Heuristics for the Bi-Objective Path Dissimilarity Problem

RAFAEL MARTÍ
Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain.
Rafael.Marti@uv.es

JOSÉ LUIS GONZÁLEZ VELARDE
Centro de Manufactura y Calidad, Tecnológico de Monterrey, México
Gonzalez.Velarde@itesm.mx

ABRAHAM DUARTE
Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Spain.
Abraham.Duarte@urjc.es

Version: April 24, 2008

Abstract

In this paper the Path Dissimilarity Problem is considered. The problem has been previously studied in several contexts, the most popular motivated by the need of selecting routes for transportation of 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 side the average length of the paths that must be kept low, and on the other side the dissimilarity among the paths in the set, that should be kept high. Previous methods are reviewed and adapted to our bi-objective problem, in this way we are able to compare the methods using the standard measures in multi-objective optimization. A new GRASP procedure is proposed and tested among the revised methods, and we show that it is able of creating better approximations of the efficient frontiers than existing methods.

Key words: metaheuristics, bi-objective programming, routing

1. Introduction

In this paper we consider a multi-objective routing problem in which it is necessary to generate a set of paths from an origin to a destination. Finding different paths in a graph is a classical optimization problem. The best known is the *k*-shortest path problem in which *k* different 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 a different approach. For example, in the context of hazmat transportation we want to obtain not only different but also very dissimilar paths that minimize the risk (distributing uniformly the risk over all zones of the crossed regions). Specifically, we consider here the Path Dissimilarity Problem, which consists of obtaining a set of *p* paths with minimum length and maximum dissimilarity.

Given a graph $G=(V, E)$ with V the set of vertices and E the set of edges with associated cost c_{ij} for $(i,j)\in E$, and a pair of vertices origin-destination, *o*-*d*, we define $P(o,d)$ as the set of all paths in G from o to d. Given an integer number $p>1$, a solution to the Path Dissimilarity Problem (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:

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

We also define its dissimilarity value $z_2(S)$ as the average of the dissimilarity between the $\binom{p}{2}$ different 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 a dissimilarity measure $dis(P_i,P_j)$. Given an origin *o*, a destination *d* and an integer number $p>1$, the Path Dissimilarity Problem can be stated as:

(PDP) Minimize $z_1(S)$
Maximize $z_2(S)$

Subject to: $S\subseteq P(o,d)$
$|S|=p$

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, if there is no solution in $P(o,d)$ better than S^* with respect to both objectives.

Most multi-objective programming techniques focus on finding the set or frontier of efficient points (E) for a given problem or, 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. 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 (Zitzler and Thiele, 1999). In this paper we first review (Section 2) the relevant procedures proposed for this and related problems, then we adapt them to the Path Dissimilarity Problem as a bi-objective approach (Section 3) and finally, we propose in Section 4 a new heuristic based on the GRASP methodology. The paper finishes with a computational testing among the revised methods as well as our new approach (Section 5) and the associated conclusions.

2. Previous Methods

Studies on bi-criteria path problems can be traced back to the early eighties due to Clímaco and Martins (1982) and Martins (1984). 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 $\hat{E}$.

Johnson et al. (1992) 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 to discourage their selection in future constructions. As a result dissimilar paths with relatively short length are obtained with this method. Advantage of the method is that it only requires a shortest path algorithm to generate paths. Shortcoming: it relies heavily on the penalization parameter.

Lombard and Church (1993) proposed the Gateway Shortest Path (GSP) method, based on the generation of 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. Advantage of the method is that a large number of alternative paths may be generated by simply using a shortest path algorithm twice. Shortcomings: the paths may contain loops, and may be impossible to identify some desirable dissimilar paths as GSPs.

Kuby et al. (1997) reformulated the problem using path variables, requiring the selection of a number of alternative paths between each *o* - *d* pair. The task reduces to generate a

small number of paths with acceptable lengths which have as few common links as possible. They also proposed the Minimax method. The idea is to generate a set of p differentiated paths by selecting a subset from a large set of k paths. Attention is paid to similarity between the selected paths and to their lengths so that the final set is mutually dissimilar and includes relatively short paths. The method performs two steps. In the first one it generates the k shortest paths. In the second it iteratively selects paths minimizing a linear combination of two objectives, path length and similarity with already selected paths.

Akgün et al. (2000) improved the Minimax method with the introduction of the p-dispersion problem. In the p-dispersion problem, p out of m given elements ($p<m$) are selected to maximize the minimum distance between any two of the selected elements. The authors proposed to employ a constructive with a local search method previously introduced for the p-dispersion problem (Erkut 1990), to select a subset of p dissimilar paths from a set of k shortest paths. Moreover they explore a variant in which the initial set with k paths is constructed with 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 $D(P_i, P_j)=1- S(P_i, P_j)$.

The two new methods as well as the three previous ones IPM, GSP and Minimax are compared when solving eleven instances (o-d pairs) from a road network with 305 nodes and 854 arcs with p ranging from 3 to 25. Solutions are evaluated in the comparison according to the following three measures:

AvLe: the average path length
AvDi: the average dissimilarity between paths
MiDi: the minimum dissimilarity

Note that the first two measures correspond to z_1 and z_2 respectively, in the definition of the PDP given in Section 1. The computational comparison indicates the superiority of the methods based on the p-dispersion problem.

Dell'Olmo et al. (2005) approached the problem from a multi-objective perspective. They pointed out that there are two major drawbacks in the method proposed in Akgün et al. (2000):

- 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, this last 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 method D2 (Glover et al. 1998) for the p-dispersion problem, to select in ND a subset of p dissimilar paths. Dell'Olmo et al. replace the length $L(.)$ with the area $A(.)$ in the computation of the dissimilarity introduced in Akgün et al. (2000). This area is determined by defining a band of 150 m around each path (that is called the Buffer Zone). The method D2 is basically a destructive procedure that starts by considering a solution which contains all the k candidate paths and, at each step, eliminates the one with minimum dissimilarity until a set with exactly p paths is obtained. This method is coupled with a local search procedure based on exchanges.

Carotenuto et al. (2007) considered to select 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 (1988). 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,…, P_K , be the ordered paths (where P_1 is the best one). The method examines in order these paths; 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 ($P^*=P^*\cup\{P_t\}$), 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 evaluate the performance of both methods on a road network with 311 nodes and 441 arcs with p ranging from 2 to 8.

3. Adaptation of Previous Methods

In this section we first specify the particular problem we are solving, including the dissimilarity definition, and we then adapt the main methods described in the previous section to our bi-objective problem.

We will assume that on the graph $G = (V, E)$ there exists a distance function $\delta(u, v)$ for every $u, v \in V$. This distance is assumed to have the triangle property. Analogously, we can define the distance from a vertex v to a path $P_1 = \{o, v_1, v_2, \ldots, v_n, d\}$ as:

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

With these definitions, we may now introduce the dissimilarity measure *dis* between the two paths P_1 and $P_2 = \{o, u_1, u_2, \ldots, u_m, d\}$, 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. It should be noted that this dissimilarity measure takes into account spatial information without introducing any arbitrary parameter. Akgün et al. (2000) measured the dissimilarity in terms of the shared edges between paths, which as mentioned by other authors do not considered spatial information ("the obtained routes may be spatially very similar to one another"). On the other hand, Dell'Olmo et al. (2005) introduced the concept of Buffer Zone to measure the dissimilarity including spatial information; however, this measure is based on a parameter (the width of the band around each path) set to 150 m. We prefer to use the distance between the nodes in the paths, as described above, because it is based on spatial information and does not include any extra parameter.

3.1 Iterative Penalty Method (IPM)

The way to construct the approximation to the efficient frontier for our specific problem is based on varying systematically the penalty factor β that is applied to the already selected edges. We have experimentally found that in some instances (with large differences between cost edges) the method can select the same path in consecutive iterations. In this case, we penalize the edges in the path repeatedly for a maximum number of *maxTrials* iterations.

Low penalties will encourage the construction of short paths. However, given that edges already used may still appear, the dissimilarity of the set of constructed paths may be low. High penalties on the contrary will discourage edges already selected favouring dissimilarity at the expense of constructing longer paths. Let IPM(β) be the IPM method with a particular value of β. We run *max_iter* times IPM(β) with $\beta = iter/max\_iter$ where $iter = 1, \ldots, max\_iter$. We finally select the set of non-dominated solutions, $\hat{E}$, from the *max_iter* solutions (sets with p paths) generated with this method.

3.2 Gateway Shortest Path (GSP)

To construct the paths using GSP, the selection of the gateway nodes is based on a randomized procedure. Initially, the shortest path is constructed, once it has been

obtained, the distance form every node in V, that is not included in the path is computed. These nodes are then sorted increasingly according to this distance in a candidate list (CL). To construct the first solution S_1 the first p-1 vertices in CL are selected as gateway nodes. This solution favours the paths with shortest lengths, but dissimilarity will be low. For the i-th construction the gateways nodes are selected randomly among the first p-1+ i nodes in the CL, obtaining solution S_i. As i increases the constructions will favor the dissimilarity at the expense of constructing longer paths. We run GSP for *max_iter* iterations, and, as in the previous method, we then select the non-dominated solutions, obtaining the approximation of the efficient frontier $\hat{E}$.

3.3 Minimax

In each run, the Minimax method iteratively selects paths minimizing a linear combination of the objectives z_1 and z_2 with the already selected paths. As in the original method by Kuby et al. (1997), it performs two steps. In the first one it generates the k shortest paths. In the second it iteratively selects paths from this set of k paths minimizing $\gamma z_1+(1-\gamma)z_2$ where γ $(0\leq\gamma\leq1)$ is the weight in this linear combination. We then run *max_iter* times Minimax(γ) with γ=*iter*/*max_iter* where *iter*=1, .., *max_iter* and select the set $\hat{E}$, from the *max_iter* solutions constructed with this method.

3.4 Iterative Penalty Method + p-Dispersion (IPM(β)_pD)

The two phase method IPM(β)_pD adapts the algorithm proposed in Akgün et al. (2000) to the bi-objective problem we are considering. In the first phase we run the IPM(β) method to obtain k paths. We select in the second phase p of them with the constructive plus the local search methods proposed in Erkut (1990) for the p-dispersion problem. We run IPM(β)_pD *max_iter* times with β= *iter*/*max_iter* where *iter*=1, …, *max_iter*, obtaining as the output of the method the set $\hat{E}$ with the non-dominated solutions.

3.5 Multicriteria Shortest Path Algorithm (MSPA)

Our adaptation of the MSPA method proposed in Dell'Olmo et al. (2005) has two phases. In the first phase we run the IPM(β) method to obtain k paths. We then select in the second phase p of them with the semi-greedy heuristic for the p-dispersion problem (which is similar to the D2 method proposed in Glover et al. (1998) plus a local search based on exchanges, D2_LS). Let MSPA(β) be the IPM(β) method with a particular value of β coupled with the D2_LS. As in the previous methods, we run MSPA(β) *max_iter* times with β= *iter*/*max_iter* where *iter*=1,…, *max_iter*, and select the non-dominated solutions to obtain the set $\hat{E}$.

3.6 Randomized Greedy Algorithm (RGD)

We adapt now the method due to Carotenuto et al (2007) to our bi-objective problem. We consider the Randomized Greedy Algorithm because their Totally Greedy Algorithm is a deterministic procedure that only obtains one single solution. Since we want to obtain an approximation of E, we need a procedure that is able to generate many different solutions.

Figure 1 shows that our implementation of the RGD method first constructs k shortest paths to set the candidate list of paths CL. Then, in each iteration, it randomly selects from the candidate list CL 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 $\hat{E}$.

```
Construct k-shortest paths
Let P1, P2, , , Pk , the ordered paths. CL = {P1, P2, , , Pk}
P* = {P1}
CL = CL \ {P1}
While (|P*| < p) {
        RCL = { Pi ∈ CL | dis(Pi , P*) ≥ dth}
        Randomly select Pj from RCL
        P* = P* ∪ { Pj }
        CL = CL \ { Pj }
}
Return P*
```

Figure 1. RGD method

4. A GRASP Algorithm

The GRASP methodology was developed in the late 1980s (Feo and Resende 1989), and the acronym was coined by Feo and Resende (1995). We refer the reader to the recent chapter (Resende and Ribeiro 2003) 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 Path Dissimilarity Problem.

Although not mentioned in Carotenuto et al (2007), their RGD algorithm uses the elements of a GRASP construction. Specifically, considering our adaptation of this method to the bi-objective problem shown in Figure 1, 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 Figure 2). In particular, we first generate a set with k initial paths and then extract p diverse paths using RDE. We have seen in the previous section 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 consider IPM(β) after preliminary experimentation among the methods in the previous section not based on the shortest path algorithm, 10 executions of this method with $\beta = 1, 10, 20, \ldots, 100$, respectively, is able to generate paths more diverse than the other methods. 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 it is 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.

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 = $\{ P_i \in P^* \mid dis(P_i, P^*) \leq dth\}$
    Randomly select P_j from RCL
    $P^* = P^* \setminus \{ P_j \}$
}
Return $S=P^*$

Figure 2. RDE method

The RDE method first constructs a set K with the k shortest paths. The partial solution P^*, which at this stage is infeasible, is initialized as $P^*=K$. Then, in each iteration, it randomly removes from P^* a path with low contribution to the objective function z_2. Specifically, the restricted candidate list RCL is formed with the paths $P_i \in P^*$ with a 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 favouring objective function z_1. This is why in algorithm RDE we consider the distance criteria to favour 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. In any of the two stages if the method is not able to improve the corresponding objective (z_1 or z_2 respectively) it stops and resorts to the other stage. Both stages are alternated until no longer 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 next 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 with the application of 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. We then apply again the first stage to the output solution of this second stage and continue in this way. The improvement method stops after *g_iter* global iterations (applications 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 $\hat{E}=\varnothing$, then after each construction, we check whether the constructed solution S is dominated or not with respect to the solutions already in $\hat{E}$, including it if it is efficient. Moreover, after each movement of the local search we check whether the obtained solution qualify to enter $\hat{E}$ or not (is a non-dominated solution). Figure 3 shows a pseudo-code of the GRASP algorithm.

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

While($iter \leq max\_iter$)
- 4. Apply the RDE constructive method $\Rightarrow S$.
- 5. Check the inclusion of S in $\hat{E}$.
- 6. *iter*2=1.
- While ($iter2 \leq g\_iter$)
  - 7. *iter*3=1.
  - While($iter3 \leq l\_iter$).
    - 8. Perform the first improving move w.r.t. $z_1 \Rightarrow S'$
    - 9. Check the inclusion of S' in $\hat{E}$
    - 10. *iter*3 = *iter*3+1
  - 11. iter4=1
  - While($iter4 \leq l\_iter$).
    - 12. Perform the first improving move w.r.t. $z_2 \Rightarrow S''$
    - 14. Check the inclusion of S'' in $\hat{E}$
    - 14. *iter*4 = *iter*4+1
  - 15. *iter*2 = *iter*2+1
- 16. *iter* = *iter* +1

Figure 3. GRASP algorithm

5. Computational Experiments

Our experimentation has two main goals. The first one is to present a comparison between existing procedures adapted to the bi-objective path dissimilarity problem, and the second one to show that the proposed GRASP procedure is capable of creating better approximations of the efficient frontiers than the 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 the standard ones in multi-objective optimization:

1. *Number of points*: This refers to the ability of finding efficient points. It is assumed that a larger number of efficient points is preferred by the decision maker.

2. *SSC*: This metric suggested by Zitzler and Thiele (1999) 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 Zitzler, Laumanns and Thiele (2001) in connection with the computational testing of SPEA2 is based on the k^{th} nearest neighbor method of Silverman (1986) . The metric is simply the distance to the k^{th} nearest efficient point. 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 (Zizler and Thiele, 1999). $C(A,B)$ 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 them harder to solve as a Path Dissimilarity Problems, we remove most of the edges in the original instances and only include those edges with a cost (distance value) lower than a 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. In all of them we remove the large edges 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 (Carotenuto et al. 2007) 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. Figures 4 and 5 show the results of these six methods on the gr431 with p=15 and on the att532 instance with p=10 respectively.

Figures 4 and 5 clearly show that the GRASP method obtains a better approximation of the efficient frontier of instances gr431 and att532 (with p=15 and 10 respectively) than the other competing methods. We can see in these diagrams that GRASP obtains a larger number and better distributed points in the efficient frontier than the other algorithms. The next experiment complements this information since it shows the performance measures described above for these six methods.

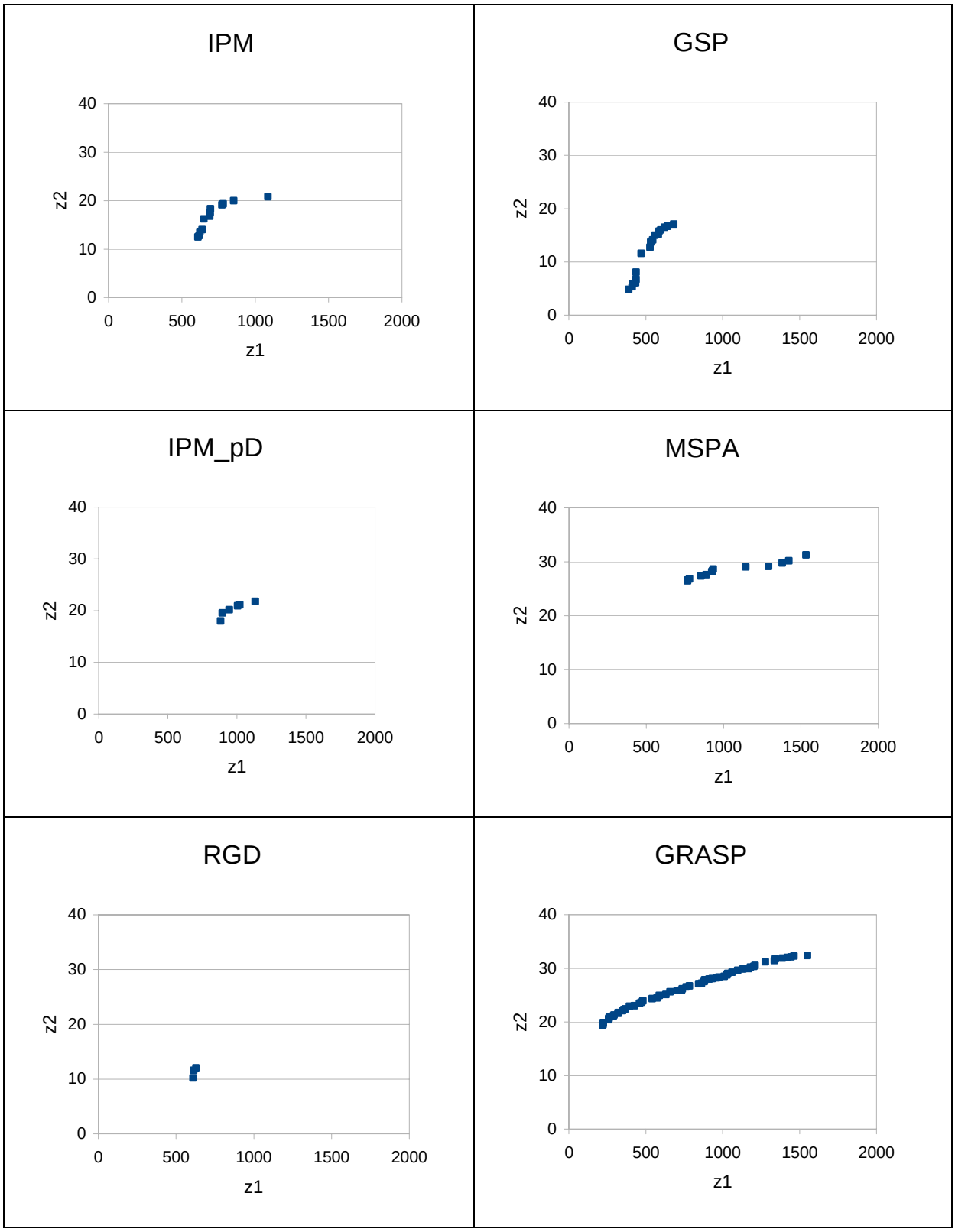

Figure 4. Approximation of the efficient frontier for gr431 with $p = 15$.

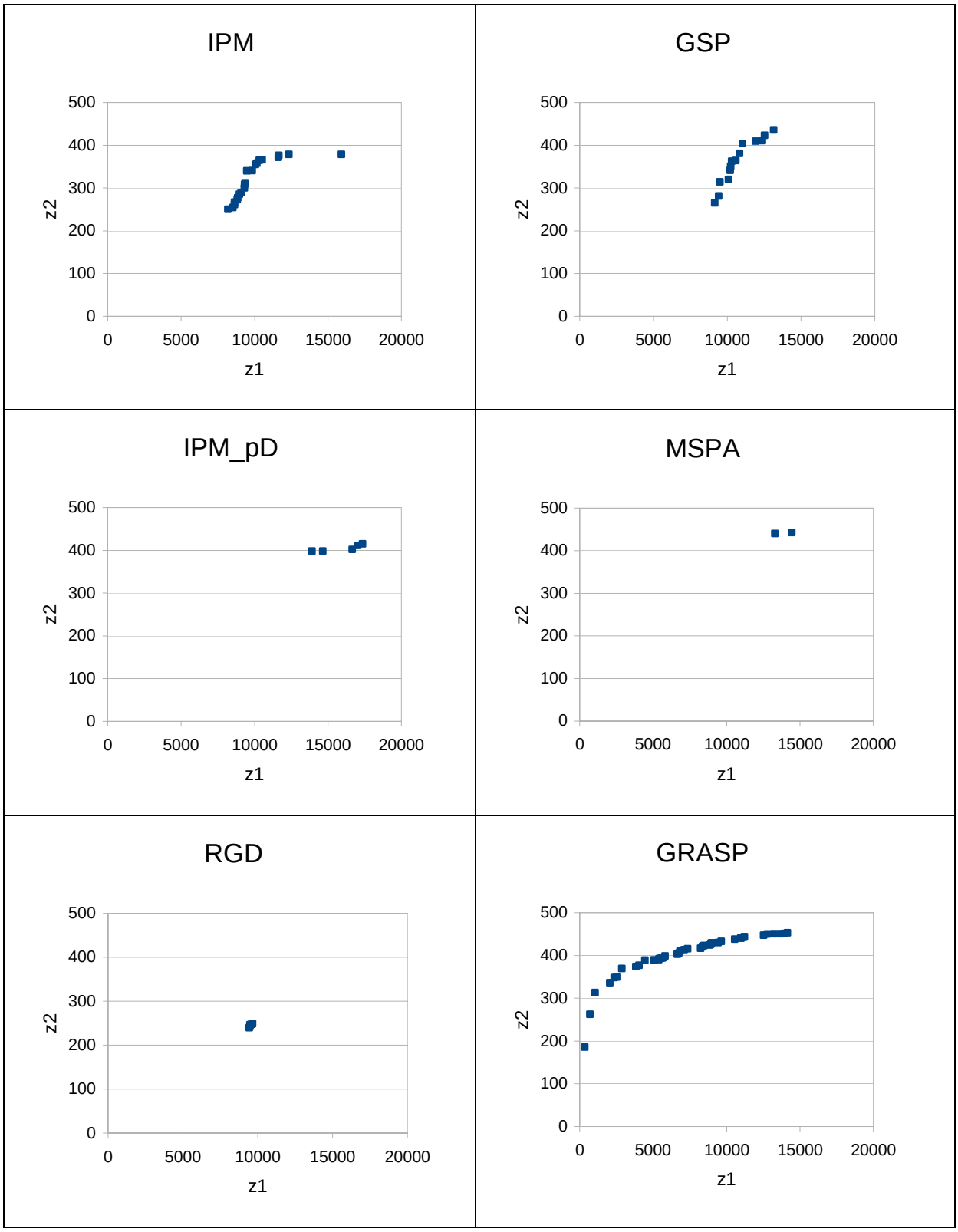

Figure 5. Approximation of the efficient frontier for att532 and $p = 10$.

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 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.

Methods	*N. 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 1. Performance measures with p=5

Methods	*N. 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 2. Performance measures with p=10

Methods	*N. 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

Table 3. Performance measures with p=15

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 5 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 a superior performance of GRASP over the competing approaches since it is significantly larger than the others in the three tables. CPU times are of a similar magnitude in all the methods (around 6 minutes) 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).

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).

***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

Table 4. Coverage of two sets

The values in Table 4 show that the frontier generated by GRASP dominates those generated by the other methods. For example, *C*(GRASP,GSP)=0.94 indicates that 94% of the point in the frontier obtained with GSP are dominated by points in the frontier obtained with GRASP. We compare this value with *C*(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 the cases we can see that *C*(GRASP,-) > *C*(-,GRASP), assessing the superiority of GRASP in terms of dominance.

6. Conclusions

We have formally stated the Bi-objective Path Dissimilarity Problem (PDP), and described the development and implementation of a GRASP procedure for its resolution. The final design is then compared to state-of-the-art methods and the outcome of our experiments seems quite conclusive in regard to the merit of the procedure that 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 one to test several procedures for the Path Dissimilarity Problem within the multi-objective programming framework and its associated evaluation measures based on the efficient frontier.

Acknowledgments

This research has been partially supported by the *Ministerio de Educación y Ciencia* of Spain (Grant Refs. TIN2006-02696 TIN2005-08943-C02-02) and by the Comunidad de Madrid – Universidad Rey Juan Carlos project (Ref. URJC-CM-2006-CET-0603), by ITESM Campus Monterrey Research Fund CAT025 and CONACYT 61903.

The authors want to thank Professor Julian Molina for their help with the computation of multi-objective performance measures.

References

Akgun, V., Erkut, E. and R. Batta (2002) "On finding dissimilar paths", *European Journal of Operational Research*, 121:232-246

Carotenuto, P., S. Giordani and S. Ricciardelli (2007) " Finding minimum and equitable risk routes for hazmat shipments", *Computers & Operations Research*, 34:1304-1327.

Clímaco, J.C. and E.Q.V Martins (1982) "A bicriterion shortest path algorithm", *European Journal of Operational Research*, 11:399-404.

Dell'Olmo, P., M. Gentili and A. Scozzari (2005) "On finding dissimilar Pareto-optimal paths", *European Journal of Operational Research*, 162:70-82

Duarte, A. and R. Martí (2007) "Tabu Search and GRASP for the Maximum Diversity Problem," *European Journal of Operational Research,* 178 (11): 71-84.

Erkut, E. (1990) "The discrete *p*-dispersion problem", *European Journal of Operational Research*, 46. 48-60.

Feo, T.A. and M.G.C. Resende (1989) "A probabilistic heuristic for a computationally difficult set covering problem", *Operations Research Letters*, 8:67-71.

Feo, T.A., and M.G.C. Resende (1995) "Greedy Randomized Adaptive. Search Procedures", *Journal of Global Optimization,* 6:109-133.

Glover, F., C. C. Kuo, and K. S. Dhir (1998) "Heuristic Algorithms for the Maximum Diversity Problem," *Journal of Information and Optimization Sciences*, 19(1): 109-132.

Johnson, P.E., D.S. Joy and D.B. Clarke (1992) "HIGHWAY 3.01, An Enhancement Routing Model: Program, Description, Methodology and Revised User's Manual", *Technical Report*, Oak Ridge National Laboratories.

Kuby, M., X. Zhongyi and X., Xiaodong (1997) "A minimax method for finding the *k* best diferentiated paths" *Geographical Analysis* 29 (4): 298-313.

Lombard, K. and R.L. Church (1994) "The gateway shortest path problem: generation of alternative routes for a corridor location problem", *Geographical System,* 1: 25-45.

Martins, E.Q.V (1984) "On a multicriteria shortest path problem", *European Journal of Operational Research*, 16: 236-245.

Resende, M.G.C., Ribeiro, C.C. (2003) "*Greedy randomized adaptive search procedures*" In: Glover, F., Kochenberger, G. (Eds.), State-of-the-art Handbook in Metaheuristics, Kluwer Academic Publishers, Boston, 219-250.

Silverman, B. W. (1986) Density estimation for statistics and data analysis. Chapman and Hall, London.

Zitzler, E. (1999) *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, Ph D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.

Zitzler, E., K. Deb and L. Thiele (2000) "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation Journal*, vol. 8, no. 2, pp. 125-148.

Zitzler, E., M. Laumanns and L. Thiele (2001) "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Zitzler, E. and L. Thiele (1999) "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271.