# Heuristics and Metaheuristics for the Maximum Diversity Problem

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

MICAEL GALLEGO
Departamento de Ciencias de la Computación
Universidad Rey Juan Carlos, Spain
Micael.Gallego@urjc.es

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

EDUARDO G. PARDO
Departamento de Ciencias de la Computación
Universidad Rey Juan Carlos, Spain
Eduardo.Pardo@urjc.es

**ABSTRACT**

This paper presents extensive computational experiments to compare 10 heuristics and 20 metaheuristics for the maximum diversity problem (MDP). This problem consists of selecting a subset of maximum diversity from a given set of elements. It arises in a wide range of real-world settings and we can find a large number of studies, in which heuristic and metaheuristic methods are proposed. However, probably due to the fact that this problem has been referenced under different names, we have only found limited comparisons with a few methods on some sets of instances.

This paper reviews all the heuristics and metaheuristics for finding near-optimal solutions for the MDP. We present the new benchmark library MDPLIB, which includes most instances previously used for this problem, as well as new ones, giving a total of 315. We also present an exhaustive computational comparison of the 30 methods on the MDPLIB. Non-parametric statistical tests are reported in our study to draw significant conclusions.

Original draft: January 5, 2010
Revision: April 21, 2011

# 1 Introduction

The maximum diversity problem (MDP) consists of selecting a subset of *m* elements from a set of *n* elements in such a way that the sum of the distances between the chosen elements is maximized. The MDP presents a challenge to heuristic methods and it has appeared in previous studies under many different names, such as:

- *maxisum*, or *max-avg, dispersion* (Kuby 1987; Ravi et al., 1994),
- *p-dispersion, or p-dispersion-sum* (Erkut, 1990; Kincaid and Yellin, 1993),
- *edge-weighted, or remote clique* (Macambira and Souza, 2000; Chandra and Halldorsson, 2001),
- *maximum edge-weighted subgraph* (Macambira 2002),
- *dense k-subgraph* (Feige et al., 2001),
- *p-defense, or p-defense-sum* (Moon and Chaudhry, 1984; Cappanera, 1999), or
- *equitable dispersion* (Prokopyev et al., 2009).

A large number of heuristics and metaheuristics have been proposed for this problem; however, probably due to its different names, only partial comparisons with a few methods have been previously published. The most recent one is presented in Aringhieri and Cordone (2011), where the authors also proposed different metaheuristics based on the Tabu Search and Variable Neighborhood Search methodologies and compare them with the 4 previous algorithms identified to be the best. As mentioned by the authors, the use of different computers makes a fair comparison a very hard task. In this paper we present extensive computational comparison with 10 heuristics and 20 metaheuristics previously published in which all of them are executed in the same computer for the same CPU time. Furthermore, we propose new large instances that are hard to solve by recent metaheuristics. Our experimental study establishes the state-of-the-art methods for this problem, and identifies the key search elements and strategies that permit them to obtain high quality solutions. Table 1 summarizes some of the main applications of this problem.

| ***Context*** | **Reference** | **Description** |
|---|---|---|
| Location | Erkut and Neuman (1991) | Locating facilities according to distance, accessibility or impacts |
| Ecological systems | Pearce (1987) | Establishing viable systems relies crucially on considerations of diversity maximization |
| Medical treatments | Kuo et al. (1993) | Combating diseases is enhance by programs with mote diverse lines of defense |
| Genetics | Porter et al. (1975) | In animal and plant genetics to obtain new varieties by controlled breeding |
| Ethnicity | Swierenga (1977) | Historical perspective |
| | McConnell (1988) | Promotion of ethnic diversity among immigrants |
| Product design | Glover et al. (1998) | Maximizing product diversity |

**Table 1**. MDP applications

Kuo et al. (1993) introduced the following straightforward formulation of the MDP as a quadratic binary problem and proposed several improvements to tackle small sized instances. In this formulation, the binary variables $x_i$ take the value 1 if element $i$ is selected and 0 otherwise, $i = 1, ..., n$. The distance values between elements $i$ and $j$ is represented with $d_{ij}$, $i, j = 1, \ldots, n$. The authors show with this formulation that the clique problem, which is known to be NP-hard, is reducible to the MDP.

$$\text{Maximize} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j$$

$$\text{Subject to} \quad \sum_{i=1}^{n} x_i = m$$

$$x_i = \{0, 1\} \qquad 1 \le i \le n$$

Martí et al. (2010) showed that the three linear integer formulations proposed in Kuo et al. (1993) are only able to solve small problems with Cplex 8.0 ($n \le 15$, and $15 \le n \le 30$ for some values of $m$). This is why they proposed a branch and bound algorithm to provably solve medium sized problems ($n \le 50$, and $50 \le n \le 150$ for some values of $m$). Independently, Erkut (1990) and Pisinger (2006) also proposed branch and bound algorithms: the former is limited to solve small problems while the latter is also able to solve medium sized problems.

We have identified two categories on the MDP methods. Early procedures, called heuristics, are based on simple selection rules, implemented as constructive procedures coupled, in some cases, with local search methods. These procedures reflect the goal of researchers of quickly obtaining solutions of reasonable quality. Recent developments, however, have shown that complex metaheuristic approaches can be successfully applied to the MDP. These methods usually outperform the simple heuristics, obtaining solutions with a significant better quality, although they require longer running times. An alternative approach to combinatorial optimization problems is given by Discrete Hopfield Neural Networks. Wang et al. (2009) hybridized this methodology with an Estimation of Distribution Algorithm for the MDP. We do not include this method in our comparison because we restrict it to heuristics and metaheuristics, which on the other hand, outperform this kind of procedures in terms of solution quality and CPU time.

In this paper we undertake to explore the behavior of the heuristic and metaheuristics developed for the MDP. With this goal in mind, we have collected all the previous instances reported for this problem (discarding the trivial ones) and we have generated some new ones (large and hard instances). The entire set of instances, which will be called the MDPLIB (`http://www.optsicom.es/mdp`), includes 315 cases with a wide range of sizes and characteristics. In the following sections we first provide short descriptions of the 10 heuristics proposed for the MDP. We then go on to present a summarized description of 20 metaheuristics. These sections are followed by results of our computational testing on the MDPLIB including statistical analysis. The paper finishes with the associated conclusions.

## 2 Heuristics

In this section we review 8 construction and 2 improvement heuristics for the MDP. The construction methods obtain a solution (i.e., a selection of *m* elements) from scratch, adding or removing iteratively one element at each step. The improvement heuristics start from the solution obtained with a construction method and iteratively improve it by performing exchanges usually referred to as *moves*. The different types of selection strategies characterize the various heuristics.

To describe these methods we will use the following notation. Let $S = \{s_i : i \in N\}$ be a set of elements where $N = \{1, 2,.., n\}$ is the set of indexes. Let $d_{ij}$ be the distance between elements $s_i$ and $s_j$, and let $m < n$ be the desired size of the diverse set.

### 2.1 ErkC - Erkut and Neuman construction method

In Erkut (1990) a constructive and an improvement method are proposed. The constructive heuristic, `ErkC`, generates a solution by first selecting the two elements with the largest distance between them. Then, in the following iterations, the element that increases the objective function the most is added to the partial solution.

### 2.2 GhC - Ghosh construction method

Ghosh (1996) proposed a multi-start algorithm. It basically consists of a construction phase and a local search post-processing. The construction phase, `GhC`, performs *m* iterations to obtain a solution. In each iteration, one element is selected according to an estimation of its contribution to the final value of the solution. As mentioned by other authors (see for instance Silva et al., 2004) this multi-start method produces results of relatively low quality. However, as will be shown, the local search phase has been used in several algorithms.

### 2.3 C2 and D2 - Glover et al. construction methods

In Glover et al. (1998) four constructive heuristics are proposed. The constructive methods `C1` and `D1`, only applicable in Euclidean sets, are based on the concept of the center of gravity of a set. The authors also proposed `C2` and `D2`, applicable in any set, in which the distance between an element $s_i$ and a set $X = \{s_j : j \in I\}$ is defined as:

$$d(s_i, X) = \sum_{j \in I} d(s_i, s_j)$$

`C2` first randomly selects an initial element. Then, it selects at each step, the element with the maximum distance to the already selected elements *Sel*. The method finishes when *m* elements have been selected. (Note that `C2` is the same method as the `ErkC` described above except for the selection of the first element.) Symmetrically, starting with all the elements selected, `D2` unselects the element with the minimum distance to the set *Sel* of selected elements at each step. It concludes when *n-m* elements have been unselected.

### 2.4 STA - Palubeckis construction method

Palubeckis (2007) proposed STA, a constructive heuristic based on the idea of performing a steepest ascent from a point within the *n*-dimensional unit cube. The procedure fixes one variable $x_i$ at either 0 or 1 at each step, defining a trajectory through the vertices in the cube. The method finishes when it reaches a feasible solution of the MDP.

### 2.5 KLD, KLDv2 and MDI - Silva et al. construction methods

In Silva et al. (2004) we can find three constructive methods: KLD, KLDv2 and MDI. KLD initially estimates the contribution of an element to any solution. This estimation is obtained as the sum of the *k* larger distances between the element and the other non-selected elements. At each iteration, a restricted candidate list *RCL* is formed with the *k* best-estimated elements, and the algorithm randomly selects one to be added to the partial solution under construction. The method finishes when *m* elements have been selected. The parameter *k* is set with a reactive mechanism to avoid its offline tuning.

KLDv2 is an improved version of KLD in which the *RCL* is built using an adaptive procedure. MDI is similar to KLDv2 with the only difference in the way that the estimation of contribution is calculated. The estimation in KLDv2 is calculated with the *k* larger distances between an element and the other non-selected elements. In MDI, the estimation is calculated as the sum of the largest distances between an element and the other non-selected elements and the distances with the selected elements.

### 2.6 BLS - Best improvement local search

The local search method BLS proposed first in Erkut (1990) and later in Ghosh (1996). It implements a straightforward local search heuristic based on performing the best available exchange. Exchanges in this context consist of replacing one selected element with an unselected one. The procedure scans the set of selected elements *Sel* (|*Sel*|=*m*) in search of the exchange that gives the largest increase of the objective function (i.e., thus maximizing the diversity). The method performs the best available exchange in each iteration until no further improvement is possible.

### 2.7 I_LS - Improved local search

Duarte and Martí (2007) adapted the so-called *first strategy* to the local search method for the MDP. Their improved local search, I_LS, is also based on exchanges, like the BLS method above; however, instead of searching for the best exchange at each iteration, it performs two stages. In the first one, I_LS selects the element $s_{i*}$ with the lowest contribution to the value of the current solution. Then, in the second stage, the method performs the first improving move to replace $s_{i*}$ (i.e., instead of scanning the whole set of *unselected elements* searching for the best exchange associated with $s_{i*}$, it performs the first improving exchange without examining the remaining unselected elements). If there is no improving move exchanging $s_{i*}$, the method resorts to the next element with the lowest contribution and so on. This improved local search method performs iterations until no further improvement is possible.

# 3 Metaheuristics

In the last two decades a series of methods have appeared under the name of metaheuristics, which aim to obtain better results than those obtained with *traditional heuristics*. The term metaheuristic was coined by Glover (1986) to refer to a set of methodologies conceptually ranked *above* the heuristics in the sense that they guide their design. Thus, facing an optimization problem, we can employ any of these methodologies to design a specific algorithm for computing an approximate solution. Some of the most promising metaheuristic methodologies have been applied to the MDP totalizing 20 different algorithms. We shall review them in the following section.

## 3.1 SA – Simulated Annealing

Simulated annealing (Kirkpatrick et al. 1983) proceeds in the same way as ordinary local search but incorporates some randomization in the move selection to avoid getting trapped in a local optimum by means of non-improving moves. These moves are accepted according to probabilities taken from the analogy with the *annealing process*.

Kincaid (1992) presented the SA algorithm for the MDP. In a given iteration, the SA method generates a random move (an exchange between a selected and an unselected element). If it is an improving move, it is automatically performed; otherwise, it may still be made with a certain probability (according to the Boltzmann distribution) using a parameter called *temperature*. The algorithm starts with an initial temperature equal to the largest distance value and this is reduced according to the factor *tfactr* = 0.89. For each temperature value, *sample_size*=10*n* moves are generated, evaluated and performed if this is the case. The SA method terminates when a maximum number of iterations *max_it* = 80 is reached.

## 3.2 GRASP – Greedy Randomized Adaptive Search Procedure

GRASP is a multi-start method (Feo and Resende, 1995) in which at each iteration we first apply a construction method and then an improvement method to find a local optimum (i.e., the final solution for that iteration). Silva et al. (2004) proposed three GRASP methods for the MDP formed respectively with the construction methods KLD, KLDv2 and MDI, described above, and the BLS as the improvement method. We will simply denote them as KLD+BLS, KLDv2+BLS and MDI+BLS.

Duarte and Martí (2007) proposed two GRASP methods randomizing the C2 and D2 construction methods (Glover et al. 1998) respectively. Both GRASP algorithms apply I_LS as the improvement method. We will call them as GRASP_C2+I_LS and GRASP_D2+I_LS respectively.

Santos et al. (2005) presented a hybrid method, GRASP_DM, combining GRASP with *data mining* techniques. The GRASP phase, based on the KLD method (Silva et al., 2004), is executed a certain number of iterations. Then, the data-mining process extracts patterns from an elite set of solutions that guide the following GRASP iterations. Solutions are represented by sets of items and patterns are defined as subsets of items that

occur in a relative large number of solutions. The process of mining these patterns is the well-known problem called frequent item set mining (Han and Kamber, 2000).

Silva et al. (2007) presented a hybrid method, GRASP_PR, combining GRASP with Path Relinking (Laguna and Martí, 1999). As in the hybrid method above, an elite set is populated with the solutions obtained with the application of a GRASP algorithm. Then, path relinking is applied from each solution in the elite set (*initial* solution) to the local optimum obtained in each new GRASP iteration (*guiding* solution). In this way, we create a path by adding elements in the guiding solutions to the initial solution (and dropping those not present in the guiding solution. The path relinking procedure terminates when the guiding solution is reached.

### 3.3 TS – Tabu Search

Tabu Search is a metaheuristic that guides a local search heuristic to explore the solution space beyond local optimality (Glover and Laguna, 1997). One of the main components of Tabu Search is its use of adaptive memory, which creates more flexible search behavior. This method is the core of what has been recently called as Adaptive Memory Programming.

Kincaid (1992) proposed a Tabu Search algorithm, K_TS, based on exchanging a selected element $s_i$ with an unselected element $s_j$. The algorithm starts from a random solution and improves it by exchanging elements. At each iteration, 10$n$ randomly generated exchanges are evaluated and the best admissible (non-tabu) move is performed. When a move is performed, the unordered pair ($s_i$, $s_j$) is labeled tabu for *tenure*=20 iterations. A selected move is admissible if it is not labeled tabu, or if its value improves upon the best-known solution (aspiration criterion). The method runs during *max_it*=65 iterations.

Macambira (2002) proposed a similar implementation of the tabu search methodology, called M_TS, to solve the MDP. The algorithm starts from a solution constructed with a method similar to C2 (Glover et al., 1998). Then, as in K_TS, it performs iterations exchanging elements. Specifically, at each iteration, M_TS performs the best admissible exchange between a selected and unselected element (even if it does not improve the solution). The tabu status is active during *tenure*=*m* iterations. The procedure stops after *num_iterations* without improvement with respect to the best solution found. Note that K_TS starts with a random solution while M_TS starts with a greedy constructed solution.

Duarte and Martí (2007) proposed a multi-start method, LS_TS, based on the tabu search methodology in both construction and improvement phases. The initial solution is obtained with a constructive heuristic Tabu_D2 derived from D2 (Glover et al., 1998) by adding memory structures to obtain diverse solutions as initial points. Once an initial solution has been constructed, LS_TS performs exchanges as the previous tabu search procedures. An iteration begins by randomly selecting a selected element $s_i$. The probability of selecting element $s_i$ is inversely proportional to the contribution of this

element to the objective value. The first improving move associated with $s_i$ is performed. (Note that if there is no improving move associated with $s_i$, the method performs the best one available, even if it is a non-improving move.) The selected elements become tabu-active for *sTenure*=0.28*m* iterations and the unselected ones become tabu-active for *uTenuere*=0.028*m*. The method performs iterations until the best solution found cannot be improved upon in *MaxIter*=0.1*n* consecutive iterations. Then, the search is re-initiated from a new initial solution. It is obtained by using the long-term memory information recorded during the search.

Palubeckis (2007) proposed an Iterated Tabu Search, `ITS`, that alternates tabu search with perturbation procedures. The algorithm is applied to the solution constructed with the `STA` method. The tabu search phase is similar to `K_TS` with *tenure* = 30. If the tabu search step finds a better solution than the previous best solution, a local search is applied to the new solution. The perturbation phase is applied in the last step of the method and basically consists of selecting (unselecting) a random number of unselected (selected) elements.

Aringhieri et al. (2008) presented `XTS`, a tabu search algorithm for the maximum diversity problem. It implements short and long term memory functions like `LS_TS`. Specifically, `XTS` first constructs a solution with a greedy algorithm similar to `ErkC` (Erkut, 1990). Then, in each iteration, it explores all possible exchanges between selected and unselected elements and performs the best one (this is different to `K_TS` and `M_TS` that only explores a subset of the neighborhood solutions). Regarding the tabu status `XTS`, like `LS_TS`, it labels the two exchanged elements as tabu independently, while `K_TS` and `M_TS` label the unordered pair as tabu (formed with both elements). The *tabu tenure* parameter is dynamically set during the execution of the algorithm (i.e., it is increased if the solution value has steadily improved and is reduced if the solution value has steadily worsened). The long-term memory maintains a set of good solutions that did not qualify for selection. The search restarts from one of these solutions when the currently explored region does not seem to contain high-quality solutions.

Finally, Aringhieri and Cordone (2011) proposed a random re-start method, `RR`, which constructs an initial solution with a greedy procedure similar to `ErkC`. Then, the constructed solution is improved by means of a simplified version of `XTS`. Specifically, in this simple tabu search, it is only considered the short-term memory and a constant size tabu list of selected and unselected elements. Then, for a pre-specified maximum number of iterations, the method periodically re-starts from a random solution, improving it with this tabu search procedure.

### 3.4 VNS – Variable Neighborhood Search

Variable neighborhood search (Hansen and Mladenovic, 2003) (VNS) is based on a simple and effective idea: a systematic change of the neighborhood within a local search algorithm.

Silva et al. (2004) proposed a simple VNS algorithm, SOMA, for the MDP. The method, based on two neighborhoods, first applies BLS (Ghosh, 1996) until no further improvement is possible. Then, a local search based on an expanded neighborhood is executed. The new neighborhood is defined as the set of all solutions obtained by replacing two elements in the solution (selected elements) by another two that are not present in the current solution (unselected elements).

Brimberg et al. (2009) proposed several VNS procedures originally devoted to the heaviest *k*-subgraph problem, which is a generalization of the MDP in which the distances are replaced with weights (that can take arbitrary values, including 0). Therefore, any algorithm designed to the *k*-subgraph is able to solve the MDP. The authors presented different VNS versions, including *skewed* VNS, *basic* VNS and a combination of a constructive heuristic followed by VNS. The best overall method according to their experimentation is the *basic* VNS, B_VNS, which consists of three main elements. The first one, called *Data Structure*, allows the algorithm to efficiently update the value of the objective function; the second one, *Shaking*, generates solutions in the neighborhood of the current solution by performing random vertex swaps; and third one is a local search procedure based on exchanges.

Aringhieri and Cordone (2011) presented four implementations of the VNS methodology to solve the MDP. They are called Basic VNS, Guided VNS, Accelerated VNS and Random VNS. In all of them, the initial solution is constructed with a greedy method similar to ErkC (Erkut, 1990). Given a solution, each iteration in any of the four methods consists of generating a new solution by replacing *k* elements in the current solution with *k* elements out of it (*Shaking* procedure). The new solution is improved with a basic Tabu Search. Only if the new solution is better or sufficiently distant than the previous one, it becomes the current solution; otherwise it is discarded and a new solution is generated with the same replacement mechanism in which *k* is augmented by one unit. The methods finish when a maximum number of iteration is reached or *k* reaches a pre-established value $k_{max}$.

In the Basic VNS, the Shaking procedure randomly unselects *k* elements in the current solution and randomly selects another *k*. In the Guided variant, the shaking procedure is deterministic and the elements are selected/unselected according to a frequency value, which records the number of times that each element has been included in previous solutions. The accelerated VNS modifies the basic variant by setting $k_{max} = \min(m, n - m)$. This strategy makes re-starts much less frequent because $k_{max}$ is considerably larger than the values used in the Basic and Guided variants. Finally, the Random VNS sets $k_{min} = k_{max} = \min(m, n - m)$, which means that the procedure always re-starts the search in the largest neighborhood, instead of gradually enlarging the neighborhood used. This approach corresponds to a nearly random restart, which only forbids the elements belonging to the current best-known solution. Experimental results show that the random VNS outperforms the other three variants. Consequently, we select this method, denoted as A_VNS, to be included in our own comparison.

### 3.5 SS – Scatter Search

Scatter search (SS) is an evolutionary or population based method (Laguna and Martí, 2003) that explores the solution space by evolving a set of reference solutions stored in the *reference set* (*RefSet*). The evolution of the reference set is induced by the application of four methods: subset generation, update, combination and improvement, where the first two have standard implementations but the last two must be designed for each specific problem.

Aringhieri and Cordone (2006) presented a scatter search procedure, A_SS, to solve the maximum diversity problem in which the *RefSet* is divided into two subsets. Subset *B* contains the best solutions computed during the search and subset *D* contains solutions, which largely differ from each other and from the best ones. The initial *RefSet* is populated by first applying a greedy algorithm similar to ErkC (Erkut, 1990) and then improved with a basic Tabu Search procedure similar to K_TS (Kincaid, 1992). The improved solutions are evaluated to enter in the *RefSet*. Only those solutions with a relatively good value or different enough from those already in the *RefSet* are admitted. The combination method applies the constructive method D2 to the union of the elements in the solutions being combined. After creating all possible combinations, the algorithm tries to insert them in the *RefSet*. If it succeeds and the *RefSet* is modified, the combination process is applied again to the pairs containing newly added solutions. When the *RefSet* is no longer modified, the current iteration terminates, a new subset *D* is generated in the same way as described before and a new iteration starts. The algorithm stops after $I_{SS}$ iterations.

Gallego et al. (2009) proposed an alternative scatter search algorithm, G_SS, for the MDP. In their approach, the distance between solutions is used to measure how diverse one solution is with respect to a set of solutions (measured as the inverse of the number of times that each selected element in a solution appears in the solutions of the set). The diversification generation and the improvement methods of G_SS are respectively the constructive method Tabu_D2 and the local search method LS_TS, proposed in Duarte and Martí (2007). The combination of solutions is performed by applying the constructive method Tabu_D2 to the union of the elements in the solutions being combined. Once the initial *RefSet* has been created, the method performs iterations as long as the combination method is able to generate new and good solutions that enter the *RefSet*. When this is no longer possible, as customary in scatter search, a re-starting is applied by rebuilding the *RefSet* by again applying the diversification generation method. The method stops after a certain number of *RefSet* rebuilding steps.

If we compare both scatter search approaches for the MDP we can observe that they implement different generation, improvement and combination methods: A_SS applies ErkC, K_TS and D2 are respectively, while G_SS employs Tabu_D2, TS_LS and Tabu_D2 respectively. Moreover, A_SS implements two reference sets (to tackle quality and diverse solutions separately), while G_SS uses a single reference set.

### 3.6 MA - Memetic Algorithm

Memetic Algorithms (MA) was introduced in the late 80s to denote a family of metaheuristics that have as central theme the hybridization of different algorithmic approaches for a given problem (Moscato, 1999). They basically combine a population-based approach, which consists of a set of cooperating and competing agents, with a local search approach, in which those agents are individually improved while they sporadically interact.

Katayama and Narihisa (2005) presented a population based method called *memetic algorithm* (`MA`) for the MDP. It basically consists of a randomized greedy method that creates initial solutions, a crossover operator that combines solutions, a repair method to reach feasibility from unfeasible solutions, and a local search method. The randomized greedy method is similar to `GRASP_C2` (Duarte and Martí, 2007). The local search is a *k*-flip method based on the *variable depth search* (VDS) (Kernighan, 1970), which consists of creating new solutions exchanging *k* elements from the selected to the unselected state and vice versa. The crossover operator is the well-known uniform crossover. Finally, the repair method consists of unselecting the elements with lower contribution to the solution (if there is more than *m* selected elements) or selecting the elements with larger contribution (if there is less than *m* selected elements).

## 4 Instance Sets

We have compiled a comprehensive set of benchmark instances representative of the collections previously used for computational experiments in the MDP. We call this benchmark MDPLIB and it is available at `http://www.optsicom.es/mdp`. Furthermore we have included new hard instances. A brief description of the origin and the characteristics of the set of instances follows.

- **SOM**: This data set consists of 70 matrices with random numbers between 0 and 9 generated from an integer uniform distribution.
  - **SOM-a** These 50 instances were generated by Martí et al. (2010) with a generator developed by Silva et al. (2004). The instance sizes are such that for $n = 25$, $m = 2$ and 7; for $n = 50$, $m = 5$ and 15; for $n = 100$, $m = 10$ and 30; for $n = 125$, $m = 12$ and 37; and for $n = 150$, $m = 15$ and 45.
  - **SOM-b**: These 20 instances were generated by Silva et al. (2004) and used in most of the previous papers (see for example Aringhieri et al. 2008). The instance sizes are such that for $n = 100$, $m = 10$, 20, 30 and 40; for $n = 200$, $m = 20$, 40, 60 and 80; for $n = 300$, $m = 30$, 60, 90 and 120; for $n = 400$, $m = 40$, 80, 120, and 160; and for $n = 500$, $m = 50$, 100, 150 and 200.
- **GKD**: This data set consists of 145 matrices for which the values were calculated as the Euclidean distances from randomly generated points with coordinates in the 0 to 10 range.

- **GKD-a**: Glover et al. (1998) introduced these 75 instances in which the number of coordinates for each point is generated randomly in the 2 to 21 range. The instance sizes are such that for $n$ = 10, $m$ = 2, 3, 4, 6 and 8; for $n$ = 15, $m$ = 3, 4, 6, 9 and 12; and for $n$ = 30, $m$ = 6, 9, 12, 18 and 24.
- **GKD-b**: Martí et al. (2010) generated these 50 matrices for which the number of coordinates for each point is generated randomly in the 2 to 21 range and the instance sizes are such that for $n$ = 25, $m$ = 2 and 7; for $n$ = 50, $m$ = 5 and 15; for $n$ = 100, $m$ = 10 and 30; for $n$ = 125, $m$ = 12 and 37; and for $n$ = 150, $m$ = 15 and 45.
- **GKD-c**: Duarte and Martí (2007) generated these 20 matrices with 10 coordinates for each point and $n$ = 500 and $m$ = 50.

▪ **MDG**: This data set consists of 100 matrices with real numbers randomly selected between 0 and 10 from a uniform distribution.

- **MDG-a**: Duarte and Martí (2007) generated these 40 matrices, 20 of them with $n$ = 500 and $m$ = 50 and the other 20 with $n$ = 2000 and $m$ = 200. These instances were used in Palubeckis (2007).

▪ **MDG-b**: This data set consists of 40 matrices generated by Duarte and Martí (2007). 20 of them have $n$ = 500 and $m$ = 50, and the other 20 have $n$ = 2000 and $m$ = 200. These instances were used in Gallego et al. (2009) and Palubeckis (2007).

▪ **MDG-c**: We are proposing here this data set with 20 matrices with $n$ = 3000 and $m$ = 300, 400, 500 and 600. These are the largest instances reported in our computational study. They are similar to those used in Palubeckis (2007).

To sum it up, the MDPLIB contains 315 instances. We refer the reader to our web site `http://www.optsicom.es/mdp` where these instances and their best-known solution values (according to our experimentation below) are available. We include here an appendix with the best values on the largest instances (MDG-c).

## 5 Computational experiments

To perform the experiments reported in this section, we run the original codes of A_SS (Aringhieri and Cordone, 2006), A_VNS (Aringhieri and Cordone, 2011), B_VNS (Brimberg et al., 2009), ITS (Palubeckis, 2007), RR (Aringhieri and Cordone, 2011), STA (Palubeckis, 2007) and XTS (Aringhieri et al., 2008). As acknowledge in Section 7, we thank these authors for sending us the executable codes of their algorithms. The remaining methods were implemented in Java 6. All experiments were performed on an Intel Core 2 Quad CPU Q 8300 with 6 GiB of RAM and Ubuntu 9.04 64 bits OS. We have divided our experimentation into three parts, according to the classification of the instances and methods introduced in the previous sections.

In the first experiment we consider the simple heuristics described in Section 2. In this experiment, for each instance and each method, we compute the relative deviation *Dev* (in percent) between the best solution value, *Value*, obtained with the method and the best-known value of this instance, *BestValue*. The *BestValue* of an instance is the best value obtained with all the methods in all the experiments reported in this paper (i.e., the best known value overall, and available at `http://www.optsicom.es/mdp`). For each method, we also report the number of instances *#Best* for which the method obtains the best value. In addition, we calculate the so-called *Score* (Ribeiro et al., 2002) associated with each method. For each instance, the *nrank* of method *M* is defined as the number of methods that found a better solution than the one found by *M*. In the event of ties, the methods receive the same *nrank*, equal to the number of methods strictly better than all of them. The value of *Score* is the sum of the *nrank* values for all the instances in the experiment, thus, the lower the *Score* the better the method.

| | | GKD-a | GKD-b | GKD-c | MDG-a | MDG-b | SOM-a | SOM-b | Summary |
|---|---|---|---|---|---|---|---|---|---|
| **C2** | **Dev** | 2.18% | 2.26% | 0.27% | 1.84% | 1.66% | 2.95% | 2.08% | 2.07% |
| | **#Best** | 28 | 2 | 1 | 0 | 0 | 5 | 0 | 36 |
| | **Score** | 196 | 186 | 67 | 46 | 42 | 85 | 38 | 660 |
| | **Time** | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.34 | 0.35 | 0.09 |
| **D2** | **Dev** | 0.10% | 0.08% | 0.01% | 1.81% | 1.88% | 2.26% | 1.49% | 1.02% |
| | **#Best** | 61 | 31 | 5 | 0 | 0 | 7 | 0 | 104 |
| | **Score** | 62 | 35 | 10 | 51 | 65 | 65 | 22 | 310 |
| | **Time** | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.10 | 0.10 | 0.03 |
| **ErkC** | **Dev** | 0.52% | 0.19% | 0.02% | 2.16% | 2.20% | 4.70% | 2.57% | 1.73% |
| | **#Best** | 47 | 22 | 6 | 0 | 0 | 5 | 0 | 80 |
| | **Score** | 80 | 62 | 15 | 101 | 110 | 167 | 65 | 600 |
| | **Time** | 0.00 | 0.01 | 0.00 | 0.00 | 0.01 | 0.46 | 0.47 | 0.13 |
| **GhoC** | **Dev** | 1.28% | 1.16% | 0.18% | 3.75% | 3.51% | 6.15% | 3.99% | 2.83% |
| | **#Best** | 35 | 6 | 0 | 0 | 0 | 6 | 0 | 47 |
| | **Score** | 162 | 181 | 69 | 200 | 198 | 197 | 94 | 1101 |
| | **Time** | 0.02 | 0.07 | 0.00 | 0.00 | 0.11 | 2.06 | 1.92 | 0.56 |
| **KLD** | **Dev** | 6.85% | 11.67% | 9.65% | 14.72% | 14.11% | 25.12% | 14.14% | 13.50% |
| | **#Best** | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| | **Score** | 411 | 313 | 124 | 278 | 273 | 323 | 128 | 1850 |
| | **Time** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **KLDv2** | **Dev** | 9.85% | 13.14% | 10.97% | 13.37% | 13.13% | 25.05% | 15.15% | 14.34% |
| | **#Best** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **Score** | 501 | 335 | 136 | 242 | 247 | 326 | 132 | 1919 |
| | **Time** | 0.00 | 0.20 | 0.00 | 0.00 | 0.23 | 5.06 | 5.02 | 1.40 |
| **MDI** | **Dev** | 0.31% | 0.18% | 0.24% | 2.39% | 2.22% | 1.29% | 1.28% | 1.05% |
| | **#Best** | 63 | 26 | 0 | 0 | 0 | 10 | 1 | 100 |
| | **Score** | 14 | 50 | 83 | 132 | 122 | 23 | 22 | 446 |
| | **Time** | 0.03 | 1.30 | 0.00 | 0.02 | 1.69 | 82.11 | 83.99 | 22.73 |
| **STA** | **Dev** | 0.83% | 1.39% | 0.08% | 1.96% | 1.88% | 3.54% | 2.02% | 1.71% |
| | **#Best** | 35 | 9 | 0 | 0 | 0 | 4 | 0 | 48 |
| | **Score** | 155 | 132 | 50 | 70 | 63 | 117 | 57 | 644 |
| | **Time** | 0.17 | 0.19 | 0.15 | 0.14 | 0.22 | 0.52 | 0.86 | 0.30 |

**Table 2** – Constructive methods

Tables 2 and 3 report the results for 8 constructive and 2 improving heuristics respectively. Table 2 shows results for GhoC (Ghosh, 1996), C2, D2 (Glover et al., 1998), ErkC (Erkut, 1990), STA (Palubeckis, 2007), KLD, KLDv2 and MDI (Silva et al., 2004). Similarly, Table 3 reports the results of BLS (Ghosh, 1996) and I_LS (Duarte and Martí, 2007) applied from random initial solutions as well as I_LS applied from the solution obtained with D2 (represented as D2+I_LS).

Table 2 shows that constructive methods D2 and MDI obtain better results than the other competing methods, since they present relative percentage deviations very close to 1% (while the other methods are all above 1.5%). On the other hand, KLD and KLDv2 provide low quality results with deviations larger than 10%. We applied the non-parametric Friedman test for multiple correlated samples to the best solutions obtained by each of the 8 constructive methods. This test computes, for each instance, the rank value of each method according to solution quality (where rank 8 is assigned to the best method and rank 1 to the worst one). Then, it calculates the average rank values of each method across all the instances solved. If the averages differ greatly, the associated *p*-value or significance will be small. The resulting *p*-value of 0.000 obtained in this experiment clearly indicates that there are statistically significant differences among the eight methods tested. Specifically, the rank values produced by this test are 6.46 (D2), 5.98 (MDI), 5.52 (ErkC), 5.49 (C2), 5.46 (STA), 3.49 (GhoC), 1.66 (KLD) and 1.49 (KLDv2).

| | | GKD-a | GKD-b | GKD-c | MDG-a | MDG-b | SOM-a | SOM-b | Summary |
|---|---|---|---|---|---|---|---|---|---|
| **BLS** | **Dev** | 0.19% | 0.45% | 0.00% | 1.72% | 1.74% | 1.98% | 1.68% | 1.04% |
| | **#Best** | 67 | 36 | 16 | 0 | 0 | 10 | 0 | 129 |
| | **Score** | 6 | 11 | 6 | 75 | 72 | 45 | 33 | 248 |
| | **Time** | 0.019 | 4.394 | 0.00 | 0.02 | 0.84 | 283.38 | 282.87 | 77.14 |
| **D2+I_LS** | **Dev** | 0.34% | 0.03% | 0.00% | 0.87% | 0.89% | 1.45% | 0.91% | 0.64% |
| | **#Best** | 67 | 41 | 13 | 0 | 0 | 9 | 3 | 133 |
| | **Score** | 6 | 13 | 10 | 14 | 14 | 34 | 10 | 101 |
| | **Time** | 0.003 | 0.653 | 0.00 | 0.00 | 0.04 | 118.57 | 142.77 | 35.48 |
| **I_LS** | **Dev** | 0.43% | 0.57% | 0.00% | 1.05% | 1.13% | 1.93% | 1.25% | 0.91% |
| | **#Best** | 66 | 32 | 12 | 0 | 0 | 13 | 2 | 125 |
| | **Score** | 5 | 19 | 12 | 31 | 34 | 40 | 16 | 157 |
| | **Time** | 0.047 | 16.143 | 0.00 | 0.08 | 5.08 | 1325.29 | 1489.00 | 383.06 |

**Table 3** – Improvement methods

The first and third main rows in Table 3 show the results of BLS and I_LS respectively, running from random initial solutions. The second main row (D2+I_LS) shows the results obtained when I_LS is applied to the solution obtained with D2 (the best constructive method according to the previous experiment). Results in Table 3 show that the two improvement methods, BLS and I_LS, provide good solutions, since both present average percentage deviations from the best-known solution close to 1% (where I_LS is slightly better than BLS). Moreover, comparing I_LS with D2+I_LS we observe that, as expected, D2+I_LS obtains better solutions than I_LS (0.64% versus 0.91%) in shorter running times (35.48 versus 383.06 CPU seconds). We apply the Wilcoxon test to compare I_LS with D2+I_LS. (This test measures whether the solutions come or not from different methods.) The *p*-value of 0.000 obtained with this

nonparametric test for pairwise comparisons confirms that the there are differences between the results of both methods.

| | | GKD-c | MDG-a | MDG-b | SOM-b | Summary |
|---|---|---|---|---|---|---|
| | **Dev** | 0.00% | 7.97% | 7.53% | 1.77% | 5.46% |
| **GRASP_C2+I_LS** | **#Best** | 15 | 0 | 0 | 5 | 20 |
| | **Score** | 29 | 183 | 184 | 71 | 467 |
| | **Dev** | 0.00% | 1.81% | 1.79% | 0.29% | 1.25% |
| **GRASP_D2+I_LS** | **#Best** | 19 | 2 | 1 | 7 | 29 |
| | **Score** | 5 | 14 | 9 | 26 | 54 |
| | **Dev** | 0.00% | 5.02% | 4.66% | 0.25% | 3.27% |
| **GRASP+DM** | **#Best** | 20 | 0 | 0 | 7 | 27 |
| | **Score** | 0 | 82 | 69 | 30 | 181 |
| | **Dev** | 0.00% | 5.84% | 5.57% | 0.58% | 3.90% |
| **GRASP+PR** | **#Best** | 20 | 0 | 0 | 7 | 27 |
| | **Score** | 0 | 126 | 141 | 40 | 307 |
| | **Dev** | 0.00% | 5.13% | 4.81% | 0.34% | 3.37% |
| **KLD+BLS** | **#Best** | 20 | 0 | 0 | 7 | 27 |
| | **Score** | 0 | 114 | 104 | 39 | 257 |
| | **Dev** | 0.00% | 5.57% | 5.22% | 0.62% | 3.70% |
| **KLDv2+BLS** | **#Best** | 19 | 0 | 0 | 5 | 24 |
| | **Score** | 6 | 140 | 157 | 68 | 371 |
| | **Dev** | 0.00% | 50.40% | 50.43% | 0.20% | 33.64% |
| **MDI+BLS** | **#Best** | 20 | 0 | 0 | 6 | 26 |
| | **Score** | 0 | 181 | 176 | 20 | 377 |

**Table 4** – GRASP metaheuristics running for 10 seconds

In our second experiment we consider the 20 metaheuristics described in Section 3 to solve most of the larger instance sets: SOM-b, GKD-c, MDG-a, MDG-b. We have executed each method for 10 seconds per instance. We have divided this experiment into three groups, in the first one we consider the seven GRASP methods, in the second one the ten methods based on trajectories and in the third one the three methods based on populations.

Table 4 reports the values *Dev*, *#Best* and *Score* obtained with the following seven GRASP methods executed for 10 seconds in each instance: `KLD+BLS`, `MDI+BLS`, `KLDv2+BLS` (Silva et al., 2004) `GRASP_C2+I_LS`, `GRASP_D2+I_LS` (Duarte and Martí, 2007), `GRASP+PR` (Silva et al., 2007), and `GRASP+DM` (Santos et al., 2005).

Results in Table 4 show that, according to the average deviation, the `GRASP_D2` coupled with `I_LS` is the best method (with a value of 1.25%), while the `KLD` variants and `MDI` coupled with the `BLS` improvement obtain lower quality results. The ranking of the Friedman statistical test is in line with these deviation values, obtaining: `GRASP_D2+I_LS` (5.38), `GRASP+DM` (4.39), `GRASP+PR` (4.28), `MDI+BLS` (4.34), `KLD+BLS` (3.76), `KLDv2+BLS` (2.97) and `GRASP_C2+I_LS` (2.88). The associated *p*-value of 0.000 indicates that there are significant differences among the results obtained with these methods.

| | | GKD-c | MDG-a | MDG-b | SOM-b | Summary |
|---|---|---|---|---|---|---|
| **A_VNS** | **Dev** | 0.00% | 0.16% | 0.14% | 0.00% | 0.10% |
| | **#Best** | 20 | 20 | 20 | 20 | 80 |
| | **Score** | 0 | 43 | 44 | 0 | 87 |
| **B_VNS** | **Dev** | 0.00% | 0.09% | 0.08% | 0.00% | 0.06% |
| | **#Best** | 20 | 16 | 15 | 20 | 71 |
| | **Score** | 0 | 20 | 25 | 0 | 45 |
| **ITS** | **Dev** | 0.00% | 0.17% | 0.16% | 0.03% | 0.12% |
| | **#Best** | 20 | 16 | 17 | 18 | 71 |
| | **Score** | 0 | 63 | 55 | 12 | 130 |
| **K_TS** | **Dev** | 0.00% | 6.51% | 6.02% | 0.67% | 4.29% |
| | **#Best** | 20 | 0 | 0 | 7 | 27 |
| | **Score** | 0 | 322 | 326 | 109 | 757 |
| **M_TS** | **Dev** | 0.00% | 1.19% | 1.10% | 0.22% | 0.80% |
| | **#Best** | 20 | 0 | 0 | 8 | 28 |
| | **Score** | 0 | 284 | 283 | 82 | 649 |
| **RR** | **Dev** | 0.00% | 0.17% | 0.17% | 0.00% | 0.11% |
| | **#Best** | 20 | 20 | 20 | 20 | 80 |
| | **Score** | 0 | 51 | 57 | 0 | 108 |
| **SA** | **Dev** | 0.00% | 0.69% | 0.61% | 0.05% | 0.44% |
| | **#Best** | 20 | 1 | 2 | 9 | 32 |
| | **Score** | 0 | 217 | 215 | 59 | 491 |
| **SOMA** | **Dev** | 0.00% | 7.02% | 6.62% | 1.09% | 4.73% |
| | **#Best** | 12 | 0 | 0 | 3 | 15 |
| | **Score** | 72 | 350 | 348 | 147 | 917 |
| **Tabu_D2+LS_TS** | **Dev** | 0.00% | 0.43% | 0.38% | 0.10% | 0.29% |
| | **#Best** | 20 | 0 | 0 | 4 | 24 |
| | **Score** | 0 | 208 | 210 | 101 | 519 |
| **XTS** | **Dev** | 0.00% | 0.23% | 0.18% | 0.00% | 0.13% |
| | **#Best** | 20 | 12 | 14 | 20 | 66 |
| | **Score** | 0 | 91 | 75 | 0 | 166 |

**Table 5 –** Local search based metaheuristics running for 10 seconds

Table 5 reports the results obtained with the following ten local search based metaheuristics executed for 10 seconds per instance: A_VNS (Aringhieri and Cordone, 2011), B_VNS (Brimberg et al., 2009), ITS (Palubeckis, 2007), K_TS (Kincaid, 1992), M_TS (Macambira, 2002), RR (Aringhieri and Cordone, 2011), SA (Kincaid, 1992), SOMA (Silva et al., 2004), Tabu_D2+LS_TS (Duarte and Martí, 2007), and XTS (Aringhieri et al., 2008). Results in Table 5 show that all the local search based metaheuristics tested, with the exception of K_TS and SOMA, are able to obtain high quality solutions, since they present low average percentage deviations. In particular B_VNS and A_VNS present 0.06% and 0.10% deviation values and 71 and 80 best-known solutions respectively out of 120 instances, which compare favorably with the rest of the methods. They also present the lowest scores (45 for B_VNS and 87 for A_VNS), therefore we conclude that they are best methods in this group (although closely followed by some of the other methods). The ranking of the Friedman test confirms this evaluation (and also provides a *p*-value of 0.000): B_VNS (7.94), A_VNS (7.53), RR (7.35), ITS

(7.24), XTS (7.00), Tabu_D2+LS_TS (4.99), SA (4.85), M_TS (3.60), K_TS (2.73) and SOMA (1.78).

Considering that the differences between B_VNS and A_VNS are small, we have applied two statistical tests for pairwise comparisons, the *Wilcoxon test* and the *Sign test*. The former tests if the two samples (the solutions of both methods) come from different populations, while the latter computes the number of instances on which an algorithm improves upon the other. The resulting *p*-values of 0.000 and 0.004 respectively, indicate that there are significant differences between the results of both methods, resulting B_VNS as the best method of this experiment. However, the pairwise comparison between A_VNS and any other method with a Friedman rank value greater than or equal to 7.0 (RR, ITS and XTS) produces a *p*-value larger than 0.05, indicating that there are no significant differences among these four methods.

In the third group of this experiment, we considered the population-based metaheuristics. Specifically, Table 6 reports the results obtained with A_SS (Aringhieri and Cordone, 2006), G_SS (Gallego et al., 2009) and MA (Katayama and Narihisa, 2005) executed in 10 seconds per instance.

| | | GKD-c | MDG-a | MDG-b | SOM-b | Summary |
|---|---|---|---|---|---|---|
| | **Dev** | 0.00% | 0.41% | 0.36% | 0.00% | 0.26% |
| **A_SS** | **#Best** | 20 | 14 | 17 | 20 | 71 |
| | **Score** | 0 | 49 | 41 | 0 | 90 |
| | **Dev** | 0.00% | 0.28% | 0.25% | 0.00% | 0.18% |
| **G_SS** | **#Best** | 20 | 17 | 16 | 18 | 71 |
| | **Score** | 0 | 20 | 19 | 3 | 42 |
| | **Dev** | 0.00% | 0.27% | 0.24% | 0.02% | 0.17% |
| **MA** | **#Best** | 20 | 3 | 10 | 15 | 48 |
| | **Score** | 0 | 34 | 25 | 10 | 69 |

**Table 6 –** Population based metaheuristics running for 10 seconds

Results on Table 6 show that the three methods are able to provide good solutions. MA and G_SS are the best, with 0.17% and 0.18% average deviations and 48 and 71 best-known solutions respectively. The ranking of the Friedman test is G_SS (2.28), MA (2.00) and A_SS (1.72) with a *p*-value of 0.000, confirming that G_SS emerges as the best method. However, if we apply the Wilcoxon and the Sign tests to compare G_SS with MA, we obtain a *p*-value of 0.374 and 0.15, respectively. The associated *p*-value indicates that there are no statistical differences between both methods. On the other hand, if we apply these tests to compare G_SS with A_SS we obtain in both of them a *p*-value of 0.000 indicating that G_SS outperforms A_SS.

In the previous experiment, we have measured the ability of 20 metaheuristics to obtain good solutions in a short-term period of time (10 seconds). In our next experiment we undertake to compare these methods on a long-term scenario (600 seconds). Tables 7, 8 and 9 show, respectively, the results of the seven GRASP methods, the ten local search

based methods and the three population-based methods when running for 600 seconds on each instance.

| | | GKD-c | MDG-a | MDG-b | SOM-b | Summary |
|---|---|---|---|---|---|---|
| | **Dev** | 1.04% | 3.77% | 3.63% | 0.06% | 2.65% |
| **GRASP_C2+I_LS** | **#Best** | 7 | 3 | 2 | 10 | 22 |
| | **Score** | 78 | 186 | 192 | 46 | 502 |
| | **Dev** | 0.00% | 0.53% | 0.54% | 0.02% | 0.36% |
| **GRASP_D2+I_LS** | **#Best** | 20 | 10 | 11 | 11 | 52 |
| | **Score** | 0 | 59 | 71 | 18 | 148 |
| | **Dev** | 0.00% | 0.61% | 0.56% | 0.05% | 0.40% |
| **GRASP+DM** | **#Best** | 20 | 7 | 4 | 10 | 41 |
| | **Score** | 0 | 95 | 112 | 29 | 236 |
| | **Dev** | 0.00% | 0.51% | 0.46% | 0.02% | 0.33% |
| **GRASP+PR** | **#Best** | 20 | 1 | 5 | 11 | 37 |
| | **Score** | 0 | 81 | 72 | 24 | 177 |
| | **Dev** | 0.00% | 0.65% | 0.58% | 0.03% | 0.42% |
| **KLD+BLS** | **#Best** | 19 | 4 | 4 | 11 | 38 |
| | **Score** | 4 | 117 | 119 | 31 | 271 |
| | **Dev** | 0.00% | 0.68% | 0.63% | 0.08% | 0.45% |
| **KLDv2+BLS** | **#Best** | 19 | 1 | 1 | 8 | 29 |
| | **Score** | 4 | 156 | 133 | 54 | 347 |
| | **Dev** | 0.00% | 0.55% | 0.50% | 0.03% | 0.35% |
| **MDI+BLS** | **#Best** | 15 | 2 | 2 | 12 | 31 |
| | **Score** | 25 | 107 | 92 | 14 | 238 |

**Table 7 –** GRASP metaheuristics running for 600 seconds

Results of Tables 7, 8 and 9 are in line with those reported in Tables 4, 5 and 6, although, as expected, after 600 seconds of CPU time, the differences between the methods are small in most cases (as compared with the differences obtained in the 10-seconds runs). Specifically, GRASP_D2+I_LS emerges again as the leading GRASP based method (with the best values of #*Best* and *Score* and with a competitive *Dev.* value). Regarding the local search based methods (Table 8), B_VNS, ITS and A_VNS are the best of them in terms of deviation, number of best and score. Although in the short runs (10 sec.) tested in Table 5 B_VNS is the clear winner, in the long runs tested in Table 8 (600 sec.) these three methods produce very similar results. As a matter of fact, the *p*-value of pairwise statistical tests between any pairs of them are larger than 0.05, indicating that there are not significant differences among them. With respect to the population based procedures shown in Table 9, the three methods exhibit again a similar performance. Nevertheless, the Friedman ranking establishes the following ordering: G_SS (2.09), A_SS (2.04) and MA (1.87) and a *p*-value 0.03, indicating statistical significant differences. As a conclusion, we will consider these five best methods, GRASP_D2+I_LS, B_VNS, ITS, A_VNS and G_SS in the final experiment.

| | | GKD-c | MDG-a | MDG-b | SOM-b | Summary |
|---|---|---|---|---|---|---|
| | **Dev** | 0.00% | 0.02% | 0.02% | 0.00% | 0.02% |
| **A_VNS** | **#Best** | 20 | 31 | 23 | 20 | 94 |
| | **Score** | 0 | 23 | 34 | 0 | 57 |
| | **Dev** | 0.00% | 0.01% | 0.02% | 0.00% | 0.01% |
| **B_VNS** | **#Best** | 20 | 31 | 27 | 20 | 98 |
| | **Score** | 0 | 19 | 21 | 0 | 40 |
| | **Dev** | 0.00% | 0.02% | 0.01% | 0.00% | 0.01% |
| **ITS** | **#Best** | 20 | 20 | 23 | 19 | 82 |
| | **Score** | 0 | 28 | 18 | 5 | 51 |
| | **Dev** | 0.00% | 2.67% | 2.44% | 0.32% | 1.76% |
| **K_TS** | **#Best** | 20 | 1 | 0 | 9 | 30 |
| | **Score** | 0 | 330 | 334 | 89 | 753 |
| | **Dev** | 0.00% | 0.55% | 0.52% | 0.04% | 0.36% |
| **M_TS** | **#Best** | 20 | 3 | 2 | 12 | 37 |
| | **Score** | 0 | 244 | 252 | 53 | 549 |
| | **Dev** | 0.00% | 0.04% | 0.04% | 0.00% | 0.03% |
| **RR** | **#Best** | 20 | 21 | 21 | 20 | 82 |
| | **Score** | 0 | 50 | 51 | 0 | 101 |
| | **Dev** | 0.00% | 0.27% | 0.24% | 0.00% | 0.17% |
| **SA** | **#Best** | 20 | 20 | 19 | 20 | 79 |
| | **Score** | 0 | 118 | 121 | 0 | 239 |
| | **Dev** | 0.00% | 1.55% | 1.42% | 0.49% | 1.07% |
| **SOMA** | **#Best** | 20 | 0 | 0 | 4 | 24 |
| | **Score** | 0 | 338 | 339 | 140 | 817 |
| | **Dev** | 0.00% | 0.28% | 0.27% | 0.07% | 0.20% |
| **Tabu_D2+LS_TS** | **#Best** | 20 | 0 | 1 | 5 | 26 |
| | **Score** | 0 | 238 | 234 | 108 | 580 |
| | **Dev** | 0.00% | 0.12% | 0.13% | 0.00% | 0.09% |
| **XTS** | **#Best** | 20 | 12 | 15 | 20 | 67 |
| | **Score** | 0 | 122 | 106 | 0 | 228 |

**Table 8** – Local search based metaheuristics running for 600 seconds

| | | GKD-c | MDG-a | MDG-b | SOM-b | Summary |
|---|---|---|---|---|---|---|
| | **Dev** | 0.00% | 0.10% | 0.12% | 0.00% | 0.07% |
| **A_SS** | **#Best** | 20 | 20 | 20 | 20 | 80 |
| | **Score** | 0 | 8 | 17 | 0 | 25 |
| | **Dev** | 0.00% | 0.14% | 0.13% | 0.00% | 0.09% |
| **G_SS** | **#Best** | 20 | 20 | 19 | 20 | 79 |
| | **Score** | 0 | 29 | 23 | 0 | 52 |
| | **Dev** | 0.00% | 0.13% | 0.13% | 0.00% | 0.09% |
| **MA** | **#Best** | 20 | 20 | 20 | 20 | 80 |
| | **Score** | 0 | 23 | 22 | 0 | 45 |

**Table 9** – Population based metaheuristics running for 600 seconds

In our final experiment we compare the methods identified to be the best in the previous experiments. Specifically we consider the best algorithms in each category (GRASP based, local search based and population based) identified above:

- GRASP based: GRASP_D2+I_LS (Duarte and Martí, 2007).
- Local Search based: B_VNS (Brimberg et al., 2009), A_VNS (Aringhieri and Cordone, 2011) and ITS (Palubeckis, 2007).
- Population based: G_SS (Gallego et al., 2009).

Instances in the GKD set seem to be very easy for these methods, even the largest in the GKD-c set, since most of them are able to match all the best known solutions. Therefore, they will no longer be considered in the final experimentation. On the other hand, we shall include the largest instances reported in our study, MDG-c with *n*=3000. Table 10 reports the *Dev*, *#Best* and *Score* values obtained with these five methods executed for 10 seconds on the SOM-b, MDG-a, MDG-b and MDG-c instances. Table 11 reports the results obtained when these methods are left to run for 600 seconds.

| **Methods** | | **MDG-a** | **MDG-b** | **MDG-c** | **SOM-b** | **Summary** |
|---|---|---|---|---|---|---|
| GRASP based | | | | | | |
| | **Dev** | 1.81% | 1.79% | 1.97% | 0.29% | 1.57% |
| **GRASP_D2+I_LS** | **#Best** | 2 | 1 | 0 | 7 | 10 |
| | **Score** | 152 | 156 | 80 | 52 | 440 |
| Local Search based | | | | | | |
| | **Dev** | 0.16% | 0.14% | 0.37% | 0.00% | 0.16% |
| **A_VNS** | **#Best** | 20 | 20 | 0 | 20 | 60 |
| | **Score** | 30 | 28 | 42 | 0 | 100 |
| | **Dev** | 0.09% | 0.08% | 0.13% | 0.00% | 0.08% |
| **B_VNS** | **#Best** | 16 | 15 | 0 | 20 | 51 |
| | **Score** | 14 | 14 | 2 | 0 | 30 |
| | **Dev** | 0.17% | 0.16% | 0.33% | 0.03% | 0.17% |
| **ITS** | **#Best** | 16 | 17 | 0 | 18 | 51 |
| | **Score** | 42 | 36 | 35 | 7 | 120 |
| Population based | | | | | | |
| | **Dev** | 0.28% | 0.25% | 0.35% | 0.00% | 0.24% |
| **G_SS** | **#Best** | 17 | 16 | 0 | 18 | 51 |
| | **Score** | 65 | 70 | 41 | 6 | 182 |

**Table 10** – Best metaheuristics running for 10 seconds

Table 10 shows that the five methods under comparison are able to obtain high quality solutions within a short period of time (10 seconds of CPU). The GRASP based method is probably the worse one in this group of best methods, since it obtains average percentage deviations larger than 1% (while the other four methods present values below 1%). On the other hand B_VNS and A_VNS seem to be the best ones with deviations of 0.08% and 0.16% and number of best solutions of 51 and 50, respectively. The Friedman test obtains a ranking in line with these observations: 4.08 (B_VNS), 3.45 (A_VNS), 3.33 (ITS), 2.97 (G_SS), and 1.17 (GRASP_D2+ILS). The Wilcoxon and Sign tests comparing the best two methods, B_VNS and A_VNS, obtain both a *p*-value of 0.000 indicating that there are significant differences between them. We can therefore conclude that B_VNS is the best method on the short runs (10 seconds).

| Methods | | MDG-a | MDG-b | MDG-c | SOM-b | Summary |
|---|---|---|---|---|---|---|
| **GRASP based** | | | | | | |
| **GRASP_D2+I_LS** | **Dev** | 0.53% | 0.54% | 1.64% | 0.02% | 0.63% |
| | **#Best** | 10 | 11 | 0 | 11 | 32 |
| | **Score** | 120 | 116 | 80 | 36 | 352 |
| **Local search based** | | | | | | |
| **A_VNS** | **Dev** | 0.02% | 0.02% | 0.07% | 0.00% | 0.03% |
| | **#Best** | 31 | 23 | 1 | 20 | 75 |
| | **Score** | 18 | 27 | 29 | 0 | 74 |
| **B_VNS** | **Dev** | 0.01% | 0.02% | 0.04% | 0.00% | 0.02% |
| | **#Best** | 31 | 27 | 5 | 20 | 83 |
| | **Score** | 16 | 17 | 10 | 0 | 43 |
| **ITS** | **Dev** | 0.02% | 0.01% | 0.06% | 0.00% | 0.02% |
| | **#Best** | 20 | 23 | 0 | 19 | 62 |
| | **Score** | 24 | 15 | 21 | 3 | 63 |
| **Population based** | | | | | | |
| **G_SS** | **Dev** | 0.14% | 0.13% | 0.27% | 0.00% | 0.13% |
| | **#Best** | 20 | 19 | 0 | 20 | 59 |
| | **Score** | 60 | 62 | 60 | 0 | 182 |

**Table 11** – Best metaheuristics running for 600 seconds

As expected, the average percentage deviations of the methods are lower in Table 11, in which they are run for 600 seconds, than in Table 10 in which they are run for 10 seconds. In this way, after 600 seconds of CPU time, the five methods under comparison present deviations lower than 1%. In line with this, the number of best solutions found increases as the running time increases. The ranking of the Friedman test on the results of Table 11 is very similar to the one obtained with the results of Table 10 with the same *p*-value of 0.000 (although now `ITS` seems to be the second method). Specifically, the rank values are: 3.68 (`B_VNS`), 3.53 (`ITS`), 3.42 (`A_VNS`), 2.88 (`G_SS`), and 1.49 (`GRASP_D2+ILS`). The Wilcoxon and Sign tests comparing the best two methods in this experiment, `B_VNS` and `ITS`, obtain both a *p*-value of 0.000 indicating that there are significant differences between them. We can therefore conclude that `B_VNS` is also the best method on the large runs (600 seconds).

To complement this analysis, Figure 1 shows the typical search profile for the best methods that we compared (average percentage deviations are reported in this diagram). It is designed to show how the average value of the best solution found improves over time. This execution corresponds to the five largest instances ($n$=3000, $m$=600) in the MDG-c set with a time limit of 30 minutes (recording the average deviation values every 5 minutes).

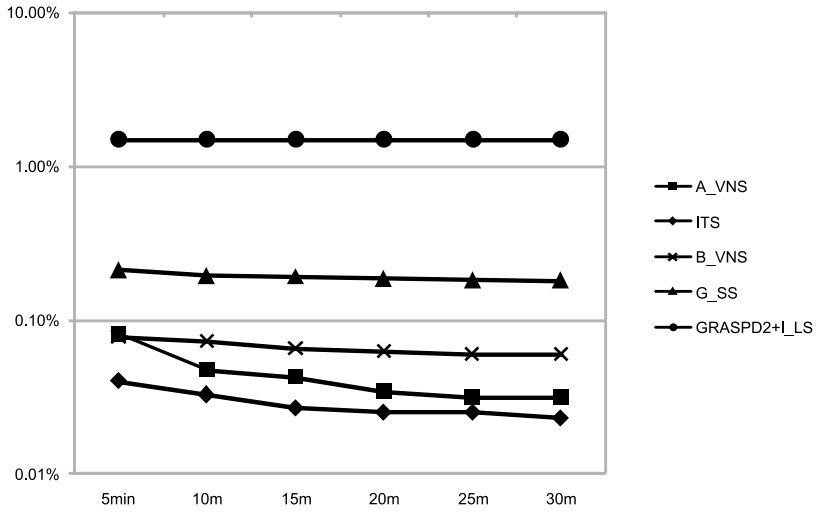


**Figure 1.** Search profile for a 30-minute run of the largest instances

The average percentage deviations shown in Figure 1 reveal a different behavior of the methods when considering only large instances and 30 minutes run than the one exhibited in the previous experiment. In this case, `ITS` consistently produces better solutions than the other methods (note that results in Table 11 show that `B_VNS` produces slightly better outcomes than `ITS`). This behavior can be partially explained by the fact that in the experiment reported in Table 11 we consider 20 matrices with $n = 3000$ and $m = 300$, 400, 500 and 600, while in the experiment shown in Figure 1 we only consider the largest 5 instances ($n = 3000$ and $m = 600$). We can then consider that both methods are the best overall, where `B_VNS` can be considered more robust and `ITS` more suited for largest instances. Regarding the rest of methods, `A_VNS` appears as the third best method with very good results, closely followed by `G_SS`. `GRASPD2+I_LS` produces average percent deviation above 1.0%.

Table 12 in the Appendix contains the best-known solutions for the 20 largest (and apparently hardest) instances in our study to set a benchmark for future comparisons. The Best Value has been obtained running the `B_VNS` and `ITS` methods for 2 hour on each instance.

## 6 Conclusions

A computational comparison of 30 methods, 10 heuristics and 20 metaheuristics, for the maximum diversity problem has been presented. Experiments with 315 instances were performed to compare the procedures when solving this NP-hard problem. Our extensive experimentation reveals that even the simplest heuristics provide good solutions to this combinatorial optimization problem. However, to obtain high-quality solutions, local search based metaheuristics seem better suited for this problem than GRASP or population based methods. Specifically, the Variable Neighborhood Search (Brimberg et al., 2009) emerges as the best overall except for the five largest problems where the Iterated Tabu Search (Palubeckis 2007) is the leader. When including the smaller

problems, Iterated Tabu Search comes in as a close second in overall standing, followed by the VNS procedure (Aringhieri and Cordone, 2011).

## 7 Acknowledgements

This research has been partially supported by the *Ministerio de Educación y Ciencia* of Spain within the OPTSICOM project (`http://www.optsicom.es`) with grant code TIN2009-07516. The authors would like to thank Profs. Aringhieri, Cordone, Melzani, Palubeckis, Brimberg, Mladenovíc, Uroševíc, and Ngai for sharing their programs with them.

## References

Aringhieri, R. and R. Cordone. (2006). "Better and faster solutions for the maximum diversity problem". Technical report, Universit degli Studi di Milano, Polo Didattico e di Ricerca di Crema.

Aringhieri, R. and R. Cordone. (2011). "Comparing local search metaheuristics for the maximum diversity problem". *Journal of the Operational Research Society* 62: 266-280.

Aringhieri, R.R. Cordone, and Y. Melzani. (2008). "Tabu search vs. grasp for the maximum diversity problem". *4OR: A Quarterly Journal of Operations Research* 6(1):45–60.

Brimberg, J., N. Mladenovic, D. Urosevic and E. Ngai. (2009). "Variable neighborhood search for the heaviest *k*-subgraph". *Computers & Operations Research* 36(11): 2885-2891.

Cappanera, P. (1999). "A survey on obnoxious facility location problems". *Technical Report TR-99-11*, Dipartimento di Elettronica ed Informazione, Politecnico di Milano, 19.

Chandra, B. and M.M. Halldorsson. (2001). "Approximation algorithms for dispersion problems". *Journal of Algorithms* 38(2): 438–465.

Duarte, A. and R. Martí. (2007). "Tabu search and grasp for the maximum diversity problem". *European Journal of Operational Research* 178: 71–84.

Erkut, E. (1990). "The discrete p-dispersion problem". *European Journal of Operational Research* 46(1): 48–60.

Erkut, E. and S. Neuman. (1991). "Comparison of four models for dispersing facilities". *INFOR Canadian Journal of Operational Research and Information Processing* 29: 68–86.

Feige, U., G. Kortsarz, and D. Peleg. (2001). "The dense *k*-subgraph problem". *Algorithmica* 29: 410–421.

Feo, T. and M.G.C. Resende. (1995). "Greedy randomized adaptive search procedures". *Journal of Global Optimization* 2: 1–27.

Gallego, M., A. Duarte, M. Laguna, and R. Martí. (2009). "Hybrid heuristics for the maximum diversity problem". *Computational Optimization and Applications* 44(3): 411-426.

Ghosh, J.B. (1996). "Computational Aspects of the Maximum Diversity Problem". *Operations Research Letters* 19(1): 175–181.

Glover, F. (1986). "Future paths for integer programming and links to artificial intelligence". *Computers and Operations Research* 13: 533–549.

Glover, F. and M. Laguna. (1997). *Tabu Search*, Kluwer Academic Publishers, Boston.

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.

Han, J. and M. Kamber. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishing.

Hansen, P. and N. Mladenovic. (2003). *Handbook of Metaheuristics*, chapter Variable neighborhood search, pp 145–184. Springer.

Katayama, K. and H. Narihisa. (2005). “An evolutionary approach for the maximum diversity problem”. In W. Hart, N. Krasnogor and J. E. Smith (eds), *Recent advances in memetic algorithms*, volume 166, pp 31–47. Springer, Berlin.

Kernighan, B.W. and S. Lin. (1970). “An efficient heuristic procedure for partitioning graphs”. *Bell System Technical Journal* 49: 291–307, 1970.

Kincaid, R.K. (1992). “Good solutions to discrete noxious location problems via metaheurísticas”. *Annals of Operations Research* 40(1): 265–281.

Kincaid, R.K. and L.G. Yellin. (1993). “The discrete p-dispersion-sum problem: results on trees and graphs”. *Computers & Operations Research* 1(2): 171–186.

Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi. (1983). “Optimization by Simulated Annealing”. *Science, New Series* 220(4598): 671-680.

Kuby, M. J. (1987). “Programming models for facility dispersion: The p-dispersion and maximum dispersion problem”. *Geographical Analysis* 19(4): 315–329.

Kuo, C.C., F. Glover, and K.S. Dhir. (1993). “Analyzing and Modeling the Maximum Diversity Problem by Zero-One programming”. *Decision Sciences* 24(6): 1171–1185.

Laguna, M. and R. Martí. (1999). “Grasp and path relinking for 2-layer straight line crossing minimization”. *INFORMS Journal on Computing*, 11: 44–52.

Laguna, M. and R. Marti. (2003). *Scatter Search methodology and implementations in C*. Kluwer Academic Publishers.

Macambira, E.M. (2002). “An application of tabu search heuristic for the maximum edge-weighted subgraph problem”. *Annals of Operations Research* 177: 175–190.

Macambira, E.M. and C.C. de Souza. (2000). “The edge-weighted clique problem: valid inequalities, facets and polyhedral computations”. *European Journal of Operational Research* 123: 346–371.

Martí, R., M. Gallego, and A. Duarte. (2010). “A branch and bound algorithm for the maximum diversity problem”. *European Journal of Operational Research* 200(1): 36–44.

McConnell, S. (1988). “The new battle over immigration”. *Fortune* 117(10): 89-102.

Moon, I.D. and S.S. Chaudhry. (1984). “An analysis of network location problems with distance constraints”. *Management Science* 30(3): 290–307.

Moscato, P. (1999). “Memetic algorithms: A short introduction”. In D. Corne, M. Dorigo, and F. Glover (eds.), *New Ideas in Optimization*, pp 219-234. McGraw-Hill, Maidenhead, Berkshire, England, UK.

Palubeckis, G. (2007). “Iterated tabu search for the maximum diversity problem”. *Applied Mathematics and Computation* 189: 371-383.

Pearce, D. (1987). “Economics and genetic diversity”. *Future* 19(6): 710-712.

Pisinger, D. (2006). “Upper bounds and exact algorithms for p-dispersion problems”. *Computers and Operations Research* 33(5): 1380–1398.

Porter, W.M., K.M. Rawal, K.O. Rachie, H.C. Wien, and R.C. Williams. (1975). *Cowpea germplasm catalog No 1*. International Institute of Tropical Agriculture, Ibadan, Nigeria.

Prokopyev, O.A., N. Kong, and D.L. Martinez-Torres. (2009). “The equitable dispersion problem”. *European Journal of Operational Research* 197(1): 59-67.

Ravi, S.S., D.J. Rosenkrantz, and G.K. Tayi. (1994). “Heuristic and special case algorithms for dispersion problems”. *Operations Research* 42(2): 299–310.

Ribeiro, C.C., E. Uchoa, and R.F. Werneck. (2002). “A hybrid grasp with perturbations for the steiner problem in graphs”. *INFORMS Journal on Computing* 14(3): 228–246.

Santos, L.F., M.H. Ribeiro, A. Plastino, and S.L. Martins. (2005). “A hybrid grasp with data mining for the maximum diversity problem”. In Maria J. Blesa, Christian Blum, Andrea Roli, and Michael Sampels (eds.), *Hybrid Metaheuristics*, volume 3636 of *Lecture Notes in Computer Science*, pp 116–127. Springer.

Silva, G.C., L.S. Ochi, and S.L. Martins. (2004). “Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem”. In *Experimental and Efficient Algorithms*, volume 3059 of *Lecture Notes in Computer Science*, pp 498–512. Springer Berlin/Heidelberg.

Silva, G.C., M.R.Q. Andrade, L.S. Ochi, S.L. Martins, and A. Plastino. (2007). “New heuristics for the maximum diversity problem”. *Journal of Heuristics* 13(4): 315–336.

Swierenga, R.P. (1977). “Ethnicity in historical perspective”. *Social Science* 52(1): 31–44.

Wang, J., Y. Zhou, J. Yin, and Y. Zhang. (2009). “Competitive Hopfield Network Combined With Estimation of Distribution for Maximum Diversity Problems”. In *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* - Special issue on cybernetics and cognitive informatics 39(4): 1048-1066.

# Appendix

| Instance | Best Value | Method |
|---|---|---|
| MDG-c_1_n3000_m300.txt | 24924685 | B_VNS |
| MDG-c_2_n3000_m300.txt | 24909199 | B_VNS |
| MDG-c_3_n3000_m300.txt | 24900820 | ITS |
| MDG-c_4_n3000_m300.txt | 24904964 | B_VNS |
| MDG-c_5_n3000_m300.txt | 24899703 | ITS |
| MDG-c_6_n3000_m400.txt | 43465087 | ITS |
| MDG-c_7_n3000_m400.txt | 43477267 | B_VNS |
| MDG-c_8_n3000_m400.txt | 43458007 | B_VNS |
| MDG-c_9_n3000_m400.txt | 43448137 | B_VNS |
| MDG-c_10_n3000_m400.txt | 43476251 | ITS |
| MDG-c_11_n3000_m500.txt | 67009114 | B_VNS |
| MDG-c_12_n3000_m500.txt | 67021888 | ITS |
| MDG-c_13_n3000_m500.txt | 67024373 | B_VNS |
| MDG-c_14_n3000_m500.txt | 67024804 | B_VNS |
| MDG-c_15_n3000_m500.txt | 67056334 | B_VNS |
| MDG-c_16_n3000_m600.txt | 95637733 | B_VNS |
| MDG-c_17_n3000_m600.txt | 95645826 | ITS |
| MDG-c_18_n3000_m600.txt | 95629207 | ITS |
| MDG-c_19_n3000_m600.txt | 95633549 | ITS |
| MDG-c_20_n3000_m600.txt | 95643586 | ITS |

**Table 12** – Best known values for MDG-c instances