GRASP with exterior path-relinking for the multidimensional two-way number partitioning problem

Francisco J. Rodriguez[a], Fred Glover[b], Carlos García-Martínez[c], Rafael Martí[d], Manuel Lozano[a]

[a]*Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain*
[b]*OptTek Systems, Boulder (Co), USA*
[c]*Department of Computing and Numerical Analysis, University of Córdoba, Córdoba, Spain*
[d]*Department of Statistics and Operations Research University of Valencia, Valencia, Spain*

Abstract

In this work, we tackle multidimensional two-way number partitioning (MDTWNP) problem by combining GRASP with Exterior Path Relinking. In the last few years, the combination of GRASP with path relinking (PR) has emerged as a highly effective tool for finding high-quality solutions for several difficult problems in reasonable computational time. However, in most of the cases, this hybridization is limited to the variant known as interior PR. Here, we couple GRASP with the "exterior form" of path relinking and perform extensive experimentation to evaluate this variant. Our computational experiments show the superiority of this approach compared with the previous best method for MDTWNP and with alternative methods for this problem that use other forms of PR.

Keywords:
multidimensional two-way number partitioning problem, GRASP, exterior path-relinking, binary combinatorial optimisation

Email address: fjrodriguez@decsai.ugr.es (Francisco J. Rodriguez)

1. Introduction

The *multidimensional* two-way number partitioning (MDTWNP) is a binary optimization problem that consists of partitioning a set of vectors S into two disjoint groups so that the maximum difference between the sum per coordinate of the elements in each group is minimised.

More formally, Kojic [1] describes the MDTWNP problem as follows. Given a set of n vectors of dimension d, $S = \{w_i \mid w_i = (w_{i1}, w_{i2}, \ldots, w_{id}), i = 1, \ldots, n)\}$, the objective is to split the elements of S into two sets S_1 and S_2 so that $S_1 \cap S_2 = \emptyset$, $S_1 \cup S_2 = S$, and t is minimum, with:

$$t = \max\{|\sum_{w_i \in S_1} w_{ij} - \sum_{w_i \in S_2} w_{ij}| : j = 1, \ldots, d\} \quad (1)$$

Example 1.1. *Given* $S = \{w_1 = (2, 6), w_2 = (-1, 5), w_3 = (3, -7), w_4 = (-2, 4), w_5 = (-2, -1)\}$, *consider the solution with* $S_1 = \{w_2, w_3\}$ $S_2 = \{w_1, w_4, w_5\}$. *In order to compute its objective function value, firstly, we calculate the sum of the vectors included in each subset per every coordinate, obtaining* $(2, -2)$ *for* S_1 *and* $(-2, 9)$ *for* S_2. *Secondly, we obtain the differences,* $(|2-(-2)|, |-2-9|) = (4, 11)$. *Therefore, the objective function value for this solution is* $max(4, 11) = 11$.

The MDTWNP problem is NP-hard [1], and it is a generalisation of the *two-way number partitioning* problem [2, 3] where the objective is to split a set of numbers. The latter has practical applications in different areas such as scheduling on multiprocessors, designing of VLSI circuits, public key cryptography, and database processing [4, 5]. Kojic [1] described two different real-world scenarios for the MDTWNP problem related to tasks distribution area. In the first one, two travel agents need to have as closely similar revenues from organized tours as possible, considering the price of the tour, the mileage due to petrol consumption, and special offers such as restaurants, shopping tours, or visits to museums. In the second one, the objective is to homogenize police patrols according to the mileage, degree of risk, and time needed for touring their beat.

We propose a new method for MDTWNP by joining GRASP and Exterior Path Relinking. We now describe their key components, and our rationale for combining them.

GRASP [6, 7] is a multistart metaheuristic consisting of two phases, construction and improvement. In the first phase, a greedy randomised procedure generates an initial solution by adding elements one at a time from a restricted candidate list to the current partial solution. The second phase performs a local search to improve the quality of the solution obtained in the first phase. Construction and improvement phases are performed iteratively until a predefined termination condition is met.

Path-relinking (PR) was originally proposed to integrate intensification and diversification strategies within the context of tabu search and scatter search [8, 9, 10]. PR creates new solutions by a process of generating paths between and beyond pairs of selected points in neighbourhood space. In spite of the widespread application of PR in combinatorial optimisation, almost all PR implementations only consider the between-form of PR. On the contrary, the beyond-form of PR introduced by Glover in [11] remains virtually unexplored. This variant, called exterior PR (ePR), only has been recently applied to address the differential dispersion problem [12], obtaining very promising results.

Based on an original proposal by Laguna and Martí [13], GRASP is usually combined with PR in order to improve its performance, and in the last few years, this combined methodology has emerged as a leading method for finding high-quality solutions for several difficult problems within reasonable computational time [13, 14, 15, 16, 17, 18]. The efficacy of this approach leads us to go a step farther by proposing an algorithm that combines GRASP with ePR to deal with the MDTWNP problem. We further enhance our algorithm by proposing a new restricted local descent method that accelerates the selection of moves and improves the quality of solutions that can be found within a given time limit.

The remainder of this paper is organised as follows. Section 2 reviews the state-of-the-art algorithms for the MDTWNP problem. In Section 3, we detail the proposed GRASP+PR algorithm for the MDTWNP problem, where our PR method embodies the ePR approach. In Section 4, we show the results of an empirical study whose aim is twofold: 1) analyse the influence of different parameters and algorithm components, 2) compare the results of a tuned GRASP+PR with those obtained by other state-of-the-art algorithms for the MDTWNP problem. Finally, in Section 5, we discuss conclusions and further work.

2. State-of-the-art Algorithms for MDTWNP problem

There are many approaches in the literature to solve the number partitioning problem such as greedy heuristics [19, 20], total enumeration [21], a hybrid algorithm that combines branch and bound, breadth first search, and beam search [3], simulated annealing [22], tabu search [9, 23], genetic and memetic algorithms [24, 25], and GRASP [26].

Unfortunately, as shown by Kojić [1], some of them cannot be directly applied to solve the MDTWNP problem. For this reason, Kojić proposes the following *integer linear programming formulation* for this problem:

$$\textbf{Minimise } t \tag{2}$$

$$\textbf{Subject to } -0.5 \cdot t + \sum_{i=1}^{n} w_{ij} \cdot x_i \leq 0.5 \cdot s_j, \ \forall j = 1, \ldots, d, \tag{3}$$

$$0.5 \cdot t + \sum_{i=1}^{n} w_{ij} \cdot x_i \geq 0.5 \cdot s_j, \ \forall j = 1, \ldots, d, \tag{4}$$

$$s_j = \sum_{i=1}^{n} w_{ij}, \ \forall j = 1, \ldots, d, \tag{5}$$

$$x_i \in \{0, 1\}, \ \forall i = 1, \ldots, n. \tag{6}$$

where s_j is the sum of the value stored in the $j-th$ position of all vectors (Equation (5)), t is necessarily the largest absolute difference between the sum per every coordinate of both groups because of Equations (2)-(4), and $x_i = 1$ if the vector $w_i \in S_1$ and 0 otherwise. Kojić carries out an experimental study to solve the above formulation with CPLEX on a set of instances presented also in this work.

Later, Pop and Matei [27] proposed a memetic algorithm (MA) to deal with the MDTWNP problem which starts from an initial random population and evolves it by means of crossover and mutation operators. The crossover operator selects two parents using a binary tournament method and applies single point crossover. At each generation, 85% of the new population is generated by crossed solutions and the remaining 15% is copied directly from the old population. Mutation is applied with a probability of 10%, flipping the group to which the vector belongs. Finally, for each new solution obtained using the genetic operators, a local improvement procedure is applied, em-

ploying three different neighbourhoods in succession consisting of 1, 2, and 3-swap moves, respectively.

Recently, Kratica et al. [28] presented a *variable neighbourhood search* (VNS) and a *electromagnetism based metaheuristic* (EM) for the MDTWNP problem. The variable neighbourhood search starts with a random solution, shakes the incumbent solution by randomly changing several elements of this solution and performs a local search procedure based on 1-swap improvements. The electromagnetism-like metaheuristic is a population based algorithm that evolves solutions through three different stages: local search, scaling and moving. Firstly the same local search procedure as in VNS is applied. Then, solutions from the population are moved toward local optima obtained by the local search procedure during the scaling step. Finally, during the moving phase, solutions are led to the most promising regions of the search space sampled by the population.

The experimental study of Kratica et al. compares VNS, EM, MA, and the results obtained by CPLEX, where the latter solves the integer linear programming model for the MDTWNP problem [1]. The authors test EM and VNS on the set of instances presented in [1] and compare them with the results reported by MA and CPLEX in their respective original publications. The study concludes that VNS and EM outperform MA and CPLEX, but finds no statistically significant differences between EM and VNS.

3. GRASP and ePR for the MDTWNP problem

In this section we present the general design of our GRASP+PR for the MDTWNP problem (Section 3.1). Then, we detail the main elements of the model: construction phase (Section 3.2), local improvement phase (Section 3.3), and PR (Section 3.4), focusing on the novel contributions made in each of these stages: constructive procedure guided by heuristic information, local search with restricted neighbourhood and ePR.

3.1. General Scheme

As previously intimated, GRASP is a multistart process that iteratively performs two main phases, construction and solution improvement, which respectively employ a greedy randomised procedure to generate an initial solution and then execute an improvement process to drive the initial solution to a local optimum. The best solution found over all iterations is returned as the result.

As it is well known, the basic GRASP scheme does not make use of any memory structure to exploit the information obtained about the search space in previous iterations. To complement this approach, GRASP is frequently hybridized with PR, as first proposed by Laguna and Martí [13]. In this hybridization a set P of so-called *elite solutions* is incorporated that stores high-quality and diverse solutions found throughout the search process. The PR method then explores trajectories between and beyond solutions obtained by the improvement procedure and solutions from the elite set.

The general scheme of our proposed GRASP+PR algorithm (which incorporates the option of using the ePR method) is shown in Figure 1. We employ six input parameters: $typePR$ identifies the form of PR, T_{max} denotes the computation time limit, $maxElite$ is the maximum number of solutions in the elite set P, $typeLS$ identifies the particular local search method applied during the improvement phase, and $typeRCL$ and α are parameters related to the construction phase. These parameters are explained in detail in the following sections.

Each iteration begins by constructing an initial solution s by procedure Construction($typeRCL, \alpha$) (Section 3.2). Then a local search improvement method is applied to the solution s in ImprovementProcedure($s, typeLS$) (Section 3.3). Next, PR is applied in procedure PathRelinking($typePR, s, maxElite$) (Section 3.4). The best solution found during the search, denoted s^*, is returned at the end.

```
   Input: T_max, maxElite, typePR, typeRCL, α, typeLS
   Output: s*
 1 s* ← NULL;
 2 P ← ∅;
 3 while computation time limit T_max not reached do
 4 |   s ← Construction(typeRCL, α);
 5 |   s ← ImprovementProcedure(s, typeLS);
 6 |   s ← PathRelinking(typePR, s, maxElite) end;
 7 |   if s* == NULL then
 8 |   |   s* ← s;
 9 |   else
10 |   |   if f(s) < f(s*) then s* ← s end;
11 |   end
12 end
```

Figure 1: Pseudocode for GRASP with PR for MDTWNP problem

3.2. Construction procedure

In the construction phase, a feasible solution is created by means of an iterative procedure that combines greediness and randomness to successively select new elements to add to the current partial solution (which begins empty). Elements are drawn from a *restricted candidate list* (RCL) and the process repeats until a complete solution is obtained. The construction procedure has two main components: a function $g(\cdot)$ that evaluates the solution elements and a strategy to manage the RCL. Below, we describe each of these components:

- Function $g(\cdot)$ is responsible for indentifying the most promising solution elements and defines the greedy behaviour of the construction procedure. Taking into account that a feasible solution is obtained by assigning each vector $w \in S$ to one group $j = \{1, 2\}$, a solution element is represented by a pair (w, j). Given a partial solution consisting of sets S_1 and S_2 of vectors already assigned to the groups 1 and 2, we calculate the greedy value $g(z, j)$ for the assignment of an unassigned vector z to the group j as follows:

$$g(z,1) = \max\{|(\sum_{p \in S_1} w_{pi} + z_i) - \sum_{q \in S_2} w_{qi}| : i = 1, \ldots, d\}. \quad (7)$$

$$g(z,2) = \max\{|\sum_{p \in S_1} w_{pi} - (\sum_{q \in S_2} w_{qi} + z_i)| : i = 1, \ldots, d\}. \quad (8)$$

  where z_i is the value of the $i-th$ position of the vector z. By this definition, the smaller $g(z, j)$ values are attached to the solution elements (z, j) that are more attractive.

- Several different ways exist to manage the RCL [29]. In this work, we have chosen two of them. In the first, denoted by B_RCL, the RCL is composed of high-quality candidate elements which are restricted either in number (cardinality-based) or by their quality (value-based). The cardinality based membership includes within RCL a percentage α of the full set of elements ordered by their attractiveness. Then, a random element from the RCL is chosen to be added to the current partial solution. In the second approach for managing the candidate

list, denoted by R_RCL, the RCL consists of randomly selected solution elements whose number is again determined as a percentage α of the full set, and the member of RCL having the best greedy function value is chosen to be added to the current partial solution. The use of randomness in various parts of the RCL management facilitates the exploration of different regions of the search space by the constructive procedure. B_RCL is the standard implementation that first applies the greedy component, evaluating the candidate elements, and then applies the random one in the selection. R_RCL, recently applied [30], performs first the random component, and then the greedy one.

The detailed pseudocode of the construction procedure described above is shown in Figure 2.

```
   Input: α, typeRCL
   Output: s
1  S̄ ← S;
2  Let s be an empty solution;
3  while S̄ ≠ ∅ do
4      if typeRCL == B_RCL then
5          For all pairs (z, j) : z ∈ S̄, j = {1, 2} compute g(z, j);
6          RCL ← {(z, j) | g(z, j) is one of the α · n best greedy values };
7          (z*, j*) ← Choose an element at random from RCL;
8      end
9      else if typeRCL == R_RCL then
10         RCL ← Choose randomly α·n elements from {(z, j) : z ∈ S̄, j = {1, 2}};
11         For all pairs (z, j) ∈ RCL compute g(z, j);
12         (z*, j*) = argmin{g(z, j) : (z, j) ∈ RCL };
13     end
14     Assign vector z* to group j* in solution s;
15     S̄ ← S̄\{z*};
16 end
```

Figure 2: Pseudocode for contruction procedure

3.3. Local improvement phase: restricted first interchange local search

The aim of the improvement phase is to explore the neighbourhood of the solution obtained by the construction procedure to find better solutions relatively close to the initial one. The construction procedure is responsible for

identifying promising areas of the search space, whereas the local improvement procedure is in charge of intensifying the search in these promising regions.

We have considered two different local search procedures for the MDTWNP problem, both of which consist of a simple descent that stops as soon as no further improving moves are detected. The first, proposed in [28], explores the search space near a particular solution by choosing two random vectors assigned to different groups in the current solution and then exchanging (swapping) the groups to which they belong. The first neighbour found that improves the current solution becomes the new current solution. We call this process *first interchange local search* (FI).

The second form of simple descent we consider is a new one that we propose to overcome a limitation of FI which stems from the fact that the neighbour solution selected by this process can have a significantly different objective function value than the current solution, and this can cause the process to become prematurely lodged in a local optimum. Our new local search descent method, which we call *restricted first interchange local search* (RFI), evaluates only a single swap move for each vector by pairing it with the most similar vector (in terms of Euclidean distance) assigned to a different group. Like in FI, the order for selecting the vectors is determined by a random permutation. As soon as such a restricted swap move is found that improves the current objective function value, it is selected to produce the new solution. This process, like the FI method, ends when there is no available move (of the form considered) that enhances the current value. The RFI method moves to nearby solutions with objective function values that are generally close to those of solutions left behind, yielding a more gradual form of descent. At the same time, by restricting the neighbourhood explored, the RFI local search is faster

It is important to highlight that, in order to improve the performance of the local search methods, the new solutions obtained after a swapping move are not evaluated from scratch but applying *delta evaluation*. With this purpose, we store the sum of the elements in each group for each coordinate ($Sum(S_i, j) : i = \{1, 2\}, j = \{1, ..., d\}$) for the current solution, and when the group of a vector w is changed, we only have to update accordingly the sums, per coordinate, of the old (S_{old}) and new (S_{new}) group. This results from the computation shown in Equations 9 and 10, enabling the new objective

function value to be obtained more quickly after a swapping move.

$$Sum(S_{old}, j) = Sum(S_{old}, j) - w_j \ \ : j = 1, ..., d. \tag{9}$$

$$Sum(S_{new}, j) = Sum(S_{new}, j) + w_j \ \ : j = 1, ..., d. \tag{10}$$

3.4. Exterior Path Relinking

The main objective of PR is to explore paths between and beyond high-quality solutions in order to find new solutions in their vicinity that may be even better. Several variants of PR have been studied in the literature [13, 31, 32, 33, 30]. Among these, two main variants yielding high quality results have been widely analysed, the first consisting of interchanging initiating and guiding solutions to generate different trajectories between these solutions and the second consisting of simultaneously moving from each of these solutions toward the other. These studies have always focused on interior PR, that is, exploring paths between solutions (see Figure 3). Here, by contrast, we explore the exterior PR (ePR) approach introduced in [11].

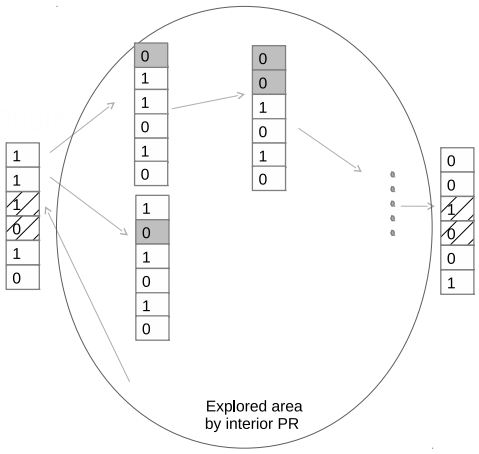

Figure 3: Interior PR scheme

The essence of ePR for binary optimization can be characterised as follows (see Figure 4). Let s_i and s_g denote an initiating and guiding solution, respectively. Starting from $s = s_i$ and defining $J = \{j : s_g^j = s_i^j\}$, we focus on changing the value of only those components s^j for $j \in J$. This way, we avoid moving either in the direction of s_i or s_g. The variable s^j for $j \in J$

that is selected to be flipped at each stage of the ePR process is typically one that produces a new solution s with the highest evaluation (best objective function value) from the accessible candidates. This is called greedy PR [30], in contrast to greedy randomized PR in which we randomly select an element among the candidates with better evaluations. In this manner, the search is drawn toward attractive solutions (and potential improving sequences) when these are available.

Given two solutions for the MDTWNP problem, ePR considers the set of moves where the vectors assigned to the same group are shifted. The process starts from the so-called initiating solution and, at each step, the best move is performed, stoping when there is no possible move left. As underscored in the tabu search literature [8, 9], diversification should not be divorced from intensification, but should aim at producing high quality solutions at the same time that it drives the solution process into new regions. The exterior form of PR is usefully structured to accomplish this.

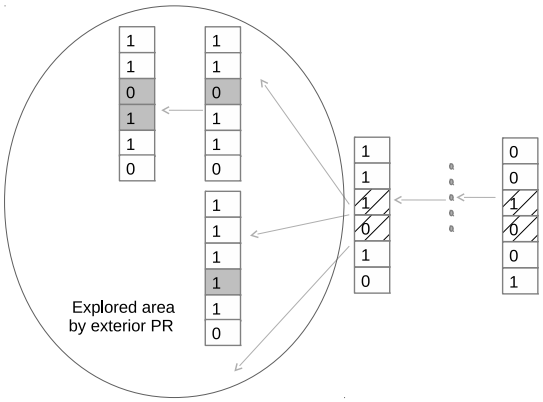

Figure 4: ePR scheme

As noted in [11], instead of starting with $s = s_i$, we can also start the exterior PR procedure from the solution s' that was selected as best during an interior PR process, yielding s' between s_i and s_g. We will call this alternative approach *alt_ePR*.

The solutions chosen to be combined by PR are drawn from a reference set consisting of high-quality and diverse solutions found throughout the execution of the algorithm. In this exterior form of PR we deem it natural to choose the initiating solution to have a better objective function value than the guiding one, on the supposition that the chance of finding high-quality

solutions should be higher near the better solutions. Therefore, we choose the initiating solution to be the higher quality solution from the two candidates consisting of the current GRASP solution and a solution selected at random from the reference set. In the following, we take the liberty of referring to the reference set as the *elite set*.

In order to have a benchmark for comparing the behaviour of the exterior form of PR, we have also developed an instance of interior PR for the MDTWNP problem. The moves of interior PR involve shifting those vectors in the initial solution that are assigned to a different group in the guiding solution. We have applied backward PR [34] for interior PR, where the initial solution is chosen as the best of the current GRASP solution and the solution from the elite set. Previous experiments carried out by Aiex et al. [15] and Resende et al. [35] for interior PR showed that backward PR usually outperforms *forward PR*. The latter, on the contrary, assigns the role of initial solution to the worst solution.

The complete pseudocode of our PR procedures for the MDTWNP problem is shown in Figure 5. The PR procedure Path-Relinking($typePR, s, maxElite$) receives three input parameters, the type of PR (interior or exterior, where alt_ePR is performed by calling PR function twice), the solution s achieved by the improvement procedure and the maximum size of the elite set. First, a random solution s' from the elite set is selected. Then, for interior PR, the less attractive (lower quality solution) of the two solutions s and s' plays the role of guiding solution s_g, whereas the more attractive becomes the initial solution s_i. Then each step selects a vector w^* from the set $\Delta(s_c, s_g)$ which consists of those vectors in s_i assigned to a different group from s_g, so that the quality of the solution resulting from changing the group to which this vector belongs is maximised. This process is repeated until $|\Delta(s_i, s_g)| > 1$, that is, next solution in the path is the guiding solution.

Similarly, the more attractive solution between s and s' is used as the initiating solution s_i for ePR. Then, at each step, w^* is chosen from the set of vectors assigned to the same group in s_i and s_g, $\delta(s_i, s_g)$. The process stops when $|\delta(s_i, s_g)| = 0$.

The solution s obtained after applying PR is considered as a candidate to be added to the elite set, provided that s is not already included in the set elite. This procedure is implemented by the function UpdateEliteSet($s, P,$ $maxElite$) and its pseudocode is shown in Figure 6. Its structure is based on the procedure proposed by Martí and Sandoya in [36]. If the current cardinality of the set P is less than its maximum capacity ($maxElite$), s is

```
   Input: typePR, s, maxElite
   Output: s
 1 e ← Random(1, |P|);
 2 s' ← P_e;
 3 s_i ← argmin{f(s), f(s')};
 4 s_g ← argmax{f(s), f(s')};
 5 if typePR == Interior then
 6 |   while |Δ(s_i, s_g)| > 1 do
 7 |   |   w* ← argmin{f(mov(s_i, w)) : w ∈ Δ(s_i, s_g)};
 8 |   |   s_i ← mov(s_i, w*);
 9 |   |   if f(s_i) < f(s) then
10 |   |   |   s ← s_i;
11 |   |   end
12 |   end
13 end
14 else if typePR == Exterior then
15 |   while |δ(s_i, s_g)| > 0 do
16 |   |   w* ← argmin{f(mov(s_g, w)) : w ∈ δ(s_i, s_g)};
17 |   |   s_i ← mov(s_i, w*);
18 |   |   if f(s_i) < f(s) then
19 |   |   |   s ← s_i;
20 |   |   end
21 |   end
22 end
23 UpdateEliteSet(s, P);
```

Figure 5: Pseudocode for PR

just added to P, provided that s is not already included in P. By contrast, if the elite set is full, s must meet at least one of the following conditions to be included in the set:

- The objective function value of s is better than the best solution in the elite set (s_b).
- The objective function value of s is better than the worst solution in the elite set (s_w) and the diversity that s would provide to the elite set $Div(s, P)$ is larger than the smallest diversity contribution of any of the solutions that are currently in the elite set.

If s fulfils either of these conditions, it is added to the elite set by replacing the solution most similar to it (in terms of distance).

```
   Input: s, P, maxElite
   Output:
 1 if s ∈ P then
 2 |  Exit();
 3 end
 4 if |P| < maxElite then
 5 |  P ← P ∪ {s};
 6 end
 7 else
 8 |  s_w ← argmax{f(t) : t ∈ P};
 9 |  s_b ← argmin{f(t) : t ∈ P};
10 |  if f(s) < f(s_b) then
11 |  |  s_r ← argmin{Dist(s,t) : t ∈ P};
12 |  |  P ← P\{s_r};
13 |  |  P ← P ∪ {s};
14 |  end
15 |  else if f(s) > f(s_w) then
16 |  |  d ← argmin{Div(t,P) : t ∈ P};
17 |  |  Div(s,P) ← Σ_{t∈P} Dist(s,t);
18 |  |  if Div(s,P) > Div(d,P) + Dist(s,d) then
19 |  |  |  s_r ← argmin{Δ(t,s) : t ∈ P};
20 |  |  |  P ← P\{s_r};
21 |  |  |  P ← P ∪ {s};
22 |  |  end
23 |  end
24 end
```

Figure 6: Pseudocode for procedure to update the elite set

The diversity contribution of a solution s to the elite set is calculated as the sum of the distances between this solution and all the solutions in this set. The diversity in the elite set becomes larger as distances among its solutions get greater. The function $Dist(s_1, s_2)$ is used to calculate the distance between any two solutions s_1 and s_2 and returns the minimum value over the number of vectors assigned to a different group and to the same one. Notice that, if we change the group to which every vector belongs, we obtain the same objective value.

From the point of view of computational complexity, we should note that diversity values are not recalculated from scratch at each iteration. For this purpose, we store $Div(s, P)$ for each $s \in P$. In this way, each time a solution s_a is added to P and replaces s_r in P if needed, diversity values are updated

as follows:

$$Div(s_a, P) = Div(s_a, P) - Dist(s_a, s_r). \tag{11}$$

$$Div(t, P) = Div(t, P) + Dist(s_a, t) - Dist(s_r, t)\ \forall t \neq s_a \in P. \tag{12}$$

4. Computational Experiments

This section details the computational experiments performed to study the behaviour of our GRASP+PR algorithm for the MDTWNP problem. First, we outline the experimental framework (Section 4.1) and then analyse the results obtained. The objective of these experiments is twofold: (1) to analyse the behaviour of the proposed method depending on its parameters and settings; (2) to compare the results of GRASP+PR with those obtained by the VNS algorithm proposed in [28] and by CPLEX, executing the latter with the integer linear programming model detailed in Section 2.

4.1. Experimental Setup

GRASP+PR is implemented in C++ and compiled with gcc 4.8.2. The experiments were carried out on a computer with an Intel® Core™ i7-930 2.8 GHz processor (8MB cache, four cores and eight threads) with 24GB of RAM running Fedora™ Linux 20. Our experiments use the instance set created in [1], which is composed of 210 instances with different combinations of the number of vectors (n) and their dimensions (d). In particular, the value n is drawn from {50, 100, 200, 300, 400, 500} and d is drawn from {2, 3, 4, 5, 10, 15, 20}. For each combination of n and d, there are 5 different instances with different random values for each element w_{ij}. We have divided this set of instances into two subsets. The first one is used exclusively for tuning the parameters and configuration of GRASP+PR (using the first two instances of each type, 84 instances in total), and the second one to perform the comparison with the state-of-the-art algorithms (126 instances, using the remaining 3 instances of each type).

We execute GRASP+PR 10 times on each problem instance. The maximum CPU time (T_{max}) allotted to each instance is fixed depending on the number of vectors involved, in particular, $n/10$ seconds.

We employ non-parametric statistical tests, as proposed in [37], to compare the different algorithms. The performance measure is the average of the

best values found over the 10 independent runs. In particular, we have employed two alternative non-parametric analysis to compare the experimental results:

- In the first case, we use the *Iman and Davenport* test [38], followed by *Holm's* method [39] as a post-hoc procedure. The first test allows us to see whether there are statistically significant differences among a group of algorithms. Provided that such differences are detected, the second test compares the best performing algorithm (control algorithm) against the remaining ones.

- The second method is the Wilcoxon matched-pairs signed-ranks test [40] to perform pairwise comparisons. In order to perform the test, we rank the differences, in absolute value, between the performance scores of the two algorithms compared for each instance. Then, we compute R^+ as the sum of ranks for the instances in which the first algorithm outperforms the second one and R^- as the sum of ranks for the reverse case. If R^+ is greater than R^- and R^- is less than or equal to the critical value, the first algorithm outperforms the second one. On the contrary, if R^+ is less than R^- and R^+ is less than or equal to the critical value, the second algorithm outperforms the first one. Otherwise, the test does not find differences between the algorithms' performances.

4.2. Parameter Study of the GRASP+PR Algorithm

Our objective in this section is to study the effect of different parameters and strategies used in our algorithm. We first attempt to obtain a competitive configuration of the GRASP+PR algorithm by performing a tuning of the parameters and strategies. Subsequently, we analyse the impact of different key mechanisms, namely those consisting of PR, the randomised construction procedure, and the local search improvement procedure.

4.2.1. Parameter Tuning of the GRASP+PR Algorithm

In this section, we perform an experimental study to find a suitable configuration for the proposed algorithm. We have carried out a full factorial experiment [41, 42] with the RCL construction procedure (B_RCL, R_RCL), the local search method (FI, RFI), the size of the elite set (6, 12), the type of PR (Interior, Exterior, and Alt_ePR), and the percentage of solution components

included into the RCL (0.05, 0.1, 0.2), exploring 72 different combinations in total. However, it is important to note that additional values can be considered for each parameter, and therefore the performance of GRASP+PR might be further improved.

Table 1 details the mean ranking of the different GRASP+PR configurations. The ranking is obtained by ordering the algorithms according to their observed results at each problem instance, assigning the ranking 1 to the best algorithm, and $|A|$ to the worst (where A is the set of algorithms). Each configuration is denoted by GRASP-<RCL_Type>-< α >-<PR_Type>-<Elite_Set_Size>-<Local_Search>.

In order to identify significant performance differences between the algorithms, we have first applied the Iman and Davenport test. The statistical value (395.19) is greater than the critical one (1.29) for a significance factor of 0.05, evidencing that performance differences really exist. Then, we have compared the behaviour of the best-performing configuration (control algorithm, GRASP-R_RCL-0.05-Exterior-6-RFI) against the remaining ones by means of Holm's test. The third column details the results of this test, showing whether there are significant performance differences (+) or not ($\sim$).

Order	**Algorithm**	**Ranking**	**Holm's test**
1	GRASP-R_RCL-0.05-Exterior-6-RFI	7.57	Control algorithm
2	GRASP-R_RCL-0.05-Exterior-12-RFI	8.04	$\sim$
3	GRASP-R_RCL-0.05-Interior-12-RFI	8.52	$\sim$
4	GRASP-R_RCL-0.1-Exterior-12-RFI	8.89	$\sim$
5	GRASP-R_RCL-0.1-Interior-12-RFI	9.34	$\sim$
6	GRASP-R_RCL-0.1-Interior-6-RFI	9.70	$\sim$
7	GRASP-R_RCL-0.1-Exterior-6-RFI	10.54	$\sim$
8	GRASP-R_RCL-0.2-Interior-6-RFI	10.66	$\sim$
9	GRASP-R_RCL-0.05-Alt_ePR-12-RFI	10.70	$\sim$
10	GRASP-R_RCL-0.05-Interior-6-RFI	10.83	$\sim$
11	GRASP-R_RCL-0.05-Alt_ePR-6-RFI	11.17	$\sim$
12	GRASP-R_RCL-0.1-Alt_ePR-6-RFI	11.37	$\sim$
13	GRASP-R_RCL-0.2-Exterior-12-RFI	11.67	$\sim$
14	GRASP-R_RCL-0.2-Exterior-6-RFI	12.11	$\sim$
15	GRASP-R_RCL-0.2-Interior-12-RFI	12.30	$\sim$
16	GRASP-R_RCL-0.1-Alt_ePR-12-RFI	13.41	$\sim$
17	GRASP-R_RCL-0.2-Alt_ePR-6-RFI	14.30	$\sim$
18	GRASP-R_RCL-0.2-Alt_ePR-12-RFI	14.70	$\sim$
19	GRASP-B_RCL-0.05-Exterior-6-RFI	23.18	+
...	...	...	+
37	GRASP-R_RCL-0.05-Alt_ePR-12-FI	39.11	+
...	...	...	+
72	GRASP-B_RCL-0.2-Interior-6-FI	66.94	+

Table 1: Mean Ranking of GRASP+PR configurations and Holm's test results

As shown in Table 1, the best performing GRASP+PR configuration uses

the R_RCL contruction procedure with $\alpha = 0.05$, ePR with 6 elements in the elite set, and restricted local search RFI. Holm's test detects statistically significant differences between the control algorithm (GRASP-R_RCL-0.05-Exterior-6-RFI) and those algorithms that use B_RCL instead of R_RCL to build the RCL (first occurrence of such configuration is at position 19 in the rankings list) and that employ the first interchange local search (FI) instead of the restricted first interchange one (RFI). Therefore, both the R_RCL scheme for the management of the RCL and the novel RFI local search appear to be two key components in the proposed algorithm.

Table 2 summarises the results of applying the Wilcoxon test, with p-value = 0.05, to compare the results of the control algorithm and the algorithms for which Holm's test does not detect significant differences. The last column indicates whether GRASP-R_RCL-0.05-Exterior-6-RFI performs statistically better (+), worse (-), or without significant differences ($\sim$) to its competitor.

GRASP-R_RCL-0.05-Exterior-6-RFI vs	R^+	R^-	**Sig. differences?**
GRASP-R_RCL-0.05-Exterior-12-RFI	1638	1932	$\sim$
GRASP-R_RCL-0.05-Interior-12-RFI	1834	1736	$\sim$
GRASP-R_RCL-0.1-Exterior-12-RFI	2192	1378	$\sim$
GRASP-R_RCL-0.1-Interior-12-RFI	1948	1622	$\sim$
GRASP-R_RCL-0.1-Interior-6-RFI	2067	1503	$\sim$
GRASP-R_RCL-0.1-Exterior-6-RFI	2192	1378	$\sim$
GRASP-R_RCL-0.2-Interior-6-RFI	2204	1366	$\sim$
GRASP-R_RCL-0.05-Alt_ePR-12-RFI	2287	1283	+
GRASP-R_RCL-0.05-Interior-6-RFI	2429	1141	+
GRASP-R_RCL-0.05-Alt_ePR-6-RFI	2405	1165	+
GRASP-R_RCL-0.1-Alt_ePR-6-RFI	2354	1216	+
GRASP-R_RCL-0.2-Exterior-12-RFI	2475	1095	+
GRASP-R_RCL-0.2-Exterior-6-RFI	2603	997	+
GRASP-R_RCL-0.2-Interior-12-RFI	2558	1012	+
GRASP-R_RCL-0.1-Alt_ePR-12-RFI	2833	737	+
GRASP-R_RCL-0.2-Alt_ePR-6-RFI	2969	601	+
GRASP-R_RCL-0.2-Alt_ePR-12-RFI	2801	769	+

Table 2: Wilcoxon's test results (critical value is 1345)

From results in Table 2, we notice that GRASP-R_RCL-0.05-Exterior-6-RFI with the ePR scheme, which explores areas beyond the initial solutions, improves the algorithm behaviour compared to the Alt_ePR approach, which combines exterior and interior PR. If we compare the performance of the approaches using the interior and exterior PR scheme, we can notice that GRASP-R_RCL-0.05-Exterior-6-RFI seems to outperform all the approaches using interior PR, although sometimes the Wilcoxon's test does not find statistically significant differences, due to R^+ value is always higher than R^-.

4.2.2. GRASP+PR performance analysis

In this section, we perform an in-depth study of the influence of the different components and parameters on the behaviour of the proposed GRASP+PR algorithm, focusing on the three key components identified in Section 4.2.1: the ePR scheme, the mechanism to choose the elements included in the RCL and the local search procedure.

First, we compare the behaviour of the GRASP+PR algorithm depending on the PR scheme. For this purpose, we show in Figure 7 the results for all the instances, grouped by number of vectors, in terms of the difference (Δ_R) between the algorithm using the ePR scheme and the algorithm with interior PR and alt_ePR schemes. In order to calculate Δ_R, we have applied the following formula:

$$\Delta_R = \frac{R_{Ex} - R}{R_{Ex}} \tag{13}$$

where R_{Ex} is the result of the algorithm using ePR and R is the result of the algorithm with other PR schemes. This means that negative Δ_R values indicate an advantage for the algorithm using ePR.

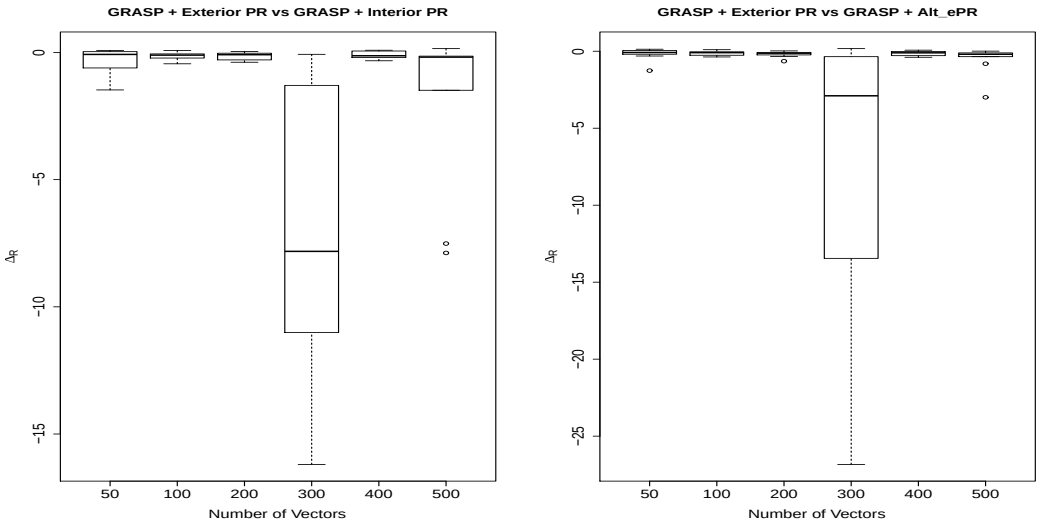

Figure 7: Difference (Δ_R) between GRASP + ePR vs GRASP + interior PR / alt_ePR

Figure 7 shows that the ePR scheme produces better quality solutions for the instances of all types, since the Δ_R value is less than 0 for all the instances

independently of the number of vectors. Interestingly, the advantage of using ePR over interior or alt_ePR is greater for instances with an intermediate number of vectors (300) and, in addition with regards to interior PR, for the instances with the largest number of vectors (500).

At this point, it is interesting to study the benefits of the optimized calculation of the elite set diversity presented in Section 3.4. For that purpose, we have measured the total time taken by the path relinking phase with and without the optimized calculation and have computed the improvement obtained as follows:

$$\Delta_{PR}(\%) = \frac{PR_{wi} - PR_i}{PR_{wi}} \cdot 100 \tag{14}$$

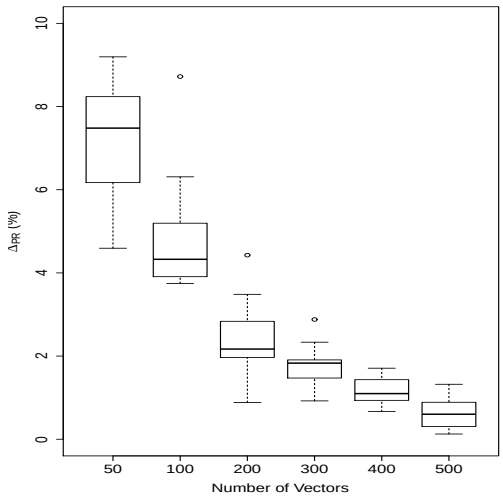

Figure 8: Improvement obtained by optimizing the computation of the elite set diversity

where PR_{wi} is the total time required for the PR phase without the improvement of the diversity calculation and PR_i is that of the PR phase with the improvement. This means that positive Δ_{PR} values indicate a reduction of the time needed for the PR phase when applying the optimized calculation of the diversity. Figure 8 shows the value of $\Delta_{PR}(\%)$ grouped by the number of vectors. As we can observe, performance improvement during the PR phase reaches values around 8% on average for instances with 50 vectors. This value gradually decreases as we increase the size of the instances due to

the relative importance of the diversity calculation is less within the full PR procedure. Notice that the complexity of the diversity calculation is mostly related to the number of dimensions of the vectors in the elite set. However, it is important to highlight that regardless of the number of vectors that enhancement always brings a positive result.

Secondly, we analyse the behaviour of our proposal depending on the mechanism used to select the elements included in the RCL. For this purpose, we compare the behaviour of different relevant configurations along the whole run. We employ convergence graphs that summarise the behaviour of the algorithms on the tuning instances set over 10 independent runs. In order to build the convergence graphs, we have performed the following steps:

- Every result is normalised to lie in the interval $[0, 1]$ by considering the highest and lowest values achieved by the algorithms on each test problem.
- The average per time instant, over the set of problem instances and 10 independent runs, is computed from previous normalised results. The result is referred to by *Avg. Normalized Fitness* in Figures 9 and 10.

Figure 9 shows the convergence graph of the best performing configuration found in the above section, GRASP-R_RCL-0.05-Exterior-6-RFI, and three more variants. The first configuration is mainly characterised by the use of the R_RCL scheme, ePR, and RFI local search. To see the effect of using the R_RCL scheme, we also plot the convergence graphs of configurations obtained from the best one by changing that component and its related parameter α. In particular, we consider one configuration with B_RCL and three configurations with different values for the parameter α.

The main difference between B_RCL and R_RCL is the number of solution elements evaluated at each step. B_RCL assesses all available solution elements while R_RCL only evaluates some of them randomly chosen. We find that the exhaustive exploration performed by B_RCL does not provide benefits over R_RCL scheme in terms of solution quality throughout the execution of the algorithm, as can be seen in Figure 9. In fact, at the end of the execution, the quality of the solutions obtained by the configuration with B_RCL is the worst among the algorithms plotted in Figure 9.

With respect to the behaviour of our proposal depending on the value of α, we note that, from the very beginning of the execution, the best results

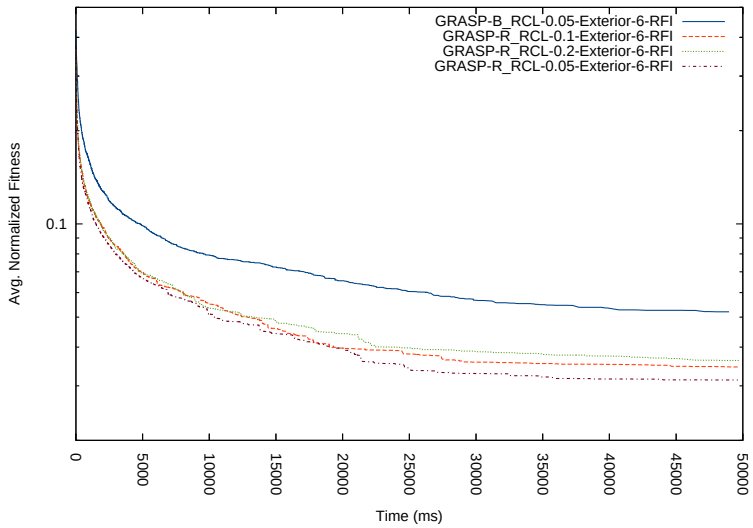

Figure 9: Convergence graphs of GRASP+PR configurations considering diferent schemes to manage the RCL and α values

are achieved by the configuration with the lowest α value, that is, at each step we are evaluating only a few components for inclusion in the current partial solution. In fact, the worst results at the end of the execution, as can be seen in Figure 9, are obtained with the higher α value (0.2). This result is in line with our finding about the mechanism to select the elements included in RCL. The reduction of the computational time spent during the construction phase allows GRASP+PR to perform more iterations, which seems to be benefitial for improving the quality of the solutions along the whole execution.

Then, we analyse the influence of the local search method employed during the local improvement phase. To do this, the convergence graphs of the best performing algorithm and a variant with FI local search are presented in Figure 10.

The results shown in Figure 10 disclose that GRASP+PR with RFI outperforms GRASP+PR with FI from the very beginning of the execution. This shows that just the pre-selection of the candidate neighbours performed by RFI is suitable for this problem, regardless of the computational effort saving achieved. This faster local improvement carried out by RFI pays off as execution progresses, increasing the performance difference between the two algorithms.

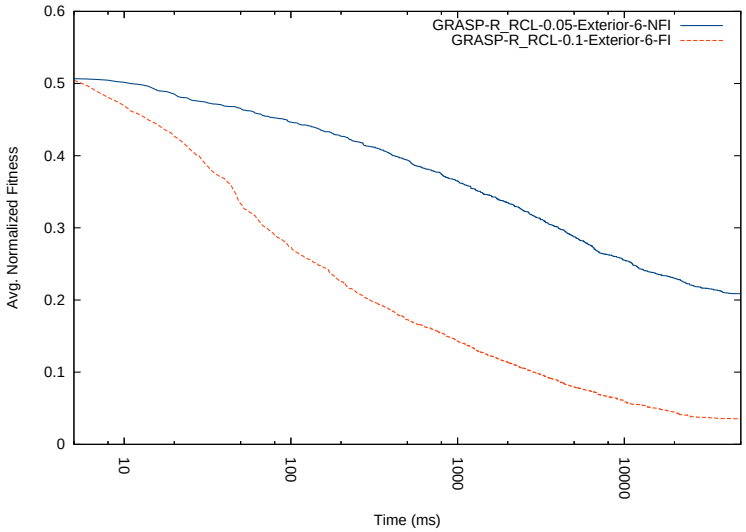

Figure 10: Convergence graphs of GRASP+PR configurations considering RFI and FI local search methods

Finally, we study the improvement obtained by using the delta evaluation in the local search procedure. With this aim, we measure the total time taken by the local search procedure with and without delta evaluation and compute the improvement obtained analogously as we did for the diversity calculation improvement during the path relinking phase:

$$\Delta_{LS}(\%) = \frac{LS_{wd} - LS_d}{LS_{wd}} \cdot 100 \tag{15}$$

where LS_{wd} is the total time required for the LS phase whithout delta evaluation and LS_i is that of the LS phase with delta evaluation. This means that positive Δ_{LS} values indicate a reduction of the time needed for the LS phase when using the delta evaluation. Figure 11 shows the value of $\Delta_{LS}(\%)$ grouped by the number of vectors. We can clearly see how the delta evaluation is a key element to improve the efficiency of the proposed algorithm, achieving reductions of up to 70% in the runtime of the LS phase. As expected, these improvements are greater as the number of vectors increases.

Taking into account the conclusions drawn from Sections 4.2.1 and 4.2.2, we base all comparisons in the following sections on the GRASP+PR configuration with R_RCL, $\alpha = 0.05$, ePR with size of the elite set equal to 6, and RFI local search. This configuration will be referred to from now on just as

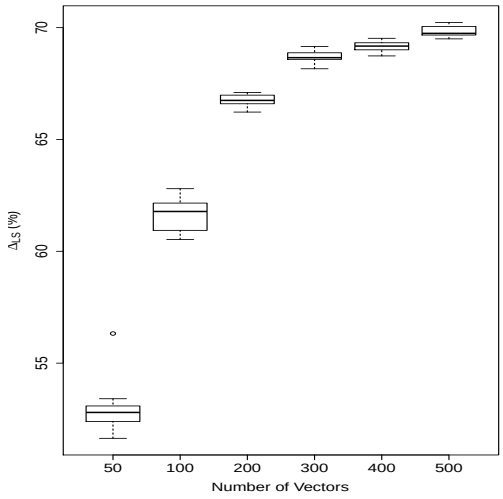

Figure 11: Improvement obtained by using the delta evaluation during the LS phase

GRASP+ePR.

4.3. Comparison with state-of-the-art methods

In this section, we compare the GRASP+PR method with the results of the integer programming model and the variable neighbourhood search (VNS) described in Section 2. VNS is the current state-of-the-art algorithm for the problem along with the EM method proposed in the same paper [28]. The VNS method is noteworthy for its simplicity, and establishes itself as an effective tool for tackling hard problems [43]. To solve the integer programming model, we have used CPLEX. Moreover, we have reimplemented VNS in C++ following the guideliness in the original publication, allowing us to perform a comparison under identical execution conditions such as computational time limit, computer features, etc. The parameters used for VNS are those recommended in its original publication. Emphasize that the experimentation in this section has been done on a different subset of instances from the above section, as explained in Section 4.1.

We have selected the GRASP+PR configuration that achieved the best results in the previous sections (GRASP-R_RCL-0.05-Exterior-6-RFI), which will be noted simply as GRASP+ePR from now on. All the algorithms but CPLEX were executed 10 times on each problem instance, alloting a computation time limit of 600 seconds for each execution. CPLEX was executed

only once due to its deterministic nature. CPLEX settings were established with a maximum memory base of 400 MB of RAM, allowing disk-backup, compression once the 400 MB limit was exceeded, and using a single thread. The computer features are described in Section 4.1.

Table 3 summarises the results achieved by GRASP+ePR and its main competitor, VNS. We compute for each instance type the average objective function value obtained (*Avg.*), the average relative deviation between the best solution value found by the method and the best known value (*Dev.*), and the number of executions for which the value of the best solution obtained by a given method matches the best known value (*#Best*). In addition, detailed results of all the algorithms on each instance are provided at `http://sci2s.ugr.es/MDTWNP/`

From results in Table 3, we highlight the following elements:

- GRASP+ePR attains the best overall results, obtaining the best *Avg.* value for 32 out of 42 instance types.
- GRASP+ePR outperforms VNS in the vast majority of instance types with more than 100 vectors. In fact, for instances with 500 vectors, GRASP+ePR outperforms VNS independently of the number of dimensions.
- *Dev.* values achieved by VNS in certain large instances are very high (for example in instances with 500 vectors and 3, 4, or 5 dimensions), which shows that VNS's results might be highly influenced by the random initial solution. On the contrary, GRASP+ePR present more stable *Dev.* values over the different types of instances.

To determine whether the observed differences in the above table are statistically significant, we have employed Wilcoxon's test. Table 4 shows the results of applying Wilcoxon's test with p-value = 0.05 to compare GRASP+ePR and its competitors (VNS and CPLEX). These results clearly show that GRASP+ePR outperforms both competitors, yielding statistically significant differences.

5. Conclusions

We have proposed a new metaheuristic for the MDTWNP problem that joins GRASP with a recently proposed Exterior Path Relinking method

		GRASP+ePR			VNS		
n	**d**	**Avg.**	**Dev.**	**#Best**	**Avg.**	**Dev.**	**#Best**
	2	**4.629**	**0.527**	**3**	7.146	1.752	0
	3	155.570	0.043	2	**154.760**	**0.031**	**4**
	4	**895.444**	**0.005**	**2**	896.665	**0.008**	1
50	5	2890.814	0.122	1	**2742.916**	**0.034**	**3**
	10	**16454.528**	**0.114**	**3**	17664.979	0.195	2
	15	**32144.936**	**0.177**	**12**	35898.863	0.329	1
	20	56729.249	0.117	6	**54801.555**	**0.076**	**9**
	2	**2.706**	**712.142**	**3**	6.149	2147.785	0
	3	**219.617**	**0.071**	**2**	225.352	0.129	**2**
	4	**839.822**	0.008	1	856.762	**0.003**	**2**
100	5	3193.905	0.188	1	**2992.667**	**0.106**	**2**
	10	16141.921	0.245	0	**14917.637**	**0.147**	**3**
	15	**32937.013**	**0.056**	**2**	34265.192	0.112	1
	20	55026.754	0.174	0	**52089.672**	**0.111**	**3**
	2	**2.643**	**5.719**	1	4.169	8.726	**2**
	3	**108.157**	**1.179**	1	155.907	41.141	**2**
	4	**768.598**	**0.772**	1	799.298	73.298	**2**
200	5	2122.008	**1.143**	0	**1853.161**	71.802	**3**
	10	**17445.241**	**0.104**	**2**	18320.442	0.163	1
	15	**32256.267**	**0.136**	**4**	33615.853	0.183	1
	20	**49214.221**	**0.101**	**3**	52598.723	0.179	0
	2	**3.597**	**3.437**	**3**	7.195	7.244	1
	3	202.577	**0.263**	0	**198.055**	0.264	**3**
	4	963.191	0.221	1	**958.827**	**0.209**	**2**
300	5	**1663.231**	**0.068**	**2**	2078.379	0.524	1
	10	**17159.752**	**0.093**	2	17160.718	0.094	**4**
	15	**31988.06**	**0.088**	**3**	33698.138	0.148	2
	20	**49556.532**	**0.213**	**2**	50771.759	0.246	1
	2	**3.328**	**3.561**	1	8.056	10.373	**2**
	3	223.126	0.931	1	**210.626**	**0.602**	**2**
	4	**909.013**	**0.170**	0	944.2	0.212	**1**
400	5	**1676.304**	**0.171**	**2**	2036.273	0.583	1
	10	**16075.187**	**0.171**	0	16258.54	0.233	**4**
	15	**31388.099**	**0.128**	1	33007.689	0.183	**2**
	20	**48794.137**	**0.131**	**3**	49992.424	0.158	0
	2	**2.412**	**7.209**	0	5.207	18.031	**2**
	3	**67.787**	**1.932**	**2**	106.619	21.662	1
	4	**641.744**	**0.707**	**2**	893.791	51.542	**2**
500	5	**1852.538**	**0.341**	1	12242.995	94.136	**3**
	10	**16106.188**	**0.120**	**2**	16162.858	0.129	**2**
	15	**30684.108**	**0.132**	1	32823,764	0.211	**2**
	20	**47524.461**	**0.132**	**2**	49250.618	0.173	1

Table 3: Summarised results of GRASP+ePR and VNS

(ePR), which examines areas beyond those embraced by interior PR solutions. The exterior form of PR proves to be a key component of our GRASP+ePR algorithm, enabling it to obtain significantly improved solutions for the MDTWNP problem.

We have also proposed a new restricted local search procedure for carrying out local descent for MDTWNP, which limits neighbours considered to lie

GRASP+ePR vs	R^+	R^-	Sig. differences?
VNS	5589	2412	+
CPLEX	7992	9	+

Table 4: Wilcoxon's test results between GRASP+ePR and CPLEX/VNS (critical value is 3195)

close to each other, to produce a smooth descent and a faster form of local search, and which proves superior to the previous local descent approach used for MDTWNP.

Our computational experiments have compared two different schemes to choose the elements included in the restricted candidate list RCL. We have shown that the scheme that does not perform an exhaustive evaluation of all solution components at each iteration substantially improves the performance of the GRASP+ePR algorithm for this problem due to its ability to run faster and hence examine more solutions in total. Finally, we have compared the results of GRASP+ePR with those of two chief methods: (1) CPLEX applied to an integer linear programming formulation and (2) VNS, which is the current state-of-the-art algorithm for the MDTWNP problem. This study has demonstrated that the GRASP+ePR algorithm significantly outperforms both approaches.

We mention two avenues for further research: first, to incorporate additional elements from interior PR (such as truncated PR and randomised PR) into ePR; and second, to build new hybrid metaheuristics for the MDTWNP problem combining the proposed GRASP+ePR with other salient metaheuristics such as tabu search and strategic oscillation.

Acknowledgements

This work was supported by the research projects TIN2012-37930-C02 and TIN2012-35632-C02. We are thankful to Jelena Kojić for providing the instances used in [1].

[1] J. Kojić, Integer linear programming model for multidimensional two-way number partitioning problem, Computers & Mathematics with Applications 60 (8) (2010) 2302–2308.

[2] S. Mertens, Phase transition in the number partitioning problem, Physical Review Letters 81 (1998) 4281–4284.

[3] J. P. Pedroso, M. Kubo, Heuristics and exact methods for number partitioning, European Journal of Operational Research 202 (1) (2010) 73 – 81.

[4] H. Bauke, S. Franz, S. Mertens, Number partitioning as random energy model, Journal of Statistical Mechanics: Theory and Experiment (2004) 1–2.

[5] M. Koyutörk, Hypergraph based declustering for multi-disk databases (2000).

[6] T. Feo, M. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Operations Research Letters 8 (2) (1989) 67–71.

[7] T. Feo, M. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (2) (1995) 109–133.

[8] F. Glover, Tabu search and adaptive memory programing advances, applications and challenges, in: Interfaces in Computer Science and Operations Research, Kluwer, 1996, pp. 1–75.

[9] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.

[10] F. Glover, M. Laguna, R. Martí, Fundamentals of scatter search and path relinking, Control and Cybernetics 39 (2000) 653–684.

[11] F. Glover, Exterior path relinking for zero-one optimization, International Journal of Applied Metaheuristic Computing 5 (3) (2014) 1 – 8.

[12] A. Duarte, J. Sánchez-Oro, M. G. Resende, F. Glover, R. Martí, Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization, Information Sciences 296 (2015) 46 – 60.

[13] M. Laguna, R. Marti, GRASP and path relinking for 2-layer straight line crossing minimization, Informs Journal on Computing 11 (1) (1999) 44–52.

[14] R. Aiex, S. Binato, M. Resende, Parallel GRASP with path-relinking for job shop scheduling, Parallel Computing 29 (4) (2003) 393–430.

[15] R. Aiex, M. Resende, P. Pardalos, G. Toraldo, GRASP with path relinking for three-index assignment, Informs Journal on Computing 17 (2) (2005) 224–247.

[16] C. Ribeiro, E. Uchoa, R. Werneck, A hybrid GRASP with perturbations for the steiner problem in graphs, Informs Journal on Computing 14 (3) (2002) 228–246.

[17] F. Rodriguez, C. Blum, C. Garcıa-Martınez, M. Lozano, Grasp with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times, Annals of Operations Research 201 (1) (2012) 383–401.

[18] R. Martí, V. Campos, M. G. Resende, A. Duarte, Multiobjective GRASP with path relinking, European Journal of Operational Research 240 (1) (2015) 54 – 71.

[19] R. E. Korf, A complete anytime algorithm for number partitioning, Artificial Intelligence 106 (2) (1998) 181–203.

[20] N. Karmarkar, R. M. Karp, The differencing method of set partitioning, Tech. Rep. CSD-83-113, University of California, Berkeley (2000).

[21] E. Horowitz, S. Sahni, Computing partitions with applications to the knapsack problem, Journal of the ACM 21 (2) (1974) 277–292.

[22] D. S. Johnson, C. R. Aragon, L. A. McGeoch, C. Schevon, Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning, Operations Research 39 (3) (1991) 378–406.

[23] B. Alidaee, F. Glover, G. Kochenberger, C. Rego, A new modeling and solution approach for the number partitioning problem, Journal of Applied Mathematics and Decision Sciences 9 (2) (2005) 135–145.

[24] W. Ruml, J. Ngo, J. Marks, S. Shieber, Easily searched encodings for number partitioning, Journal of Optimization Theory and Applications 89 (2) (1996) 251–291.

[25] R. Berretta, C. Cotta, P. Moscato, Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm: Results on the number partitioning problem, Springer US, 2004, pp. 65–90.

[26] M. F. Argello, T. A. Feo, O. Goldschmidt, Randomized methods for the number partitioning problem, Computers and Operations Research 23 (2) (1996) 103 – 111.

[27] P. C. Pop, O. Matei, A memetic algorithm approach for solving the multidimensional multi-way number partitioning problem, Applied Mathematical Modelling 37 (22) (2013) 9191 – 9202.

[28] J. Kratica, J. Kojić, A. Savić, Two metaheuristic approaches for solving multidimensional two-way number partitioning problem, Computers and Operations Research 46 (0) (2014) 59 – 68.

[29] M. G. C. Resende, R. M. A. Silva, GRASP: Greedy Randomized Adaptive Search Procedures, John Wiley & Sons, Inc., 2010.

[30] M. Resende, R. Marti, M. Gallego, A. Duarte, GRASP and path relinking for the max-min diversity problem, Computers & Operations Research 37 (3) (2010) 498 – 508.

[31] C. R. Reeves, T. Yamada, Genetic algorithms, path relinking, and the flowshop sequencing problem, Evolutionary Computation 6 (1) (1998) 45–60.

[32] F. Glover, M. Laguna, R. Marti, Scatter search and path relinking: Advances and applications, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Vol. 57 of International Series in Operations Research & Management Science, 2003, pp. 1–35.

[33] S. Ho, M. Gendreau, Path relinking for the vehicle routing problem, Journal of Heuristics 12 (1-2) (2006) 55–72.

[34] C. C. Ribeiro, M. G. Resende, Path-relinking intensification methods for stochastic local search algorithms, Journal of Heuristics 18 (2) (2012) 193–214.

[35] M. Resende, C. Ribero, A GRASP with path-relinking for private virtual circuit routing, Networks 41 (2003) 104–114.

[36] R. Martí, F. Sandoya, GRASP and path relinking for the equitable dispersion problem, Computers & Operations Research 40 (12) (2013) 3091 – 3099.

[37] S. Garcia, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization, Journal of Heuristics 15 (2008) 617–644.

[38] R. Iman, J. Davenport, Approximations of the critical region of the Friedman statistic, in: Communications in Statistics, 1980, pp. 571–595.

[39] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

[40] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.

[41] I. G. del Amo, D. A. Pelta, J. R. González, A. D. Masegosa, An algorithm comparison for dynamic optimization problems, Applied Soft Computing 12 (10) (2012) 3176 – 3192.

[42] B. Naderi, R. Ruiz, A scatter search algorithm for the distributed permutation flowshop scheduling problem, European Journal of Operational Research 239 (2) (2014) 323 – 334.

[43] P. Hansen, N. Mladenović, J. A. Moreno Prez, Variable neighbourhood search: methods and applications, Annals of Operations Research 175 (1) (2010) 367–407.