Computers & Operations Research 37 (2010) 521 – 533

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

Speed-up techniques for solving large-scale biobjective TSP

T. Lust[a,*], A. Jaszkiewicz[b]

[a]Faculté Polytechnique de Mons, Laboratory of Mathematics and Operational Research, 9, rue de Houdain, 7000 Mons, Belgium
[b]Poznań University of Technology, Institute of Computing Science, Ul. Piotrowo 3A, 60-965 Poznań, Poland

ARTICLE INFO

Available online 23 January 2009

Keywords:
Multiobjective combinatorial optimization
Hybrid metaheuristics
TSP
Local search
Speed-up techniques

ABSTRACT

In this paper, we present the Two-Phase Pareto Local Search (2PPLS) method with speed-up techniques for the heuristic resolution of the biobjective traveling salesman problem. The 2PPLS method is a state-of-the-art method for this problem. However, because of its running time that strongly grows with the instances size, the method can be hardly applied to instances with more than 200 cities. We thus adapt some speed-up techniques used in single-objective optimization to the biobjective case. The proposed method is able to solve instances with up to 1000 cities in a reasonable time with no, or very small, reduction of the quality of the generated approximations.

1. Introduction

Considering more than one objective in combinatorial optimization considerably increases the complexity of resolution, even if the multiobjective problem is derived from a single-objective problem solvable in polynomial time. Thus, during the last two decades, many papers have been published on the adaptation of metaheuristics to multiobjective problems [7,9].

The multiobjective metaheuristics are usually based on relatively simple versions of single-objective methods, while the state-of-the-art results in single-objective optimization are often achieved by methods using a number of advanced components, e.g. speed-up techniques that can substantially reduce the running time of local-search-based methods. In the opinion of the authors, multiobjective metaheuristics should fully utilize the most powerful techniques proposed for the single-objective case. Of course, these techniques have to be properly adapted to the multiobjective case.

In this paper, we adapt two techniques used in single-objective local search to the biobjective traveling salesman problem (bTSP). The techniques are "candidate list" and "don't look bits". To our knowledge those techniques have not yet been applied in multiobjective metaheuristics.

We apply the speed-up techniques within the Two-Phase Pareto Local Search (2PPLS) method proposed by Lust and Teghem [17,18]. Initially, this method benefits to the maximum from very efficient heuristics developed for the resolution of the corresponding single-objective problems. Then, it uses the adaptation of one of the most simple metaheuristics: the hill-climbing method. This approach implies that no new numerical parameters are introduced. Lust and Teghem applied 2PPLS to the bTSP and obtained better results on several indicators than previous state-of-the-art algorithms. However, a weak point of the method is the running time that becomes high when larger instances are tried to be solved. We show that our proposed method is able to solve instances with up to 1000 cities in a reasonable time with no, or very small, reduction of the quality of the generated approximations. Note that before only instances with up to 200 cities were solved.

The paper is organized as follows: in the next section, we present general definitions relating to multiobjective combinatorial optimization. In the third section, we define the bTSP and the instances used in this work. The different quality indicators necessary to evaluate the quality of the results found by the different methods are presented at Section 4. Section 5 is dedicated to a brief presentation of the two phases of the 2PPLS method. After that we present the main contribution of this paper, speed-up techniques applied to the 2PPLS method for solving large-scale bTSP. Finally, the results obtained with the new method are discussed at Section 7.

2. Multiobjective combinatorial optimization

A multiobjective combinatorial optimization problems is defined as follows:

$$\begin{array}{ll} \text{"}\min_{x}\text{"} & z(x) = Cx \\ \text{subject to} & x : Ax = b \\ & x \in \{0,1\}^n \end{array}$$

* Corresponding author.
E-mail address: thibaut.lust@fpms.ac.be (T. Lust)
URL: http://www.ig.fpms.ac.be/~lustt (T. Lust).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.01.005

$x \in \{0,1\}^n \longrightarrow n$ variables, $i = 1,\ldots,n$

$C \in \mathbb{N}^{p\times n} \longrightarrow p$ objective functions, $k = 1,\ldots,p$

$A \in \mathbb{N}^{m\times n}$ and $b \in \mathbb{N}^{m\times 1} \longrightarrow m$ constraints, $j = 1,\ldots,m$

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 two 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, $u_k \leqslant v_k,\ \forall k \in \{1,\ldots,p\} \wedge \exists k \in \{1,\ldots,p\} : u_k < v_k$. We denote this relation by $u \prec v$.

Definition 2 (*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, $u_k \leqslant v_k,\ \forall k \in \{1,\ldots,p\}$. We denote this relation by $u \preccurlyeq v$.

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

Definition 3 (*Efficient solution*). A feasible solution $x^* \in X$ is called efficient if there does not exist any other feasible solution $x \in X$ such as $z(x) \prec z(x^*)$.

Definition 4 (*Non-dominated point*). The image $z(x^*)$ in objective space of an efficient solution x^* is called a non-dominated point.

Definition 5 (*Efficient set*). The efficient set denoted by X_E contains all the efficient solutions.

Definition 6 (*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 can distinguish two types of efficient solutions: supported efficient solutions and non-supported efficient solutions [8].

- 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 with all positive components ($\lambda_k > 0, \forall k \in \{1,\ldots,p\}$). 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 $(\text{conv } Z) + \mathbb{R}^p_+$. We can obtain all supported solutions by varying the weight set λ and by solving the corresponding weighted sum single-objective problems.
- 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 $(\text{conv } Z) + \mathbb{R}^p_+$.

We can also make a distinction between supported efficient solutions and define extreme supported efficient solutions and non-extreme supported efficient solutions [8].

- 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 $(\text{conv } Z) + \mathbb{R}^p_+$.
- 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 $(\text{conv } Z) + \mathbb{R}^p_+$.

It is also important to introduce the following classification of the set X_E [13].

Definition 7 (*Equivalent solutions*). Two solutions $x_1, x_2 \in X_E$ are equivalent if $z(x_1) = z(x_2)$.

Definition 8 (*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 9 (*Minimal complete set*). A minimal complete set X_{Em} is a complete set without equivalent solutions. Every complete set contains a minimal complete set.

In this work, we will only try to find an approximation of a minimal complete set: no equivalent solution generated will be thus retained.

3. The bTSP

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

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

Hence, two values are associated to a tour realized by a traveling salesman, who has to visit each city exactly once and to return to the starting city. We are interested here only in the symmetric bTSP, that is $c_k(v_i, v_j) = c_k(v_j, v_i)$ for $1 \leqslant i, j \leqslant N$. In this paper, we use biobjective instances of size going from 100 to 1000, called KroAB100, …, KroAB1000. The instances with less than or equal to 200 cities have been generated on the basis of single-objective TSP instances of the TSPLIB library [22]. The costs between the cities are computed by calculating the euclidean distance between each city. Two files of randomly generated Cartesian coordinates are available for each biobjective instance. For the instances of at least 300 cities, we have generated ourselves the bTSP instances, by randomly generating coordinates. The costs between the cities are computed in the same way than the instances with less than or equal to 200 cities. All the instances and results are available on the web site of the first author (http://www.ig.fpms.ac.be/~lustt).

4. Quality indicators

4.1. 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 [27], because the solutions are represented by a trade-off surface. Consequently, we use several indicators to measure the quality of an approximation A of the efficient set. We call the representation in objective space of an approximation A a non-dominated set (NS). An approximation A has the following property: $\forall x_1, x_2 \in A, z(x_1) \not\leq z(x_2) \wedge z(x_2) \not\leq z(x_1)$ (no solution of the approximation weakly dominates another).

We use the following indicators in this work:

- The hypervolume $\mathscr{H}$ (to be maximized) [26]: Approximation of the volume included under the curve formed by the NS.
- The R measure (normalized between 0 and 1, to be maximized) [15]: Evaluates a NS 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) [6,25] between a reference set of good quality and the NS, by using the euclidean distance. Ideally, the reference set is the Pareto front.
- The number of potentially efficient solutions found, noted $|PE|$.

Both distances D_1 and D_2 are rather good indicators, provided that the reference set is of good quality. The distance D_1 reflects the capacity of an algorithm to reach solutions close to the reference set, so it can be interpreted as an indicator to measure the intensification property of a multiobjective algorithm. On the other hand, the distance D_2 is a good indicator to measure the diversification property of an algorithm.

4.2. Reference set

As said before, to compute the D_1 and D_2 indicators, it is important to have a reference set of excellent quality. To compute the reference set, we use the notion of *ideal set* [17], which is a lower bound of the Pareto front [10]. The ideal set is defined as the best potential Pareto front that can be produced 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. 1 the representation of five extreme supported non-dominated points 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_{ij}^k values of the bTSP are supposed to belong to $\mathbb{N}$, and so $Z \subset \mathbb{N}^2$, 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. 1, we have added these ideal non-extreme supported non-dominated points and ideal non-supported non-dominated points, represented by the circles.

So, 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.

For generating the extreme supported non-dominated points, we use the method proposed by Przybylski et al. [21]. 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 (bMST) problem 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 problem is a relaxation of the bTSP, all feasible solutions of the bTSP remain weakly dominated by the solutions of the ideal set of the bMST.

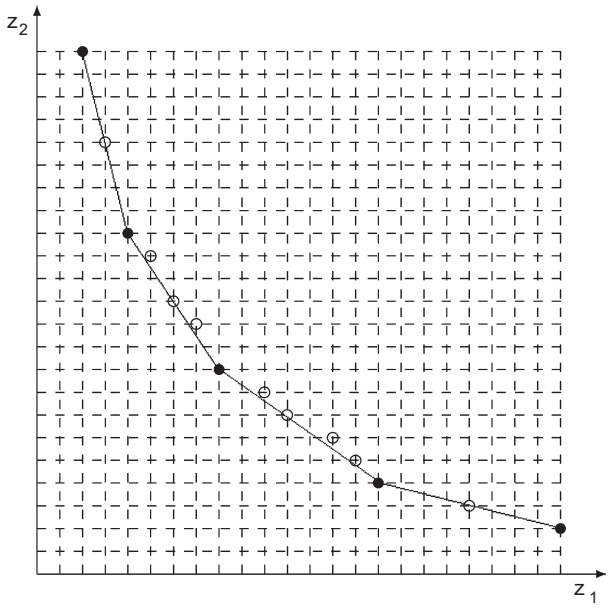

Fig. 1. Ideal set produced on the basis of five extreme supported non-dominated points.

5. Two-Phase Pareto Local Search

The 2PPLS has been developed recently by Lust and Teghem [17,18] and has been applied to the bTSP. The spirit of the two phases of 2PPLS is similar to that of the exact Two-Phase method developed by Ulungu and Teghem [24], 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 generated by resolution of weighted sum single-objective problems obtained by applying a linear aggregation of the objectives. Only a good approximation of a minimal complete set of the extreme supported efficient solutions is sought. To this aim, Lust and Teghem have heuristically adapted the method of Aneja and Nair [1], initially proposed for the resolution of a biobjective transportation problem. The 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. Each single-objective problem is solved with one of the best heuristics for the single-objective TSP: the Lin–Kernighan heuristic. They use the chained Lin–Kernighan version of Applegate et al. [3].
(2) Phase 2: Find non-supported efficient solutions located between the supported efficient solutions. In this phase, they use the Pareto Local Search (PLS) method, used and developed by different authors [2,4,19]. The PLS method is a purely local search algorithm, generalization in the multiobjective case of the most simple metaheuristic: the hill-climbing method. The PLS method does not require any objectives aggregation nor any numerical parameters. In PLS, the neighborhood of every

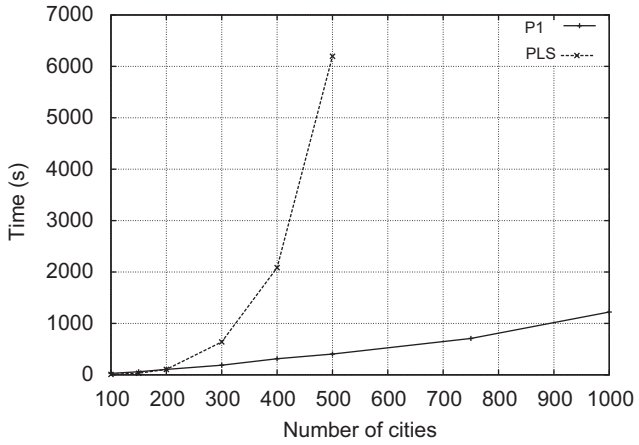

Fig. 2. Evolution of the running time of the two phases.

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-dominated neighbors starting from a solution of the population. Two different versions of PLS are known depending on how the population is updated: the version of Angel et al. [2] and the version of Paquete et al. [19]. Lust and Teghem use the version of PLS of Angel et al., version also been used by Basseur [4] as local search in a memetic algorithm. This version presents two main advantages: not being dependent on the order in which the solutions of the population are examined (contrarily to the version of Paquete et al.) and giving better results than the version of Paquete et al. in similar running times [17].

Lust and Teghem have compared the 2PPLS method with state-of-the-art algorithms and have showed that the 2PPLS method is better on several indicators for instances with up to 200 cities.

We have represented in Fig. 2 the evolution of the running time of the two phases (P1 and PLS) of the 2PPLS method according to the instance size. We use the instances KroAB from size 100 to 1000.

We remark that the running time of the first phase increases more or less linearly according to the instance size. On the other hand, the running time of the second phase, the PLS method, strongly increases. Indeed, in the second phase, Lust and Teghem totally explore the neighborhood of every solution of a population, by making two-exchange moves. Since for each solution of the population the number of neighbors generated is equal to $N(N-3)/2$, it takes $O(n^2)$ time to generate neighbors from one solution of the population.

Therefore, solving instances of more than 500 cities with the 2PPLS method without speed-up techniques is practically impossible. Effectively, we did not manage to solve the instances of 750 and 1000 cites in a reasonable time with the 2PPLS method (for the 500 cities instance, the second phase already takes more than 6000 s). So, speed-up techniques will be useful to reduce the running time of 2PPLS, while keeping better quality results than state-of-the-art methods.

Many speed-up techniques have been developed for the single-objective TSP [5], but to our knowledge, none of these techniques have been adapted to the resolution of the bTSP (excluding biobjective instances resolved by a method using aggregation functions to transform the biobjective problem into several single-objective problems).

Hence, we present at the next section speed-up techniques for solving the bTSP with the 2PPLS method, to reduce the running time of the second phase.

6. Speed-up techniques

6.1. Introduction

Before applying speed-up techniques, let us take a look at the edges used by the solutions of an efficient set. As for biobjective instances, the edges can be represented in a two-dimensional graph (the x-coordinate and y-coordinate of the graph are, respectively, given by the costs 1 and 2 of the edges), we will employ such representation to study what are the edges of a biobjective instance used by the efficient set.

We have represented in Fig. 3, on the left, all the 4950 edges of the biobjective instance KroAB100. On the right, we have represented the edges used by a near-efficient set, which is a very good approximation of the efficient set, obtained with a method presented in [17].

It is noted that only a small proportion of the edges are used by the near-efficient set, and the edges that are bad for both costs are not used at all. So, it clearly seems that it will be possible to implement efficient speed-up techniques. In the right graph, we also add frequencies with which the edges are used by the solutions of the near-efficient set, and we remark that well-located edges (both costs are low) are intensively employed (near to 100% for certain edges, what means that almost all solutions use these edges) while other are slightly used. But the relation between the location of the edges and the frequencies is not clear and would be difficult to take into account.

6.2. Candidate list for the bTSP

A classic speed-up technique for solving single-objective TSP is the candidate list. This speed-up technique is based on the observation of Steiglitz and Weiner [23]: for an improving two-exchange move where (t_1,t_2) and (t_3,t_4) are the leaving edges and where (t_1,t_4) and (t_2,t_3) are the entering edges (see Fig. 4), it must be the case that

$$\text{Either } c_1(t_1,t_2) > c_1(t_2,t_3) \text{ or } c_1(t_3,t_4) > c_1(t_1,t_4) \text{ or both} \quad (1)$$

(where c_1 represents the single-objective cost). That is to say, one of the entering edges must be cheaper than one of the leaving edges.

To take advantage of this observation, a first step is to compute for each city v_i a static list containing the cities in order of increasing cost with v_i. The size of the list is limited to a reasonable size. All the cities are then considered as starting cities for the two-exchange moves. For a starting city t_1 with t_2 the next city in the tour, to consider candidates for t_3 (see Fig. 4), we only need to start at the beginning of the t_2 list and proceed down it until $c_1(t_2,x) > c_1(t_1,t_2)$ or when the end of the list has been reached. To check all possibilities, it is also necessary to start at the beginning of the t_1 list and proceed down it until $c_1(t_1,x) > c_1(t_1,t_2)$ or when the end of the list has been reached. So, for each starting city, two lists are explored: the candidate list of the starting city and the candidate list of the city following the starting city in the tour. As each city of the current tour is considered as starting city, each list is explored two times.

We can then consider two different techniques: to consider the first improvement move, or among all the moves, the best improvement move [14]. If the first improvement technique is used, the examination of the candidate lists is stopped as soon as an improvement move has been found; if there is one. If the best improvement technique is used, among all the improving moves, the move that allows to produce the best improvement is considered.

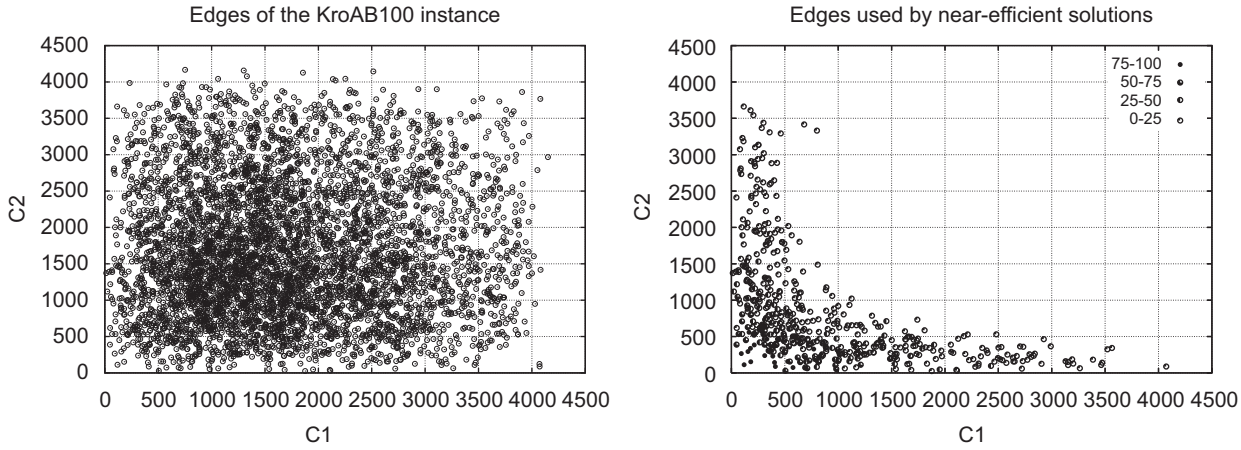

Fig. 3. Edges of the KroAB100 instance.

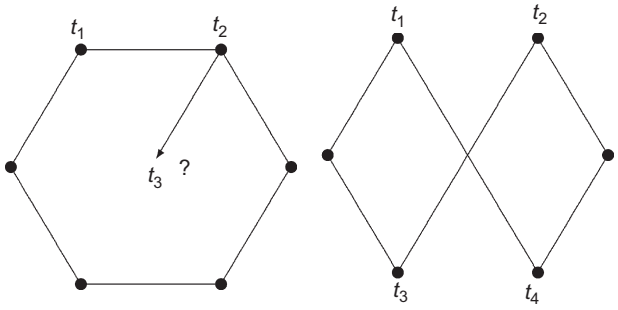

Fig. 4. Two-exchange move.

The candidate list technique is very efficient, and allows to considerably reduce the running time of single-objective heuristics with very small degradations of the quality of the solutions obtained [16].

For the bTSP, for an improving two-exchange move where (t_1, t_2) and (t_3, t_4) are the leaving edges and where (t_1, t_4) and (t_2, t_3) are the entering edges, it must be the case that

$$\text{Either } c(t_1,t_2) \not\prec c(t_2,t_3) \text{ or } c(t_3,t_4) \not\prec c(t_1,t_4) \text{ or both} \quad (2)$$

(where c represents the cost vector, of dimension 2 in the biobjective case). That is to say, at least one of the entering edges must not to be dominated by one of the leaving edges. Otherwise, the move will not lead to a new non-dominated tour as the cost vector of the entering edges (equal to $c(t_2,t_3)+c(t_1,t_4)$) will be dominated by the cost vector of the leaving edges (equal to $c(t_1,t_2)+c(t_3,t_4)$).

In the biobjective case, it is not more possible to sort out each candidate list, since there is no more total order between the cost vectors c. We have thus to explore each candidate list of each city until the end of the list has been reached. We see that this technique will not be as effective as in the single-objective case since each candidate list has to be explore until the end. For this reason, we do not take into account the relation (2) in the exploration of the candidate lists. In this way, to check all possibilities, each list has to be explore only one time.

We present below how to create the candidate lists in the biobjective case. Two different techniques are presented.

6.2.1. k-Nearest neighbors

A simple way to consider both costs is to create first two lists for each city. The first list contains the cities in order of increasing cost, for the objective 1, and the second, the cities in order of increasing cost, for the objective 2. We then merge the two lists to obtain a unique candidate list of size $2*k$ by paying attention to two things: the same city cannot appear twice in the same list and if the city v_j is in the candidate list of the city v_i, the city v_i cannot be in the candidate list of the city v_j, to avoid double evaluation of the same move.

We have represented in Fig. 5 the edges that are taken into account for the two-exchange moves (called candidate edges) for $k=1$ and 5. We see that with this technique edges having high costs for both objectives are not candidates.

6.2.2. Data dominance relations

Another way to create candidate lists is to use data dominance relations. Indeed, as we have seen in Fig. 3, edges that are Pareto dominated by many other edges do not appear in the near-efficient set.

So, to determine the edges that will be used, we associate to each edge a rank, based on the dominance ranking developed by Goldberg [12].

All the non-dominated edges have a rank equal to 0. These edges are then removed, and the following non-dominated edges obtain a rank equal to 1. This process is repeated until a rank equal to the value given by a parameter D has been obtained.

We show in Fig. 6 the representation of the edges, for $D=0$, 1, 10 and 20. We remark that with this technique, the set of candidate edges visually better fits to the edges used by the near-efficient set, than with the k-nearest neighbors technique (Fig. 5).

At the end, we create for each city a candidate list by only considering the candidate edges given by the data dominance relations. To do that, we explore the set of candidate edges, and for each candidate edge $\{v_i, v_j\}$, we add the city v_j to the candidate list of the city v_i (if v_i is not already in the candidate list of v_j).

6.3. "Don't look bits"

In the previous speed-up techniques, all the cities are always considered as starting cities for the two-exchange moves. It is possible to not consider all the cities, with a simple rule, often implemented in singe-objective heuristics. This rule is known under the name "don't look bits" [5]. Here the observation is that if a starting city t_1 previously failed to find an improving move (a move that generates a new potentially non-dominated tour), and if the neighbors of t_1 in the tour have not changed since that time, the probability that an improved move will be found starting from t_1 is low.

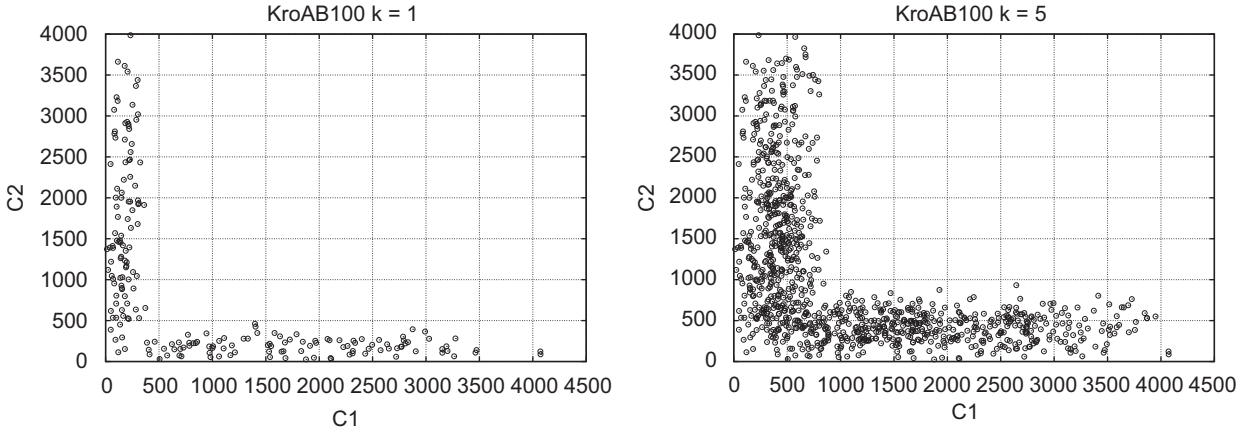

Fig. 5. k-Nearest neighbors.

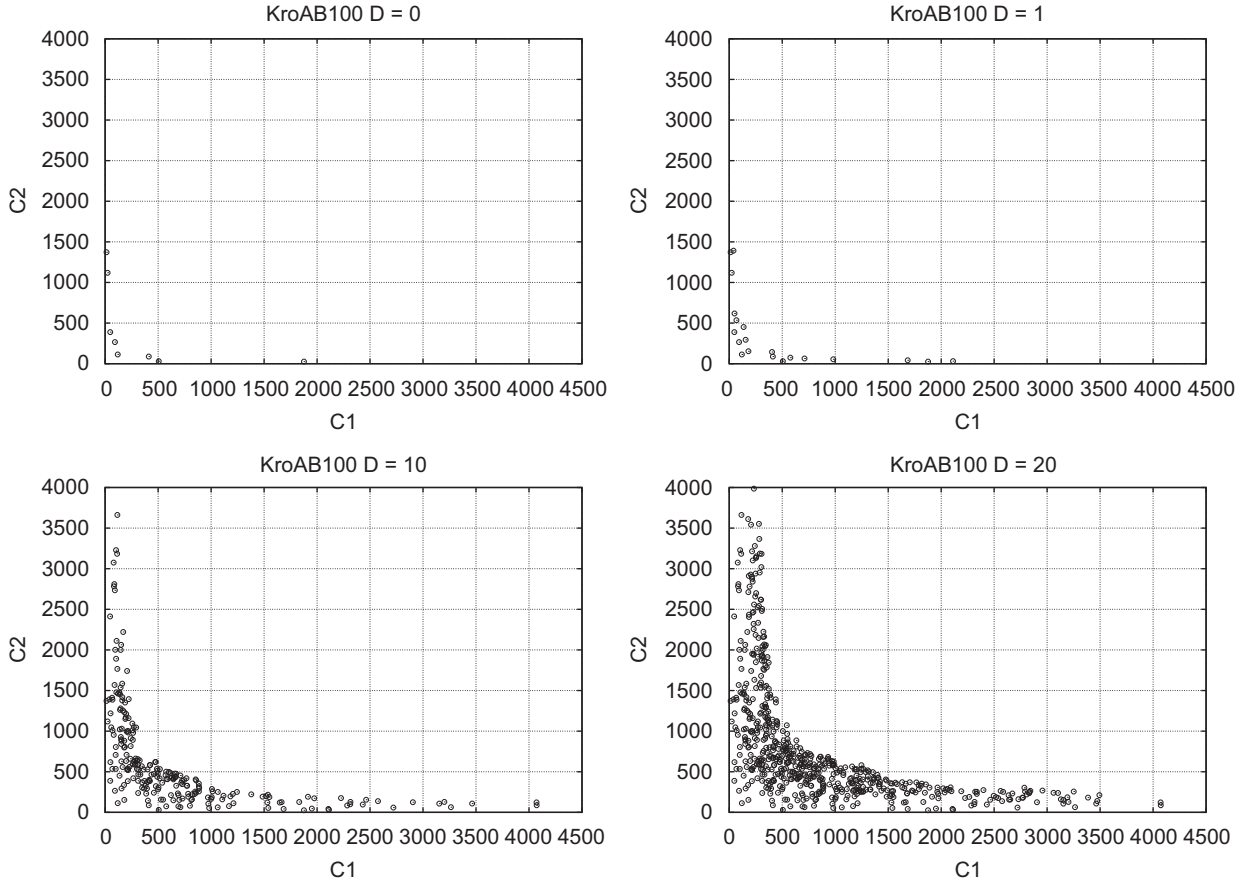

Fig. 6. Data dominance relations KroAB100.

We exploit this observation by means of special bits for each city. Before applying the PLS method, we associate to each solution of the population a boolean array "don't look bits", containing bits all turned off. For a solution, the bit for city c is turned on whenever a search for an improving move with $t_1 = c$ fails and is turned off whenever its predecessors and successors have changed, that is to say an improving move is performed in which c is an endpoint of one of the deleted edges.

When we try to generate non-dominated tours from a solution of the population, we ignore all starting cities t_1 whose bits given by the array "don't look bits" of the solution are switched on.

7. Results

We first present in this section the results of the comparison between the speed-up techniques, based on different figures showing the evolution of the D_1 and R indicators according to the running time of the second phase of 2PPLS (the speed-up techniques have no influence on the running time of the first phase). As the 2PPLS method is stochastic (the stochasticity comes from the first phase only), we make the average of the indicators over three executions. This number of executions is enough since the aim of the different figures is to see which speed-up

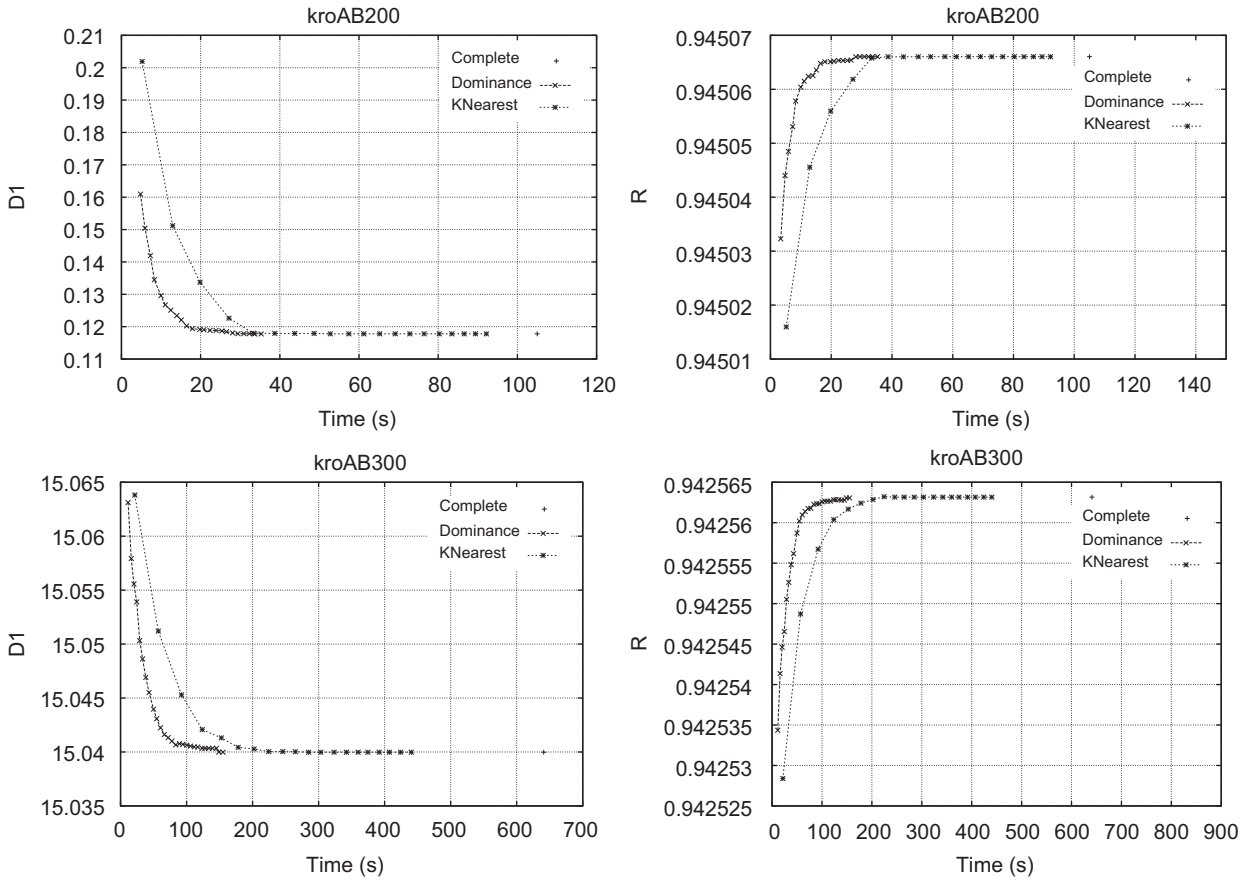

Fig. 7. Comparison between k-nearest and data dominance relations.

techniques dominate the others for different lengths of the candidate lists.

We then statistically compare the speed-up techniques between them, with the standard `2PPLS` method (with no speed-up techniques) and with another state-of-the-art method. For these comparisons, we make the average of the indicators over 20 executions.

All the algorithms experimented in this work have been run on a Pentium with 3 Ghz and 512 Mo of memory. The running time of our implementation of the algorithms corresponds to the wall clock time.

7.1. Comparison between k-nearest and data dominance relations

We compare two different techniques to create candidate lists: candidate list based on k-nearest neighbors and candidate list based on data dominance relations.

To do that, we vary the value of k for the k-nearest technique and the value of D for the data dominance relations technique.

We have represented in Fig. 7 the values obtained by the k-nearest neighbor technique (under the name `KNearest`) and the data dominance relations technique (under the name `Dominance`) for the D_1 and R indicators according to the running time, for the KroAB200 and KroAB300 instances.

The running time is controlled by the value of k for `KNearest` and by the value of D for `Dominance`.

We can see that the `Dominance` technique is more efficient than the `KNearest` technique: for the same running time, the values of the D_1 and R indicators obtained with `Dominance` are better than the values obtained with `KNearest`. Obviously, when a certain value for D or k is reached, the results with these two techniques are similar, since the sets of candidate edges become the same.

We have also represented the results of the complete exploration of the neighborhood (under the name `Complete`) and we can see that both techniques allow to improve the running time (with a speed-up factor equal to about 3) by keeping results of same quality.

The value of D that allows to obtain the same quality result than the complete exploration is equal to 60 for the KroAB200 instance and 70 for the KroAB300 instance. For k, the values are, respectively, equal to 12 and 16.

7.2. Use of the edges found after `P1`

Fixing a good value for D is not necessarily obvious, so we propose here a simple way to determine candidate edges without having to fix a value for D.

The idea comes from the following observation: after application of the `P1` of `2PPLS`, a set of potentially efficient solutions is already discovered and it would be relevant to retrieve information from this set.

We have represented in Fig. 8 all the edges used in at least one solution generated in `P1` and edges used by a near-efficient set for the KroAB100 instance. We can see that both sets of candidate edges are very close. So, it seems that using only the edges found after `P1` for the creation of the candidate lists will already give good results.

We have represented in Fig. 9 the comparison for the D_1 and R indicators according to the running time between candidate lists created with the `Dominance` technique and candidate lists created

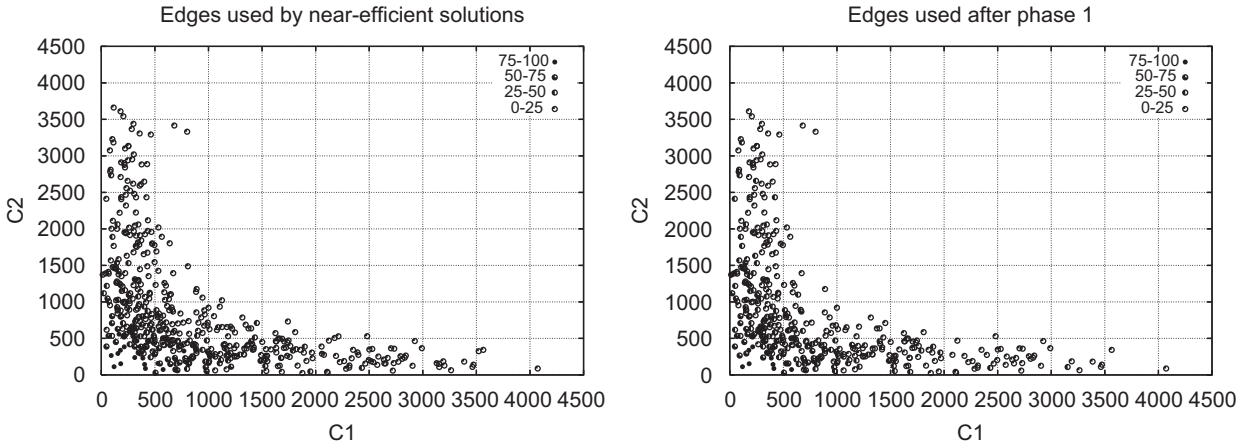

Fig. 8. Edges used by a near-efficient set and by the set obtained after `P1` (KroAB100 instance).

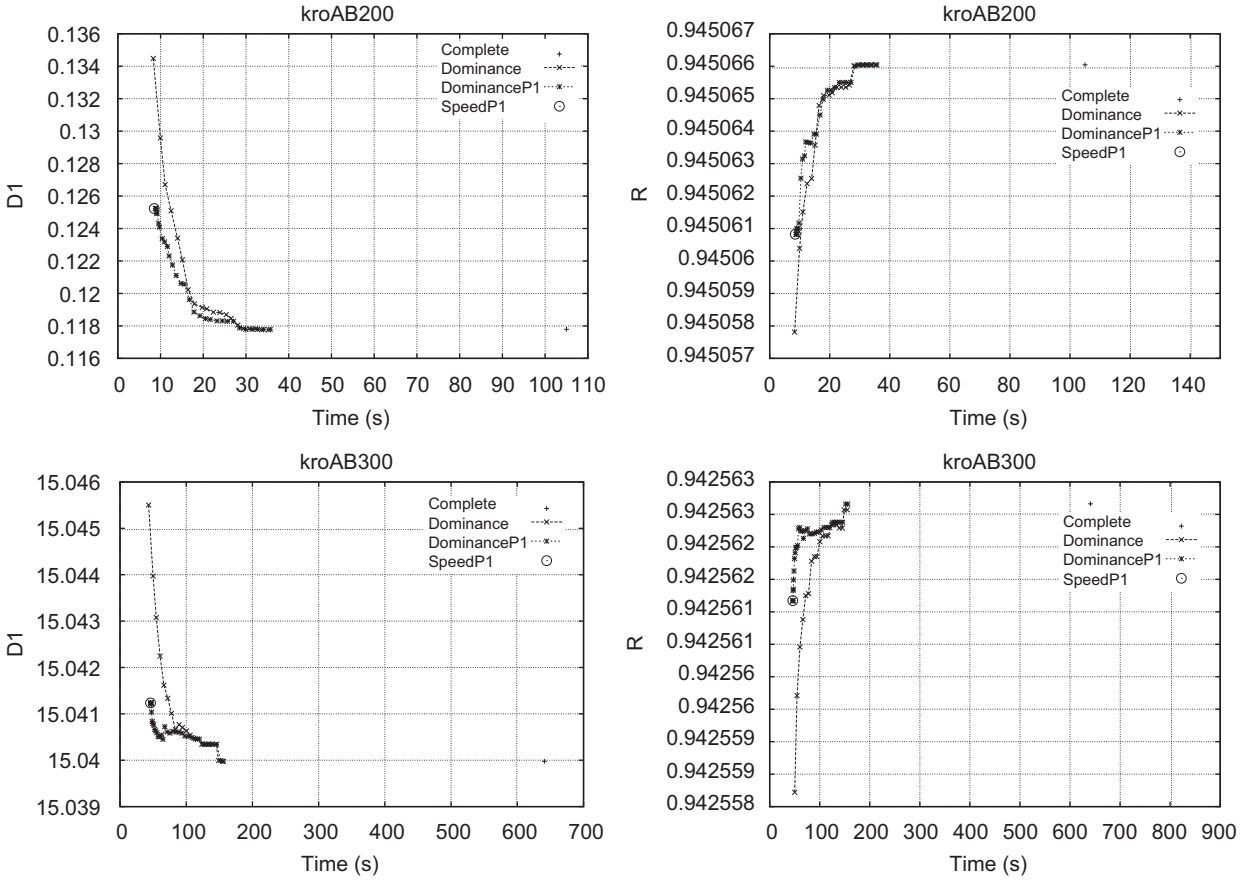

Fig. 9. Comparison between data dominance relations and data dominance relations based on P1.

from the edges found after `P1` (under the name `SpeedP1`), for the KroAB200 and KroAB300 instances (note that the scale of this figure is different to the scale of the preceding ones). We can see that the `SpeedP1` technique allows to obtain better results than the `Dominance` technique: for the same running time the indicators D_1 and R found by `SpeedP1` are better than the indicators found by `Dominance`.

Comparing to the complete exploration of the neighborhood, the results given by the `SpeedP1` technique are very close with a much lower running time (the speed-up factor is equal to about 10). On the other hand, we can improve the results given by `SpeedP1` by adding edges that were not used after `P1`, in the order given by the data dominance relations. We can see these results on the same figure under the name `DominanceP1`. By increasing D and therefore the running time, the results given by `SpeedP1` are better and are finally as good as the complete exploration, always with a lower running time. Also, after a certain running time, the results given by `DominanceP1` and `Dominance` are the same given that from a

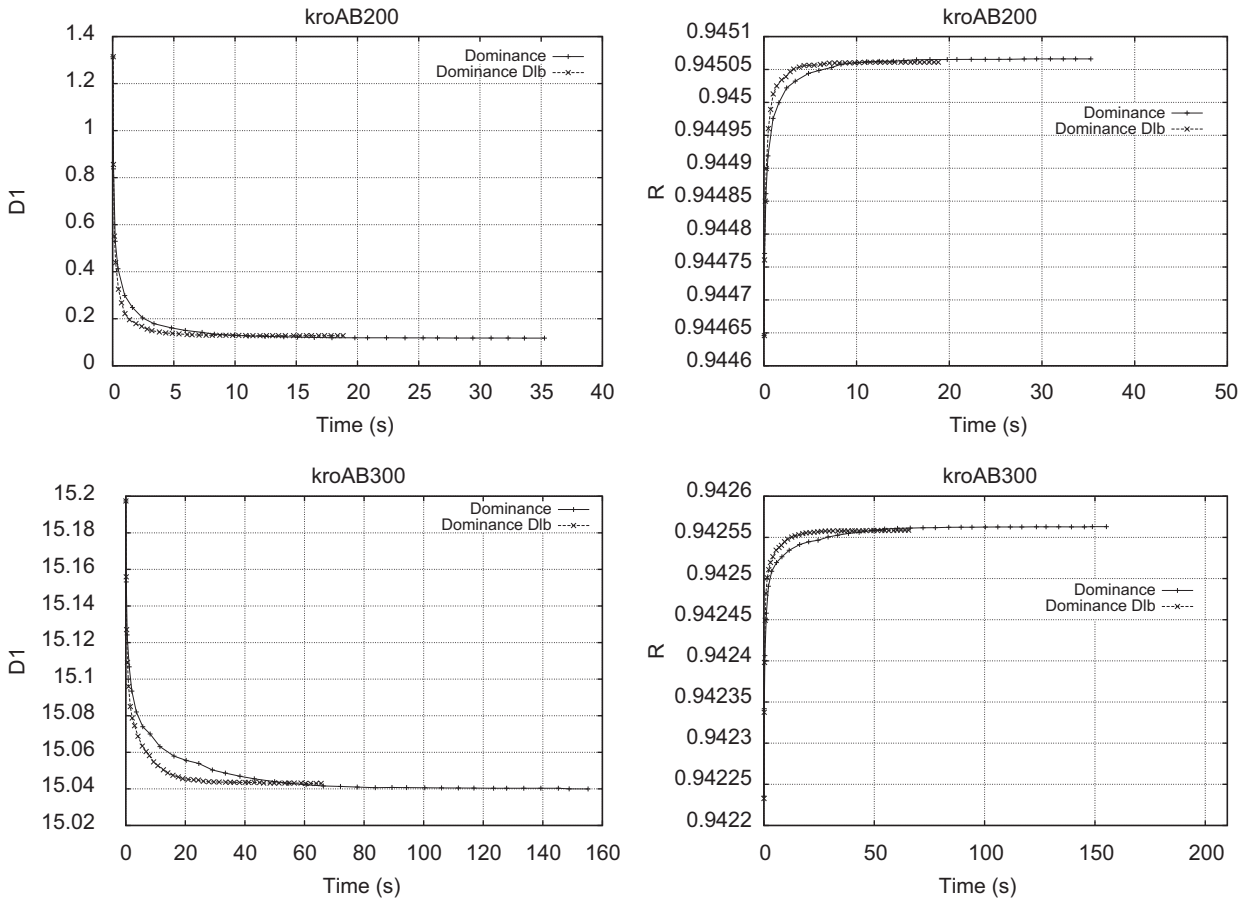

Fig. 10. Data dominance relations and "don't look bits".

certain value of D, the sets of candidate edges are equal for both techniques.

7.3. Use of "don't look bits"

We can see in Fig. 10 the results obtained with the data dominance relations without and with the "don't look bits" technique (respectively, under the name `Dominance` and `Dominance Dlb`).

We remark that for a low running time, the "don't look bits" technique gives better results on the D_1 and R indicators for the KroAB200 and KroAB300 instances. But by increasing D and therefore the running time, the performances without "don't look bits" are better. Furthermore, with the "don't look bits" technique, we do not manage to reach the same quality results than without, even when the value of D is increased. But the difference remains low.

We can see in Fig. 11 the results obtained with the `speedP1` technique without and with the "don't look bits" technique (respectively, under the name `SpeedP1` and `SpeedP1 Dlb`) and the results obtained by adding edges that were not used after `P1`, in the order given by the data dominance relations, without and with the "don't look bits" technique (respectively, under the name `DominanceP1` and `DominanceP1Dlb`).

We remark that the results obtained with the "don't look bits" technique are of worse quality than without (for a same running time, the indicators without "don't look bits" technique are better). But this technique remains interesting, since it allows to give approximations of relatively good quality in a low running time.

7.4. Summary

7.4.1. Statistical comparison between speed-up techniques

We can see in Table 1 the results obtained with the different methods, that is to say:

- the standard `2PPLS` method, that is the method without speed-up techniques;
- `2PPLS` with the "don't look bits" technique (`Dlb`);
- `2PPLS` with the speed-up technique based on the edges found after `P1` (`SpeedP1`); and
- `2PPLS` with both techniques (`SpeedP1+Dlb`) on the KroAB200, KroAB300, KroAB400 and KroAB500 instances.

We remark that the values of the indicators are practically the same but the running time of `PLS` (second phase of `2PPLS`) is strongly reduced by employing the speed-up techniques. We also observe that the `SpeedP1` technique is more efficient than the `Dlb` technique.

We also tried to solve instances of size equal to 750 and 1000. For these instances, it was not possible to apply the standard `2PPLS` method while keeping reasonable running times. We have represented these results in Table 2. Even with the `SpeedP1` technique, the running time is high (more than 5000 s for the KroAB1000 instance). But by applying the "don't look bits" technique coupled to the `SpeedP1` technique, it is possible to strongly reduce the running time.

To take into account the variations in the results of the algorithms, as we do not know the distributions of the indicators, we also carried out the non-parametric statistical test of Mann–Whitney [11].

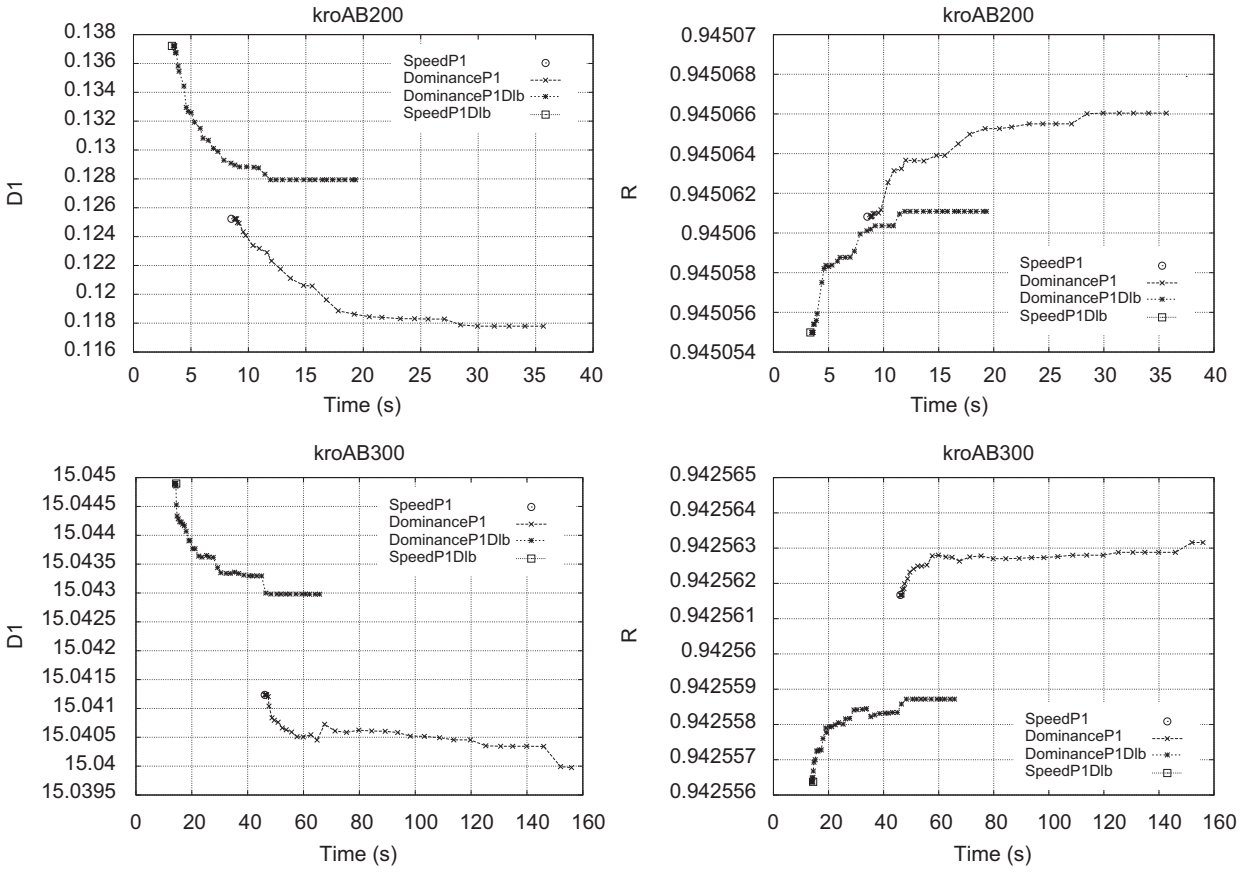

Fig. 11. SpeedP1 and "don't look bits".

Table 1
Comparison between speed-up techniques (1).

Instance	Speed-up	$\mathscr{H}(10^8)$	R	D_1	D_2	$\|PE\|$	Time (s) (P1+PLS)
KroAB200	/	1076.0781	0.945067	0.115	4.410	6736.50	106 + 106
	Dlb	1076.0600	0.945062	0.125	4.410	6168.00	106 + 26
	SpeedP1	1076.0619	0.945062	0.123	4.410	6210.75	106 + 9
	SpeedP1+Dlb	1076.0411	0.945056	0.134	4.410	5569.95	106 + 3
KroAB300	/	1952.8212	0.942563	15.040	18.692	14 853.10	188 + 640
	Dlb	1952.9939	0.942558	15.045	18.716	13 446.85	188 + 125
	SpeedP1	1952.1064	0.942561	15.042	19.048	14 101.25	188 + 44
	SpeedP1+Dlb	1952.2790	0.942556	15.047	19.052	12 514.85	188 + 14
KroAB400	/	3461.5350	0.944511	16.781	36.307	22 068.25	312 + 2085
	Dlb	3461.4723	0.944505	16.788	36.309	19 249.45	312 + 339
	SpeedP1	3461.5105	0.944509	16.784	36.583	21 098.90	312 + 122
	SpeedP1+Dlb	3461.4436	0.944503	16.790	36.585	18 092.00	312 + 33
KroAB500	/	5495.8583	0.946919	17.332	22.559	33 690.80	404 + 6197
	Dlb	5495.7553	0.946913	17.339	22.575	28 459.75	404 + 784
	SpeedP1	5495.8286	0.946917	17.334	22.568	32 245.95	404 + 327
	SpeedP1+Dlb	5495.7208	0.946911	17.340	22.580	26 496.60	404 + 73

Table 2
Comparison between speed-up techniques (2).

Instance	Speed-up	$\mathscr{H}(10^8)$	R	D_1	D_2	$\|PE\|$	Time(s)(P1+PLS)
KroAB750	Dlb	12 924.4508	0.952319	17.181	31.616	50 162.40	708 + 3367
	SpeedP1	12 924.6259	0.952323	17.174	32.233	59 345.00	708 + 1593
	SpeedP1+Dlb	12 924.3840	0.952317	17.184	31.414	46 332.20	708 + 266
KroAB1000	Dlb	22 656.9005	0.954128	17.538	23.300	79 793.85	1222 + 10872
	SpeedP1	22 658.5914	0.954132	17.531	23.581	97 119.10	1222 + 5984
	SpeedP1+Dlb	22 658.2416	0.954127	17.540	23.382	73 990.75	1222 + 798

This test allows to compare the distributions of the indicators of the standard 2PPLS method with the indicators of 2PPLS+SpeedP1.

The results of the comparison of 2PPLS with 2PPLS+SpeedP1 are given in Table 3. We only use the D_1 and R indicators, particularly revealing. We test the following hypothesis: "the two samples come from identical populations" for the D_1 or R indicator on a given instance. When the hypothesis is satisfied, the result "=" is indicated (no differences between the indicators of the algorithms). When the hypothesis is not satisfied, the sign ">" (2PPLS+SpeedP1 is better) or "<" (2PPLS+SpeedP1 is worse) is indicated. The level of risk of the test has been fixed to 5%.

We remark that for the KroAB200 instance, the standard 2PPLS method is statistically better than the 2PPLS+SpeedP1 method on the D_1 and R indicators. For the KroAB300, the 2PPLS method is better only on the R indicator. For the rest of the instances, that is the large-scale instances, they are no statistically differences between both methods.

Furthermore, we show in Fig. 12 the results of the comparison between the potentially non-dominated points found by 2PPLS and by 2PPLS+SpeedP1, for the 300 and 500 cities instances. To create these box-plot graphs, we compare the points found by the 2PPLS+SpeedP1 method with the points found by the 2PPLS method. Four cases can occur: a point of 2PPLS+SpeedP1 is dominated by at least one point of 2PPLS (Dominated), a point of 2PPLS+SpeedP1 dominates at least one point of 2PPLS (Dominate), a point of 2PPLS+SpeedP1 is equal to an another point of 2PPLS (Commons), or the result of the comparison belongs to none of these three possibilities (Others). As the algorithms are run 20 times, we compare the potentially non-dominated points of the execution i of 2PPLS with the potentially non-dominated points of the execution i of 2PPLS+SpeedP1, with $1 \leqslant i \leqslant 20$. We choose this scheme because the execution i of 2PPLS is launched with the same initial population than the execution i of 2PPLS+SpeedP1.

Table 3
Results of the Mann–Whitney test for the D_1 and R indicators (2PPLS+SpeedP1 compared to 2PPLS).

Instance	D_1	R
KroAB200	<	<
KroAB300	=	<
KroAB400	=	=
KroAB500	=	=

We can see thanks to these box-plot graphs that about 80% of the solutions obtained with the 2PPLS+SpeedP1 method are also generated by 2PPLS. Only about 15% of the solutions of 2PPLS+SpeedP1 are dominated by 2PPLS. Moreover, the 2PPLS+SpeedP1 method allows to generate some new solutions that dominate solutions obtained with 2PPLS.

7.4.2. Gain in running time and number of neighbors generated

In Fig. 13, we have represented the evolution of the gain in running time by applying the SpeedP1 technique comparing to the standard 2PPLS method.

We can see than the gain in running time is increasing according to the instance size.

In Fig. 14, we show two things: the evolution of the ratio equal to the number of edges found after P1 and the total number of edges, and the ratio between the number of neighbors generated by 2PPLS+SpeedP1 and the number of neighbors generated by 2PPLS.

As we can see, both rations are decreasing, what can explain why the gain in running time is increasing.

7.4.3. Comparison between 2PPLS and PD-TPLS

The PD-TPLS method is an efficient method for the resolution of the bTSP proposed by Paquete and Stützle [20]. In [18], Lust and Teghem have shown that the standard 2PPLS method generates results of better quality than PD-TPLS on instances with up to

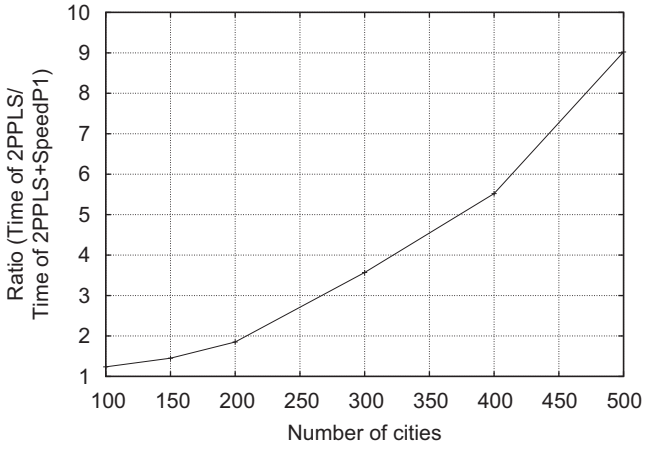

Fig. 13. Evolution of the gain in running time between SpeedP1 and the standard 2PPLS method.

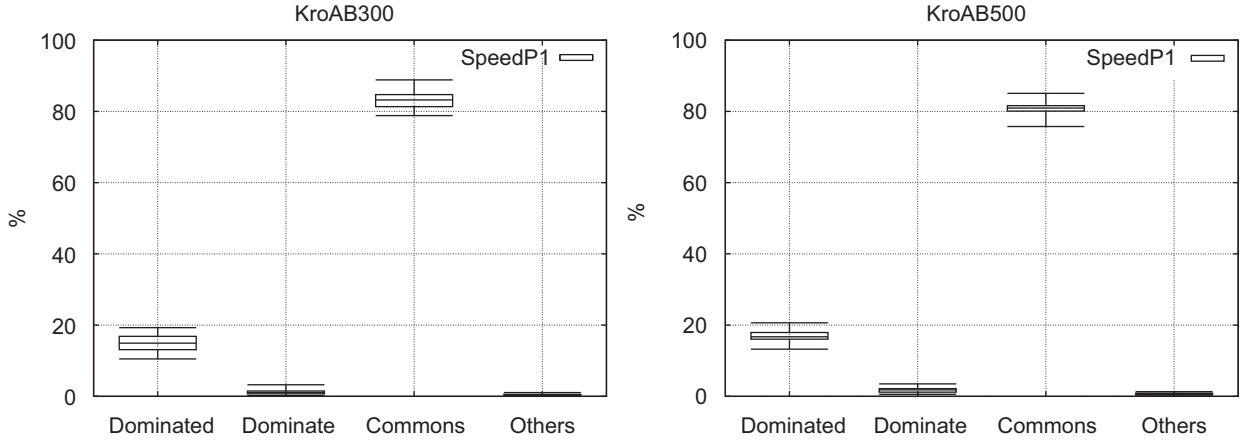

Fig. 12. Comparison between the solutions.

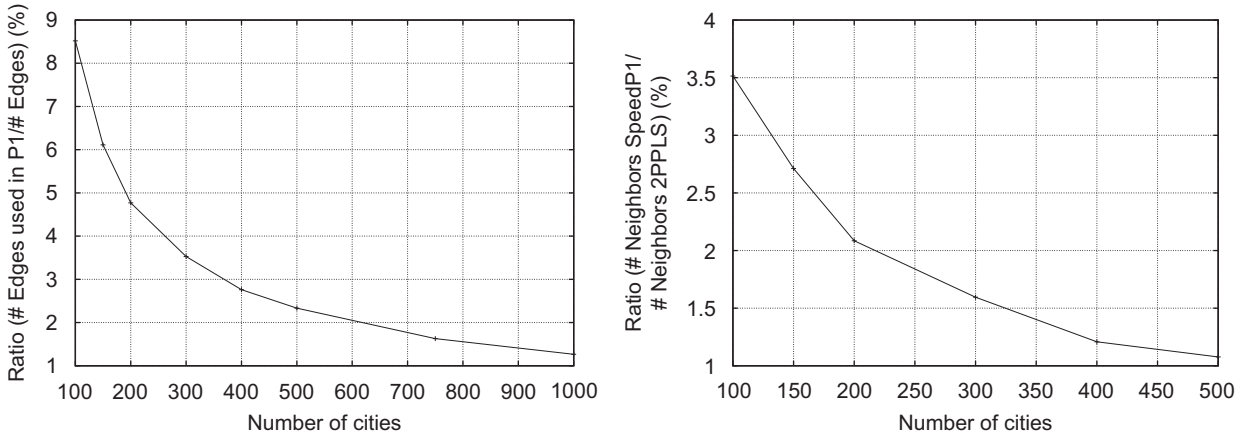

Fig. 14. Evolution of the ratio between the number of edges found after P1 and the total number of edges (%) and evolution of the ratio between the number of neighbors generated by 2PPLS+SpeedP1 and the number of neighbors generated by 2PPLS (%).

Table 4
Comparison between 2PPLS+SpeedP1, 2PPLS+SpeedP1+Dlb and PD-TPLS.

Instance	Method	$\mathcal{H}(10^8)$	R	D_1	D_2	$\|PE\|$	Time (s)
KroAB300	2PPLS+SpeedP1	1952.1064	**0.942561**	**15.042**	19.048	**14 101.25**	232
	2PPLS+SpeedP1+Dlb	**1952.2790**	0.942556	15.047	19.052	12 514.85	**202**
	PD-TPLS	1929.8330	0.942473	15.155	**18.903**	4464.90	258
KroAB400	2PPLS+SpeedP1	**3461.5105**	**0.944509**	**16.784**	**36.583**	**21 098.90**	434
	2PPLS+SpeedP1+Dlb	3461.4436	0.944503	16.790	36.585	18 092.00	**345**
	PD-TPLS	3460.5673	0.944428	16.912	38.874	6478.75	591
KroAB500	2PPLS+SpeedP1	**5495.8286**	**0.946917**	**17.334**	**22.568**	**32 245.95**	731
	2PPLS+SpeedP1+Dlb	5495.7208	0.946911	17.340	22.580	26 496.60	**477**
	PD-TPLS	5493.4948	0.946848	17.468	22.751	8634.45	1137
KroAB750	2PPLS+SpeedP1	**12 924.6259**	**0.952323**	**17.174**	32.233	**59 345.00**	2301
	2PPLS+SpeedP1+Dlb	12 924.3840	0.952317	17.184	31.414	46 332.20	**974**
	PD-TPLS	12 921.4842	0.952270	17.298	**30.861**	14 511.80	3579
KroAB1000	SpeedP1	**22 658.5914**	**0.954132**	**17.531**	23.581	**97 119.10**	7206
	SpeedP1+Dlb	22 658.2416	0.954127	17.540	**23.382**	73 990.75	**2020**
	PD-TPLS	22 493.5355	0.954090	17.653	24.598	21 984.25	8103

200 cities. But for instances with more than 200 cities, the running times of the standard 2PPLS method are higher than the running times of PD-TPLS. Thus, we compare our method with speed-up techniques with our own implementation of the PD-TPLS method. For this implementation, we have fixed the parameters of this method as follows, as done in [20]:

- The number of iterations for the number of perturbation steps is equal to the number of cities N minus 50.
- The number of aggregations is equal to N.

We can see the results in Table 4 for the instances with more than 200 cities, and we remark, that the 2PPLS method with the SpeedP1 technique with or without the Dlb technique allows to obtain better results in lower running times, for all the indicators considered, except for the D2 indicator in some cases.

The results of the comparison of 2PPLS+SpeedP1 with PD-TPLS for the Mann–Whitney test are given in Table 5.

We remark that the 2PPLS+SpeedP1 method finds statistically better results than the PD-TPLS method on the D_1 and R indicators. We can thus affirm with a very low risk that, for the D_1 and R indicators, 2PPLS+SpeedP1 is better than PD-TPLS on the large-scale instances experimented in this work.

Table 5
Results of the Mann–Whitney test for the D_1 and R indicators (2PPLS+SpeedP1 compared to PD-TPLS).

Instance	D_1	R
KroAB300	>	>
KroAB400	>	>
KroAB500	>	>
KroAB750	>	>
KroAB1000	>	>

8. Conclusion

We have proposed different speed-up techniques to solve large-scale bTSP instances with the 2PPLS method. With the candidate lists we are able to achieve in much shorter time results as good as the standard 2PPLS method with a complete exploration of the neighborhood. With further reduction of the running time, e.g. if only the edges found after the phase 1 are considered in the second phase, the quality of the generated solutions only slightly deteriorates and are statically comparable for instances with at least 400 cities. So, this simple parameter-free method gives a good compromise between performances and running time. Further reduction of the running

time, at the additional cost of the quality, may be obtained by adding the "don't look bits" technique. Such method may be used to obtain in a very short time a reasonably good approximation of the Pareto front of large-scale instances.

Our method has been compared to another state-of-the-art method for bTSP, the PD-TPLS method, on large-scale instances. We have shown that our new method outperforms PD-TPLS on both quality of results and running time.

We also gave by this work state-of-the-art results for biobjective instances of the TSP with more than 200 cities, until 1000 cities, while before only instances with up to 200 cities have been tackled. Among the research perspectives, automatically fixing a good value for D for the speed-up technique based on data dominance relations will be interesting, as well as taking into account the frequencies with which the edges are used. Another interesting and obvious future work would be to test the 2PPLS method on more than two objective instances. The speed-up techniques could also be adapted to another large-scale biobjective combinatorial optimization problems.

Acknowledgment

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

References

[1] Aneja YP, Nair KPK. Bicriteria transportation problem. Management Science 1979;25:73–8.
[2] 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, editors. Metaheuristics for multiobjective optimisation. Lecture notes in economics and mathematical systems, vol. 535. Berlin: Springer; 2004. p. 153–76.
[3] Appletgate D. Chained Lin–Kernighan for large traveling salesman problems. INFORMS Journal on Computing 2003;15:82–92.
[4] Basseur M. Design of cooperative algorithms for multi-objective optimization: application to the flow-shop scheduling problem. 4OR 2006;4(3):255–8.
[5] Bentley JL. Fast algorithms for geometric traveling salesman problems. ORSA Journal on Computing 1992;4:387–411.
[6] Czyzak P, Jaszkiewicz A. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis 1998;7:34–47.
[7] Deb K. Multi-objective optimization using evolutionary algorithms. Wiley-interscience series in systems and optimization. Chichester: Wiley; 2001.
[8] Ehrgott M. Multicriteria optimization. 2nd ed., Berlin: Springer; 2005.
[9] Ehrgott M, Gandibleux X. Multiple criteria optimization: state of the art annotated bibliographic surveys. Boston: Kluwer Academic Publishers; 2002.
[10] Ehrgott M, Gandibleux X. Bound sets for biobjective combinatorial optimization problems. Computers and Operations Research 2007;34:2674–94.
[11] Ferguson TS. Mathematical statistics, a decision theoretic approach. New-York and London: Academic Press; 1967.
[12] Goldberg DE. Genetic algorithms for search, optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.
[13] Hansen P. Bicriterion path problems. In: Lecture notes in economics and mathematical systems, vol. 177, 1979, p. 109–27.
[14] Hansen P, Mladenovic N. First vs. best improvement: an empirical study. Discrete Applied Mathematics 2006;154:802–17.
[15] Jaszkiewicz A. On the performance of multiple-objective genetic local search on the 0/1 Knapsack problem—a comparative experiment. IEEE Transactions on Evolutionary Computation 2002;6(4):402–12.
[16] Johnson DS, McGeoch LA. The traveling salesman problem: a case study. In: Aarts EHL, Lenstra JK, editors. Local search in combinatorial optimization. New York: Wiley; 1997. p. 215–310.
[17] Lust T, Teghem J. Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics 2009; doi:10.1007/s10732-009-9103-9.
[18] Lust T, Teghem J. Two-phase stochastic local search algorithms for the biobjective traveling salesman problem. Technical Report, IRIDIA, Technical Report No.TR/IRIDIA/2007-014; 2007.
[19] 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, editors. Metaheuristics for multiobjective optimisation. Lecture notes in economics and mathematical systems, vol. 535. Berlin: Springer; 2004. p. 177–99.
[20] Paquete L, Stützle T. A two-phase local search for the biobjective traveling salesman problem. In: Fonseca CM, Fleming PJ, Zitzler E, Deb K, Thiele L, editors. Evolutionary multi-criterion optimization. Second international conference, EMO 2003. Lecture notes in computer science, vol. 2632. Faro, Portugal: Springer; April 2003. p. 479–93.
[21] Przybylski A, Gandibleux X, Ehrgott M. Two-phase algorithms for the biobjective assignment problem. European Journal of Operational Research 2008;185(2):509–33.
[22] Reinelt G. Tsplib—a traveling salesman problem library. ORSA Journal of Computing 1991;3(4):376–84.
[23] Steiglitz K, Weiner P. Some improved algorithms for computer solution of the traveling salesman problem. In: Proceedings 6th annual allerton conference on communication, control, and computing. Urbana: Department of Electrical Engineering and the Coordinated Science Laboratory, University of Illinois; 1968. p. 814–21.
[24] Ulungu EL, Teghem J. The two phases method: an efficient procedure to solve biobjective combinatorial optimization problems. Foundation of Computing and Decision Science 1995;20:149–56.
[25] Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D. MOSA method: a tool for solving multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis 1999;8(4):221–36.
[26] Zitzler E. Evolutionary algorithms for multiobjective optimization: methods and applications. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland; November 1999.
[27] Zitzler E, Laumanns M, Thiele L, Fonseca CM, Grunert da Fonseca V. Why quality assessment of multiobjective optimizers is difficult. In: Langdon WB, Cantú-Paz E, Mathias K, Roy R, Davis D, Poli R et al., editors. Proceedings of the genetic and evolutionary computation conference (GECCO'2002). San Francisco, California: Morgan Kaufmann Publishers; July 2002. p. 666–73.