Computers & Operations Research 36 (2009) 2905 – 2912

Contents lists available at ScienceDirect

Computers & Operations Research

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

Heuristics for the bi-objective path dissimilarity problem

Rafael Martí[a,*], José Luis González Velarde[b], Abraham Duarte[c]

[a]*Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain*
[b]*Centro de Manufactura y Calidad, Tecnológico de Monterrey, México*
[c]*Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Spain*

ARTICLE INFO

Available online 10 January 2009

Keywords:
Metaheuristics
Bi-objective programming
Routing

ABSTRACT

In this paper the path dissimilarity problem is considered. The problem has previously been studied within several contexts, the most popular of which is motivated by the need to select transportation routes for hazardous materials. The aim of this paper is to formally introduce the problem as a bi-objective optimization problem, in which a single solution consists of a set of p different paths, and two conflicting objectives arise, on one hand the average length of the paths must be kept low, and on the other hand the dissimilarity among the paths in the set should be kept high. Previous methods are reviewed and adapted to this bi-objective problem, thus we can compare the methods using the standard measures in multi-objective optimization. A new GRASP procedure is proposed and tested against the revised methods, and we show that it is able to create better approximations of efficient frontiers than existing methods.

1. Introduction

In this paper we consider a multi-objective routing problem in which a set of paths from an origin to a destination must be generated. Finding different paths in a graph is a classical optimization problem. The best known is the k-shortest path problem in which the shortest, second shortest, … , kth shortest paths from an origin o to a destination d are obtained in a graph. However, many of these alternative paths are likely to share a large number of edges. This is why in some applications we need to consider an alternative approach. For example, in the context of hazmat transportation we want to obtain spatially dissimilar paths that minimize the risk (distributing the risk over all regional zones to be crossed uniformly). Specifically, we consider here the path dissimilarity problem (PDP), which involves obtaining a set of p paths with minimum length and maximum dissimilarity.

Given a graph $G=(V, E)$ with V the set of vertices or nodes and E the set of edges with associated cost c_{ij} for $(i,j)\in E$, and a pair of origin–destination vertices, o–d, we define $P(o,d)$ as the set of all paths in G from o to d. Note that in most applications the cost c_{ij} of edge (i,j) is its Euclidean distance. Given an integer number $p>1$, a feasible solution to the PDP is a set $S\subseteq P(o,d)$ such that $|S|=p$.

Given a solution $S=\{P_1,P_2,\ldots,P_p\}$, we define its cost value $z_1(S)$ as the average of the costs of the paths in S:

$$z_1(S)=\frac{\sum_{t=1}^{p}c(P_t)}{p} \quad \text{where } c(P_t)=\sum_{(i,j)\in P_t}c_{ij}$$

We also define its dissimilarity value $z_2(S)$ as the average of the dissimilarity between the $\binom{p}{2}$ distinct pairs of paths in S:

$$z_2(S)=\frac{\sum_{i=1}^{p-1}\sum_{j=i+1}^{p}dis(P_i,P_j)}{\binom{p}{2}}$$

Note that to compute $z_2(S)$ we need to define an appropriate dissimilarity measure $dis(P_i,P_j)$. Given an origin o, a destination d and an integer number $p>1$, the PDP can be stated as:

$$\begin{array}{ll}(\text{PDP}) & \text{Minimize } z_1(S)\\ & \text{Maximize } z_2(S)\\ & \text{Subject to: } \ S\subseteq P(o,d)\\ & |S|=p\end{array}$$

A solution S^* of PDP is efficient if there is no other solution $S\subseteq P(o,d)$ such that $z_1(S^*)>z_1(S)$ and $z_2(S^*)<z_2(S)$. In other words, S^* is efficient if there is no solution in $P(o,d)$ better than S^* with respect to both objectives.

Most multi-objective programming techniques [1] focus on finding the set or frontier of efficient points (E) for a given problem or,

* Corresponding author. Tel.: +34 96 386 4362; fax: +34 96 386 4735.
E-mail addresses: Rafael.Marti@uv.es (R. Martí), Gonzalez.Velarde@itesm.mx (J. Luis González Velarde), Abraham.Duarte@urjc.es (A. Duarte).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.01.003

in the case of heuristic procedures, an approximation of the efficient frontier ($\hat{E}$). In this paper, we describe the development and testing of a metaheuristic procedure for the PDP as a bi-objective problem with the goal of finding an approximation to the efficient frontier. Although several heuristics have been applied to different versions of this problem, they usually provide the user with a good solution or a few good solutions (in terms of both objectives), but they do not study $\hat{E}$ with the standard measures in multi-objective optimization, such as $|\hat{E}|$, the size of the space covered or the dominance between efficient sets [2]. In this paper first we review (Section 2) the relevant procedures proposed for this and related problems and adapt them to the PDP as a bi-objective approach. In Section 3, we introduce a new dissimilarity function and compare it with previous measures. Finally, in Section 4 we propose a new heuristic based on the GRASP methodology. The paper finishes with a computational testing (Section 5) of the revised methods, as well as our new approach and the subsequent conclusions.

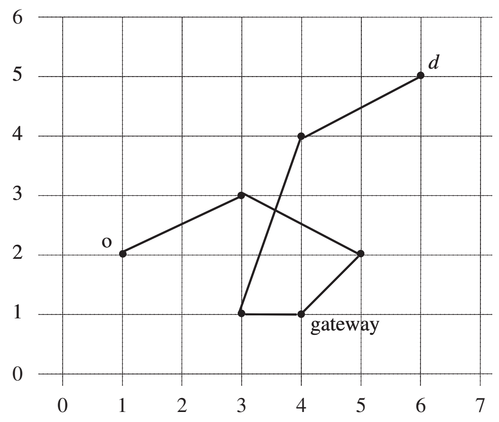

Fig. 1. Illustration of a loop in GSP method.

2. Previous methods

Studies on bi-criteria path problems can be traced back to the early eighties, namely to the work by Clímaco and Martins [3] and Martins [4]. They considered cost and time as the two criteria in conflict, and proposed truncated enumerative methods to obtain a few solutions in the efficient frontier $\hat{E}$. In this section we review previous approaches to the PDP and adapt them to generate multiple solutions for this problem (i.e. an approximation of the efficient frontier).

Johnson et al. [5] introduced the *iterative penalty method* (IPM) in which a shortest-path algorithm is iteratively applied. After each application of the method, the weight of the edges in the constructed path is penalized (adding a penalty factor β) to discourage their selection in future constructions. As a result this method gives rise to dissimilar paths that are relatively short in length. Low β values encourage short paths to be constructed. However, given that edges already used may still appear, the dissimilarity between the constructed paths may be low. Alternatively, high values of β discourage already selected edges, favoring dissimilarity at the expense of constructing longer paths. The main advantage of the method is that it only requires a shortest-path algorithm to generate paths. A drawback is that it relies heavily on the penalization parameter, for example, a small penalty may not achieve the goal of dissimilarity, while a large penalty may eliminate a great many viable paths from consideration.

Let IPM(β) be the IPM method with a particular value of β. To construct an approximation to the efficient frontier for the PDP, we propose systematically varying the penalty factor β. We run *max_iter* times IPM(β) with $\beta = iter/max\_iter$ where $iter = 1, \ldots, max\_iter$. Finally, we select the set of non-dominated solutions, $\hat{E}$, from the *max_iter* solutions (sets with p paths) generated by this method. It should be noted that this adaptation of IPM restricts the value of β to be equal or lower than 1 because we have empirically found that larger values provide low quality results (i.e. do not add new solutions to $\hat{E}$).

Lombard and Church [6] proposed the *gateway shortest path* (GSP) method, based on generating the shortest paths between an o–d pair computed to go through a specific node (called gateway). At each step, the GSP method first selects a gateway node, and then computes two paths, one from the origin to this node and another from this node to the destination. Linking both paths it obtains the final path from o to d. To evaluate the dissimilarity between two paths, the concept of "area under a path" is introduced. An advantage of the method is that a large number of alternative paths may be generated by simply using a shortest-path algorithm twice. A shortcoming is that the final paths may contain loops, as illustrated in Fig. 1, because the linked paths are computed independently. Possibly some good paths do not contain nodes proposed as gateways, thus they are not identified by this method.

In order to approximate the efficient frontier of the PDP with the GSP method, the gateway nodes are selected via a randomized procedure. We construct a candidate list of nodes, CL, with those not present in the shortest path from o to d. Nodes in CL are ordered according to their distance to the shortest path. We run GSP for *max_iter* iterations, randomly selecting the gateway nodes among the first nodes in CL (at iteration i we select among the first $p-1+i$ nodes).

Kuby et al. [7] proposed the *minimax method* whose aim is to produce a set of p differentiated paths by selecting a subset of a large set of paths aiming to achieve a final set that is mutually dissimilar and includes relatively short paths. The method starts by generating a set of *k-shortest* paths between o and d. From this set a dissimilar subset (DS) is constructed iteratively using a dissimilarity index. This measure is defined paying attention to the similarity between the selected paths and to their lengths. The first path included is the shortest path P_1, the second path to be selected is the one that minimizes the similarity index. The third path is selected to minimize the maximum (hence their method is called minimax) of the indices between the candidates and the first two paths. The procedure continues thus until the desired number of paths is reached. Our implementation of the minimax method to approximate the efficient frontier of the PDP, incorporates a parameter γ to weight the similarity index. This enables us to run the method *max_iter* times, obtaining different solutions (sets of p paths) to this problem.

Akgün et al. [8] noted that the minimax method is just a greedy algorithm for solving the p-dispersion problem, so they proposed an improvement. In the p-dispersion problem [9], given a set M ($|M| = m$), called the candidate set, p elements ($p < m$) must be selected so as to maximize the minimum distance between any two of the selected elements. The authors proposed employing a constructive semi-greedy algorithm with a local search method previously employed in the p-dispersion problem [10], to select a subset of p dissimilar paths from a set of *k-shortest* paths. The candidate set M may be constructed in several ways. In their study they proposed two different methods: k-shortest paths, and the IPM method described above. Given two paths P_i and P_j, their similarity $S(P_i, P_j)$ is measured as

$$S(P_i, P_j) = \frac{1}{2}\left(\frac{L(P_i \cap P_j)}{L(P_i)} + \frac{L(P_i \cap P_j)}{L(P_j)}\right)$$

where $L(P_i)$ denotes the length of path P_i. Then, the dissimilarity of two paths P_i and P_j, $D(P_i, P_j)$, is simply computed as $D(P_i, P_j) = 1-S(P_i, P_j)$. Computational experimentation showed the methods based on the *p*-dispersion problem to be superior. In our experimentation to approximate the efficient frontier of the PDP we ran the variant with the IPM method coupled with *p*-dispersion *max_iter* times and called it IPM_pD.

Dell'Olmo et al. [11] approached the problem from a multi-objective perspective. They pointed out that there are two major drawbacks in the two methods proposed in Akgün et al. [8]:

- they only consider one single criterion, edge length, in the first step of their method,
- the definition of dissimilarity only considers the edges, and the obtained routes may be spatially very similar to one another.

Although other previous approaches, as the minimax method, considered two criteria to construct the set of k initial paths, they added both in a single objective function. In the new approach, called *multicriteria shortest path algorithm* (MSPA), Dell'Olmo et al. computed a set ND of non-dominated paths according to the multi-objective concept with respect to both length and risk. Each arc has two associated values, length and risk. Given two paths P_i and P_j, let $l(P_i)$ and $l(P_j)$ be their lengths, and let $r(P_i)$ and $r(P_j)$ be their risks, respectively. It is said that P_i dominates P_j if $l(P_i) < l(P_j)$ and $r(P_i) < r(P_j)$. In the first step, the MSPA method computes a set of k-shortest paths (according to the length). At each iteration, it checks whether the shortest path is admitted to the non-dominated set ND or not. To do this, it is compared with the previous paths in ND. If none of them dominates the new one the latter is included; otherwise, it is discarded and it resorts to the next path generated with the k-shortest path algorithm. At the end of the first step it obtains the set ND with non-dominated paths with respect to both length and risk.

In the second step, MSPA applies the D2 method [12] for the *p*-dispersion problem, to select a subset of p dissimilar paths in ND. Dell'Olmo et al. replace the length $L(.)$ with the area $A(.)$ in the computation of the dissimilarity introduced in Akgün et al. [8]. This area is determined by defining a band of 150 m around each path (called the buffer zone). The D2 method is basically a destructive procedure that starts by considering a solution that contains all the k candidate paths and, at each step, eliminates the one with minimum dissimilarity until a set S with exact p paths is obtained. This method is coupled with a local search procedure that, at each iteration, tries to exchange an element in S with an element outside S until it is not possible to improve the objective function further.

Thyagarajan et al. [13] considered the problem of determining dissimilar paths for military aircraft spatially and temporally during mission ingress. The authors first showed how to generate a network for a region from terrain information, and then considered a heuristic algorithm to obtain the paths. The terrain information is available in the form of a digital elevation model (DEM), from which a network is generated observing the concept of "keep-out" zones (zones that must be avoided to decrease the chance of detection). The heuristic is a modification of the Akgün et al. [8] method to consider not only spatial but also temporal information. The authors show the performance of their method in a case study, which considering its geographical features they considered a challenging instance.

Carotenuto et al. [14] considered selecting p distinct simple *o–d* paths on a given network, so as to minimize the total path risk while satisfying a risk threshold constraint in the traversed links. The authors proposed a measure of risk based on the damage function introduced in Batta and Chiu [15]. They proposed two constructive algorithms: a greedy GD and a randomized greedy RGD.

The GD algorithm first constructs *k-shortest* paths according to the risk values. Let $P_1, P_2, \ldots, P_K$, be the ordered paths (where P_1 is the best one). The method examines these paths in order; if it can add a path P_t to the set of selected paths P^* satisfying the risk threshold constrained, it is selected and included in the set ($P^* = P^* \cup \{P_t\}$),

```
Construct $k$-shortest paths
Let $P_1, P_2, , , P_k$ , the ordered paths. CL = $\{P_1, P_2, , , P_k\}$
$P^* = \{P_1\}$
CL = CL \ $\{P_1\}$
While ($|P^*| < p$) {
        RCL = $\{ P_i \in$ CL | $dis(P_i , P^*) \geq dth\}$
        Randomly select $P_j$ from RCL
        $P^* = P^* \cup \{ P_j \}$
        CL = CL \ $\{ P_j \}$
}
Return $P^*$
```

Fig. 2. RGD method.

otherwise P_t is discarded. In *the RGD method*, instead of this greedy selection, a restricted candidate set RC is considered at each iteration with the "good" paths for selection (those satisfying the risk threshold constrained). The RGD randomly selects a path from RC and adds it to P^* ($P^* = P^* \cup \{P_j\}$). The authors evaluated the performance of both methods on a road network with 311 nodes and 441 arcs with p ranging from 2 to 8.

To approximate the efficient frontier of the PDP we adapt the RGD algorithm because the GD is a deterministic procedure that only obtains one single solution and we need a procedure that can generate many different solutions. Fig. 2 shows how we implement the RGD method first to construct *k-shortest* paths to set the candidate list of paths CL. Then, in each iteration, from the candidate list CL it randomly selects the path with a distance $dis(P_i,P^*)$ from already selected paths P^*, above a distance threshold dth, where the dis value between a path and a set of paths P^* is computed as

$$dis(P_i, P^*) = \sum_{P_j \in P^*} dis(P_i, P_j)$$

The threshold dth is computed as a percentage α between the maximum, $dmax$, and minimum, $dmin$, distances in CL.

$$dth = dmin + \alpha(dmax - dmin)$$

As in the previous methods, we run RGD *max_iter* times and select the non-dominated solutions to obtain the set $\hat{E}$.

3. Dissimilarity

We will assume that there is a distance function $\delta(u, v)$ for every point u, v in the domain of our problem. This function can be the Euclidean distance or any other having a triangle property. From this distance between two points, we define the distance from a vertex $v \in V$ to a path $P_1 = \{v_1, v_2, \ldots, v_n\}$ as

$$\delta(v, P_1) = \min_{v_j \in P_1} \delta(v, v_j)$$

We may now introduce the dissimilarity measure *dis* between the two paths $P_1 = \{v_1, v_2, \ldots, v_n\}$ and $P_2 = \{u_1, u_2, \ldots, u_m\}$, as follows:

$$dis(P_1, P_2) = \frac{1}{2}\left(\frac{\sum_{v_i \in P_1} \delta(v_i, P_2)}{|P_1|} + \frac{\sum_{u_j \in P_2} \delta(u_j, P_1)}{|P_2|}\right)$$

and we will use it to compute $z_2(S)$ for a solution S. Fig. 3 shows an example of two paths, P_1 and P_2, on a grid (P_1 is depicted with dashed lines) to illustrate the computation of our dissimilarity measure and to compare it with previous measures.

To compute $dis(P_1, P_2)$ in the example shown in Fig. 3, we first compute the Euclidean distance between each point in P_1 and its closest point in P_2. For example, $u_1 = (3,1) \in P_2$ is the closest point to $v_1 = (3,3) \in P_1$. Then, $\delta(v_1, P_2) = \delta(v_1, u_1) = 2$. In this way we obtain $\delta(v_2,P_2) = \delta(v_2,u_3) = 2.2$, $\delta(v_3, P_2) = \delta(v_3,u_4) = 0.0$,

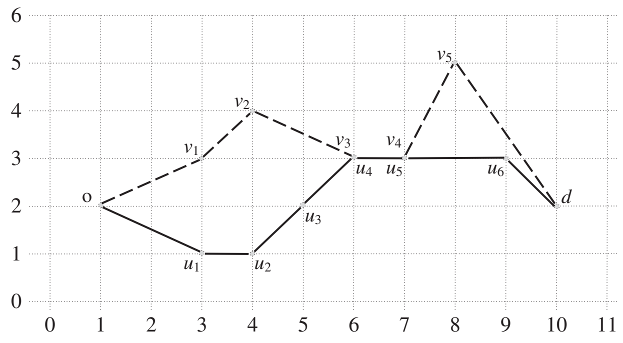

Fig. 3. Two paths in a grid.

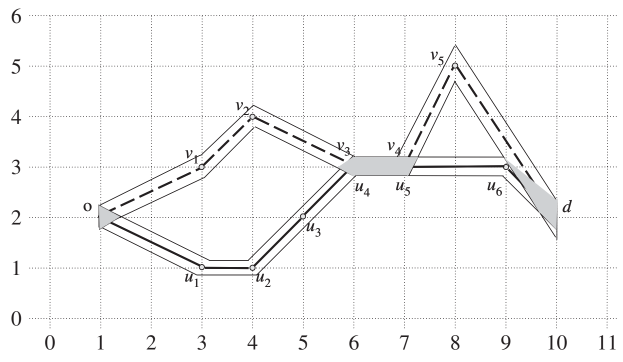

Fig. 4. Two paths with buffer zone in a grid.

$\delta(v_4,P_2) = \delta(v_4,u_5) = 0.0$ and $\delta(v_5,\ P_2) = \delta(v_5,\ u_6) = 2.2$. With the similar computations for the vertices in P_2 and their closest vertices in P_1, we obtain $dis(P_1,P_2) = 1.28$.

Akgün et al. [8] measured the dissimilarity in terms of the shared edges between paths, which as mentioned by other authors, do not consider spatial information ("the obtained routes may be spatially very similar to one another"). As reviewed in the previous section the expression to compute the dissimilarity between paths P_1 and P_2 is

$$D(P_1, P_2) = 1 - \frac{1}{2}\left(\frac{L(P_1 \cap P_2)}{L(P_1)} + \frac{L(P_1 \cap P_2)}{L(P_2)}\right)$$

In the example in Fig. 3, $P_1 \cap P_2 = \{(v_3,v_4)\} = \{(u_4,u_5)\}$, then $L(P_1 \cap P_2) = 1$. It is easy to compute $L(P_1) = 12.6$ and $L(P_2) = 10.4$. Then, $D(P_1,P_2) = 0.91$.

Dell'Olmo et al. [11] introduced the concept of *buffer zone* to include spatial information in the measure of dissimilarity. This measure is based on a parameter w (the width of the band around each path), that the authors set to 150 m. Let $A(P_1,w)$ be the area of the band around path P_1 with a width of w. Then, the dissimilarity between the paths P_1 and P_2, $D_{band}(P_1,P_2,w)$ is computed for a given value of w as

$$D_{band}(P_1, P_2, w) = 1 - \frac{1}{2}\left(\frac{A(P_1 \cap P_2, w)}{A(P_1, w)} + \frac{A(P_1 \cap P_2, w)}{A(P_2, w)}\right),$$

where the intersection of P_1 and P_2 is interpreted as the intersection of the two bands.

Fig. 4 shows the same example given in Fig. 3, but to which we have added the bands around each path (considering that the units in the grid are kilometers) and paint in grey the common area of both bands. Since $L(P_1) = 12.6$ and $w = 0.15$ we can approximate $A(P_1,0.15)$ as $12.6 \times 0.15 = 1.89$. Similarly we can compute $A(P_2,0.15) \approx 10.4 \times 0.15 = 1.56$. It is more difficult to approximate the area in the intersection of both bands accurately. For the sake of simplicity, we use a combination of triangles and rectangles and obtain $A(P_1 \cap P_2,0.15) \approx 0.435$. Therefore, $D_{band}(P_1,P_2,0.15) \approx 0.74$.

If we compare our dissimilarity measure $dis(P_1,P_2)$ with the measure of Agkün et al. $D(P_1,P_2)$, it is clear that their dissimilarity value does not reflect the distance between the vertices or edges between different paths, outside the common edges, while our measure does consider them. On the other hand, the measure introduced by Dell'Olmo et al. $D_{band}(P_1,P_2,w)$, despite incorporating spatial information, it strongly depends on parameter w, which in some applications could be artificial. Therefore we prefer to use the $dis(P_1,P_2)$ measure, based on the distance between the nodes in the paths, because it contains spatial information, does not include any extra parameter and is easy to compute.

Thyagarajan et al. [13] proposed an interesting extension of the dissimilarity measure. The authors consider the measure of Agkün et al. $D(P_1,P_2)$, but include the time context. In their application of mission detection involving aircraft, time differences between routes must be considered. (There is an inverse relationship between time and similarity). The authors modified $D(P_1,P_2)$ to introduce this factor.

4. A GRASP algorithm

The GRASP methodology was developed in the late 1980s [16], and the acronym was coined by Feo and Resende [17]. We refer the reader to the recent chapter [18] for a survey on this metaheuristic. Each GRASP iteration consists of constructing a trial solution and then applying a local search procedure to find a local optimum (i.e. the final solution for that iteration). The construction phase is iterative, randomized greedy, and adaptive. In this section we describe our adaptation of the GRASP methodology for the PDP.

Although Carotenuto et al. [14] do not mention it, their RGD algorithm uses the elements of a GRASP construction. Specifically, considering our adaptation of this method to the bi-objective problem shown in Fig. 4, we can see a candidate list CL of paths, and a Restricted Candidate List RCL of the most promising paths to be added to the partial solution under construction.

Our new GRASP algorithm for the PDP has the two standard phases: construction and improvement. In the construction phase we obtain a solution with the randomized destructive method RDE (see Fig. 5). In particular, we first generate a set with k initial paths and then extract p diverse paths using RDE. In the previous section we have already seen that some methods, such as the minimax or the RGD, compute the *k-shortest* paths and then select p diverse paths from this set. We found these approaches limited in the sense that the construction of these k paths does not consider the dissimilarity and only focuses on objective z_1. We therefore consider our initial set of $k = k_1 + k_2$ paths generated from two sources: we compute the k_1-shortest paths and then apply the IPM(β) to obtain k_2 paths. We select the IPM(β) method to obtain the k_2 paths because we found that it outperforms (in terms of finding diverse paths) the other methods described in Section 2 not based on the shortest-path algorithm. Our GRASP algorithm starts generating, by means of both methods, the set of $k = k_1 + k_2$ initial paths from which it then extracts p diverse paths using RDE.

In the improvement phase, we apply a local search method to the constructed solution, as customary in GRASP. This two-phase algorithm is replicated *max_iter* times and from the set of solutions obtained (sets of p paths), dominated solutions are discarded to obtain the approximation of the efficient frontier.

The RDE method first constructs a set K with the *k-shortest* paths. The partial solution P^*, which at this stage is unfeasible, is initialized as $P^* = K$. Then, in each iteration, it randomly removes a path with low contribution from P^* to the objective function z_2. Specifically, the restricted candidate list RCL is formed with the paths $P_i \in P^*$ with a

```
Construct $k=k_1+k_2$ initial paths
Let $P^* = \{P_1, P_2, \ldots, P_k\}$, the set of $k$ ordered paths.
While ($|P^*| > p$) {
        RCL = $\{ P_i \in P^* \mid dis(P_i, P^*) \le dth\}$
        Randomly select $P_j$ from RCL
        $P^* = P^* \setminus \{ P_j \}$
}
Return $S=P^*$
```

Fig. 5. RDE method.

distance value $dis(P_i, P^*)$ below a distance threshold dth. The threshold is computed as a percentage α between the maximum, $dmax$, and minimum, $dmin$, distances in CL, which is randomly selected at each iteration with diversification purposes.

$$dth = dmin + \alpha(dmax - dmin)$$

As in the RGD method, $dis(P_i, P^*)$ is computed with the expression:

$$dis(P_i, P^*) = \sum_{P_j \in P^*} dis(P_i, P_j)$$

When the number of paths in P^* equals p, the RDE method stops and returns the final P^* as the constructed solution S. It should be noted that the first step of our method constructs the k-shortest paths, thus favoring objective function z_1. This is why in algorithm RDE we consider the distance criteria to favor objective function z_2.

The second phase of our solving method is a local search procedure. Given a solution $S=\{P_1,P_2,\ldots,P_p\}$, with values $z_1(S)$ and $z_2(S)$ we apply an exchange procedure to improve it. The method alternates two stages for *g_iter* global iterations; in the first one it tries to improve (to reduce) the value of z_1 regardless the value of z_2 for a maximum of *l_iter* local iterations. In the second one it tries to improve (to increase) z_2 regardless the value of z_1 for a maximum of *l_iter* local iterations. If the method is unable to improve the corresponding objective (z_1 or z_2, respectively) at either stage it stops and resorts to the other stage. Both stages are alternated until no further improvement is possible or the maximum number *g_iter* of global iterations is reached.

In the first stage the p paths are examined in order, according to their contribution to z_1, where the path P_i with largest $c(P_i)$ value is examined first. We try to exchange path P_i with a new path $Q \in K\setminus S$ obtaining a new solution $S' = S\setminus\{P_i\}\cup\{Q\}$ with $z_1(S') < z_1(S)$. We perform the first improving exchange, replace S with S' and resort to the path P_j in the ordered list of paths. The first stage stops when no more exchanges are possible in any of the paths in the current solution or the number *l_iter* of maximum local iterations is reached.

In the second stage of the improvement method the p paths in the current solution S' (obtained by applying the first stage) are examined in the order given by their contribution to z_2, where the path P_i with lowest $dis(P_i,S')$ value is examined first. We try to exchange path P_i with a new path $Q \in K\setminus S$ obtaining a new solution $S'' = S'\setminus\{P_i\}\cup\{Q\}$ with $z_2(S'') > z_2(S')$. We perform the first improving exchange, replace S' with S'' and resort to next the path P_j in the ordered list of paths. The second stage stops when no more exchanges are possible in any of the paths in the current solution or the number *l_iter* of maximum local iterations is reached. Then we apply the first stage again to the output solution of this second stage and continue in this way. The improvement method stops after *g_iter* global iterations (application of both stages).

As in the previous methods, we run the GRASP algorithm *max_iter* times and select the non-dominated solutions to obtain the set $\hat{E}$. At the beginning of the method, we initialize $\hat{E} = \emptyset$, then after each construction, we check whether the constructed solution S is dominated or not with respect to the solutions already in $\hat{E}$, and if it

```
1. Set max_iter equal to the number of global iterations.
2. $\hat{E}=\emptyset$.
3. $iter=1$.

While( $iter \le max\_iter$ )
    4. Apply the RDE constructive method $\Rightarrow S$.
    5. Check the inclusion of $S$ in $\hat{E}$.
    6. $iter2=1$.
    While ($iter2 \le g\_iter$ )
            7. $iter3=1$.
            While($iter3 \le l\_iter$ ).
                8. Perform the first improving move w.r.t. $z_1 \Rightarrow S'$
                9. Check the inclusion of $S'$ in $\hat{E}$
                10. $iter3 = iter3+1$
            11. iter4=1
            While($iter4 \le l\_iter$ ).
                12. Perform the first improving move w.r.t. $z_2 \Rightarrow S''$
                14. Check the inclusion of $S''$ in $\hat{E}$
                14. $iter4 = iter4+1$
            15. $iter2 = iter2+1$
    16. $iter = iter +1$
```

Fig. 6. GRASP algorithm.

is efficient we include it. Moreover, after each movement of the local search we check whether the solution obtained qualifies to enter $\hat{E}$ or not (a non-dominated solution). Fig. 6 shows a pseudo-code of the GRASP algorithm.

5. Computational experiments

Our experimentation has two main goals. The first one is to compare between existing procedures adapted to the bi-objective PDP, and the second one is to show that the proposed GRASP procedure is able to create better approximations of the efficient frontiers than adaptation of existing methods. Specifically, we are interested in comparing the performance of IPM, GSP, minimax, IPM coupled with p-dispersion (IPM_pD), MSPA, RGD and our GRASP procedure. We have implemented these seven methods in C and all the experiments were conducted on a Pentium 4 computer at 3 GHz with 3 GB of RAM. The performance measures that we employ are standard in multi-objective optimization:

1. *Number of points*: This refers to the ability of finding efficient points. The decision-maker is assumed to prefer a larger number of efficient points.
2. *SSC*: This metric suggested by Zitzler and Thiele [2] measures the size of the space covered (SSC). In other words, SSC measures the volume of the dominated points. Hence, the larger the SSC value the better.
3. *k-distance*: This density estimation technique used by [19] in connection with the computational testing of SPEA2 is based on [20] kth nearest neighbor method. The metric is simply the distance to the kth nearest efficient points. We use $k=5$ and calculate both the mean and the max of k-distance values. The k-distance measure is such that a smaller value denotes a better approximation in terms of frontier density.
4. $C(A,B)$: This is known as the coverage of two sets measure [2]. $C(A,B)$ represents the proportion of points in the estimated efficient frontier B that are dominated by the efficient points in the estimated frontier A.

The test problems in our experimentation are taken from the well known TSP Library and are available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. In order to make

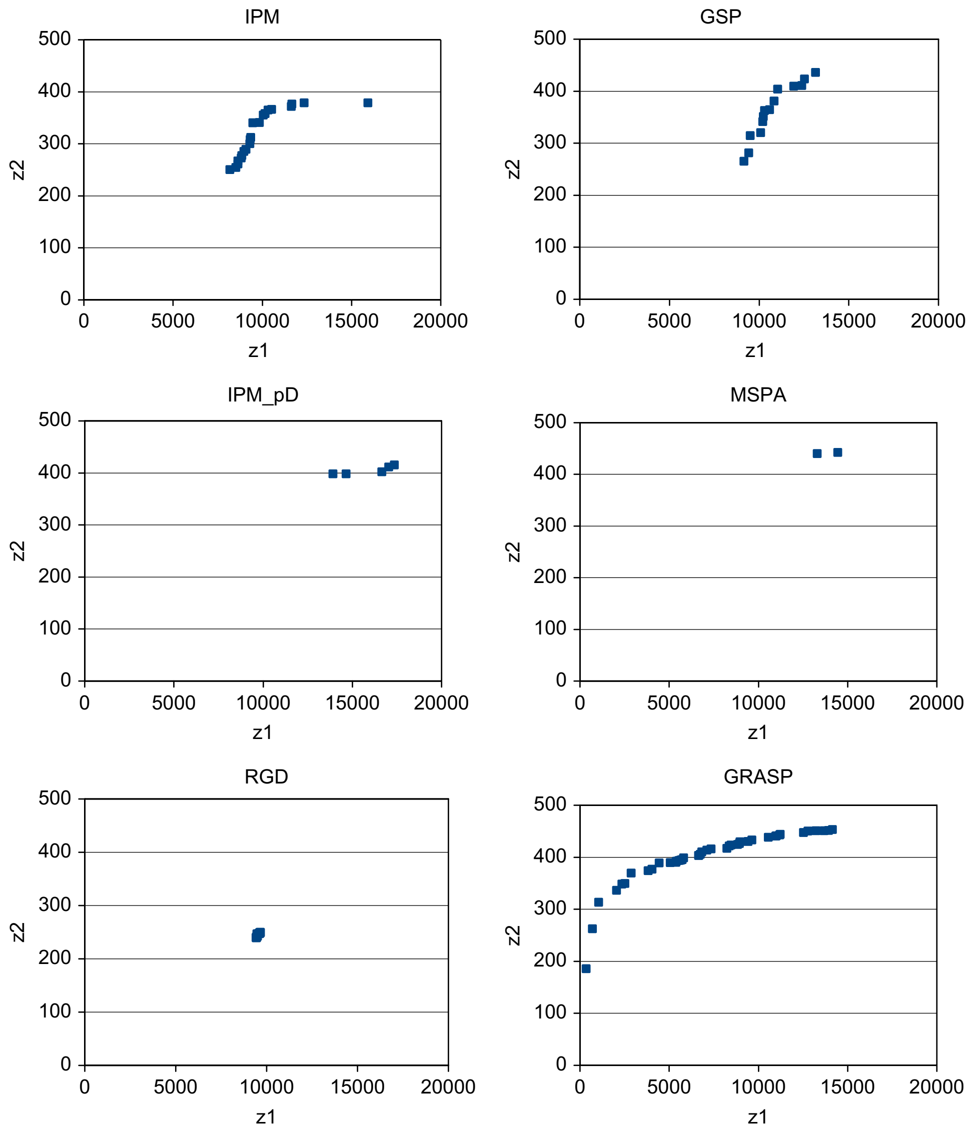

Fig. 7. Approximation of the efficient frontier for att532 and $p = 10$.

them harder to solve as PDPs, we remove most of the edges in the original instances and only include those edges with a cost (distance value) lower than 10% of the maximum distance value in each instance. With this value of 10%, the final instance is sparse and connected (in the cases considered in our experimentation). Since previous studies in this problem consider instances with approximately 300 vertices, we select the following ten instances in the TSPLIB with approximately 500 vertices: ali535, att532, d493, d657, fl417, gr666, gr431, rat575, u574 and pcb442. We remove the large edges in all of them, as described above, and select the farthest points as the origin and destination. The resulting instances are available at www.uv.es/rmarti.

As in previous studies we employ Yen's algorithm [14] to compute the initial *k-shortest* paths in all the tested methods. We set $k = 1000$ in the minimax, IPM(β)_pD, MSPA and $k_1 = k_2 = 1000$ in our GRASP algorithm. After preliminary experimentation we also set *maxTrials* $= 50$ in the IPM method and *g_iter* $= 10$ and *l_iter* $= 5$ in GRASP. In our first experiment we represent the approximation of the efficient frontier obtained with five of the six adapted algorithms and with our GRASP method, all run with *max_iter* set to 1000. We do not depict the results obtained with the minimax method because it only produces a single point in the efficient frontier. Fig. 7 shows the results of these six methods on the att532 instance with $p = 10$.

Fig. 7 shows that the GRASP method obtains a better approximation of the efficient frontier of instance att532 than the other competing methods. We can see in these diagrams that GRASP obtains a larger number of points with better distribution in the efficient frontier than the other algorithms (note that good solutions have a low value of z_1 and a large value of z_2).

Fig. 8 shows together the results of three of these methods, IPM_pD MSPA and GRASP, on the gr431 instance with $p = 15$. In line with the results on Fig. 7, we can see that GRASP obtains a significantly larger number of solutions than the other two methods. Moreover, the 56 GRASP solutions clearly dominate the 6 IPM_pD solutions, for which $z_1 > 600$ and $z_2 < 25$. Only one of the 14 MSPA solutions dominates the GRASP solutions.

In our second experiment we compare the solutions found with each of the six methods that we have implemented (as in the previous experiment we do not include the minimax method in the comparison since it only obtains a single point). Table 1 shows the values of the measurements described above. Specifically, it shows the *number of points*, the mean and maximum of the *k-distance* and the *SSC*. We have added a "CPU Time" column to show the CPU

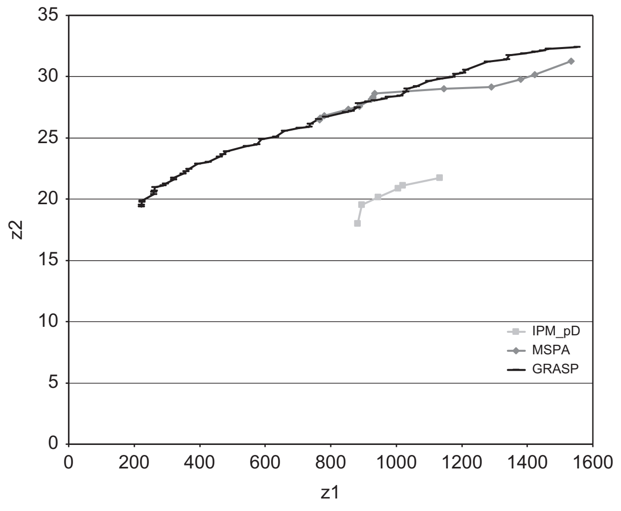

Fig. 8. Approximation of the efficient frontier for gr431 with $p = 15$.

Table 1
Performance measures with $p = 5$.

Methods	No. of points	k-distance (mean)	k-distance (max)	SSC	CPU time
IPM	29.30	0.09	0.29	0.58	407.2
GSP	21.10	0.08	0.20	0.61	130.3
IPM_pD	11.70	0.07*	0.14*	0.60	480.4
MSPA	26.90	0.09*	0.20*	0.61	348.5
RGD	9.30	0.10*	0.20*	0.43	63.5
GRASP	35.20	0.09	0.24	0.84	294.7

*Not available in the 10 instances.

Table 2
Performance measures with $p = 10$.

Methods	No. of points	k-distance (mean)	k-distance (max)	SSC	CPU time
IPM	26.70	0.07	0.25	0.47	414.1
GSP	20.10	0.06	0.16	0.48	264.3
IPM_pD	11.00	0.06*	0.19*	0.48	469.8
MSPA	27.91	0.11*	0.18*	0.47	343.5
RGD	5.60	0.02*	0.13*	0.45	67.1
GRASP	56.50	0.06	0.27	0.85	307.8

*Not available in the 10 instances.

Table 3
Performance measures with $p = 15$.

Methods	No. of points	k-distance (mean)	k-distance (max)	SSC	CPU time
IPM	28.90	0.08	0.24	0.42	485.3
GSP	20.00	0.06	0.16	0.45	441.0
IPM_pD	13.50	0.08*	0.17*	0.44	476.4
MSPA	30.01	0.11*	0.17*	0.42	323.4
RGD	6.10	0.01*	0.13*	0.42	78.1
GRASP	62.40	0.03	0.21	0.85	499.5

*Not available in the 10 instances.

seconds associated with each procedure. The statistics are calculated over the 10 TSPLIB instances summarized above with $p = 5$. Tables 2 and 3 report the same information with $p = 10$ and 15, respectively.

The results in Tables 1, 2 and 3 indicate that GRASP is capable of finding efficient frontiers with a large number of points and high density, as indicated by the small k-distance values. Some k-distance values in these tables are followed with the "*" symbol indicating that the associated method is not able to produce five points or more in the associated frontier. Specifically, IPM_pD MSPA and RGD obtain more than five points in the efficient frontier in 20, 26 and 18 out of the 30 instances solved, respectively. We therefore compute the average and max of the k-distance values considering only those cases. The SSC values clearly show GRASP outperforms the competing approaches, since it is significantly larger than the others in all three tables. CPU times are similar for all the methods (around 6 min) with the exception of the RGD method, which is much faster than the others (although it obtains a reduced number of points in the efficient frontier).

Table 4
Coverage of two sets.

C(A/B)	IPM	GSP	IPM_pD	MSPA	RGD	GRASP
IPM	0.01	0.34	0.23	0.02	0.47	0.00
GSP	0.44	0.06	0.46	0.13	0.58	0.01
IPM_pD	0.20	0.05	0.00	0.00	0.03	0.00
MSPA	0.34	0.14	0.75	0.00	0.00	0.04
RGD	0.04	0.06	0.03	0.00	0.00	0.00
GRASP	0.96	0.94	0.95	0.83	0.84	0.00

In our third experiment we compare the $C(A,B)$ measure over the entire set of test problems. This measure allows us to make a comparison according to the dominance of one efficient frontier over another. Table 4 shows the average $C(A,B)$ values over the 30 instances (10 with $p = 5$, 10 with $p = 10$ and 10 with $p = 15$).

The values in Table 4 show that the frontier generated by GRASP dominates those generated by the other methods. For example, $C(\text{GRASP,GSP}) = 0.94$ indicates that 94% of the points in the frontier obtained with GSP are dominated by points in the frontier obtained with GRASP. We compare this value with $C(\text{GSP,GRASP}) = 0.01$, which indicates that only the 1% of the points in the frontier obtained with GRASP are dominated by points in the frontier obtained with GSP. In all cases we can see that $C(\text{GRASP,-}) > C(\text{-,GRASP})$, assessing the superiority of GRASP in terms of dominance.

In our last experiment we consider an application of the PDP to the transportation of hazardous chemical materials in Spain. According to Spanish law, these materials pose a risk of mass-disaster and are highly toxic and contaminant. Therefore, any company wishing to transport these types of hazardous substances must obtain a transport approval, which involves a routing plan. A large number of routing alternatives is highly recommended to obtain it.

In this case study we consider the particular situation of a chemical company with a plant in Catalonia (North–East of Spain) that has to deliver chemical material (alkyl series) to their customer in Huelva (South–West of Spain). In their monthly planning they need five routes from their origin (plant) to their destination (customer). They annually present an application for transport approval with different monthly plans. Therefore, they need to solve the PDP with $p = 5$. We apply our GRASP algorithm to this problem and obtain seven different solutions (note that each solution is a set with five origin–destination paths). Fig. 9 shows, in two maps for the sake of clarity, the five routes of one of the seven solutions obtained. It has $z_1 = 2777.64$ and $z_2 = 50{,}704.68$ km.

If we solve this problem with two of the best methods mentioned here, which are IPM_pD and MSPA according to our previous experiments, we obtain worse solutions than with the GRASP algorithm. Specifically, IPM obtains three solutions and MSPA five solutions. Moreover, the solutions obtained with our method dominate the solutions obtained with these other two methods, as shown in Fig. 10. In short, GRASP provides the company with more and better alternatives for transportation than the previous methods.

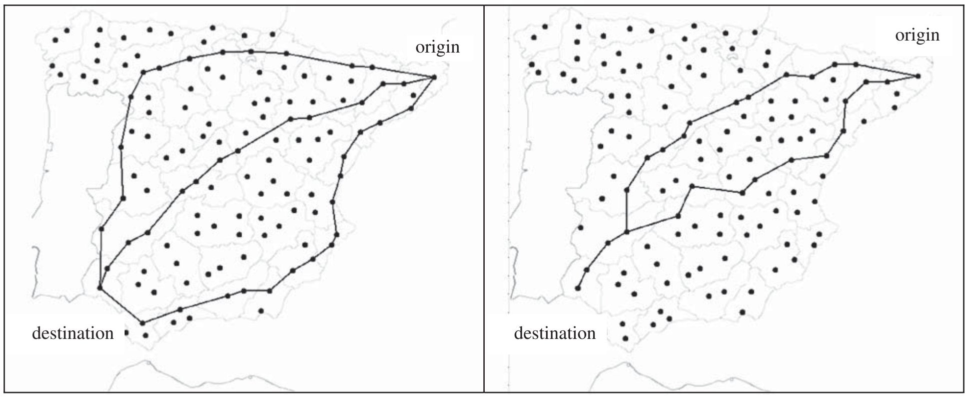

Fig. 9. One solution (five routes) obtained with GRASP in the case study.

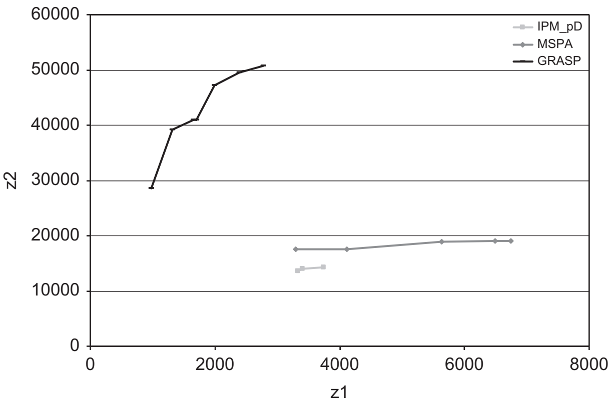

Fig. 10. Approximation of the efficient frontier for the case study.

6. Conclusions

Herein we have formally stated the bi-objective path dissimilarity problem (PDP), and described the development and implementation of a GRASP procedure to resolve it. The final design is then compared to state-of-the-art methods and the outcome of our experiments seems quite conclusive in conferring merit to the procedure we propose. Our first contribution is to adapt previous methods originally designed for different versions of this problem to the bi-objective variant we are considering. Our second contribution has been to propose a new algorithm based on the GRASP methodology. To the best of our knowledge, our work is the first to test several procedures for the PDP within the multi-objective programming framework and its associated evaluation measures based on the efficient frontier.

The definition of the dissimilarity is still an open question. For this reason, as pointed out by an anonymous referee, we could study an extension and/or generalization of the measure we are proposing in this paper. For instance, the path between the origin and the destination can be parameterized and then integrate the distance from a point on the path to another path over the length. We could introduce artificial vertices in lengthy edges to implement this extension. This measure, however, involves an extra computational effort that could compromise its applicability.

Acknowledgments

This research has been partially supported by the *Ministerio de Educación y Ciencia* of Spain (Grant Ref. TIN2006-02696), by the Comunidad de Madrid—Universidad Rey Juan Carlos project (CCG08-URJC/TIC-3731) and by Tecnológico de Monterrey Research Fund CAT128 and CONACYT 61903.

The authors wish to thank Professor Julian Molina for his help in computing the multi-objective performance measures. The authors also wish to thank Professors Thyagarajan, Batta, Karwan and Szczerba for provided us with the details of their case study.

References

[1] Zitzler E. Evolutionary algorithms for multiobjective optimization: methods and applications. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland; 1999.
[2] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 1999;3(4):257–71.
[3] Clímaco JC, Martins EQV. A bicriterion shortest path algorithm. European Journal of Operational Research 1982;11:399–404.
[4] Martins EQV. On a multicriteria shortest path problem. European Journal of Operational Research 1984;16:236–45.
[5] Johnson PE, Joy DS, Clarke DB. HIGHWAY 3.01, an enhancement routing model: program, description, methodology and revised user's manual. Technical Report, Oak Ridge National Laboratories; 1992.
[6] Lombard K, Church RL. The gateway shortest path problem: generation of alternative routes for a corridor location problem. Geographical Systems 1993;1:25–45.
[7] Kuby M, Zhongyi X, Xiaodong X. A minimax method for finding the *k* best differentiated paths. Geographical Analysis 1997;29(4):298–313.
[8] Akgün V, Erkut E, Batta R. On finding dissimilar paths. European Journal of Operational Research 2000;121:232–46.
[9] Duarte A, Martí R. Tabu search and GRASP for the maximum diversity problem. European Journal of Operational Research 2007;178(11):71–84.
[10] Erkut E. The discrete *p*-dispersion problem. European Journal of Operational Research 1990;46:48–60.
[11] Dell'Olmo P, Gentili M, Scozzari A. On finding dissimilar Pareto-optimal paths. European Journal of Operational Research 2005;162:70–82.
[12] Glover F, Kuo CC, Dhir KS. Heuristic algorithms for the maximum diversity problem. Journal of Information and Optimization Sciences 1998;19(1):109–32.
[13] Thyagarajan K, Batta R, Karwan MH, Szczerba RJ. Planning dissimilar paths for military units. Military Operations Research 2005;10(1):25–42.
[14] Carotenuto P, Giordani S, Ricciardelli S. Finding minimum and equitable risk routes for hazmat shipments. Computers & Operations Research 2007;34: 1304–27.
[15] Batta R, Chiu SS. Optimal obnoxious paths on a network: transportation of hazardous materials. Operations Research 1988;36:84–92.
[16] Feo TA, Resende MGC. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 1989;8:67–71.
[17] Feo TA, Resende MGC. Greedy randomized adaptive search procedures. Journal of Global Optimization 1995;6:109–33.
[18] Resende MGC, Ribeiro CC. Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G, editors. State-of-the-art handbook in metaheuristics. Boston: Kluwer Academic Publishers; 2003. p. 219–50.
[19] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland; 2001.
[20] Silverman BW. Density estimation for statistics and data analysis. London: Chapman & Hall; 1986.