A Hybrid Metaheuristic Method for the Maximum Diversity Problem

Qinghua Wu [a,b] and Jin-Kao Hao [b,*]

[a] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China*

[b] *LUNAM Université, Université d'Angers, LERIA, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France*

Accepted to European Journal of Operational Research, June 2013.

DOI: http://dx.doi.org/10.1016/j.ejor.2013.06.002

Abstract

The Maximum Diversity Problem (MDP) consists in selecting a subset of m elements from a given set of n elements ($n > m$) in such a way that the sum of the pairwise distances between the m chosen elements is maximized. We present a hybrid metaheuristic algorithm (denoted by MAMDP) for MDP. The algorithm uses a dedicated crossover operator to generate new solutions and a constrained neighborhood tabu search procedure for local optimization. MAMDP applies also a distance-and-quality based replacement strategy to maintain population diversity. Extensive evaluations on a large set of 120 benchmark instances show that the proposed approach competes very favorably with the current state-of-art methods for MDP. In particular, it consistently and easily attains all the best known lower bounds and yields improved lower bounds for 6 large MDP instances. The key components of MAMDP are analyzed to shed light on their influence on the performance of the algorithm.

Keywords: Maximum Diversity Problem, solution combination, local search, constrained neighborhood, population diversity.

* Corresponding author.
Email addresses: wu@info.univ-angers.fr (Qinghua Wu), hao@info.univ-angers.fr (Jin-Kao Hao).

1 Introduction

Let $N = \{s_1, s_2, ..., s_n\}$ be a set of elements and d_{ij} be the distance between elements s_i and s_j (d_{ij}=d_{ji}), with $d_{ij} > 0$ if $i \neq j$ and $d_{ij} = 0$ otherwise. The Maximum Diversity Problem (MDP for short) consists in selecting a subset $M \subset N$ of a given cardinality m ($m < n$) from N, such that the sum of the distances between every two elements in M is maximized. Formally, the problem can be stated as the following quadratic zero-one integer program [27]:

$$Maximize\ f(x) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_ix_j \tag{1}$$

subject to

$$\sum_{i=1}^{n} x_i = m \tag{2}$$

where x_i is a binary variable indicating whether an element s_i is selected to be a member of the subset M.

MDP is known to be NP-hard and has a high computational complexity [16]. In addition to its theoretical significance as a difficult combinatorial problem, MDP is notable for its ability to formulate a number of practical applications: location of undesirable or mutually competing facilities [11], decision analysis with multiple objectives [36], composing jury panels [28], genetic engineering [33], medical and social sciences [27], and product design [18]. During the past three decades, MDP has been studied under many different names such as maxisum dispersion [26], MAX-AVG dispersion [39], edge-weighted clique [1,31], remote-clique [9], maximum edge-weighted subgraph [30], and dense k-subgraph [8,12].

The computational challenge of the MDP has motivated a variety of solution approaches including exact methods, approximation algorithms and metaheuristic methods. Examples of approximation algorithms are described in [12,22]. These approaches provide a performance guarantee, but do not compete well with other methods in computational testing. Three recent examples of exact methods are described in [5,32,38]. While these methods have the theoretical advantage of finding optimal solutions to a given problem, their applications are generally limited to problems with some 150 elements.

For larger problem instances, heuristics and metaheuristics are often used to find approximate solutions of good quality with a reasonable computing time. This includes tabu search [1,4,6,10,30,44], iterated tabu search [36], simulated annealing [25], iterated greedy algorithm [28], estimation of distribution algorithms [43], genetic algorithms [13], variable neighborhood search [6,8], scatter search [15,21], path-relinking method [2,3] and memetic search [24]. Another approach that has received considerable attention in the solution of the MDP is greedy randomized adaptive search procedure (GRASP) [2,3,10,16,40–42]. Finally, a comprehensive survey and an interesting comparison of the most significant heuristic and metaheuristic methods for MDP can be found in [6,33].

This paper presents MAMDP, a hybrid metaheuristic algorithm integrating a tabu search procedure with a population-based evolutionary algorithm for solving the Maximum Diversity Problem. The proposed algorithm integrates three complementary key components to ensure the high efficiency of the search process. First, to generate promising new solutions, we introduce a dedicated crossover operator which tries to preserve common elements that are shared by parent solutions. The design of this crossover operator is motivated by an experimental observation that high quality solutions share a large number of common elements. Second, to allow the algorithm to explore efficiently the search space around each newly generated solution by crossover, we devise a tabu search optimization procedure which relies on a constrained neighborhood and a dynamic tabu list management strategy. Finally, to maintain the population diversity, we employ a quality-and-distance replacement strategy for population updates.

To assess the performance and the competitiveness of our algorithm in terms of both solution quality and computing efficiency, we provide computational results on a total of 120 MDP benchmark instances with up to 5000 elements, showing that the proposed algorithm achieves highly competitive results with respect to the best existing MDP heuristics. Moreover, for 6 large MDP instances, the proposed algorithm is able to provide new improved results.

The remaining part of the paper is organized as follows. In Section 2, the ingredients of our algorithm are described, including the dedicated crossover operator, the constrained neighborhood tabu search procedure and the quality-and-distance based pool updating rule. Section 3 is dedicated to the computational results. Section 4 investigates several important components of the proposed MAMDP algorithm and concluding remarks are given in Section 5.

2 A hybrid metaheuristic algorithm for MDP

Our hybrid metaheuristic algorithm follows the general memetic framework which combines the population-based evolutionary search and neighborhood-based local search [34,35]. The basic idea is to take advantage of both a recombination (or crossover) operator that discovers unexplored promising regions of the search space, and a local search operator that finds good solutions by concentrating the search around these regions. In order to be effective, the general memetic framework needs to be carefully adapted to the given problem and to integrate problem-specific knowledge within its search operators and strategies [23]. As discussed in [17] and [19] (Chapter 9), the MA framework shares ideas with scatter search [20]. In particular, scatter search provides unifying principles for joining solutions by structured combinations of elite solutions from a reference set. Additionally, scatter search pays special attention to both distance and quality when it updates the reference set. Finally, scatter search typically uses tabu search to improve each new solution. In this sense, the proposed algorithm can be considered as a simplified scatter search algorithm.

The general procedure of our hybrid metaheuristic algorithm for MDP (called MAMDP) is summarized in Algorithm 1. It is composed of four main basic components: a population initialing procedure, a tabu search procedure, a crossover operator and a population management strategy. Starting from an initial population of local optima obtained by the tabu search procedure (Section 2.2), MAMDP performs a series of cycles called generations. At each generation, two solutions S^1 and S^2 are randomly chosen in the population to serve as parents. The crossover is then used to produce an offspring solution S^0 from S^1 and S^2 (Section 2.4). The tabu search procedure is applied to improve S^0 for a fixed number of iterations (Section 2.3). Afterward, the population updating rule decides whether the improved solution S^0 should be inserted into the population and which existing solution should be replaced (Section 2.5). This process repeats until a stop condition is verified, such as a time limit or a fixed number of generation (Section 3.3). In the following, we describe the four main components of the proposed algorithm.

Algorithm 1 Memetic algorithm for the Maximum Diversity Problem

```
Require: A set of $n$ elements $N = \{s_1, s_2, ..., s_n\}$, distance matrix $[d_{ij}]_{n \times n}$, cardinality of the subset $m$ $(m < n)$, population size $p$
Ensure: The best solution $S^*$ found
 1: Initialize population $Pop = \{S_1, ..., S_p\}$ /* Section 2.2 */
 2: $S^* \leftarrow Best(Pop)$ /* $S^*$ records the best solution encountered until now */
 3: while Stop condition is not verified do
 4:    Randomly select 2 parent solutions $S^1$ and $S^2$ from $Pop = \{S_1, ..., S_p\}$
 5:    $S^0 \leftarrow Cross\_Over(S^1, S^2)$ /* Section 2.4, generate a new solution from parents */
 6:    $S^0 \leftarrow Tabu\_Search(S^0)$ /* Section 2.3, improve the offspring */
 7:    if $f(S^0) > f(S^*)$ then
 8:       $S^* \leftarrow S^0$ /* Update the best solution found so far */
 9:    end if
10:    $Pop \leftarrow PopulationUpdate(S^0, Pop)$ /* Section 2.5, update population using a distance-and-quality rule */
11: end while
```

2.1 Search space and evaluation function

Before presenting the components of the MAMDP algorithm, we define first the search space explored by the algorithm as well as the evaluation function to measure the quality of a candidate solution.

Given that the objective of MDP is to determine a subset $M \subset N$ of size m (called m-subset) while maximizing the sum of the distances between every two elements in M, we define our search space Ω to be the collection of all possible subsets of N of cardinality m, i.e., $\Omega = \{S \subset N : |S| = m\}$. It is clear that Ω has a size of $C_n^m = \frac{n!}{m!(n-m)!}$ which may be very large for reasonable values of n and m.

To evaluate the quality of a solution $S \in \Omega$ (i.e., a m-subset), we just sum up the distances between every two elements in S as follows:

$$f(S) = \sum_{s_u, s_v \in S, u<v} d_{uv} \tag{3}$$

It is easy to see that this function is strictly equivalent to the function defined in Formula (1).

Finally, we mention that to represent each subset S of Ω, we use a binary vector $\overline{S}$ of length n containing exactly m 1s: $\overline{S}[i] = 1$ if element $s_i \in N$ belongs to subset S, $\overline{S}[i] = 0$ otherwise. For simplicity reasons, hereafter we will use the set notion S, instead of its vector representation $\overline{S}$, to designate a solution of Ω even if both are semantically equivalent.

2.2 Generation of initial solutions

Our algorithm begins with an initial population composed of p solutions (p is the population size which is fixed by a parameter). There are different ways to obtain the initial population. One basic technique is random generation which, though easy to apply, can hardly lead to initial solutions of good quality. In this paper, we initialize the population with locally optimal solutions as follows. Starting from a random m-subset S ($S \in \Omega$), we apply the tabu search procedure (see Section 2.3) to improve S until a local optimum S is reached. The resulting improved solution is added to the population if it is not identical to any solution currently in the population. The insertion condition can be stated formally using the distance measure defined in Section 2.5 as follows: the new solution is added to the population if its distance to any existing solution of the population is greater than zero. This procedure is repeated until the population is filled up with $3 \times p$ solutions from which we finally retain the p

best ones with the largest objective values to form the initial population. This procedure allows us to obtain an initial population of relatively high quality. Notice that for some small (or easy) MDP instances, our algorithm can even reach the optimal solutions (or solutions with previous best known objective values) during the phase of initialization of the population due to the high efficiency of the tabu search procedure.

2.3 The constrained neighborhood tabu search procedure

One key element of our hybrid MAMDP algorithm is its tabu search procedure which ensures the critical role of intensified search of a limited region. In addition to being applied to generate the initial population as explained in the previous section, the tabu procedure is in particular used to improve the offspring solutions created by the crossover operator (see Section 2.4). Adopting the general method of tabu search [19], our tabu procedure (see Algorithm 2) is specifically adapted to the MDP problem by introducing a dedicated constrained neighborhood and a dynamic tabu list management mechanism, which are developed in this section.

Algorithm 2 Constrained neighborhood tabu search procedure for MDP

```
Require: A set of n elements $N = \{s_1, s_2, ..., s_n\}$, initial solution $S$, number $MaxIter$ of tabu search iterations
Ensure: The best solution $S^*$ found and $f(S^*)$
 1: $S^* \leftarrow S$   /* Records the best solution found so far */
 2: $Iter \leftarrow 0$   /* Iteration counter */
 3: Compute the potential $p_v$ according to Eq. 4 for each element $s_v \in N$.
 4: Initiate the tabu list and tabu tenure
 5: $dmax \leftarrow max\{d_{ij} \mid 1 \leq i < j \leq n\}$   /* $dmax$ is the maximum distance between two elements in $N$*/
 6: while $Iter < MaxIter$ do
 7:    $dMinInS \leftarrow min\{p_i \mid s_i \in S\}$ /* $dMinInS$ is the smallest potential in $S$ */
 8:    Identify subset $X = \{s_i \in S \mid p_i \leq dMinInS + dmax\}$
 9:    $dMaxOutS \leftarrow max\{p_i \mid s_i \in N \setminus S\}$ /* $dMaxOutS$ is the largest potential in $N \setminus S$ */
10:    Identify subset $Y = \{s_i \in N \setminus S \mid p_i \geq dMaxOutS - dmax\}$
11:    Choose a best admissible $swap(s_u, s_v)$ move from the constrained neighborhood $CN(S)$ defined by $X$ and $Y$
12:    $S \leftarrow S \backslash \{s_u\} \cup \{s_v\}$ /* Move to the new solution */
13:    Update the tabu list and the potential $p_v$ for each $s_v \in N$
14:    if $f(S) > f(S^*)$ then
15:       $S^* \leftarrow S$     /* Update the best solution found so far */
16:    end if
17:    $Iter \leftarrow Iter + 1$
18: end while
```

2.3.1 Constrained swap move and neighborhood

As explained in Section 2.1, our search space Ω is composed of all possible m-subsets from the given set N. To explore this space, one simple and basic way is to start with any initial m-subset S and subsequently swap an element of S with another element of $N \setminus S$ such that the objective value is improved. One advantage of this swap move is that it maintains solution feasibility. Nevertheless, this unconstrained swap leads to a large neighborhood of size $m \cdot (n - m)$. This unconstrained swap and its associated neighborhood were used in [16] and further explored in [10]. As indicated in [10], although this unconstrained swap is valuable for local search algorithms, the evaluation of the neighborhood can be computationally expensive. This is particularly the case with tabu search given that at each iteration of the algorithm, we wish to select the *best* swap move among all $m.(n-m)$ possible moves induced by S and $N \setminus S$.

To reduce the computing time needed to examine neighboring solutions and improve the computational efficiency of our tabu search procedure, we devise a *constrained* neighborhood which is both more focused and smaller-sized. The idea of our constrained neighborhood is to limit the swap move to two specifically identified subsets $X \subseteq S$ and $Y \subseteq N \setminus S$ such that $|X|$ and $|Y|$ are as small as possible, and the resulting neighborhood contains always the best solutions of the unconstrained neighborhood induced by S and $N \setminus S$.

The constrained neighborhood is based on the notion of *potential* defined for each element of the current solution. Precisely, let $S \in \Omega$ be a solution (i.e., a m-subset of N), we define, for each element $s_i \in N$, its potential p_i with respect to the objective value $f(S)$ as follows:

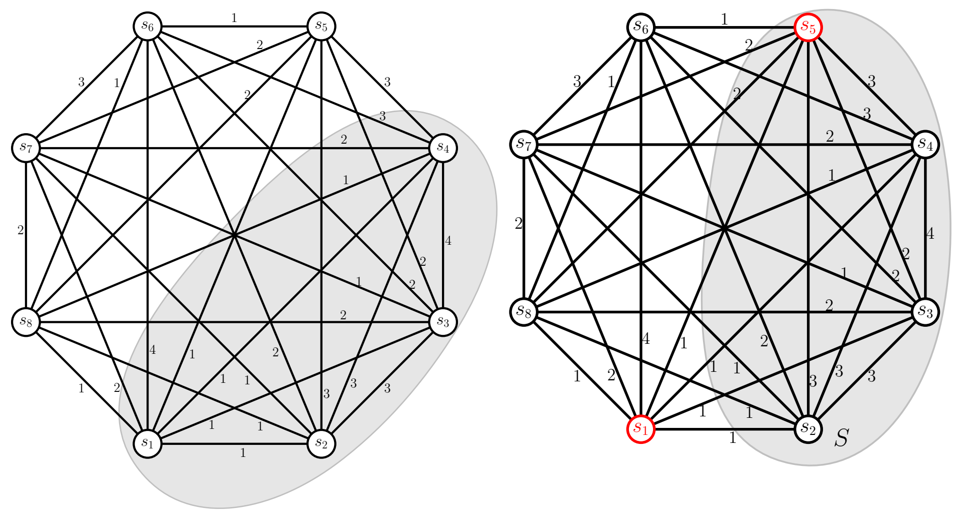
$$p_i = \sum_{s_j \in S} d_{ij}, for \ s_i \in N \tag{4}$$

Let $swap(s_u, s_v)$ designate the move which swaps $s_u \in S$ and $s_v \in N \setminus S$. Then, when $swap(s_u, s_v)$ is applied, the objective variation Δ_{uv}, also called the 'move gain', can be conveniently computed by:

$$\Delta_{uv} = f(S') - f(S) = p_v - p_u - d_{uv} \tag{5}$$

where $S' = S \setminus \{s_u\} \cup \{s_v\}$ while p_v and p_u are respectively the potential of s_v and s_u according to Formula (4).

From Formula 5, we observe that the move gain Δ_{uv} of $swap(s_u, s_v)$ depends on p_v, p_u and d_{uv}. For the purpose of maximizing the objective function f, we should prefer an element $s_v \in N \setminus S$ with a large potential and inversely an element $s_u \in S$ with a small potential. In addition, we also need to consider the distance d_{uv} between s_u and s_v. To maximize f, we constrain s_u to belong to a specific subset $X \subseteq S$ containing the elements in S with small potentials, and s_v to belong to a specific subset $Y \subseteq N \setminus S$ including the elements in $N \setminus S$ with large potentials.



$S = \{s_1, s_2, s_3, s_4\},\ N \setminus S = \{s_5, s_6, s_7, s_8\},\ dmax = 4$
$d_1 = 3,\ d_2 = 7,\ d_3 = 8, d_4 = 8,\ d_5 = 9,\ d_6 = 11,\ d_7 = 6,\ d_8 = 5$
$dMinInS = 3,\ dMaxOutS = 11,\ X = \{s_1, s_2\},\ Y = \{s_5, s_6\}$

$\Delta_{15} = 5,\ \Delta_{16} = 4,\ \Delta_{25} = -1, \Delta_{26} = 2$
swap(s_1, s_5) *is the best swap move*

Fig. 1. An example for the constrained neighborhood defined by swap move.

Let $dMinInS = min\{p_i \mid s_i \in S\}$ and

Let $dMaxOutS = max\{p_i \mid s_i \in N \setminus S\}$.

Then we define subsets X and Y as follows:

$X = \{s_i \in S \mid p_i \leq dMinInS + dmax\}$ and

$Y = \{s_i \in N \setminus S \mid p_i \geq dMaxOutS - dmax\}$

where $dmax$ is the maximum distance between two elements in N, i.e., $dmax = max\{d_{ij} \mid 1 \leq i < j \leq n\ \}$.

To obtain a neighboring solution S' from S, we swap one element $s_u \in X$ with another element $s_v \in Y$. All possible swap moves induced by X and Y define our constrained neighborhood $CN(S)$, i.e.,

$CN(S) = \{S' \mid S' = S \backslash \{s_u\} \cup \{s_v\},\ s_u \in X, s_v \in Y\}$.

Fig. 1 shows an illustrative example where S has four possible neighboring solutions and we assume that the tabu list is empty (i.e., no element is forbidden for swap).

Our tabu search procedure explores the search space Ω by following this constrained neighborhood. At each iteration, instead of examining all the swap moves induced by S and $N \setminus S$, our tabu search first identifies the two subsets

X and Y associated to S and then selects the best admissible $swap(s_u, s_v)$ ($s_u \in X$, $s_v \in Y$) with the highest move gain Δ_{uv} (ties broken randomly). The resulting solution replaces S to become the new current solution. A swap move $swap(s_u, s_v)$ is admissible if it is not classified tabu (i.e. neither s_u nor s_v is in the tabu list) or if it verifies the aspiration criterion. The aspiration criterion simply states the tabu status of a move is revoked if the move leads to a solution better than any solution found so far.

We can see that our constrained neighborhood is a strict subset of the unconstrained neighborhood. Furthermore, assume that S'' is a best admissible neighboring solution in the unconstrained neighborhood, it is easy to verify that $S'' \in CN(S)$. In other words, our constrained neighborhood $CN(S)$ contains all the best admissible neighboring solutions in the unconstrained neighborhood, while its size is generally much smaller than the unconstrained neighborhood.

When a $swap(s_u, s_v)$ move is performed to give a new solution, the potential associated with each element s_i in N can be efficiently updated using the following formula, as shown in [4,6]:

$$p_i = \begin{cases} p_i + d_{iv}, & if\ s_i = s_u, \\ p_i - d_{iu}, & if\ s_i = s_v, \\ p_i + d_{iv} - d_{iu}, & if\ s_i \neq s_u \text{ and } s_i \neq s_v. \end{cases}$$

Thus, the updating of the potentials associated with the n elements in N can be performed in linear time $O(n)$.

To compute subset X (see also line 7–8 in Algorithm 2), we first examine all the elements in S and identify $dMinInS$ to be the element with the minimum potential among the elements in S. We then check all the elements in S once again, a vertex u is added into X if its potential $p_u \leq dMinInS + dmax$. Obviously, the procedure for computing subset X can be performed in linear time $O(|S|)$. Similarly, we compute subset Y by examining all the vertices in $N \setminus S$ (see also line 9–10 in Algorithm 2) with a time complexity of $O(|N \setminus S|)$. When the two subsets X and Y are identified, we need to examine all the swap moves induced by X and Y to select the best admissible move with the highest move gain in the constrained neighborhood. This can be achieved with a time complexity of $O(|X| \times |Y|)$. In addition, the time needed to update the potentials associated with the n elements in N is bounded by $O(n)$. Therefore, the total time of each iteration of our algorithm using the constrained neighborhood is bounded by $O(n) + O(|X| \times |Y|)$. As shown in Section 4.1, X and Y of our constrained neighborhood are much smaller than S and $N \setminus S$, reducing drastically the computing time of our algorithm.

2.3.2 Tabu list and tenure management

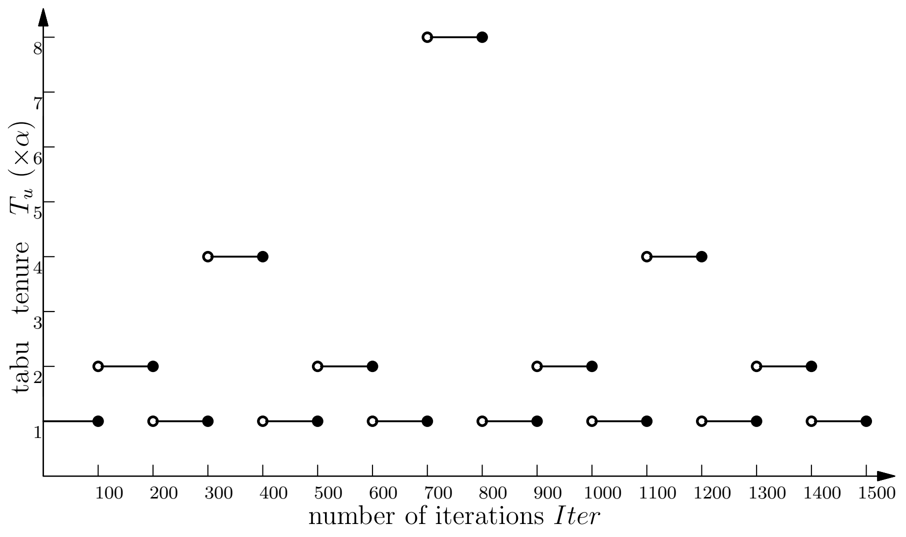


Fig. 2. An illustration of the step function (one period) used for tuning the tabu tenure T_u [45].

As previously explained, a neighboring solution of S is obtained by applying a $swap(s_u, s_v)$ move to the current solution. To prevent the search from short-term cycling, when such a move is performed, element s_u is marked tabu for the next T_u (called tabu tenure) iterations, during which s_u cannot be put back into solution S (except the aspiration criterion is satisfied). Similarly, element s_v is also marked tabu for the next T_v iterations and s_v cannot be removed from S during this period (except the aspiration criterion is satisfied).

It is well known that the performance of a tabu search algorithm depends on the way the tabu tenure is determined [19]. A too short tabu tenure may lead the search to revisit solutions previously encountered while a too long tabu tenure may exclude high quality solutions during the search. Unfortunately, there does not exist a general way to optimally tune the tabu tenure. In this paper, we adopt a dynamic tabu list management technique. This tabu list management technique was first proposed in [14] and recently explored in [45]. With this technique, the tabu tenure T_u are dynamically adjusted by a periodic step function $T_u(Iter)$ defined over the number of iterations $Iter$: $Iter \rightarrow T_u(Iter)$ where $Iter$ is the number of iterations and $T_u(Iter)$ is the tabu tenure for T_u at iteration $Iter$. Each period of the step function is composed of 1500 iterations divided into 15 steps.

Fig. 2 provides an illustration of this step function. As shown in Fig. 2, the tabu tenure T_u is equal to α (α is a parameter) for the first 100 iterations [1..100], then $2 \times \alpha$ for iterations from [101..200], followed by α again for iterations [201..300] and $4 \times \alpha$ for iterations [401..500] etc. After reaching the largest value $8 \times \alpha$ for iterations [701..800], T_u drops again to α for the next 100 iterations and so on. This function repeats periodically this variation scheme every 1500 iterations. Similarly, we use the same strategy to tune the tabu tenure T_v. At each iteration of the tabu search, T_v is set equal to $0.7 * T_u$.

One notices that the tabu tenure T_u for s_u (the element leaving S) is larger than the tabu tenure T_v for s_v (the element joining S). This can be explained by the simple fact that in general, there are much fewer elements contained in S than in $N \setminus S$ ($m < n$). As a consequence, when an element becomes a part of the solution, we try to keep it in the solution for a longer period.

As one observes, this tabu mechanism involves four different tabu tenures (α, $2\times\alpha$, $4\times\alpha$ and $8\times\alpha$) which are applied with quite different frequencies: 8/15 for α, 4/15 for $2\times\alpha$, 2/15 for $4\times\alpha$ and 1/15 for $8\times\alpha$. Since a shorter (longer) tabu tenure generally implies a more intensified (diversified) search, this tabu mechanism ensures most of the time an intensified examination of the search space followed by punctual diversification phases of different intensities. So if the search with the shortest tabu tenure α is trapped in a local optimum, the subsequent longer tabu tenure $2 \times \alpha$ is expected to bring the search out of the trap. If this is not sufficient, a still longer tabu tenures $4 \times \alpha$, then $8\times\alpha$ is applied to break the trap. Each time the search escapes from the local optimum, an intensification phase is resumed with the shortest tenure.

Finally, we also experimented two additional tabu tenure methods in addition to the above dynamic tabu tenure method based on the periodic step function. The first one applies in a static way each of the four tabu tenures (α, $2\times\alpha$, $4\times\alpha$ and $8\times\alpha$) during the whole search. The second method uses, at each iteration of the algorithm, a tabu tenure randomly taken from the 15-tuple $(\alpha, 2\times\alpha, \alpha, 4\times\alpha, \alpha, 2\times\alpha, \alpha, 8\times\alpha, \alpha, 2\times\alpha, \alpha, 4\times\alpha, \alpha, 2\times\alpha, \alpha)$. This second method shares similarities with our dynamic method in the sense that the shortest tabu tenure is used more frequently than the three other values. Yet this second method follows a random scheme to alter among the different tabu tenures while the dynamic method method uses the periodic step function shown in Fig. 2. Experimental results showed that the dynamic method dominates both cases. Even if other tabu tenure methods would be envisaged, we will see in Section 3 that the adopted method is effective and robust and allows our algorithm to deliver highly competitive computational outcomes.

2.4 Crossover operator

Within the hybrid memetic framework, the crossover operator constitutes another important search operator [35]. Its main goal is to create new promising candidate solutions by blending existing parent solutions, a solution being promising if it can potentially lead the search process to new search regions where better solutions may be found. For this reason, crossover plays basically an exploratory role which comes to complement the intensification role of the tabu search procedure. It is well known that even if random crossover operators (uniform, one-point etc.) can be easily applied in the context of binary representation, such a blind operator can rarely guide the search process to effectively explore the most promising search regions. To be effective, a meaningful crossover operator is usually based on some pertinent properties (building blocks) of the given problem and a recombination mechanism to preserve these properties from parents to offspring [23].

To identify "good properties" for MDP, we carried out a detailed analysis of samples of locally optimal solutions (see Section 4). This analysis discloses that high quality solutions share a large number of common elements that have high chances to be part of an optimal solution. Therefore, given two high quality solutions, it seems pertinent to preserve the shared elements (building blocks). Our proposed crossover operator follows this idea and operates as follows.

Algorithm 3 The crossover operator of the MAMDP algorithm

```
Require: Two parent solutions $S^1$ and $S^2$
Ensure: One offspring solution $S^0$
 1: $S^0 \leftarrow S^1 \cap S^2$   /* Build first a partial solution by preserving the common
    elements shared by $S^1$ and $S^2$ */
 2: while $|S^0| < m$ do
 3:    Select from $S^1 \setminus S^0$ the element $u$ with the highest potential with respect to
       $S^0$
 4:    $S^0 \leftarrow S^0 \cup \{u\}$, $S^1 \leftarrow S^1 \backslash \{u\}$
 5:    if $|S^0| = m$ then
 6:       Return $S^0$ and Stop
 7:    end if
 8:    Select from $S^2 \setminus S^0$ the element $v$ with the highest potential with respect to
       $S^0$
 9:    $S^0 \leftarrow S^0 \cup \{v\}$, $S^2 \leftarrow S^2 \backslash \{v\}$
10: end while
11: Return $S^0$
```

Given two parent solutions S^1 and S^2 which are chosen randomly from the population, one offspring solution S^0 is constructed as follows (see Algorithm 3). We build first a partial solution by preserving the common elements shared by the two selected parents, i.e., $S^0 = S^1 \cap S^2$. Then we complete S^0 to obtain a feasible solution with a greedy procedure based on the potential p_i defined in Section 2.3.1. The greedy procedure extends S^0 in a step-by-step way by adding at each step one element to S^0 until S^0 contains exactly m elements. At the first step, we examine all the elements in $S^1 \setminus S^0$ to identify the element with the highest potential with respect to S^0 and displace it from $S^1 \setminus S^0$ to S^0. Afterward, we consider the elements in $S^2 \setminus S^0$, identify the element with the largest potential in $S^2 \setminus S^0$ and add it to S^0. Then at each step of this greedy procedure, we consider the elements in $S^1 \setminus S^0$ and $S^2 \setminus S^0$ in turn until

S^0 reaches the size of m. This offspring solution is usually of relatively high quality and has approximatively the same distance to its two parent solutions.

2.5 Population updating rule

The population updating rule decides whether an offspring solution, which is generated by the crossover operator and further improved by the tabu search procedure, should become a member of the population, and in the affirmative, which existing solution of the population should be replaced. Population management is an important issue since the population updating rule impacts directly the population diversity which conditions the convergence of the search algorithm.

In our case, we wish to maintain a healthy diversity of the population during the search. For this purpose, we adopt an updating strategy which takes into account both quality and the distance between the solutions of the population [29,37]. While the notion of quality can be easily understood with respect to the objective function, we need to formally define the notion of distance between solutions.

Definition 1. Distance between two solutions: Given two solutions S^a and S^b, the distance $dist(S^a, S^b)$ between S^a and S^b is the minimum number of swap moves necessary to transform S^b to S^a, i.e., $dist(S^a, S^b) = m - |S^a \cap S^b|$.

Definition 2. Distance between one solution and a population: Given a population $Pop = \{S^1, ..., S^p\}$ and the distance $dist(S^i, S^j)$ between any two solutions S^i and S^j ($i, j \in \{1, ..., p\}$ and $i \neq j$), the distance between a solution S^i ($i = 1, ..., p$) and the population Pop is defined as the minimum distance between S^i and any other solution in the population:

$$D_{S^i,Pop} = min\{dist(S^i, S^j) | S^j \in Pop, S^j \neq S^i\} \tag{6}$$

To update the population, we adopt the strategy which was originally proposed in [29], and has shown to be very effective in maintaining the balance between the diversity and quality of the population. This strategy uses a quality-and-distance scoring function (denoted by $H(S^i, Pop')$) to rank the solutions of the population $Pop' = Pop \cup \{S^0\}$. Formally, $H(S^i, Pop')$ is defined as follows:

$$H(S^i, Pop') = \beta \widetilde{A}(f(S^i)) + (1 - \beta)\widetilde{A}(D_{S^i,Pop'}) \tag{7}$$

where β is a parameter set to 0.6 according to [29], $f(S^i)$ is the objective value of S^i, and $\widetilde{A}(.)$ is a normalized function defined as follows:

$$\widetilde{A}(y) = \frac{y - y_{min}}{y_{max} - y_{min} + 1} \tag{8}$$

where y_{max} and y_{min} are respectively the maximum and minimum of y in the population Pop. "+1" is used for the purpose of avoiding the possibility of a 0 denominator.

Algorithm 4 Population updating strategy

```
Require: Offspring solution $S^0$ and Population $Pop = \{S^1, ..., S^p\}$
Ensure: Updated population $Pop = \{S^1, ..., S^p\}$
1: Tentatively add $S^0$ to the population: $Pop' = Pop \cup \{S^0\}$
2: for $i = 0, ..., p$ do
3:     Calculate the distance between $S^i$ and $Pop'$ according to Eq. (6)
4:     Calculate the goodness score $H(S^i, Pop')$ of $S^i$ according to Eq. (7)
5: end for
6: Identify the solution $S^w$ with the smallest goodness score in $Pop'$: $S^w = arg\ min\{H(S^i, Pop')|i = 0, ..., p\}$
7: if ($S^w \neq S^0$) then
8:     Replace $S^w$ with $S^0$: $Pop = Pop \cup \{S^0\} \backslash \{S^w\}$
9: end if
```

Our population updating strategy is described in Algorithm 4. To update the population, we first tentatively add S^0 to the population Pop, then we use the scoring function H to identify the solution S^w with the smallest goodness score $H(S^i, Pop')$ in Pop'. If S^w is not the offspring S^0, then the population is updated by replacing S^w with S^0.

3 Computational experiments

In this section, we present an extensive assessment of the proposed MAMDP. For this purpose, we show experimental results obtained by MAMDP on a large collection of benchmark instances and make comparisons with the best performing MDP algorithms published in the literature.

3.1 Benchmark instances

To evaluate the efficiency of the proposed approach, we carry out extensive experiments on the same sets of 120 test instances as in [36], which are frequently used to assess algorithms for MDP. The details of the instance sets are described as follows [10,15,33]:

- Silva instances: This data set consists of 20 instances, which were generated by Silva et al [41]. 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. These instances can be downloaded from http://www.optsicom.es/mdp/.

- Random Type *I* instances (Type1_55, Type1_22 and Type1_52): These instances (60 in total) are based on matrices with real numbers generated from a (0, 10) uniform distribution. Random Type *I* was introduced by Duarte and Martí [10] and includes 3 sets of instances. The first set (Type1_55) consists of 20 instances with $n = 500$ and $m = 50$, the second set (Type1_52) includes 20 instances with $n = 500$ and $m = 200$, and the third set contains 20 instances with $n = 2000$ and $m = 200$. Random Type *I* instances are available at http://www.uv.es/∼rmarti/paper/mdp.html.
- Random Type *II* instances (Type2): These instances (20 in total) are based on matrices with real numbers generated from a (0, 1000) uniform distribution. This data set was introduced by Duarte and Martí [10] and have size of $n = 500$ and $m = 50$. Random Type *II* instances can be downloaded from http://www.uv.es/∼rmarti/paper/mdp.html.
- Beasley instances (b2500): This data set consists of 10 instances, which were taken from the OR-Library [7]. All the instances have 10% density with $m = 2500$ and $n = 1000$. These instances were used in [8,28,36,44] to assess the MDP algorithms and are available at http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/bqpinfo.html.
- Random larger instances (p3000 and p5000): These instances (10 in total) are based on matrices with integer numbers generated from a $[0, 100]$ uniform distribution. For these instances, the distance between some elements is equal to zero and the density of these instances (i.e., the percentage of the non-zero entries) is 10%, 30%, 50%, 80%, 100% respectively. There are five instances with $n = 3000$ and $m = 1500$, and five instances with $n = 5000$ and $m = 2500$. These instances were tested in [8,28,36,44]. See http://www.soften.ktu.lt/∼gintaras/max_div.html for the sources of the generator and input files to replicate these instances.

3.2 Parameter settings

The settings of the parameters used by our MAMDP algorithm are given in Table 1. These parameter values have been determined by performing a preliminary experiment on a selection of 10 hard instances (Type1_22.1, Type1_22.2, b2500-1, b2500-3, b2500-5, b2500-7, p3000_1, p3000_3, p5000_2 and p5000_3) which are randomly taken without bias from three groups of the largest benchmark instances (Type1_22, b2500, p3000 and p5000). Instances from these 3 groups are usually hard for the existing MDP algorithms (see Table 7) and are thus appropriate for the purpose of comparisons. We use these 10 selected instances for tuning our parameters as well as for the analysis of Section 4.

To set the parameters, we first fix those required by the TS procedure (*MaxIter*,

α) and then determine those used by the memetic algorithm (p,β). For ($MaxIter$, α), we test values for α in the range [5..25] and $MaxIter$ in the range [5000..200000]. For each of the 10 instances, we run the TS procedure 10 times, each run being limited to 60 seconds. Table 2 reports, for each instance and each parameter combination, the average solution gap to the previous best objective values (i.e., $f_{prev} - f_{avg}$) (where f_{avg} represents the average objective value obtained with the TS procedure).

To see whether there exists significant performance differences in terms of solution quality among the compared parameter combinations of $MaxIter$ and α, we apply the Friedman non-parametric statistical test followed by the Post-hoc test on the results from Table 2. This test calculates, for each problem

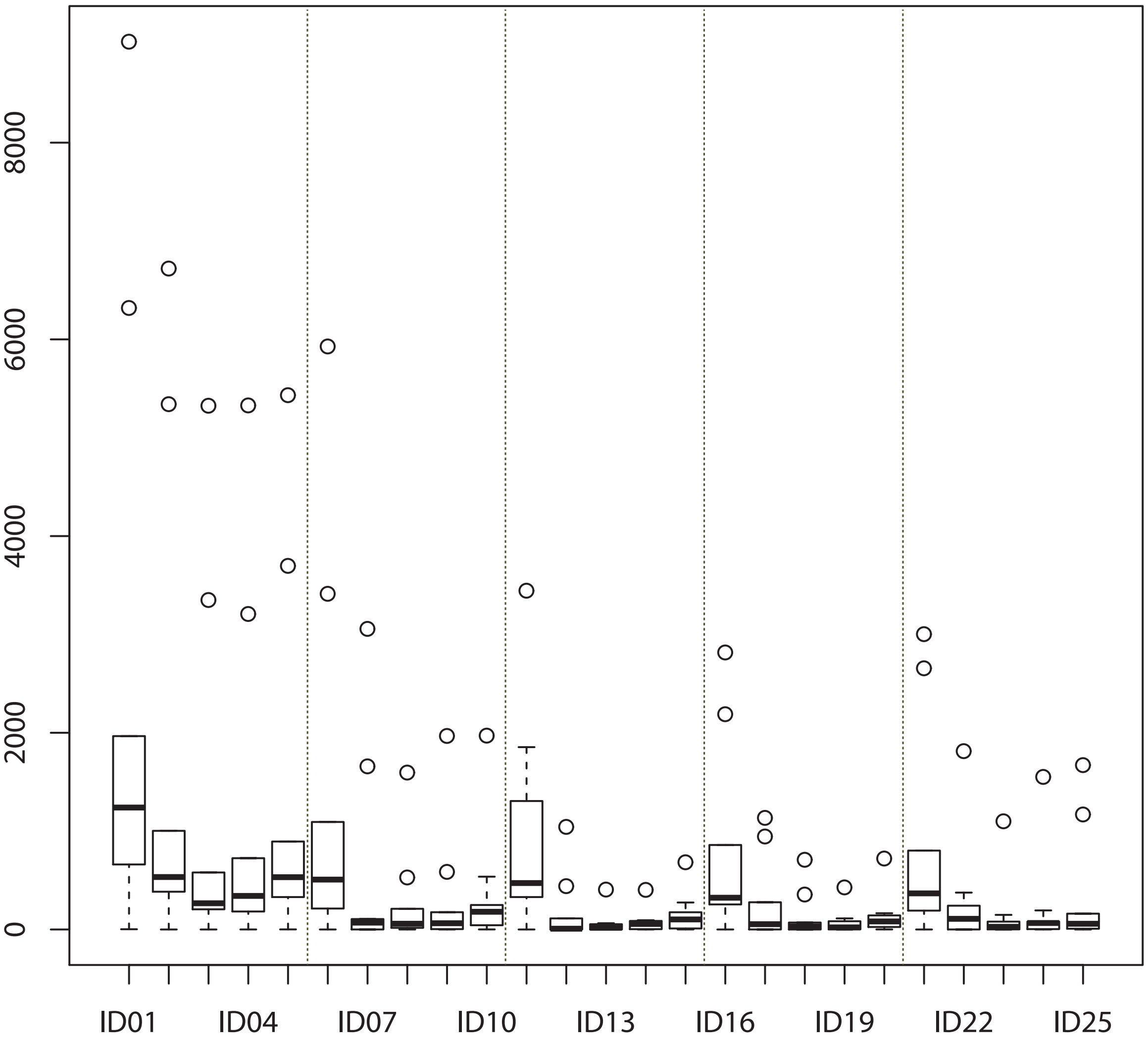


Fig. 3. Box and whisker plot of the results obtained with different parameter combinations ($MaxIter$ and α). The vertical lines separate the groups of five settings in which $MaxIter$ has the same value to ease the appreciation of the variations in each group and among the groups.

Table 1
Settings of parameters

Parameters	Section	Description	Values
p	2.2	size of population	10
α	2.3	tabu tenure management factor	15
MaxIter	2.3	number of TS iterations after recombination	50000
β	2.5	goodness score coefficient	0.6 [29]

instance, the rank value of each combination according to solution quality (where rank 25 is assigned to the best combination and rank 1 to the worst one). Then, it computes the average rank values of each combination across all the tested problem instances. The Friedman test reveals that there are statistically significant differences among the 25 tested parameter combinations (with *p-value* $= 6.457 \times 10^{-8}$). Moreover, the largest 5 associated rank values produced in this experiment are ID13(19.9), ID12(19.5), ID18(18.5), ID23(17.9) and ID19(17.7). This test gives evidence for our choice to select the parameter combination $ID13$ ($MaxIter = 50000$, $\alpha = 15$) for the TS procedure (see also Table 1). Furthermore, the Post-hoc analysis shows that our chosen setting ID13 ($MaxIter = 50000$, $\alpha = 15$) is significantly different from some other parameter combinations such as $ID1$ to $ID7$, $ID10$, $ID11$, $ID16$ and $ID21$.

To further investigate the performance of our TS procedure with 25 combinations of *MaxIter* and α, we show in Figure 3 the box and whisker plots which indicate, for each tested parameter combination, the distribution and range of the obtained results for the 10 used instances. For the sake of clarity, these results are expressed as the average solution gap to the previous best objective values f_{best} reported in the literature. From the box and whisker plot in Figure 3, we observe a visible difference in the distribution of results among the data sets obtained with the compared combinations. From Figure 3, we can also see that the five combinations $ID13$ (with *MaxIter* $= 50000$ and $\alpha = 15$), ID12 (with *MaxIter* $= 50000$ and $\alpha = 10$), $ID18$ (with *MaxIter* $= 100000$ and $\alpha = 15$), $ID23$ (with *MaxIter* $= 2000000$ and $\alpha = 15$) and $ID19$ (with *MaxIter* $= 1000000$ and $\alpha = 20$) seem to be the most robust ones, since for these combinations, the average solution gaps to the previous best objective values are generally small and the deviations from the best-known results do not vary much from one instance to another.

In addition to *MaxIter* and α, our algorithm requires two other parameters p and β. We fixed $p = 10$ (small populations are often used in memetic algorithms) while $\beta = 0.6$ is chosen according to [29]. As we see below, the adopted parameter settings are good and robust enough to allow our algorithm to yield very competitive results for the sets of the tested instances compared with those reported in the literature.

Table 2
Parameter tuning

ID	Parameter combination		Type1_22.1	Type1_22.2	b2500_1	b2500_3	b2500_5	b2500_7	p3000_1	p3000_3	p5000_2	p5000_3
	MaxIter	α										
1	5000	5	12.70	2.20	1966.00	1904.20	1512.00	967.80	661.10	859.00	6317.90	9027.70
2	5000	10	4.90	0.00	724.40	1002.80	663.20	403.40	383.80	384.10	5340.20	6720.20
3	5000	15	2.30	0.00	579.40	570.80	282.80	219.60	250.40	205.00	3348.30	5325.60
4	5000	20	7.60	0.30	725.00	723.80	320.60	182.80	342.90	339.50	3208.70	5327.90
5	5000	25	22.10	0.70	894.20	587.60	475.80	329.40	444.00	598.90	3698.10	5432.20
6	20000	5	8.90	0.00	1093.00	897.00	606.60	213.00	341.50	407.60	3414.30	5927.90
7	20000	10	0.20	0.00	83.40	71.80	56.60	66.80	106.00	0.00	1659.50	3055.10
8	20000	15	0.60	0.00	210.20	77.60	16.60	54.40	56.90	60.50	529.00	1596.00
9	20000	20	3.50	0.00	175.60	3.60	60.20	66.60	89.20	52.60	584.20	1967.10
10	20000	25	9.90	0.60	248.80	43.00	205.60	139.20	154.40	228.80	536.90	1970.70
11	50000	5	3.50	0.00	732.20	1306.00	364.80	342.00	329.50	579.30	1853.90	3444.80
12	50000	10	0.00	0.00	19.20	0.00	15.80	0.00	112.50	0.00	439.00	1043.60
13	50000	15	0.80	0.00	9.40	14.40	63.20	25.00	19.70	0.00	53.20	406.80
14	50000	20	3.00	0.00	53.40	1.20	81.60	58.00	86.80	23.20	93.80	402.80
15	50000	25	8.40	0.60	33.80	10.20	175.00	140.00	148.20	63.80	274.20	683.10
16	100000	5	10.80	0.00	539.40	859.00	312.20	254.80	333.40	295.10	2188.10	2815.60
17	100000	10	0.20	0.00	132.20	276.80	80.40	45.40	62.50	0.00	946.80	1135.20
18	100000	15	1.10	0.00	0.00	70.40	44.20	18.40	62.80	0.00	354.40	708.20
19	100000	20	4.80	0.00	37.40	1.20	41.60	112.40	83.80	0.00	5.30	428.10
20	100000	25	8.70	0.60	25.20	45.40	165.00	97.00	142.90	66.30	126.00	721.10
21	200000	5	15.00	0.00	361.60	802.00	592.80	372.60	258.20	191.70	2655.90	3004.00
22	200000	10	0.30	0.00	241.80	374.80	105.40	111.20	68.70	0.00	114.40	1813.90
23	200000	15	1.00	0.00	148.60	80.00	25.00	28.80	53.70	0.00	17.80	1099.70
24	200000	20	3.60	0.00	79.20	52.40	50.80	79.20	80.40	0.00	193.70	1551.70
25	200000	25	7.60	0.00	25.60	51.20	5.60	155.60	161.00	64.70	1168.90	1671.90

3.3 Reference Algorithms and Experimental Protocol

Our MAMDP algorithm is programmed in C and compiled using GNU GCC. All the experiments were carried out on a PC running Windows XP with an Intel Xeon E5440 processor (2.83 GHz and 8 GB of RAM).

In order to evaluate the relative effectiveness and efficiency of our proposed algorithm, we compared our MAMDP algorithm with 4 recent and best-performing MDP algorithms in the literature:

- ITS: Iterated robust tabu search algorithm (2007) [36].
- VNS: Variable neighborhood search algorithm (2009) [8].
- TIG: An fine-tuning iterated greedy algorithm (2011) [28].
- LTS-EDA: Robust learnable tabu search guided by estimation of distribution algorithm (2012) [44].

As reviewed and compared in the most recent survey [33], ITS and VNS seem to be the most powerful algorithms for the MDP among 30 heuristic algorithms. TIG and LTS-EDA are two recently proposed algorithms and thus not included in the recent review [33]. However, experimental results show that TIG and LTS-EDA obtain better or competitive performance than VNS and ITS. Especially, LTS-EDA is able to reach new best solutions [44] for some larger random instances. Thus, these 4 reference algorithms are among the most successful approaches for solving MDP actually available in the literature.

Moreover, these 4 reference algorithms are tested and compared very recently in [44] under the same time limit on a Pentium R Dual-Core CPU E5400

Table 3
Experimental Protocol: cutoff times of our MAMDP algorithm which are equal to the cutoff times of [44] divided by 1.17

Instance Family	Time limit(s)	Independent runs
Silva	17	30
Type1_55	17	30
Type1_22	17	30
Type1_52	17	30
Type2	17	30
b2500	256	30
p3000	512	15
p5000	1538	15

Table 4
Performance of MAMDP on the 20 random Type1_22 instances ($n = 2000; m = 200$)

Instance	f_{prev}	MAMDP Algorithm					
		f_{best}	g_{best}	g_{avg}	deviation	success	time(s)
Type1_22.1	114,271	114,271	0	10.23	14.28	13/30	9.44
Type1_22.2	114,327	114,327	0	0	0	30/30	4.88
Type1_22.3	114,195	114,195	0	10.23	13.85	16/30	7.91
Type1_22.4	114,093	114,093	0	7.10	11.33	13/30	9.29
Type1_22.5	114,196	114,196	0	21.90	24.19	12/30	7.25
Type1_22.6	114,265	114,265	0	7.53	9.66	16/30	8.53
Type1_22.7	114,361	114,361	0	0	0	30/30	6.34
Type1_22.8	114,327	114,327	0	6.93	25.94	28/30	6.93
Type1_22.9	114,199	114,199	0	0	0	30/30	6.78
Type1_22.10	114,229	114,229	0	10.70	14.16	15/30	8.00
Type1_22.11	114,214	114,214	0	12.60	12.24	13/30	8.52
Type1_22.12	114,214	114,214	0	14.46	8.15	5/30	15.54
Type1_22.13	114,233	114,233	0	1.00	4.02	27/30	6.67
Type1_22.14	114,216	114,216	0	0.90	4.16	28/30	5.37
Type1_22.15	114,240	114,240	0	0.60	0.48	12/30	9.56
Type1_22.16	114,335	114,335	0	2.30	5.78	24/30	6.18
Type1_22.17	114,255	114,255	0	3.83	7.17	23/30	8.09
Type1_22.18	114,408	114,408	0	0.53	1.02	23/30	8.24
Type1_22.19	114,201	114,201	0	0	0	30/30	6.06
Type1_22.20	114,349	114,349	0	14.40	29.30	24/30	8.82

processor (2.70 GHZ CPU and 2 GB of RAM). According to the Standard Performance Evaluation Cooperation (www.spec.org), this computer is 1.17 time slower than the computer we used for our experiments. To make the comparisons as fair as possible, we divide the cutoff times used in [44] by 1.17 and use the reduced times as the stop condition for our MAMDP algorithm (see Table 3).

3.4 Computational results

First, let us comment that for the four groups of test instances (Silva, Type1_55, Type1_52 and Type2), our MAMDP algorithm can easily and consistently reach all the previous best known results with a success rate of 100% within

Table 5
Performance of MAMDP on the 10 Beasley instances ($n = 2500; m = 1000$)

Instance	f_{prev}	MAMDP Algorithm					
		f_{best}	g_{best}	g_{avg}	deviation	success	time(s)
b2500-1	1153,068	1153,068	0	0	0	30/30	106.46
b2500-2	1129,310	1129,310	0	23.46	43.30	23/30	167.12
b2500-3	1115,538	1115,538	0	0	0	30/30	99.28
b2500-4	1147,840	1147,840	0	0	0	30/30	107.42
b2500-5	1144,756	1144,756	0	0	0	30/30	84.02
b2500-6	1133,572	1133,572	0	0	0	30/30	56.41
b2500-7	1149,064	1149,064	0	1.46	7.89	29/30	125.37
b2500-8	1142,762	1142,762	0	0	0	30/30	98.94
b2500-9	1138,866	1138,866	0	0.66	2.02	27/30	124.86
b2500-10	1153,936	1153,936	0	0	0	30/30	108.29

Table 6
Performance of MAMDP on the 10 large random instances ($m = 0.5 * n$)

Instance	f_{prev}	MAMDP Algorithm					
		f_{best}	g_{best}	g_{avg}	deviation	success	time(s)
p3000_1	6502,308	6502,330	-22	23.20	36.96	6/15	402.78
p3000_2	18272,568	18272,568	0	0	0	15/15	80.33
p3000_3	29867,138	29867,138	0	0	0	15/15	124.16
p3000_4	46915,044	46915,044	0	0	0	15/15	217.52
p3000_5	58095,467	58095,467	0	0	0	15/15	113.37
p5000_1	17509,215	17509,369	-154	-137.80	40.12	6/15	980.22
p5000_2	50102,729	50103,092	-363	-253.60	108.13	2/15	1209.90
p5000_3	82039,686	82040,316	-630	-345.47	279.45	2/15	1574.50
p5000_4	129413,112	129413,710	-598	-461.54	123.72	5/15	817.27
p5000_5	160597,781	160598,156	-375	-276.47	74.80	5/15	894.11

the given time limits. These instances are easy for our MAMDP algorithm (and other state-of-the-art approaches). Consequently we don't show the detailed results of these instances in the paper.

Tables 4 to 6 respectively show the computational statistics of the MAMDP algorithm on the other three groups of instances (Type1_22, b2500, p3000 and p5000) which are of larger sizes and harder to solve. In these tables, columns 1 and 2 respectively give the name of the instance and the previous best objective values (f_{prev}). Note that the previous best objective values (f_{prev}) are compiled from Tables 4, 6 and 7 presented in [36] and Tables 6 and 11 presented in [44]. To the best of our knowledge, these lower bounds are the current best known results for these instances.

Columns 3 to 8 show our results: the best objective value (f_{best}), the best solution gap to the previous best known objective values g_{best} (i.e., $f_{best} - f_{prev}$), the average solution gap to the previous best objective values g_{avg} (i.e., $f_{avg} - f_{prev}$) (where f_{avg} represents the average objective value), the standard deviation over the tested runs, the success rate (success) for reaching its f_{best}

value and the average CPU time in seconds (time) over the runs on which the f_{best} value is reached. Given that the MDP is a maximization problem, a negative value for g_{best} and g_{avg} indicates an improved outcome compared to the current best known result.

From Tables 4 to 6 as well as the results for the other four groups of instances (Silva, Type1_55, Type1_52 and Type2), we can see that for *all* these 120 instances, our MAMDP algorithm can reach the previous best known results within the time limits given in Table 3. Specifically, for 100 out of 120 instances (83%), MAMDP has a successful rate of 100%, attaining the best known objective value for each of its runs. More importantly, for 6 large and very challenging instances (p3000_1, p5000_1, p5000_2, p5000_3, p5000_4 and p5000_5), our MAMDP algorithm is able to improve on the previous best objective values.

3.5 Comparison with other algorithms

In order to further evaluate our MAMDP algorithm, in this section we compare our results with four MDP algorithms in the literature: ITS [36], VNS [8], TIG [28] and LTS-EDA [44]. As stated previously in Section 3.3, these 4 reference algorithms are the best performing approaches for MDP currently available.

Table 7 shows the best and average results of our MAMDP algorithm compared with the reference algorithms. The results of these 4 reference algorithms are compiled from Tables 6 and 11 from [44]. Note that the results of all these algorithms are obtained under the same time limit (see Section 3.3). Table 7 summarizes the solution difference between the best objective values and the average objective values obtained by these 5 algorithms with the best known objective values on the 40 large size benchmark instances with 2,000 to 5,000 elements.

From Table 7, it may be observed that the MAMDP algorithm outperforms the 4 reference algorithms, named ITS, VNS, TIG and LTS-EDA. In terms of the best solution, MAMDP matches the best known values on 34 instances and finds new best solutions for 6 out of the 40 instances, while ITS, VNS, TIG and LTS-EDA matched the best known solutions on 2, 10, 5, 19 instances respectively. Concerning the average solution value, the results of our MAMDP algorithm remains competitive when compared with these 4 reference algorithms. Indeed, for each of these 40 instances, our MAMDP algorithm is able to reach an average solution value better than each of those 4 reference algorithms.

In order to estimate the validity of our conclusion, we have applied two statistical tests to compare the average results of our MAMDP algorithm with those

Table 7
Comparison of MAMDP with four best performing MDP algorithms in the literature.

Instance	f_{pre}	MAMDP		ITS [36]		VNS [8]		TIG [28]		LTS-EDA [44]	
		g_{best}	g_{avg}	g_{best}	g_{avg}	g_{best}	g_{avg}	g_{best}	g_{avg}	g_{best}	g_{avg}
Type1_22.1	114,271	0	10.23	65	209.87	48	150.60	48	101.57	5	60.73
Type1_22.2	114,327	0	0	29	262.27	0	168.87	0	69.90	0	89.87
Type1_22.3	114,195	0	10.23	69	201.40	19	110.83	5	117.77	0	98.97
Type1_22.4	114,093	0	7.10	22	200.53	70	188.13	58	141.93	0	79.87
Type1_22.5	114,196	0	21.90	95	273.27	87	184.10	99	194.70	51	134.47
Type1_22.6	114,265	0	7.53	41	168.17	30	99.30	9	96.20	0	40.17
Type1_22.7	114,361	0	0	12	167.47	0	56.30	0	71.27	0	18.20
Type1_22.8	114,327	0	6.93	25	256.40	0	163.33	0	193.60	0	159.10
Type1_22.9	114,199	0	0	9	139.83	16	78.47	16	80.37	0	70.97
Type1_22.10	114,229	0	10.70	24	204.93	7	139.33	35	121.43	0	56.20
Type1_22.11	114,214	0	12.60	74	237.77	42	145.13	59	139.57	3	69.87
Type1_22.12	114,214	0	14.46	55	249.53	95	143.30	88	156.00	15	84.93
Type1_22.13	114,233	0	1.00	93	279.87	22	168.07	42	167.40	6	85.30
Type1_22.14	114,216	0	0.90	92	248.50	117	194.30	64	202.80	0	81.00
Type1_22.15	114,240	0	0.60	11	117.50	1	62.87	6	80.53	0	22.03
Type1_22.16	114,335	0	2.30	11	225.40	42	215.43	35	167.90	0	36.47
Type1_22.17	114,255	0	3.83	56	217.53	0	170.00	18	144.53	6	57.07
Type1_22.18	114,408	0	0.53	46	169.97	0	57.10	2	117.37	2	22.83
Type1_22.19	114,201	0	0	34	243.20	0	124.60	0	144.37	0	35.87
Type1_22.20	114,349	0	14.40	151	270.67	65	159.43	45	187.23	0	95.40
b2500-1	1153,068	0	0	624	3677.33	96	1911.93	42	1960.20	0	369.20
b2500-2	1129,310	0	23.46	128	1855.33	88	1034.33	1096	1958.47	154	454.53
b2500-3	1115,538	0	0	316	3281.93	332	1503.67	34	2647.87	0	290.40
b2500-4	1147,840	0	0	870	2547.93	436	1521.07	910	1937.13	0	461.73
b2500-5	1144,756	0	0	356	1800.27	0	749.40	674	1655.87	0	286.07
b2500-6	1133,572	0	0	250	2173.47	0	1283.53	964	1807.60	80	218.00
b2500-7	1149,064	0	1.46	306	1512.60	116	775.47	76	1338.73	44	264.60
b2500-8	1142,762	0	0	0	2467.73	96	862.47	588	1421.53	22	146.47
b2500-9	1138,866	0	0.66	642	2944.67	54	837.07	658	1020.60	6	206.33
b2500-10	1153,936	0	0	598	2024.60	278	1069.40	448	1808.73	94	305.27
p3000_1	6502,308	-22	23.20	444	1465.53	251	887.80	114	692.67	74	345.93
p3000_2	18272,568	0	0	0	1321.60	0	924.20	0	991.07	140	387.00
p3000_3	29867,138	0	0	1442	2214.73	328	963.53	820	1166.13	0	304.33
p3000_4	46915,044	0	0	1311	2243.93	254	1068.47	426	2488.20	130	317.07
p3000_5	58095,467	0	0	423	1521.60	0	663.00	278	1353.27	0	370.40
p5000_1	17509,215	-154	-137.80	2046	3410.93	848	1817.27	1000	2391.80	37	417.00
p5000_2	50102,729	-363	-253.60	2568	4444.80	1136	2277.00	183	2172.73	184	550.80
p5000_3	82039,686	-630	-345.47	4822	7612.33	1284	3064.40	1526	5377.13	74	828.53
p5000_4	129413,112	-598	-461.54	1032	4478.90	915	2367.90	1098	3276.80	260	677.20
p5000_5	160597,781	-375	-276.47	1682	4058.90	816	1903.30	914	1753.90	204	642.90

of the four reference algorithms. The first test is the non-parametric Friedman test. The resulting *p-value* of 2.2×10^{-16} obtained by this test clearly indicates that there are statistically significant differences among the average results obtained with the five compared methods. Moreover, the associated rank values produced in this experiment 5.0(MAMDP), 3.975(LTS-EDA), 2.275(TIG), 2.7(VNS) and 1.05(ITS) shows that our MAMDP algorithm outperforms the reference algorithms. The second test is the non-parametric Wilcoxon test for pairwise comparisons. For this test, the associated *p-value* < 0.05 indicates that there are significant differences between two compared methods. When applying the Wilcoxon test to compare MAMDP with each of the four reference algorithms, we obtain a *p-value* of 1.176×10^{-14} for ITS, 1.176×10^{-14} for VSN, 1.176×10^{-14} for TIG and 1.994×10^{-14} for LTS-EDA respectively. These outcomes indicate that the observed differences between MAMDP and the 4 reference methods are statistically significant.

4 Analysis of main components of MAMDP

4.1 Influence of the constrained neighborhood

In this section, we investigate the computational gain of the constrained neighborhood compared to the unconstrained neighborhood. In order to isolate the effect of the neighborhood, we use only the tabu search component of our memetic algorithm in this experiment. Precisely, we use CTS to denote the TS procedure using the constrained neighborhood as described in Section 2.3 and UTS to denote the TS procedure using the conventional unconstrained neighborhood. For UTS, at each iteration, it examines all the swap moves induced by S and $N \setminus S$ and selects the overall best swap move to generate the next solution. All the other ingredients are kept the same for both CTS and UTS.

Table 8 summarizes the comparative results between CTS and UTS on the selection of the 10 hard instances. Both algorithms were run with the same number of neighborhood explorations (with $MaxIter = 50000$) on each instance. In Table 8, we indicate the computing time in seconds required by both algorithms to realize 50000 iterations of neighborhood explorations. In addition, since the performance of CTS depends on the size of the two specifically identified subsets X and Y (see Section 2.3), we also indicate the size of these two subsets (averaged over 50000 iterations) in Table 8. In the last column of Table 8, we show the ratio of the computing time between UTS and CTS.

From Table 8, we can see that the constrained neighborhood drastically reduces the computing time of the tabu search procedure compared with the unconstrained neighborhood. Indeed, the reduction ratio ranges from 114.96 to 1280.83 for the 10 tested instances. Furthermore, the sizes of the two subsets X and Y of the constrained neighborhood are very small compared to sizes of S and $N \setminus S$. This experiment confirms clearly the interest of the constrained neighborhood in accelerating the tabu search procedure.

4.2 Contribution of crossover to the performance of MAMDP

As indicated in Section 2.4, our proposed MAMDP algorithm employs a dedicated crossover operator which tries to preserve common elements that are

Table 8
Comparative results between the unconstrained neighborhood tabu search (UTS) and the constrained neighborhood tabu search (CTS) on 10 problem instances

Instance	UTS			CTS			$\frac{T_{UTS}}{T_{CTS}}$
	$T_{UTS}(s)$	$\|S\|$	$\|N \setminus S\|$	$T_{CTS}(s)$	$\|\bar{X}\|$	$\|\bar{Y}\|$	
Type1_22.1	225.33	200	1800	1.96	13.12	16.03	114.96
Type1_22.2	225.38	200	1800	1.95	13.25	16.09	115.58
b2500-1	1261.16	1000	1500	3.02	30.29	30.96	417.60
b2500-3	1279.41	1000	1500	3.02	30.91	31.52	423.64
b2500-5	1272.32	1000	1500	3.07	30.18	30.91	414.43
b2500-7	1275.88	1000	1500	3.06	31.00	31.27	416.95
p3000_1	2501.47	1500	1500	3.56	30.68	30.98	702.66
p3000_3	2512.06	1500	1500	3.26	14.38	14.58	770.57
p5000_2	5753.75	2500	2500	4.93	17.00	16.95	1167.08
p5000_3	6237.62	2500	2500	4.87	14.50	14.57	1280.83

Table 9
Comparative results of MAMDP and TS on 10 problem instances

Instance	f_{pre}	MAMDP		TS	
		g_{best}	g_{avg}	g_{best}	g_{avg}
Type1_22.1	114,271	0	0	50	110.45
Type1_22.2	114,327	0	0	35	121.35
b2500-1	1153,068	0	278.50	43476	49295.30
b2500-3	1115,538	0	176.00	41840	48236.00
b2500-5	1144,756	0	7.90	40390	47986.70
b2500-7	1149,064	0	129.00	41860	47943.60
p3000_1	6502,308	-22	163.15	65751	71014.15
p3000_3	29867,138	0	253.70	126700	134585.60
p5000_2	50102,729	-285	1220.15	727769	757700.85
p5000_3	82039,686	-414	2293.95	848126	863409.20

shared by parent solutions. We carried out additional experiments to examine the influence of the crossover operator over the performance of our hybrid algorithm. For this purpose, we compare the performance of the MAMDP algorithm with its underlying TS algorithm. Furthermore, in order to highlight the role of the crossover operators, we weaken the underlying TS of MAMDP by reducing the number of tabu search iterations to $MaxIter = 500$.

Experiments were carried out on the selection of the 10 hard instances (Type1_22.1, Type1_22.2, b2500-1, b2500-3, b2500-5, b2500-7, p3000_1, p3000_3, p5000_2 and p5000_3). To solve each instance, we run both methods 20 times under exactly the same timeout limit, which was set to be 300 seconds for each run. In order not to penalize the TS algorithm, we use a multi-start technique to restart it every 500 iterations whenever the timeout limit is not reached.

Table 9 presents the comparative results between MAMDP and TS on the 10 instances. For each instance, the following statistics are provided: the best solution gap to the previous best known objective values g_{best} (i.e., $f_{best} - f_{prev}$) and the average solution gap to the previous best objective value g_{avg} (i.e., $f_{avg} - f_{prev}$). From Table 9, we observe that for each of these 10 instances, MAMDP performs far better than its underlying TS in terms of both best and average solution values. Furthermore, we also note that although the performance of its underlying TS algorithm is poor, the results of the hybrid MAMDP algorithm remain competitive compared with the current best known results. Indeed, for 3 instances, the MAMDP algorithm is able to improve on the current best known results. This further confirms our conclusion that the crossover operator makes an interesting contribution to the overall performance of the hybrid algorithm.

4.3 Why does the proposed crossover work

We demonstrated above that our proposed crossover operator makes a meaningful contribution to the overall performance of the hybrid algorithm. In this section, we provide empirical motivations for this crossover operator. For this purpose, we show an analysis on the structural similarity between local optima of various quality in terms of commonly shared elements. For two local optima S^t and S^s, we define their similarity as $sim(S^t, S^s) = \frac{|S^t \cap S^s|}{m}$. We can see that the larger the similarity between two solutions, the more common elements they share.

For this analysis, we employ the selection of the 10 hard instances used in the previous sections. For each instance, we collect 1000 local optima of different quality using our memetic algorithm as well as its underlying tabu search. Among these 1000 local optima, we select the top 10% (100) with the largest objective values and call them 'high-quality solutions'. Similarly, we take the bottom 10% (100) with the smallest objective values and call them 'low-quality solutions'.

Table 10 contains the data related to the similarity between our 1000 local optima. Columns S_{hq}, S_{all} and S_{lo} report respectively the average degree of similarity between the 100 high-quality solutions, the average degree of similarity between all the 1000 sampled local optima, and the average degree of similarity between the 100 low-quality solutions. From Table 10, we observe that in most cases, the degree of similarity between high-quality solutions is generally very large, from 0.52 to 0.89. High similarity indicates high quality solutions share a large number of common elements. Assume that high-quality solutions are close to an optimal solution or could themselves be optimal solutions, it is wise for a recombination operator to preserve the common elements shared by two (or more) high quality solutions. This is exactly what the pro-

Table 10
Analysis of structural similarity between high-quality solutions for 10 maximum diversity instances

Instance	S_{hq}	S_{all}	S_{lo}
Type1_22.1	0.63	0.20	0.14
Type1_22.2	0.52	0.19	0.14
b2500-1	0.81	0.62	0.44
b2500-3	0.77	0.63	0.44
b2500-5	0.79	0.63	0.44
b2500-7	0.82	0.64	0.44
p3000_1	0.86	0.68	0.53
p3000_3	0.89	0.69	0.53
p5000_2	0.86	0.59	0.51
p5000_3	0.84	0.59	0.52

posed crossover operator undertakes to do.

4.4 Population updating strategy

As shown in Section 2.5, our MAMDP algorithm uses a quality-and-distance replacement strategy (denoted by *DisQual*) for population updates to maintain the population diversity. In order to assess this strategy, we compare it with a traditional strategy (denoted by *PoolWorst*) which simply replaces the worst solution of the population by the new offspring solution. We show experimental evidences on the selection of the 10 hard instances to confirm the interest of the quality-and-distance replacement strategy. While keeping other ingredients unchanged in the MAMDP algorithm, Table 11 summarizes the comparative results between these two population updating strategies. For both strategies, we use the same running conditions as described in Table 3.

From Table 11, it can be observed that on the tested instances, MAMDP with *DisQual* matches or outperforms MAMDP with *PoolWorst* in terms of the best solution found. For one instance (p5000_2), MAMDP with *DisQual* is able to achieve a better objective value than with *PoolWorst*. Concerning the average solution value, MAMDP with *DisQual* is able to reach an average solution value better than with *PoolWorst* for 3 instances while a worse result only for one case. The comparative results indicate that MAMDP with *DisQual* is able to reach a slightly better performance than MAMDP with *PoolWorst*. However, the associated *p-value* of 0.179 with the Friedman test does not confirm a clear dominance of one strategy over the other.

Table 11
Comparative results two population updating strategies on 10 problem instances

Instance	f_{pre}	$DisQual$		$PoolWorst$	
		g_{best}	g_{avg}	g_{best}	g_{avg}
Type1_22.1	114,271	0	10.23	0	9.33
Type1_22.2	114,327	0	0	0	0
b2500-1	1153,068	0	0	0	0
b2500-3	1115,538	0	0	0	0
b2500-5	1144,756	0	0	0	0
b2500-7	1149,064	0	1.46	0	4.60
p3000_1	6502,308	-22	23.20	-22	36.40
p3000_3	29867,138	0	0	0	0
p5000_2	50102,729	-363	-253.60	-342	-206.50
p5000_3	82039,686	-630	-345.47	-630	-317.00

5 Conclusion

This paper deals with the NP-hard Maximum Diversity Problem. To approximate this hard combinatorial problem, we proposed a hybrid memetic algorithm (MAMDP) mixing a dedicated crossover operator and a constrained neighborhood tabu search procedure. The proposed crossover operator tries to preserve the elements shared by the parent solutions which hopefully belong to the optimal solution. Offspring solutions are improved with the tabu search optimization procedure which relies on a constrained neighborhood. To maintain a healthy population diversity, MAMDP applies a pool updating strategy that considers both the quality of an offspring solution and its distance to the solutions of the population.

Experimental evaluations on a large collection of 7 sets of 120 instances from the literature showed that our MAMDP algorithm attains consistently the previous best known results within a time limit ranging from 17 seconds (for problems with 100 to 500 elements) to 25 minutes (for instances with 5000 elements). Specifically, for 100 out of 120 cases (84%), MAMDP reaches the previous best known objective value for each of its runs (a successful rate of 100%). More importantly, for 6 large and very challenging instances (p3000_1, p5000_1, p5000_2, p5000_3, p5000_4 and p5000_5), our MAMDP algorithm is able to yield improved solutions (larger lower bounds) with respect to the current best known results. We also compared MAMDP with 4 best performing MDP algorithms published recently (2007, 2009, 2011, 2012) and showed that MAMDP dominates these reference algorithms in terms of solution quality under comparable experimental conditions.

We also investigated the impact of several essential components of MAMDP. We carried out experiments to demonstrate the beneficial role of the proposed crossover operator and showed an analysis of structural similarity between local optima which provides motivations for the crossover. Moreover, we also demonstrated the important role of the distance-and-quality pool updating strategy which allows MAMDP to maintain population diversity.

Acknowledgment

We are grateful to the anonymous referees for valuable suggestions and comments which helped us to improve the paper. The work is partially supported by the RaDaPop (2009-2013) and LigeRO projects (2009-2013) from the Region of Pays de la Loire, France.

References

[1] B. Alidaee, F. Glover, G. Kochenberger, H. Wang, Solving the maximum edge weight clique problem via unconstrained quadratic programming, European Journal of Operational Research 181(2) (2007) 592–597.

[2] P.M.F. de Andrade, A. Plastino, L.S. Ochi, S.L. Martins, GRASP for the maximum diversity problem. in: Proceedings of the Fifth Metaheuristics International Conference (MIC 2003), 2003, Kyoto, Japan, CD-ROM Paper: MIC0315.

[3] M.R.Q. de Andrade, P.M.F. de Andrade, S.L. Martins, A. Plastino, GRASP with path-relinking for the Maximum Diversity Problem, in: S.E. Nikoletseas (Ed.), 4th International Workshop on Experimental and Efficient Algorithms (WEA 2005), Santorini Island, Greece, Lecture Notes in Computer Science, vol. 3503, Springer, Berlin, (2005) 558–569.

[4] R. Aringhieri, R. Cordone, Y. Melzani, Tabu search versus GRASP for the Maximum Diversity Problem, 4OR-A Quarterly Journal of Operations Research, 6(1) (2008) 45–60.

[5] R. Aringhieri, M. Bruglieri, R. Cordone. Optimal results and tight bounds for the Maximum Diversity Problem, Foundations of Computing and Decision Sciences, 34(2) (2009) 73–86.

[6] R. Aringhieri, R. Cordone, Comparing local search metaheuristics for the Maximum Diversity Problem, Journal of the Operational Research Society, 62 (2) (2011) 266–280.

[7] J.E. Beasley, Obtaining test problems via internet, Journal of Global Optimization, 8 (1996) 429–433.

[8] J. Brimberg, N. Mladenović, D. Urošević, E. Ngai, Variable neighborhood search for the heaviest k-subgraph, Computers & Operations Research, 36(11) (2009) 2885–2891.

[9] B. Chandra, M.M. Halldórsson, Approximation algorithms for dispersion problems, Journal of Algorithms, 38(2) (2001) 438–465.

[10] A. Duarte, R. Martí, Tabu search and GRASP for the maximum diversity problem, European Journal of Operational Research, 178(1) (2007) 71–84.

[11] E. Erkut, S. Neuman, Comparison of four models for dispersing facilities, Information Systems and Operational Research, 29(2) (1991) 68-?86.

[12] U. Feige, G. Kortsarz, D. Peleg, The dense k-subgraph problem, Algorithmica, 29(3) (2001) 410–421.

[13] B. Feng, Z.Z. Jiang, Z.P. Fan, N. Fu, A method for member selection of crossfunctional teams using the individual and collaborative performances, European Journal of Operational Research, 203(3) (2010) 652–661.

[14] P. Galinier, Z. Boujbel, M.C. Fernandes, An efficient memetic algorithm for the graph partitioning problem, Annals of Operations Research, 191(1) (2011) 1–22.

[15] M. Gallego, A. Duarte, M. Laguna, R. Martí, Hybrid heuristics for the maximum diversity problem, Computational Optimization and Applications, 44(3) (2009) 411–426.

[16] J.B. Ghosh, Computational aspects of the Maximum Diversity Problem, Operations Research Letters, 19(4) (1996) 175–181.

[17] F. Glover. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). Discrete Applied Mathematics, 49 (1994) 231–255.

[18] F. Glover, C.C. Kuo, K.S. Dhir, Heuristic algorithms for the Maximum Diversity Problem, Journal of Information and Optimization Sciences, 19(1) (1998) 109–132.

[19] F. Glover, M. Laguna, Tabu search. Kluwer Academic Publishers, 1997.

[20] F. Glover, M. Laguna, R. Marti. Fundamentals of scatter search and path-relinking. Control and Cybernetics 39 (2000) 654–684.

[21] F. Gortázar, A. Duarte, M. Laguna, R. Martí, Black box scatter search for general classes of binary optimization problems, Computers & Operations Research 37(11) (2010) 1977–1986.

[22] R. Hassin, S. Rubinstein, A. Tamir, Approximation algorithms for maximum dispersion, Operations Research Letters 21 (1997) 133–137.

[23] J.K. Hao, Memetic algorithms in discrete optimization. In F. Neri, C. Cotta, P. Moscato (Eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Chapter 6, (2012) 73–94.

[24] K. Katayama, H. Narihisa, An evolutionary approach for the maximum diversity problem, in: W. Hart, N. Krasnogor, J.E. Smith (Eds.), Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, Springer, Berlin, 2005, pp. 31–47.

[25] R.K. Kincaid, Good solutions to discrete noxious location problems via metaheuristics, Annals of Operations Research, 40 (1–4) (1992) 265–281.

[26] M.J. Kuby, Programming models for facility dispersion: the p-dispersion and maxisum dispersion problems, Geographical Analysis, 19(4) (1987) 315–329.

[27] C.C. Kuo, F. Glover, K.S. Dhir, Analyzing and modeling the maximum diversity problem by zero-one programming, Decision Sciences, 24(6) (1993) 1171–1185.

[28] M. Lozano, D. Molina, C. García-Martínez, Iterated greedy for the Maximum Diversity Problem, European Journal of Operational Research, 214(1) (2011) 31-?38.

[29] Z. Lü, F. Glover, J.K. Hao, A hybrid metaheuristic approach to solving the UBQP problem, European Journal of Operational Research, 207(3) (2010) 1254–1262.

[30] E.M. Macambira, An application of tabu search heuristic for the maximum edge-weighted subgraph problem, Annals of Operations Research, 117 (1–4) (2002) 175–190.

[31] E.M. Macambira, C.C. de Souza, The edge-weighted clique problem: valid inequalities, facets and polyhedral computations, European Journal of Operational Research, 123(2) (2000) 346–371.

[32] R. Martí, M. Gallego, A. Duarte, A branch and bound algorithm for the Maximum Diversity Problem, European Journal of Operational Research, 200(1) (2010) 36–44.

[33] R. Martí, M. Gallego, A. Duarte, E.G. Pardo, Heuristics and metaheuristics for the Maximum Diversity Problem, Accepted to Journal of Heuristics DOI 10.1007/s10732-011-9172-4.

[34] P. Moscato, C. Cotta, A Gentle Introduction to Memetic Algorithms. In F. Glover and G. A. Kochenberger (Eds.), Handbook of Metaheuristics. (2012) Kluwer, Norwell, Massachusetts, USA.

[35] F. Neri, C. Cotta, P. Moscato (Eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Springer, 2012.

[36] G. Palubeckis, Iterated tabu search for the Maximum Diversity Problem, Applied Mathematics and Computation, 189(1) (2007) 371–383.

[37] D.C. Porumbel, J.K. Hao, P. Kuntz, An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. Computers and Operations Research, 37(10) (2010) 1822–1832.

[38] O.A. Prokopyev, N.K. Dayna, L. Martinez-Torres, The equitable dispersion problem, European Journal of Operational Research, 197(1) (2009) 59–67.

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

[40] L.F. Santos, M.H. Ribeiro, A. Plastino, S.L. Martins, A hybrid GRASP with data mining for the Maximum Diversity Problem, in: M. Blesa, C. Blum, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristics, Lecture Notes in Computer Science, vol. 3636, Springer, Berlin, 2005, pp. 116–127.

[41] G.C. Silva, L.S. Ochi, S.L. Martins, Experimental comparison of greedy randomized adaptive search procedures for the Maximum Diversity Problem, in: C. Ribeiro, C. Martins, L. Simone (Eds.), Experimental and Efficient Algorithms, Lecture Notes in Computer Science, vol. 3059, Springer, Berlin, 2004, pp. 498–512.

[42] G.C. Silva, M.R.Q. De Andrade, L.S. Ochi, S.L. Martins, A. Plastino, New heuristics for the Maximum Diversity Problem, Journal of Heuristics, 13(4) (2007) 315–336.

[43] J. Wang, Y. Zhou, J. Yin, Y. Zhang, Competitive Hopfield network combined with estimation of distribution for Maximum Diversity Problems, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39(4) (2009) 1048–1066.

[44] J. Wang, Y. Zhou, Y. Cai, J. Yin, Learnable tabu search guided by estimation of distribution for Maximum Diversity Problems, Soft Computing 16 (2012) 711–728.

[45] Q. Wu, J.K. Hao, Memetic search for the max-bisection problem, Computers & Operations Research, 40(1) (2013) 166–179.