a set of solutions very close to a minimal complete set of extreme supported efficient solutions.

We first present the exact method for generating a minimal complete set of extreme supported efficient solutions X_{SE1_m}. It should be noted that Przybylski et al. (2008) have recently developed a method allowing to generate all the supported efficient solutions of a biobjective problem, but that requires the integration of an enumerative method (to generate all the optimal solutions of a single-objective problem).

After that, we present the adaptation of this method to find a good approximation of the extreme supported efficient solutions, noted $\widehat{X_{SE1_m}}$.

Exact method for generating X_{SE1_m} To find a minimal complete set of extreme supported efficient solutions, we follow the dichotomic scheme proposed by Aneja and Nair (1979). The details of the algorithm are given by the Algorithm 3 called `BTSP_Phase1Exact`, containing the Algorithm 4 called `SolveRecursionExact`. First, the set S containing the supported efficient solutions is initialized with both lexicographic optimal solutions corresponding to $\text{lexmin}_{x\in X}(z_1(x), z_2(x))$ and $\text{lexmin}_{x\in X}(z_2(x), z_1(x))$, respectively.

To compute a lexicographic optimal solution, we use the method proposed by Przybylski et al. (2008), which consists of solving successively two single-objective problems. For computing the lexicographic optimal solution x'_1 corresponding to $\text{lexmin}_{x\in X}(z_1(x), z_2(x))$, we first resolve the TSP problem by only considering the first objective; the weight set is this equal to $(1, 0)$ (the optimal solution of this problem is called x_1). Then, to improve the second objective without degrading the first objective, we solve the single-objective problem with a weight set defined by the normal to the line through $(z_1(x_1), z_2(x_1))$ and $(z_1(x_1) + 1, -1)$, that is $\lambda = (z_2(x_1) + 1, 1)$, since the $c^k_{i,j}$ values are supposed to belong to $\mathbb{N}$, and thus $Z \subset \mathbb{N}^2$. The optimal solution x'_1 obtained is an optimal solution of $\text{lexmin}_{x\in X}(z_1(x), z_2(x))$.

The second lexicographic optimal solution x'_2 corresponding to $\text{lexmin}_{x\in X}$ $(z_2(x), z_1(x))$ is found in the same way by using x_2, solution of the TSP problem with only the second objective considered, and solving an additional single-objective problem with $\lambda = (1, z_1(x_2) + 1)$.

Once both lexicographic optimal solutions have been generated, the dichotomic scheme starts and the Algorithm 4, called `SolveRecursionExact`, is launched.

The dichotomic scheme is initialized with both lexicographic solutions x'_1 and x'_2 ($x_r = x'_1$ and $x_s = x'_2$). Then, a single-objective problem with a weight set defined by the normal to the line through $(z(x_r), z(x_s))$ is solved. The corresponding weight set is equal to $(z_2(x_r) - z_2(x_s), z_1(x_s) - z_1(x_r))$. The solution of this problem is called x_t.

Let us note the line segment joining two points y_r and y_s in $\mathbb{R}^2$, by $\overline{y_r y_s}$.

If $z(x_t) \cap \overline{z(x_r)z(x_s)} = \emptyset$, that is if $z(x_t)$ is not on the line segment joining both points $z(x_r)$ and $z(x_s)$, we start again the dichotomic scheme two times: with x_r and x_t, and with x_t and x_s as starting solutions of the `SolveRecursionExact` algorithm, as we can see at Fig. 2.

Algorithm 3 `BTSP_Phase1Exact`

Parameters $\downarrow$: The cost matrices C_1 and C_2 of the bTSP
Parameters $\uparrow$: The set X_{SE1_m}

--| Compute the lexicographically optimal solution $x_{1'}$ for (z_1, z_2)

--| Solve the single-objective TSP by only considering the first objective
`SolveTSP`$(C^1 \downarrow, x_1 \uparrow)$
--| Create a new cost matrix $C^{1'}$
$C^{1'} = [\lambda_1 c^1_{i,j} + \lambda_2 c^2_{i,j}]$ with $\lambda_1 = z_2(x_1) + 1$ and $\lambda_2 = 1$
--| Solve the single-objective TSP by considering the cost matrix $C^{1'}$
`SolveTSP`$(C^{1'} \downarrow, x_{1'} \uparrow)$

--| Compute the lexicographically optimal solution $x_{2'}$ for (z_2, z_1)

--| Solve the single-objective TSP by only considering the second objective
`SolveTSP`$(C^2 \downarrow, x_2 \uparrow)$
--| Create a new cost matrix $C^{2'}$
$C^{2'} = [\lambda_1 c^1_{i,j} + \lambda_2 c^2_{i,j}]$ with $\lambda_1 = 1$ and $\lambda_2 = z_1(x_2) + 1$
--| Solve the single-objective TSP by considering the cost matrix $C^{2'}$
`SolveTSP`$(C^{2'} \downarrow, x_{2'} \uparrow)$

--| Compute X_{SE1_m}

$S \leftarrow \{x_{1'}, x_{2'}\}$
`SolveRecursionExact`$(x_{1'} \downarrow, x_{2'} \downarrow, S \updownarrow)$
$X_{SE1m} \leftarrow S$

Algorithm 4 `SolveRecursionExact`

Parameters $\downarrow$: x_r, x_s
Parameters $\updownarrow$: S

--| Create a new cost matrix C^λ
$C^\lambda = [\lambda_1 c^1_{ij} + \lambda_2 c^2_{ij}]$ with $\lambda_1 = z_2(x_r) - z_2(x_s)$, and $\lambda_2 = z_1(x_s) - z_1(x_r)$
--| Solve the single-objective TSP by considering the cost matrix C^λ
`SolveTSP`$(C^\lambda \downarrow, x_t \uparrow)$
$S \leftarrow S \cup \{x_t\}$

if $z(x_t) \cap \overline{z(x_r)z(x_s)} = \emptyset$ **then**
  `SolveRecursionExact`$(x_r \downarrow, x_t \downarrow, S \updownarrow)$
  `SolveRecursionExact`$(x_t \downarrow, x_s \downarrow, S \updownarrow)$

Otherwise, the `SolveRecursionExact` algorithm is stopped. Indeed, as shown in Fig. 3, the supported solution x_t is not an extreme one, and the weight set generated from x_r and x_t or from x_t and x_s will be the same as that defined by x_r and x_s. Therefore, no new extreme supported solution will be generated.

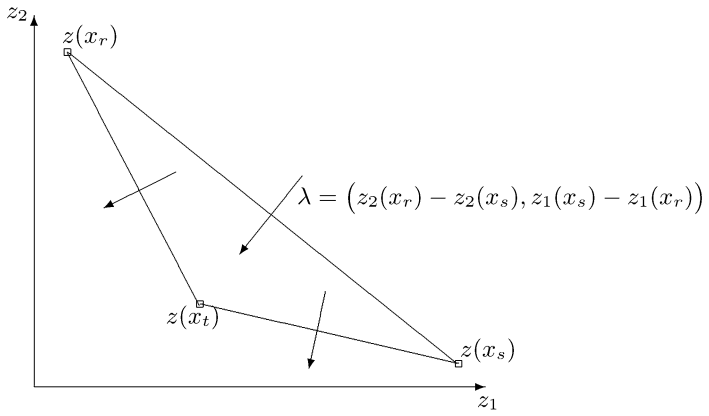

Fig. 2 Dichotomic scheme

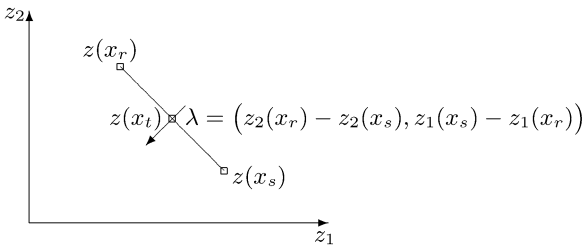

Fig. 3 Stop criterion of the exact method for generating the extreme supported efficient solutions

Approximation of the extreme supported efficient solutions To find an approximation $\widehat{X_{SE1_m}}$ of the extreme supported efficient solutions, we follow the same scheme as the exact method, except than a heuristic method is used to solve the different single-objective problems. In addition, two simplifications are considered:

- A search for the lexicographic solutions is not realized. We are satisfied with the solutions given by the heuristic for the resolution of single-objective problems by considering only one objective.
- Because the values taken by the weight sets given by the exact method can be very large, we normalize the weight sets such that $\lambda_1 + \lambda_2 = 1$. In consequence, we round the coefficients of the matrix C^{λ} to the nearest integer value, as λ is a real vector.

The details of the algorithm are given by the Algorithm 5 called `BTSP_Phase1-Heuristic` which uses the Algorithm 6 called `SolveRecursionHeuristic`.

The stop criterion of the `BTSP_Phase1Heuristic` algorithm is slightly different than in the `BTSP_Phase1Exact` algorithm. When we seek for a new solution, to start again the `SolveRecursionHeuristic` algorithm, the solution must be located in the interior of the triangle defined by the solutions x_r and x_s plus the local ideal point formed by these two solutions (see Fig. 4). Indeed, with a heuristic method, the solution x_t can also be located outside of this triangle. Launching the procedure from solutions which are certainly not supported, would reduce the chances

Algorithm 5 `BTSP_Phase1Heuristic`

Parameters ↓: The cost matrices C_1 and C_2 of the bTSP
Parameters ↑: An approximation $\widehat{X_{SE1_m}}$ of X_{SE1_m}

--| Solve the single-objective TSP by only considering the first objective
`HeuristicTSP`($C^1 \downarrow, x_1 \uparrow$)

--| Solve the single-objective TSP by only considering the second objective
`HeuristicTSP`($C^2 \downarrow, x_2 \uparrow$)

--| Compute an approximation of X_{SE1_m}

$\widehat{S} \leftarrow \{x_1, x_2\}$
`SolveRecursionHeuristic`($x_1 \downarrow, x_2 \downarrow, \widehat{S} \updownarrow$)
$\widehat{X_{SE1_m}} \leftarrow \widehat{S}$

Algorithm 6 `SolveRecursionHeuristic`

Parameters ↓: x_r, x_s
Parameters ↕: $\widehat{S}$

--| Create a new cost matrix C^λ
$C^\lambda = \text{round}([\lambda_1 c^1_{ij} + \lambda_2 c^2_{ij}])$ with $\lambda_1 = \frac{z_2(x_r) - z_2(x_s)}{z_2(x_r) - z_2(x_s) + z_1(x_s) - z_1(x_r)}$ and $\lambda_2 = 1 - \lambda_1$
--| Solve the single-objective TSP by considering the cost matrix C^λ
`HeuristicTSP`($C^\lambda \downarrow, x_t \uparrow$)
--| Add x_t to $\widehat{S}$
`AddSolution`($\widehat{S} \updownarrow, x_t \downarrow, z(x_t) \downarrow$)

if $z(x_t) \cap \text{int} \triangle z(x_r)z(x_s) \neq \emptyset$ **then**
  `SolveRecursionHeuristic`($x_r \downarrow, x_t \downarrow, \widehat{S} \updownarrow$)
  `SolveRecursionHeuristic`($x_t \downarrow, x_s \downarrow, \widehat{S} \updownarrow$)

of finding new supported efficient solutions. We note the interior of the triangle by $\text{int} \triangle z(x_r)z(x_s)$. Therefore, if $z(x_t) \cap \text{int} \triangle z(x_r)z(x_s) = \emptyset$, the `SolveRecursionHeuristic` algorithm is stopped.

Also, as the new solution x_t can be dominated by other solutions already generated, we use the Procedure 1 to update the set $\widehat{S}$ containing the potentially efficient solutions.

5.2 Second phase: find non-supported efficient solutions

In this phase, we use the `PLS` method. However, Paquete et al. (2004) and Angel et al. (2004) start their method from a randomly generated solution, whereas we use each solution of the population $\widehat{X_{SE1_m}}$ generated in phase 1 as initial solutions.

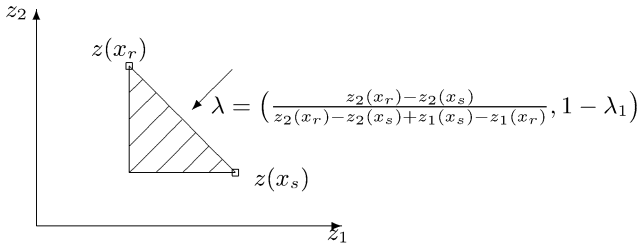

Fig. 4 Stop criterion of the heuristic method for generating an approximation of the extreme supported efficient solutions

6 Quality indicators

6.1 Quality indicators used

In single-objective optimization, it is quite easy to measure the quality of a solution or to compare the solutions obtained by various methods. That is more difficult in the multicriteria case and it remains an open problem (Zitzler et al. 2002), because the solutions are represented by a trade-off surface composed of a solution set. Consequently, we use several indicators to measure the quality of an approximation A of the efficient set.

We use the following unary indicators in this work:

- the hypervolume $\mathcal{H}$ (to be maximized) (Zitzler 1999): approximation of the volume included under the curve formed by the non-dominated set, that is the representation of the solutions of the approximation in objective space.
- the R measure (normalized between 0 and 1, to be maximized) (Jaszkiewicz 2000): evaluates a non-dominated set by the expected value of the weighted Tchebycheff utility function over a set of normalized weight vectors.
- the average distance D_1 and maximal distance D_2 (to be minimized) (Czyzak and Jaszkiewicz 1998; Ulungu et al. 1999) between a reference set of good quality and the non-dominated set, by using the Euclidean distance. Ideally, the reference set is the Pareto front.

We also take into account the binary ϵ-indicator, based on the ϵ-dominance relation. If we consider two approximations A and B, the ϵ-indicator I_ϵ (Zitzler et al. 2003) gives the ϵ factor by which the approximation A set is worse (if $\epsilon > 1$) or better (if $\epsilon < 1$) than the approximation B with respect to all objectives:

$$I_\epsilon(A, B) = \inf_{\epsilon \in \mathbb{R}^+} \{\forall b \in B, \exists a \in A : a \preceq_\epsilon b\}.$$

This indicator can also be used as an unary indicator if we use as set B a reference set of good quality. We denote this unary indicator by $I_{\epsilon 1}$.

To interpret the powerfulness of these indicators, let us consider that an approximation A is better than an another approximation B if every $b \in B$ is weakly dominated by at least one $a \in A$ and $A \neq B$ (Zitzler et al. 2003). We denote this by $A \triangleleft B$.

We have the following relations (Zitzler et al. 2003):

- $A \lhd B \Rightarrow I_{\epsilon 1}(A) \leq I_{\epsilon 1}(B)$.
- $A \lhd B \Rightarrow D_1(A) \leq D_1(B)$.
- $A \lhd B \Rightarrow D_2(A) \leq D_2(B)$.
- $A \lhd B \Rightarrow \mathcal{H}(A) \geq \mathcal{H}(B)$.
- $A \lhd B \Rightarrow R(A) \geq R(B)$.

Unfortunately, none of these unary indicators allow to affirm that $A \lhd B$, that is to say we can not inverse the preceding relations. For example, we can have two incomparable approximation sets A and B (neither A weakly dominates B nor B weakly dominates A) with $I_{\epsilon 1}(A) < I_{\epsilon 1}(B)$.

In conclusion, as pointed by Zitzler et al. (2003), with these unary indicators, we will not be able to say that an approximation A is better than an approximation B but just A is not worse than B, that means that either A is better than B or incomparable to B.

Even though, an approximation A that finds better values for the indicators used in this work is often preferred to an approximation B, since these indicators reflect well the preferences of the decision maker.

6.2 Reference set

As said before, to compute the D_1, D_2 and $I_{\epsilon 1}$ indicators, it is important to have a reference set of excellent quality. Two typical techniques to compute this set are as follows (Knowles et al. 2006):

- Generate the exact Pareto front.
- Merge all the approximations generated by state-of-the-art algorithms. The dominated points are removed from this union and the remaining points, which are not dominated by any of the approximations, form the reference set.

Concerning the first approach, we have tried to resolve the bTSP instances used in this work with an exact method, but we did not manage to obtain the Pareto front, because of a too high running time.

One advantage of the second approach is that the reference set weakly dominates all the approximations under consideration. But this is not the case if a new approximation is compared to this reference set. In this case, the reference set should be normally re-computed whenever additional approximations are included in the comparison, which is not very convenient. Furthermore, for some instances used in this work, it was not possible to generate good reference sets with the approximations generated by state-of-the-art algorithms, because of too few results published. Therefore, the indicator D_1 and D_2 would be biased if reference sets of bad quality are used.

As both techniques are not satisfactory, we introduce the notion of *ideal set* for biobjective problems, which is a lower bound of the Pareto front (Ehrgott and Gandibleux 2007). The ideal set is defined as the best potential Pareto front that we can produce from a minimal complete set of extreme supported efficient solutions. Extreme supported efficient solutions are used since these solutions are easier to generate than non-extreme supported efficient solutions and non-supported efficient solutions. For instance, we can see at Fig. 5 the representation in objective

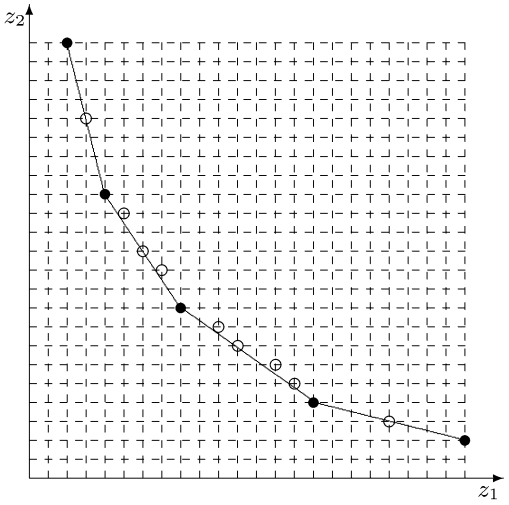

Fig. 5 Ideal set produced on the basis of five extreme supported non-dominated points

space of five extreme supported efficient solutions of a biobjective problem (filled black points). This set can only be improved by adding non-extreme supported non-dominated points or non-supported non-dominated points. Since the $c^k_{i,j}$ values of the bTSP are supposed to belong to $\mathbb{N}$, and thus $Z \subset \mathbb{N}^2$, it is easy to compute the best places than non-dominated points can possibly take. The coordinates of ideal non-extreme supported non-dominated points are the integer values located on the line between two consecutive extreme supported non-dominated points, and the coordinates of ideal non-supported non-dominated points are the integer values located the closest possible to this line. In Fig. 5, we have added these ideal non-extreme supported non-dominated points and ideal non-supported non-dominated points, represented by the circles.

Hence, it is impossible to improve this set with feasible solutions, and that is why this set is called ideal set. It gives an excellent lower bound of the Pareto front. At final, to pass from one solution to another, only a step of one unity is produced, for the objective 1 or 2, what depends on the gradient of the line between two consecutive extreme supported non-dominated points.

All feasible solutions are weakly dominated by a solution of the ideal set. Therefore it is impossible to find a feasible solution that dominates a solution of the ideal set. In addition, the ideal set has the advantage of being fixed, and does not have to be re-computed each time a new approximation is generated. It is thus a very interesting property and the distance between the ideal set and an approximation will be not biased.

An obvious disadvantage of the ideal set is that the cardinality of the set can be huge. A crude upper bound of this cardinality is equal to $\max(z_1^N - z_1^I + 1, z_2^N - z_2^I + 1)$. Let's call x_1 and x_2 two lexicographic optimal solutions of the problem, respectively equal to $\text{lexmin}_{x \in X}(z_1(x), z_2(x))$ and $\text{lexmin}_{x \in X}(z_2(x), z_1(x))$. To attain this upper bound, one property must be met: $z_1(x_1) + z_2(x_1) = z_1(x_2) + z_2(x_2)$ such that the line drawn between $z(x_1)$ and $z(x_2)$ has a linear coefficient equal to -1. As a result, $z(x_1)$ and $z(x_2)$ are the only two extreme non-dominated points of

the biobjective problem. Thus, $z_1(x^*) + z_2(x^*) = K$ for each solution x^* of the ideal set, where K is a constant value. For the biobjective TSP, this propriety occurs when $c_1(i, j) + c_2(i, j) = K$ for each couple of cities (i, j). Moreover, in this case, each feasible solution is efficient.

Nevertheless, it is not necessary to save all the solutions of the ideal set, as the ideal set is easy to compute. Also, the distance between the solutions of the ideal set and the solutions of an approximation can be exactly generated quickly as it is easy to sort out the solutions of a non-dominated set according to one of the objective by using a simple propriety of a non-dominated set in the biobjective case: for two solutions x_1 and x_2 belonging to a non-dominated set, $z_1(x_1) < z_1(x_2)$ implies that $z_2(x_1) > z_2(x_2)$. Therefore, no computational difficulties due to the high cardinality of the ideal set has appeared during the experimentations realized in this work.

7 Results

We experiment the Two-Phase Pareto Local Search (2PPLS) method on several instances of the biobjective TSP problem. We consider four different types of instances:

- Euclidean instances: the costs between the edges correspond to the Euclidean distance between two points in a plane, randomly located from a uniform distribution.
- Random instances: the costs between the edges are randomly generated from a uniform distribution.
- Mixed instances: the first cost comes from the Euclidean instance while the second cost comes from the random instance.
- Clustered instances: The points are randomly clustered in a plane, and the costs between the edges correspond to the Euclidean distance.

In this work, we use the following instances:

- Euclidean instances: six Euclidean instances of 100 cities (KroAB100, KroAC100, KroAD100, KroBC100, KroBD100, KroCD100), one instance of 150 cities (KroAB150) and one instance of 200 cities (KroAB200). We use three other Euclidean instances of respectively 100, 300 and 500 cities called EuclAB100, EuclAB300 and EuclAB500.
- Random instances: three random instances of respectively 100, 300 and 500 cities called RdAB100, RdAB300 and RdAB500.
- Mixed instances: three mixed instances of respectively 100, 300 and 500 cities called MixedAB100, MixedAB300 and MixedAB500.
- Clustered instances: three clustered instances of respectively 100, 300 and 500 cities called ClusteredAB100, ClusteredAB300 and ClusteredAB500.

The "Kro" instances have been generated on the basis of single-objective TSP instances of the TSPLIB library (Reinelt 1991). We use these instances because state-of-the-art results have been published for these instances.

We have generated ourselves the clustered instances with the generator available from the 8th DIMACS Implementation Challenge site[1] with the following property:

[1] http://www.research.att.com/~dsj/chtsp/download.html.

the number of clusters is equal to the number of cities divided by 25 and the maximal coordinate for a city is equal to 3163 (as done by Paquete for the Euclidean instance).

The other instances have been generated and published by Paquete (2005). Paquete used these instances in his thesis but he has not published the results obtained with its algorithms.

For generating the extreme supported non-dominated points necessary to produce the ideal sets, we have applied the Algorithm 3. However, for the instances of more than 200 cities, numerical problems were encountered. 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 bMST is a relaxation of the bTSP, each feasible solution of the bTSP is weakly dominated by at least one of the solutions of the ideal set of the bMST.

All the algorithms experimented in this work have been run on a Pentium with 3 GHz and 512 Mb of memory. We make the average of the indicators on 20 executions. The running time of our implementation of the algorithms corresponds to the wall clock time. For the "Kro" instances, the reference points used for computing the R and $\mathcal{H}$ indicators, can be found in Jaszkiewicz (2000) for the 100 cities instances and in Paquete and Stützle (2003) for the bigger instances. For the other instances, the reference points are determined according to the reference sets. For the R indicator, the number of weight sets used is equal to 101, for all the instances. This indicator has been normalized between 0 and 1, where 1 corresponds to the best value.

We first experiment different single-objective solvers for the first phase of `2PPLS`. Then, we test the two versions of the `PLS` method. After that, we compare the results of `2PPLS` with state-of-the-art algorithms. We finally study the influence of the size of the initial population on the results of `2PPLS`.

7.1 Method and algorithm for the first phase

The aim of the first phase is to find a good approximation of a minimal complete set of extreme supported efficient solutions, called $\widehat{X_{SE1_m}}$.

For the single-objective solver used in the Algorithm 5 (`BTSP_Phase1-Heuristic`) of the dichotomic scheme, we use one of the best heuristics for the single-objective TSP: the Lin-Kernighan heuristic (`LK`). The `LK` heuristic was initially proposed by Lin and Kernighan (1973), and is always considered as one of the most effective methods for generating near-optimal solutions of the single-objective symmetric TSP. Many different versions of the `LK` heuristic have already been implemented. In this work, we use two very efficient versions of the `LK` heuristic, both giving better results that the original version of Lin and Kernighan:

- The version implemented by Helsgaun (2000), that we call `LKH`. The code source of this method has been published on http://www.akira.ruc.dk/~keld.
- The chained Lin-Kernighan version of Applegate (2003), the code source of this method has been published on http://www.tsp.gatech.edu/concorde through the Concorde package, that includes various methods for solving exactly or heuristically the single-objective TSP. We call this method `LKC`.

In each solver, we use the default parameters given by the authors, except for the LKH solver, where the maximum number of candidate edges associated with each node has been fixed to 3 in the place of 5 for the instances of more than 200 cities, to reduce the running time of the first phase.

We can see the results of the application of the dichotomic scheme with the two different solvers at Table 1 (under the name Di-LKC and Di-LKH), for small instances. We also add the results generated by the exact method (Di-EXA) where the exact single-objective solver comes from the Concorde package. Four 100 cities instances of the bTSP are employed, of different types: kroAB100, RdAB100, MixedAB100 and ClusteredAB100, one instance of 150 cities KroAB150 and one instance of 200 cities KroAB200. We use different indicators to measure the quality of the approximations obtained: the $D_1(SE)$, $D_2(SE)$ and $I_{\epsilon 1}(SE)$ indicators that have the same meaning than the D_1, D_2 and $I_{\epsilon 1}$ indicators presented at Sect. 6.1, but as we are seeking in phase 1 an approximation of a minimal complete set of extreme supported efficient solutions, we use the exact minimal complete set given by the exact method as reference set. We also indicate the proportion PSE of extreme non-dominated points generated and as additional information, the number $|PE|$ of potentially efficient solutions obtained and the running time in seconds. The conclusions of this analysis are as follows:

- The LKH solver gives better results than the LKC solver. The differences in the values of the indicators are not large, but the differences are accentuating when the size of the instance increases. On the other hand, the running time with the LKC solver is lower, except for the RdAB100 and ClusteredAB100 instances.
- Presumable the most important result is that the quality of the approximations obtained with the heuristic method is very good. With the LKH solver, we find, at worst 79.95 percent of the solutions belonging to the extreme supported non-dominated points set (for the KroAB200 instance). For the instances of 100 cities, the worst results were obtained for the ClusteredAB100 instance (87.48) which remains good. The saving in term of running time ratio comparing to the exact method goes from 7 (ClusteredAB100 instance) to 24 (KroAB100 instance).

We can also remark than the number of potentially efficient solutions generated by a heuristic method can be higher than the number of non-dominated points generated by the exact method. It is because, with a heuristic method, some potentially non-supported efficient solutions are sometimes generated.

As we can see there are differences in the results between LKC and LKH although both solvers are known to be able to find the optimum of small single-objective TSP problems. However, for some weight sets, it seems that "difficult" instances of the TSP are generated. We have represented at Fig. 6 the histogram showing the repartition of the running time needed to solve with the exact method the single-objective instances obtained with the dichotomic scheme, for the KroAB100 instance. We remark that about 75% of the instances are easy to solve while they are some instances that are more difficult since the running time is much higher. That was also highlighted by Borges and Hansen (1998). The differences between LKC and LKH are thus probably accentuated on these difficult instances.

At Table 2, we show the results obtained with the Di-LKC and Di-LKH algorithms for instances of more than 200 cities, that is the EuclAB300, EuclAB500,

Table 1 Comparison of the LKC and LKH solvers for the first phase

Instance	Algorithm	$I_{\epsilon 1}(SE)$	$D_1(SE)$	$D_2(SE)$	PSE	$\|PE\|$	Time (s)
KroAB100	Di-EXA	1.000000	0.000	0.000	100.00	109.4	772.09
	Di-LKC	1.025895	0.342	6.867	88.21	109.60	**27.77**
	Di-LKH	**1.019012**	**0.197**	**7.158**	**91.65**	**110.85**	31.85
KroAB150	Di-EXA	1.000000	0.000	0.000	100.00	164.00	1226.90
	Di-LKC	1.024320	0.622	11.746	78.29	**160.60**	**59.46**
	Di-LKH	**1.021176**	**0.406**	**9.318**	**84.63**	154.90	93.21
KroAB200	Di-EXA	1.000000	0.000	0.000	100.00	222.00	2606.02
	Di-LKC	1.024008	0.709	10.451	65.61	**216.85**	**105.76**
	Di-LKH	**1.017208**	**0.452**	**9.025**	**79.95**	212.40	186.33
RdAB100	Di-EXA	1.000000	0.000	0.000	100.00	77.00	217.02
	Di-LKC	1.049386	1.015	13.786	76.95	**87.55**	25.28
	Di-LKH	**1.012554**	**0.117**	**5.832**	**96.56**	76.95	**22.76**
MixedAB100	Di-EXA	1.000000	0.000	0.000	100.00	98.00	358.28
	Di-LKC	1.029337	0.607	8.474	80.97	**106.00**	**25.97**
	Di-LKH	**1.025061**	**0.351**	**7.988**	**92.24**	93.00	26.57
ClusteredAB100	Di-EXA	1.000000	0.000	0.000	100.00	109.00	185.61
	Di-LKC	1.044280	1.251	17.072	76.24	94.00	48.49
	Di-LKH	**1.020360**	**0.379**	**9.156**	**87.48**	**104.60**	**25.41**

MixedAB300, MixedAB500, RdAB300, RdAB500, ClusteredAB300 and ClusteredAB500 instances. For theses instances, as obtaining the exact minimal complete set of extreme supported efficient solutions is very time-consuming, we use as reference set the ideal set defined in Sect. 6.2. We use five different indicators to measure the quality of the approximations obtained: the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2 indicators. We also add as additional information the number $|PE|$ of potentially efficient solutions generated and the running time.

We remark than none of the two solvers is better than the other on all the instances. On the other hand, the running time with LKH is lower than the running time with LKC on all the instances, except on the RdAB500 instances.

7.2 Method and algorithm for the second phase

We experiment here the two different versions of the PLS method presented at Sect. 4.2. We start both versions on four different populations: a solution randomly generated (Rd) as done by Angel et al. (2004) and Paquete et al. (2004) and on three other populations generated by the dichotomic scheme with the LKC solver, the LKH solver and the exact method. Therefore, for each instance, 8 methods are experimented.

In PLS, we use the well-known 2-exchange neighborhood, as suggested by Borges and Hansen in their work about the global convexity in bTSP (Borges and Hansen

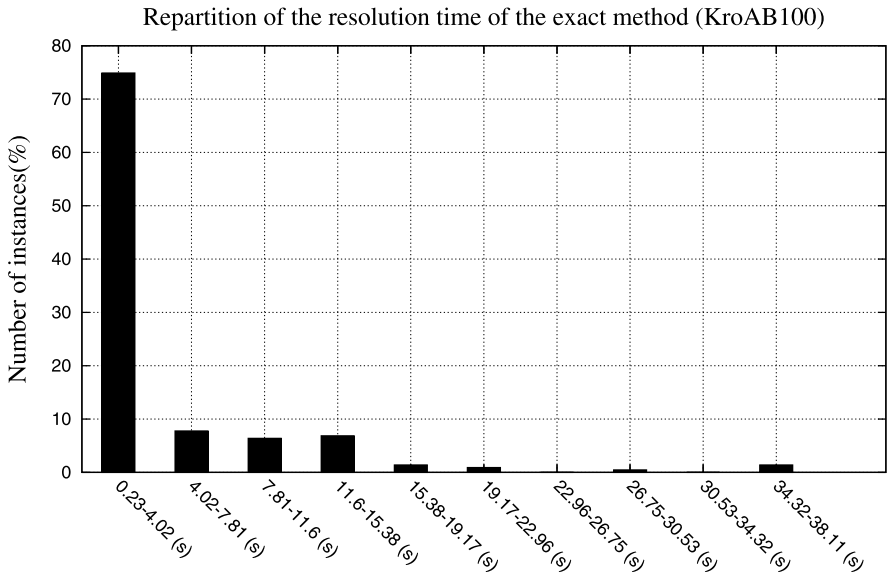

Fig. 6 Repartition of the running time of the exact method to solve single-objective TSP instances obtained with the dichotomic scheme

1998). The results of the version 2 of PLS, depend on the order according to which the solutions of the population P are examined. The rule is to take the first solution of the population, which corresponds to the solution of the population that minimizes the first objective given that the solutions of the population are continuously sorted out according to the first objective to accelerate the AddSolution procedure.

We can see the results at Table 3 for the KroAB100, MixedAB100, RdAB100 and ClusteredAB100 instances. The running time indicated in the table corresponds only to the running time of the second phase.

If we first compare the results of the two different versions of PLS, we can make the following conclusions:

- If PLS is launched from a random solution, for the four instances, PLS1 takes more time but is better in term of quality, which is logical given that in PLS1, dominated solutions are kept in the population.
- If PLS is launched from a good initial population (Di-LKC, Di-LKH or Di-EXA), the differences between the two versions of PLS in term of running time and quality are not so marked. Anyway, for the three instances, we find slightly better results with PLS1, in similar running times.
- The results when PLS is launched from a good initial population are much better in term of quality and running time than if PLS is launched from a random solution.

If we now compare the results given by the different populations as starting point of the PLS method, we can say that:

- The results given by the three different populations are very similar.

Table 2 Comparison of the LKC and LKH solvers on bigger instances for the first phase

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\lvert PE\rvert$	Time (s)
EuclAB300	Di-LKC	**2306.36**	**1.125952**	**0.941247**	**17.044**	**22.008**	**271.35**	185.29
	Di-LKH	2304.65	1.133814	0.941132	17.240	25.563	178.20	**97.26**
EuclAB500	Di-LKC	**7158.14**	**1.129153**	**0.948687**	**17.477**	**24.061**	301.00	385.99
	Di-LKH	7156.62	1.130715	0.948667	17.544	26.200	**269.00**	**367.58**
RdAB300	Di-LKC	4802.68	1.728705	0.966741	21.088	36.425	236.60	303.19
	Di-LKH	**4804.41**	**1.719164**	**0.966868**	**20.814**	**29.448**	**285.20**	**205.39**
RdAB500	Di-LKC	13983.98	1.790381	0.973544	19.766	**55.228**	260.25	**804.66**
	Di-LKH	**13990.42**	**1.771297**	**0.973761**	**19.205**	59.552	**472.15**	848.71
MixedAB300	Di-LKC	**3407.49**	1.778788	**0.956197**	**20.371**	**31.135**	240.65	225.96
	Di-LKH	3407.11	**1.767418**	0.956167	20.384	34.223	**228.15**	**149.14**
MixedAB500	Di-LKC	10433.74	1.710660	0.960407	23.088	**32.256**	292.00	700.75
	Di-LKH	**10435.53**	**1.691078**	**0.960507**	**22.949**	33.371	**426.15**	**669.68**
ClusteredAB300	Di-LKC	**2306.36**	**1.244220**	**0.941247**	18.607	**34.272**	**177.10**	215.85
	Di-LKH	2304.65	1.246607	0.941132	**18.514**	43.553	137.50	**72.98**
ClusteredAB500	Di-LKC	8502.10	1.204523	**0.962036**	18.280	80.133	**233.05**	384.08
	Di-LKH	**8508.50**	**1.182572**	0.962032	**17.470**	**40.188**	228.55	**294.83**

– To use the exact method—or using a better solver than the ones used in this work or with better parameters than the default parameters—will not have an important impact on the quality results of 2PPLS, for the 100 instances of this work.

For bigger instances, we do not show the results in this paper, but we found no significant differences in term of quality and running time between the two different populations experimented, that is the ones coming from Di-LKC and Di-LKH.

7.3 Comparison with state-of-the-art algorithms

To our knowledge, only two groups of authors have published approximations of the efficient set of the bTSP, and only for the "Kro" instances: Paquete et al. with the PLS (Paquete et al. 2004) and PD-TPLS (Paquete and Stützle 2003) methods and Jaszkiewicz with the MOGLS method (Jaszkiewicz 2002). But Paquete and Stützle have shown that the PD-TPLS method gives better results than the MOGLS method (Paquete and Stützle 2003), what makes the comparison with MOGLS not relevant.

Other authors have made experimentations on the "Kro" instances, without having published the results:

– Angel et al. (2004) have experimented a PLS method, version 1, by starting the method from a solution randomly generated and by using a reduced exploration of

Table 3 Comparison of the different alternatives for the second phase

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}(SE)$	R	D_1	D_2	$\|PE\|$	Time PLS (s)
KroAB100	RD+PLS1	224.84	1.045446	0.934207	1.461	30.154	2373.35	159.80
	RD+PLS2	224.39	1.078011	0.934066	1.676	45.438	2521.35	22.00
	Di-LKC+PLS1	**226.11**	**1.006617**	**0.935259**	**0.280**	2.892	2541.70	7.80
	Di-LKC+PLS2	226.10	1.006641	**0.935259**	0.281	2.892	2494.15	7.19
	Di-LKH+PLS1	226.10	**1.006617**	0.935255	0.283	**2.884**	**2559.30**	6.75
	Di-LKH+PLS2	226.10	**1.006617**	0.935252	0.289	**2.884**	2493.85	7.22
	Di-EXA+PLS1	226.10	**1.006617**	0.935255	0.282	2.896	2548.00	**6.72**
	Di-EXA+PLS2	226.10	**1.006617**	0.935252	0.288	2.896	2495.00	7.16
RdAB100	RD+PLS1	417.38	1.464269	0.943355	7.220	111.929	**704.90**	76.89
	RD+PLS2	416.29	1.669674	0.943110	8.302	143.237	626.15	7.40
	Di-LKC+PLS1	**429.14**	**1.021462**	0.948489	**0.810**	**14.170**	515.75	1.06
	Di-LKC+PLS2	**429.14**	1.021510	0.948491	**0.810**	**14.170**	515.15	1.20
	Di-LKH+PLS1	428.30	1.022070	**0.948499**	0.824	14.654	499.45	1.04
	Di-LKH+PLS2	428.29	1.022070	0.948496	0.827	14.654	497.65	1.13
	Di-EXA+PLS1	428.30	1.022070	0.948497	0.829	14.297	491.00	**1.00**
	Di-EXA+PLS2	428.29	1.022070	0.948494	0.832	14.297	491.00	1.12
MixedAB100	RD+PLS1	327.28	1.383581	0.934599	3.617	60.439	**1080.20**	112.81
	RD+PLS2	325.85	1.529007	0.933849	4.747	84.643	1057.05	9.78
	Di-LKC+PLS1	**331.99**	1.015601	0.936814	0.529	5.768	882.55	2.31
	Di-LKC+PLS2	**331.99**	1.015432	**0.936818**	**0.527**	5.768	870.15	2.32
	Di-LKH+PLS1	331.98	**1.013461**	0.936801	0.547	**4.486**	873.75	2.33
	Di-LKH+PLS2	**331.99**	**1.013461**	0.936806	0.546	**4.486**	861.70	2.33
	Di-EXA+PLS1	**331.99**	**1.013461**	0.936807	0.539	4.522	876.00	**2.29**
	Di-EXA+PLS2	**331.99**	**1.013461**	0.936812	0.540	4.522	862.00	2.31
ClusteredAB100	RD+PLS1	266.35	1.027901	0.949188	1.129	27.539	2464.05	183.59
	RD+PLS2	265.70	1.055536	0.948876	1.502	33.459	2333.25	19.14
	Di-LKC+PLS1	267.30	1.007167	0.950017	0.229	4.937	**2490.30**	7.19
	Di-LKC+PLS2	267.30	1.007390	0.950017	0.233	4.937	2435.90	7.70
	Di-LKH+PLS1	**267.31**	1.006992	**0.950027**	**0.218**	**2.773**	2489.80	7.06
	Di-LKH+PLS2	**267.31**	**1.006983**	0.950025	0.223	**2.773**	2436.75	7.47
	Di-EXA+PLS1	**267.31**	1.007068	**0.950027**	**0.218**	**2.773**	2473.00	**6.88**
	Di-EXA+PLS2	**267.31**	1.007059	0.950026	0.222	**2.773**	2433.00	7.49

the neighborhood, which can only give results in an order of magnitude equal to the results given by the RD+PLS1 alternative, whose results are given at Table 2. These results are strongly worse in term of quality and running time than the results obtained with the 2PPLS method.

– Recently, Kumar and Singh (2007) have experimented a memetic algorithm and they have compared their results with the results given by MOGLS and PD-TPLS.

They find comparable results, it is thus not worth to use the approximations given by this method.
– Jaszkiewicz and Zielniewicz have improved the results of the MOGLS algorithm by using a Pareto Memetic Algorithm (PMA) (Jaszkiewicz and Zielniewicz 2006). But they have only published for these results, the running time and the R indicator.

We thus first compare the results of the 2PPLS method, for the "Kro" instances, with PD-TPLS and with the PLS method of Paquete et al. (2004), which is a Pareto Local Search method version 2, starting from a randomly generated solution and using the 3-exchange neighborhood.

For 2PPLS, we choose to use the version 1 of PLS which has various advantages: not being dependent on the order in which the solutions of the population are examined (contrarily to both other versions) and to give better results than the version 2. For the single-objective solver, we choose to use the LKC solver.

We can see the results at Table 4.

We remark that on five indicators, $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2, 2PPLS finds better results than PD-TPLS and PLS-3opt. The PLS-3opt method finds more potentially efficient solutions, but of worse quality than the potentially efficient solutions generated by 2PPLS. Concerning the running time, the comparison is biased because the PD-TPLS and the PLS-3opt methods have been run on a Dual Athlon with 1.2 GHz. But we can see that the running time of PLS-3opt is much higher than the running time of 2PPLS. On the other hand, the running time of PD-TPLS is lower than the running time of 2PPLS. However, Paquete and Stützle have tried to improve the results of PD-TPLS by increasing the number of aggregations used in their method, and consequently the running time. They only do that for the KroAB100 instance, and we can see the results under the name PD-TPLS Best. We notice that although the results of PD-TPLS are better, they remain lower than 2PPLS, and with a relative higher running time.

We now compare the results of 2PPLS to PMA, only for the R indicator. We can see the results at Table 5. We remark that the R indicator of PMA is better on five of the six instances of 100 cities, but worse on the 150 and 200 cities instances. On the other hand, the running time of PMA is higher (on a Pentium with 2.8 GHz). Therefore, a future objective is to improve the results of 2PPLS. We will see at Sect. 8 a simple way to do that, even if at the end of 2PPLS we are blocked in a minimal critical nondominated Pareto local optimum set with respect to the 2-exchange neighborhood.

7.4 Comparison with PD-TPLS on the other instances

We now compare 2PPLS with PD-TPLS on the other instances. For these instances, no state-of-the-art results have been published, we have thus implemented ourselves the PD-TPLS method of Paquete and Stützle, which is a method presenting few parameters contrarily to the PMA method. For this implementation, we have fixed the parameters of this method as follows:

– The number of iterations for the number of perturbation steps is equal to the number of cities N minus 50, as done by Paquete et al. for the "Kro" instances.
– The number of aggregations is equal to N.

Table 4 Comparison with state-of-the-art algorithms on "Kro" instances

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time (s)
KroAB100	2PPLS	**226.11**	**1.006617**	**0.935259**	**0.280**	**2.892**	2541.70	35.57
	PLS-3opt	225.66	1.028526	0.934996	0.549	22.096	**2735.70**	3000.00
	PD-TPLS	225.96	1.017594	0.935103	0.524	9.332	896.54	**14.14**
	PD-TPLS Best	226.05	1.010028	0.935209	0.377	3.308	1015.12	279.88
KroAC100	2PPLS	**226.32**	**1.004604**	**0.932513**	**0.270**	**2.625**	1960.70	30.3
	PLS-3opt	225.83	1.026469	0.932251	0.555	25.579	**2133.64**	3000.00
	PD-TPLS	226.16	1.017585	0.932329	0.518	6.706	850.42	**13.72**
KroAD100	2PPLS	**227.41**	**1.006104**	**0.934623**	**0.283**	**5.251**	1788.10	26.44
	PLS-3opt	227.11	1.014947	0.934354	0.565	15.977	**2096.06**	3000.00
	PD-TPLS	227.26	1.019599	0.934436	0.543	9.442	772.48	**13.69**
KroBC100	2PPLS	**227.38**	**1.003957**	**0.936215**	**0.203**	**5.280**	2103.20	35.82
	PLS-3opt	227.00	1.017446	0.935975	0.447	10.134	**2403.08**	3000.00
	PD-TPLS	227.27	1.013191	0.936085	0.413	6.016	911.64	**14.65**
KroBD100	2PPLS	**226.12**	**1.004826**	**0.934800**	**0.231**	**5.588**	1956.20	37.18
	PLS-3opt	225.53	1.027412	0.934419	0.647	19.244	**2372.96**	3000.00
	PD-TPLS	226.01	1.013345	0.934663	0.429	6.763	879.20	**14.86**
KroCD100	2PPLS	**230.89**	**1.006435**	**0.939158**	**0.275**	**3.472**	1633.25	27.56
	PLS-3opt	230.31	1.025861	0.938809	0.691	39.410	**1846.08**	3000.00
	PD-TPLS	230.76	1.018205	0.939006	0.494	9.078	809.06	**14.02**
KroAB150	2PPLS	**592.51**	**1.003442**	**0.942127**	**0.156**	**2.441**	**3884.45**	**91.21**
	PD-TPLS	592.27	1.017489	0.942013	0.387	10.095	1558.62	116.87
KroAB200	2PPLS	**1076.08**	**1.003257**	**0.945067**	**0.115**	**4.410**	**6736.50**	**211.76**
	PD-TPLS	1075.78	1.012449	0.944988	0.293	7.229	2345.22	477.89

We can see the results at Table 6. We remark that 2PPLS finds better results than PD-TPLS on all the indicators, for all the instances, except for the ClusteredAB300 and ClusteredAB500 instances. But the running time of 2PPLS is higher, except for the RdAB500 instance. For instances of 500 cities, the running time of 2PPLS can be very high on the Euclidean (6309 s) and clustered instances (13495 s). On the other hand, the running time of PD-TPLS remains reasonable, and seems constant according to the instance type.

7.5 Mann-Whitney test

To take into account the variations in the results of the algorithms, as we do not know the distributions of the indicators, we carried out the non-parametric statistical test of Mann-Whitney (Ferguson 1967). This test allows to compare the distributions of the indicators of 2PPLS with these of PD-TPLS.

Table 5 Comparison of 2PPLS with PMA on "Kro" instances

Instance	Algorithm	R	Time (s)
KroAB100	2PPLS	0.935259	**36**
	PMA	**0.935280**	67
KroAC100	2PPLS	0.932513	**30**
	PMA	**0.932521**	67
KroAD100	2PPLS	**0.934623**	**26**
	PMA	0.934617	69
KroBC100	2PPLS	0.936215	**36**
	PMA	**0.936221**	73
KroBD100	2PPLS	0.934800	**37**
	PMA	**0.934809**	74
KroCD100	2PPLS	0.939158	**28**
	PMA	**0.939166**	73
KroAB150	2PPLS	**0.942127**	**91**
	PMA	0.942081	223
KroAB200	2PPLS	**0.945067**	**212**
	PMA	0.943312	567

We test the following hypothesis: "the two samples come from identical populations" for the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 or D_2 indicators 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, there are differences between the indicators of the algorithms: the sign ">" is indicated if the mean value obtained with 2PPLS is better than the mean value obtained with PD-TPLS and the sign "<" is indicated if the mean value obtained with 2PPLS is worse than the mean value obtained with PD-TPLS.

As five hypothesis are tested simultaneously, the levels of risk α of the tests have been adjusted with the Holm sequential rejective method (Holm 1979). This method works as follows: the p-values obtained with the Mann-Whitney test for each indicator are sorted out by increasing value and the smallest p-value is compared to α/k (where k is the number of indicators, equal to five in our case). If that p-value is less than α/k, the hypothesis related to this p-value is rejected. The second smallest p-value is then compared to $\alpha/(k-1)$, the third to $\alpha/(k-2)$, etc. The procedure proceeds until each hypothesis has been rejected or when a hypothesis cannot be rejected; at this point all hypothesis that have not yet been rejected are accepted.

The starting level of risk of the test has been fixed to 1%.

The results of the comparison of 2PPLS with PD-TPLS are given at Table 7.

The results show that we can affirm with a very low risk that, for the indicators considered in this work, 2PPLS is better or equal than PD-TPLS on all the instances

Table 6 Comparison of 2PPLS with PD-TPLS on the other instances

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time (s)
EuclAB100	2PPLS	**168.88**	**1.006386**	**0.921875**	**0.325**	**4.938**	**1303.70**	28.60
	PD-TPLS	168.76	1.013736	0.921737	0.521	8.063	744.70	**11.57**
EuclAB300	2PPLS	**2309.69**	**1.124524**	**0.941704**	**16.865**	**21.612**	**14749.60**	783.14
	PD-TPLS	2308.97	1.124950	0.941598	17.020	21.921	4415.40	**255.13**
EuclAB500	2PPLS	**7165.39**	1.128923	**0.948997**	**17.244**	**21.616**	**34861.95**	6309.10
	PD-TPLS	7163.87	**1.128899**	0.948927	17.395	21.829	9306.75	**1056.43**
RdAB100	2PPLS	**429.14**	**1.021462**	**0.948489**	**0.810**	**14.170**	**515.75**	26.34
	PD-TPLS	428.78	1.057139	0.947983	1.458	27.297	372.00	**12.34**
RdAB300	2PPLS	**4804.58**	**1.728526**	**0.966936**	**20.902**	**36.295**	**2006.55**	351.27
	PD-TPLS	4790.35	1.893817	0.966152	22.848	60.815	1238.70	**305.41**
RdAB500	2PPLS	**13988.05**	**1.790178**	**0.973703**	**19.494**	**54.692**	**3492.85**	**1095.22**
	PD-TPLS	13951.01	2.057356	0.972940	21.954	95.618	2081.55	1324.55
MixedAB100	2PPLS	**331.99**	**1.015601**	**0.936814**	**0.529**	**5.768**	**882.55**	28.28
	PD-TPLS	331.63	1.034143	0.936690	0.753	12.094	536.75	**11.89**
MixedAB300	2PPLS	**3410.44**	**1.778754**	**0.956496**	**20.132**	**28.411**	**5633.70**	447.06
	PD-TPLS	3406.12	1.943372	0.956254	20.722	37.980	2299.80	**288.04**
MixedAB500	2PPLS	**10440.41**	**1.710616**	**0.960654**	**22.856**	**31.708**	**13636.70**	2682.45
	PD-TPLS	10429.11	1.907713	0.960452	23.517	35.546	4816.00	**1303.07**
ClusteredAB100	2PPLS	**267.30**	**1.007167**	**0.950017**	**0.229**	**4.937**	**2490.30**	55.68
	PD-TPLS	267.15	1.019662	0.949873	0.534	8.792	848.10	**13.17**
ClusteredAB300	2PPLS	2565.46	1.242723	**0.956624**	17.751	22.229	**16885.00**	1554.69
	PD-TPLS	**2566.38**	**1.232286**	0.956616	**17.702**	**21.917**	4540.15	**293.72**
ClusteredAB500	2PPLS	8516.69	1.183202	**0.962417**	**16.959**	24.443	**42743.25**	13495.33
	PD-TPLS	**8516.71**	**1.181800**	0.962394	16.974	**23.230**	9678.15	**1239.56**

experimented in this work, except for the ClusteredAB300 instance (on $\mathcal{H}$, $I_{\epsilon 1}$ and D_1) and for the ClusteredAB500 instance (on $I_{\epsilon 1}$).

7.6 Outperformance relations

We now compare in term of outperformance relations (Hansen and Jaszkiewicz 1998) the solutions obtained with 2PPLS and PD-TPLS.

We show at Fig. 7 the results of the comparison between the potentially non-dominated points obtained with 2PPLS and with PD-TPLS, for the 300 cities instances.

Table 7 Results of the Mann-Whitney test for the D_1 and $I_{\epsilon 1}$ indicators

Instance	$\mathcal{H}$	$I_{\epsilon 1}$	R	D_1	D_2
KroAB100	>	>	>	>	>
KroAB150	>	>	>	>	>
KroAB200	>	>	>	>	>
KroAC100	>	>	>	>	>
KroAD100	>	>	>	>	>
KroBC100	>	>	>	>	>
KroBD100	>	>	>	>	>
KroCD100	>	>	>	>	>
EuclAB100	>	>	>	>	>
EuclAB300	>	>	>	>	>
EuclAB500	>	=	>	>	>
RdAB100	>	>	>	>	>
RdAB300	>	>	>	>	>
RdAB500	>	>	>	>	>
MixedAB100	>	>	>	>	>
MixedAB300	>	>	>	>	>
MixedAB500	>	>	>	>	>
ClusteredAB100	>	>	>	>	>
ClusteredAB300	<	<	=	<	=
ClusteredAB500	=	<	>	>	=

To create these box-plot graphs, we compare the points obtained with the `2PPLS` method with the points obtained with the `PD-TPLS` method. Four cases can occur: a point of `2PPLS` is dominated by at least one point of `PD-TPLS` (`Dominated`), a point of `2PPLS` dominates at least one point of `PD-TPLS` (`Dominate`), a point of `2PPLS` is equal to an another point of `PD-TPLS` (`Commons`), or the result of the comparison belongs to none of these three possibilities (`Others`).

We can see thanks to these box-plot graphs, that many points obtained with the `2PPLS` method dominate points of `PD-TPLS`, for the EuclAB300, RdAB300 and MixedAB300 instances. However, on the ClusteredAB300 instance, there are more points of `2PPLS` that are dominated than points that dominate points of `PD-TPLS`, but the difference remains low.

7.7 Influence of the size of the initial population

We study here the influence of the size of the initial population on the results of `2PPLS` for the Euclidean instances. To do that, we use the `Di-LKH` alternative for the first phase, and we limit the number of potentially efficient solutions generated by `Di-LKH` to m. We vary the value of the parameter m from 1 to the maximum number of potentially efficient solutions obtained with the `Di-LKH` alternative.

We can see the results at Fig. 8 for the kroAB100 instance. We have represented the influence of m on the D_1 and $I_{\epsilon 1}$ indicators, and on the number $|PE|$ of poten-

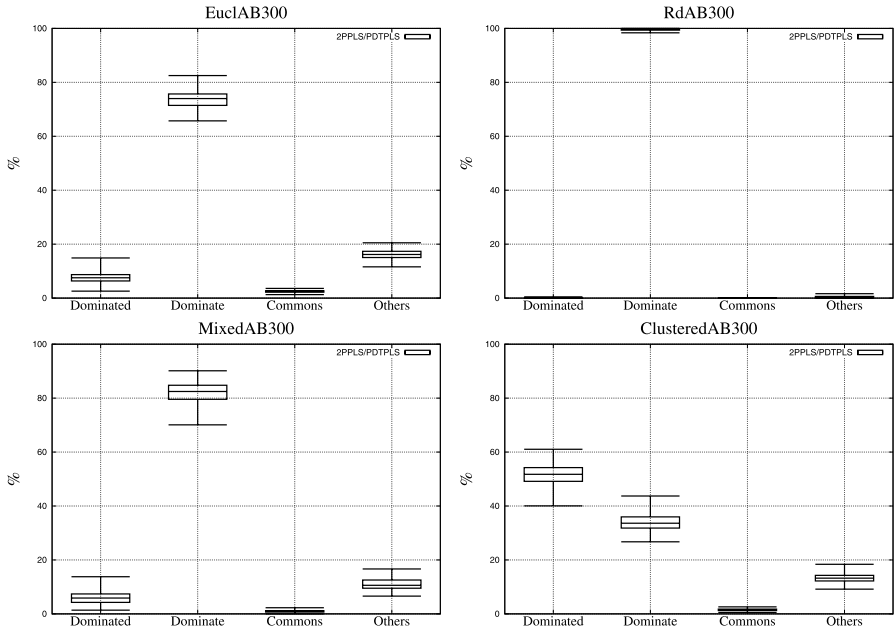

Fig. 7 Comparison between the potentially non-dominated points obtained with `2PPLS` and with `PD-TPLS`

tially efficient solutions generated for only one execution of `2PPLS`. We have also represented the influence of m on the running time of the phase 1 (`P1`), on the running time of `PLS` and on the running time of `2PPLS` (equal to the sum of the running times of `P1` and `PLS`).

We notice that by increasing m, the D_1 and $I_{\epsilon 1}$ indicators are generally improving. But there are exceptions: the second solution added increases the D_1 indicator (but reduces the $I_{\epsilon 1}$ indicator) and the 61th solution increases the $I_{\epsilon 1}$ indicator (but reduces the D_1 indicator).

Another interesting observation is that the improvements are done by step, of more or less important values. For instance, the 61th solution makes it possible to strongly improve the D_1 indicator of `2PPLS` and to increase the number of potentially efficient solutions generated. On the other hand, there are many solutions generated during the first phase that do not allow to make improvements.

Let us define a cluster as a set of solutions that are reachable from each other by applying k elementary moves with respect to a given neighborhood (Paquete and Stützle 2006). We can thus say that some solutions generated during the first phase allow to attain a new cluster of potentially efficient solutions by applying elementary moves, and other new solutions of the first phase are not worthwhile because they are already connected (that is in the same cluster) to previously generated solutions.

If we look at the $|PE|$ information, we can see that, for instance, the 61th solution allows to generate a new cluster of potentially efficient solutions, since the number of potentially efficient solutions is increasing following the addition of this solution.

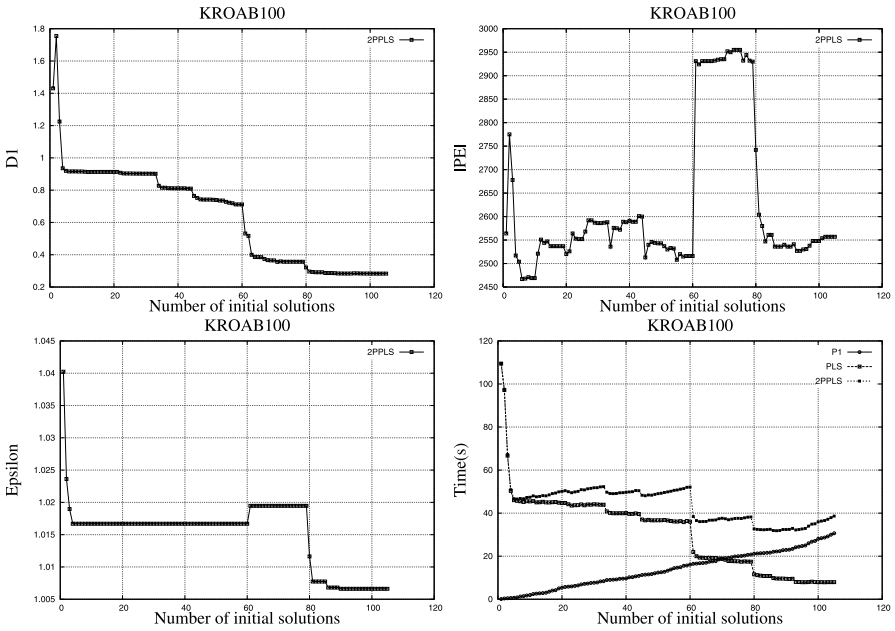

Fig. 8 Influence of the number of initial solutions for the KroAB100 instance

But most of the solutions of this cluster are then eliminated following the addition of a new solution generated during the first phase.

It seems thus that it is possible to improve the initial population generated during the first phase by using a method that allows to generate non-connected solutions with respect to the neighborhood used.

For the running times, we remark that the running time of the first phase is linearly increasing, but that the running time of PLS is decreasing, and always by step. The total running time of 2PPLS presents thus many local optima, and the global optimum is attained when about 85 solutions are generated. It is thus interesting, especially if the running time of 2PPLS is critical. Therefore, for instances of more than 100 cities, where the running time of 2PPLS becomes important, it seems that is possible to reduce it by generating the best initial population according to the running time.

8 Improvement of the initial population

As shown in Sect. 7.7, a better quality initial population gives better results for 2PPLS and reduces the running time of the second phase.

Therefore, we present here a simple technique to improve the results of 2PPLS, that should be applied after the dichotomic scheme, and before the application of the PLS method.

Algorithm 7 `Perturbation`

Parameters ↓: Number of iterations K, the perturbation parameters LP, SF and SV, the cost matrices C_1 and C_2 of the bTSP
Parameters ↕: An approximation $\widehat{X_E}$ of the efficient set

for k=1 to K **do**
  $\lambda_1 = 1 - \frac{(k-1)}{(K-1)}$
  $\lambda_2 = 1 - \lambda_1$
  --| Create a new cost matrix C^λ
  $C^\lambda = [\lambda_1 c^1_{ij} + \lambda_2 c^2_{ij}]$
  --| Perturb the cost matrix C^λ
  $SC \leftarrow SF + \text{random}(SV)$
  for $i = 1$ to $n - 1$ **do**
    for $j = i + 1$ to n **do**
      $\epsilon \leftarrow \text{random}(LP)$
      $C^{\lambda P}_{i,j} = \text{round}(C^\lambda_{i,j} \pm C^\lambda_{i,j} \cdot \frac{\epsilon}{SC})$
  --| Solve the single-objective TSP by considering the cost matrix $C^{\lambda P}$
  `SolveTSP`($C^{\lambda P}$ ↓, x_t ↑)
  `AddSolution`($\widehat{X_E}$ ↕, x_t ↓, $z(x_t)$ ↓)

As the initial population already contains many supported efficient solutions, the improvement of this population can only be done by generating potentially non-supported efficient solutions.

To do that, we use the "Data Perturbation" (DP) technique, originally proposed by Codenotti et al. for the single-objective TSP (Codenotti et al. 1996). Instead of modifying the starting solution (as carried out, for instance, in the Iterated Local Search method (Lourenço et al. 2002)), DP suggests to modify input data.

The functioning is as follows: we start the Perturbation algorithm (see Algorithm 7) from an approximation $\widehat{X_E}$ of the efficient set. Here the initial set is given by the procedure `BTSP_Phase1Heuristic` algorithm and corresponds thus to $\widehat{X_{SE1_m}}$. The input parameters of the Algorithm 7 are the number K of iterations, three parameters that determine the perturbation scheme (SF, SV and LP) and the cost matrices C_1 and C_2 of the bTSP. The set $\widehat{X_E}$ is at the same time an input and an output parameter.

During an iteration k, we first compute a weight set λ by following a linear scheme; K uniformly distributed weight sets are thus generated. We then create a new cost matrix C^λ. If we solve this single-objective problem, the chances that the optimal solution of this problem is a solution already generated during the phase 1 are large. Therefore, we slightly perturb each cost $C^\lambda_{i,j}$ of the matrix C to find new potentially efficient solutions. To generate a perturbed instance of the bTSP, we proceed in the following way. We first fix a scaling variable, called SC. This variable makes it possible to modulate the perturbations. A large value for SC gives a slightly perturbed instance while a low value gives greater perturbations. The value of SC is equal to a fixed scaling parameter SF plus a value randomly generated between 0 and SV, where SV is the variable scaling parameter. Then, we add or withdraw from

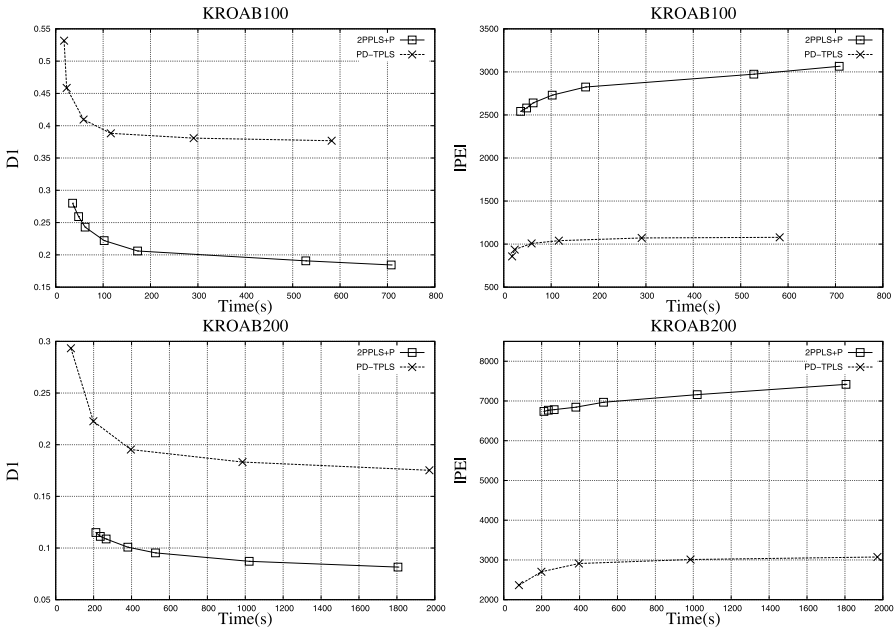

Fig. 9 Influence of the number iterations K of the perturbation algorithm

$C^{\lambda}_{i,j}$ a small fraction $\frac{\epsilon}{SC}$ of each cost $C^{\lambda}_{i,j}$, for thus obtaining the perturbed costs $C^{\lambda P}_{i,j}$. The value of ϵ is randomly generated between 0 and the local perturbation parameter LP. A single-objective solver is then applied by considering the perturbed cost matrix $C^{\lambda P}$. We finally update the set $\widehat{X_E}$ with the Procedure 1, by considering the original cost matrices C.

In this way, by application of a single-objective solver with perturbed data, new solutions, essentially potentially non-supported efficient, could be found since the data given by the linear aggregation are perturbed.

We note by `2PPLS+P` the `2PPLS` method plus the perturbation algorithm. The perturbation algorithm is applied after the phase 1 and before the `PLS` method.

We have represented at Fig. 9 the evolution of the D_1 and $|PE|$ indicators of the `2PPLS+P` method according to the running time for the KroAB100 and KroAB200 instances (the running time is controlled by the number of iterations K varying from 0 to 5000). We use as solver for the phase 1 and for the perturbations, the `LKC` solver. We choose the version 1 of the `PLS` method. The parameters of the perturbation algorithm have experimentally been fixed to 1000 for SF, 2000 for SV and 100 for LP.

We remark that the increase of the number K of iterations gives an important improvement of the indicators and allows to reach excellent results, since the number of potentially efficient solutions $|PE|$ is increasing while the distance D_1 is decreasing. We can also see on this figure, the effect of the increase of the running time of `PD-TPLS` by increasing the number of aggregation function (from N to 5000). It

Table 8 Comparison of 2PPLS+P with PMA

Instance	Algorithm	R	Time (s)
KroAB100	2PPLS+P	**0.935294**	**62**
	PMA	0.935280	67
KroAC100	2PPLS+P	**0.932532**	**55**
	PMA	0.932521	67
KroBC100	2PPLS+P	**0.936225**	**64**
	PMA	0.936221	73
KroBD100	2PPLS+P	**0.934818**	**68**
	PMA	0.934809	74
KroCD100	2PPLS+P	**0.939182**	**52**
	PMA	0.939166	73

is noticed, that in spite of this increase, the performances of PD-TPLS stagnate and never reach the performances of 2PPLS+P.

The interest of using the perturbation algorithm after the phase 1 of 2PPLS seems thus important, since thanks to these perturbations, new solutions are constantly found and the quality of the approximations is increasing.

We finally compare the results obtained by the 2PPLS+P method with $K = 200$ to the results obtained by PMA for the "Kro" 100 cities instances for which PMA obtained a better R indicator than 2PPLS. We have represented the results at Table 8, and we remark that now, the 2PPLS+P method manages to obtain a better value for the R indicator, in similar running times.

9 Conclusion

We showed in this paper that by using two different techniques—a very good initial population composed of an approximation of the extreme supported efficient solutions and the Pareto Local Search method—we can obtain very good results for the biobjective TSP.

- Advantages of 2PPLS:
  - Simple method with better results than complex methods including many parameters not easy to fix. No numerical parameters is an interesting advantage comparing to multiobjective genetic algorithms which often ask a lot.
  - Improving the results is quite easy by using the perturbation algorithm.
- Disadvantages of 2PPLS:
  - Dependent on the quality of the neighborhood used in second phase.
  - The method stops when no more improvements with the considered neighborhood is possible.
  - High running time for bTSP instances of more than 200 cities.
- Future works:

- Better analysis of the behavior of the 2PPLS method: what solutions are important in the initial population and how to produce initial solutions allowing good improvements with the Pareto Local Search.
- In the biobjective TSP, it is clear that using the 3-exchange neighborhood would give better results in term of quality, but the running time would be higher. Therefore, it would be interesting to use more efficient speed-up techniques to produce a very efficient neighborhood for the biobjective TSP, and being able to resolve efficiently larger size instances of the bTSP.

An another interesting and obvious future work would be to test this method on more than two objective instances. A reason why this work has not yet been realized is that the first phase of the 2PPLS with three or more objective is more complicated, because the correspondance between the geometrical and the algebraic interpretation of the weight sets is not more valid (Przybylski et al. 2007). Experiments should be done to see if the use of a sophisticated method as first phase will be yet worthwhile, or if a simple method based on randomly or uniformly generated weight sets would be sufficient. The more thorough study about the influence of the initial population will also help us to answer this question.

References

Aneja, Y.P., Nair, K.P.K.: Bicriteria transportation problem. Manag. Sci. **25**, 73–78 (1979)

Angel, E., Bampis, E., Gourvès, L.: A dynasearch neighborhood for the bicriteria traveling salesman problem. In: Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (eds.) Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 153–176. Springer, Berlin (2004)

Applegate, D.: Chained Lin-Kernighan for large traveling salesman problems. INFORMS J. Comput. **15**, 82–92 (2003)

Basseur, M.: Design of cooperative algorithms for multi-objective optimization: application to the flow-shop scheduling problem. 4OR **4**(3), 255–258 (2006)

Borges, P.C., Hansen, M.P.: A basis for future success in multiobjective combinatorial optimization problems. Technical report, Technical University of Denmark, Lingby, Denmark (1998)

Codenotti, B., Manzini, G., Margara, L., Resta, G.: Perturbation: An efficient technique for the solution of very large instance of the euclidean tsp. INFORMS J. Comput. **8**, 125–133 (1996)

Czyzak, P., Jaszkiewicz, A.: Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. J. Multi-Criteria Decis. Anal. **7**, 34–47 (1998)

Ehrgott, M.: Multicriteria Optimization, 2nd edn. Springer, Berlin (2005)

Ehrgott, M., Gandibleux, X.: Bound sets for biobjective combinatorial optimization problems. Comput. Oper. Res. **34**, 2674–2694 (2007)

Ehrgott, M., Klamroth, K.: Connectedness of efficient solutions in multiple criteria combinatorial optimization. Eur. J. Oper. Res. **97**, 159–166 (1997)

Ferguson, T.S.: Mathematical Statistics, a Decision Theoretic Approach. Academic, New York (1967)

Hamacher, H.W., Ruhe, G.: On spanning tree problems with multiple objectives. Ann. Oper. Res. **52**, 209–230 (1994)

Hansen, M.P., Jaszkiewicz, A.: Evaluating the quality of approximations of the nondominated set. Technical report, Technical University of Denmark, Lingby, Denmark (1998)

Hansen, P.: Bicriterion path problems. Lect. Notes Econ. Math. Syst. **177**, 109–127 (1979)

Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**, 106–130 (2000)

Holm, S.: A simple sequentially rejective multiple test procedure. Scand. J. Stat. **6**, 65–70 (1979)

Jaszkiewicz, A.: On the performance of multiple-objective genetic local search on the 0/1 knapsack problem—a comparative experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznan University of Technology, Poznań, Poland (July 2000)

Jaszkiewicz, A.: Genetic local search for multiple objective combinatorial optimization. Eur. J. Oper. Res. **137**(1), 50–71 (2002)

Jaszkiewicz, A., Zielniewicz, P.: Efficient adaptation of the Pareto memetic algorithm to the multiple objective travelling salesperson problem. In: Proceedings of the 7th International Conference devoted to Multi-Objective Programming and Goal Programming, Tours (June 2006)

Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (February 2006)

Kumar, R., Singh, P.K.: Pareto evolutionary algorithm hybridized with local search for biobjective tsp. In: Grosan, C., Abraham, A., Ishibuchi, H. (eds.) Hybrid Evolutionary Algorithms. Springer, New York (2007). Chap. 14

Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**, 498–516 (1973)

Lourenço, H.R., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics. International Series in Operations Research and Management Science, vol. 57, pp. 321–353. Kluwer Academic, Norwell (2002)

Paquete, L.: Stochastic local search algorithms for multiobjective combinatorial optimization: methods and analysis. PhD thesis, FB Informatik, TU Darmstadt (2005)

Paquete, L., Stützle, T.: A two-phase local search for the biobjective traveling salesman problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Faro, Portugal, April 2003. Lecture Notes in Computer Science, vol. 2632, pp. 479–493. Springer, Berlin (2003)

Paquete, L., Stützle, T.: Clusters of non-dominated solutions in multiobjective combinatorial optimization. In: Proceedings of the 7th International Conference devoted to Multi-Objective Programming and Goal Programming, Tours (June 2006)

Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: an experimental study. In: Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (eds.) Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 177–199. Springer, Berlin (2004)

Paquete, L., Schiavinotto, T., Stützle, T.: On local optima in multiobjective combinatorial optimization problems. Ann. Oper. Res. **156**(1), 83–97 (2007)

Przybylski, A., Gandibleux, X., Ehrgott, M.: A recursive algorithm for finding all extremal supported nondominated points in the outcome set of a multi-objective integer linear problem. Research report LINA (2007)

Przybylski, A., Gandibleux, X., Ehrgott, M.: Two-phase algorithms for the biobjective assignement problem. Eur. J. Oper. Res. **185**(2), 509–533 (2008)

Reinelt, G.: Tsplib—a traveling salesman problem library. ORSA J. Comput. **3**(4), 376–384 (1991)

Ulungu, E.L., Teghem, J.: The two phases method: an efficient procedure to solve biobjective combinatorial optimization problems. Found. Comput. Decis. Sci. **20**, 149–156 (1995)

Ulungu, E.L., Teghem, J., Fortemps, Ph., Tuyttens, D.: MOSA method: a tool for solving multiobjective combinatorial optimization problems. J. Multi-Criteria Decis. Anal. **8**(4), 221–236 (1999)

Zitzler, E.: Evolutionary algorithms for multiobjective optimization: methods and applications. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (November 1999)

Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C.M., Grunert da Fonseca, V.: Why Quality Assessment of Multiobjective Optimizers Is Difficult. In: Langdon, W.B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M.A., Schultz, A.C., Miller, J.F., Burke, E., Jonoska, N. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002), pp. 666–673, San Francisco, California, July 2002. Morgan Kaufmann, San Mateo (2002)

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)