Opposition-Based Memetic Search for the Maximum Diversity Problem

Yangming Zhou, Jin-Kao Hao, and Béatrice Duval

***Abstract*—As a usual model for a variety of practical applications, the maximum diversity problem (MDP) is computational challenging. In this paper, we present an opposition-based memetic algorithm (OBMA) for solving MDP, which integrates the concept of opposition-based learning (OBL) into the well-known memetic search framework. OBMA explores both candidate solutions and their opposite solutions during its initialization and evolution processes. Combined with a powerful local optimization procedure and a rank-based quality-and-distance pool updating strategy, OBMA establishes a suitable balance between exploration and exploitation of its search process. Computational results on 80 popular MDP benchmark instances show that the proposed algorithm matches the best-known solutions for most of instances, and finds improved best solutions (new lower bounds) for 22 instances. We provide experimental evidences to highlight the beneficial effect of OBL for solving MDP.**

***Index Terms*—Learning-based optimization, maximum diversity, memetic search, opposition-based learning (OBL), tabu search (TS).**

I. Introduction

GIVEN a set N of n elements where any pair of elements are separated by a distance, the maximum diversity problem (MDP) aims to select a subset S of m (m is given and $m < n$) elements from N in such a way that the sum of pairwise distances between any two elements in S is maximized. Let $N = \{e_1, e_2, \ldots, e_n\}$ be the given set of elements and $d_{ij} \in \mathbb{R}$ be the distance between e_i and e_j ($d_{ij} = d_{ji}$). Formally, MDP can be formulated as the following quadratic binary problem [28]:

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

$$\text{s.t.} \ \sum_{i=1}^{n} x_i = m \tag{2}$$

$$x \in \{0, 1\}^n \tag{3}$$

Manuscript received September 17, 2016; revised January 2, 2017 and February 8, 2017; accepted February 13, 2017. Date of publication February 24, 2017; date of current version September 28, 2017. *(Corresponding author: Jin-Kao Hao.)*

Y. Zhou and B. Duval are with the Department of Computer Science, LERIA, University of Angers, 49045 Angers, France (e-mail: yangming@info.univ-angers.fr; bd@info.univ-angers.fr).

J.-K. Hao is with the Department of Computer Science, LERIA Laboratory, University of Angers, 49045 Angers, France, and also with the Institut Universitaire de France, 75231 Paris, France (e-mail: hao@info.univ-angers.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2017.2674800

where the binary variable $x_k = 1$ ($k = 1, \ldots, n$) if element $e_k \in N$ is selected; and $x_k = 0$ otherwise. Equation (2) ensures that a feasible solution x exactly contains m elements.

MDP belongs to a large family of diversity or dispersion problems whose purpose is to identify a subset S from a set N of elements while optimizing an objective function defined over the distances between the elements in S [39]. Over the past three decades, MDP has been widely studied under different names, such as max-avg dispersion [46], edge-weighted clique [32], dense k-subgraph [15], maximum edge-weighted subgraph [33], and equitable dispersion [39]. In addition, MDP also proves to be a useful model to formulate a variety of practical applications including facility location, molecular structure design, agricultural breeding stocks, composing jury panels and product design [22], [35]. In terms of computational complexity, MDP is known to be NP-hard [18].

Given the interest of MDP, a large number of solution methods for MDP have been investigated. These methods can be divided into two main categories: 1) exact algorithms and 2) heuristic algorithms. In particular, exact algorithms like [3] and [34] are usually effective on small instances with $n < 150$. To handle larger instances, heuristic algorithms are often preferred to find suboptimal solutions in an acceptable time frame. Existing heuristic algorithms for MDP include construction methods [18], [22], greedy randomized adaptive search procedure (GRASP) [2], [13], [47], iterative tabu search (ITS) [38], variable neighborhood search (VNS) [4], [8], fine-tuning iterated greedy algorithm (TIG) [30], and memetic and hybrid evolutionary algorithms (MSES [11], G_SS [17], MAMDP [53], and TS/MA [52]). Comprehensive surveys and comparisons of some important heuristic algorithms prior to 2012 for MDP can be found in [4] and [35].

Recently, research into enhancing search algorithms via machine learning techniques has gained increasing interest in artificial intelligence and operations research. Machine learning is one of the most promising and salient research areas in artificial intelligence, which has experienced a rapid development and has become a powerful tool for a wide range of applications. Researchers have made much effort on using machine learning techniques to design, analyze, and select heuristics to solve large-scale combinatorial search problems [6], [26], [29], [45], [55]. Among the existing heuristics for MDP, two methods involve hybridization of heuristics and machine learning techniques. In [47], the proposed GRASP_DM algorithm combines GRASP with data mining technique (i.e., frequent itemset mining). After each GRASP phase, the data mining process extracts useful patterns from recorded elite solutions to guide the following GRASP iterations. These patterns correspond to items that are shared by a significant number of elite solutions. Another learning-based heuristic is LTS-EDA [51], which uses data mining techniques (k-means clustering and estimation of distribution algorithms) to extract useful information from the search history of tabu search (TS) in order to guide the search procedure to promising search regions. These learning-based methods have reported competitive results when they were published.

In this paper, we propose a new learning-based optimization method for solving MDP. The proposed "opposition-based memetic algorithm (OBMA)" integrates the concept of opposition-based learning (OBL) into the popular memetic algorithm (MA) framework. OBMA brings several improvements into the original MA framework. First, we employ OBL to reinforce population initialization as well as the evolutionary search process, by simultaneously considering a candidate solution and its corresponding opposite solution. Second, we apply a TS procedure for local optimization which relies on an improved parametric constrained neighborhood. Third, we propose a rank-based quality-and-distance (RBQD) pool updating strategy to maintain a healthy population diversity. We identify the main contributions of this paper as follows.

1) From an algorithmic perspective, we explore for the first time the usefulness of OBL to enhance a popular method (i.e., MA) for combinatorial optimization. We investigate how OBL can be beneficially integrated into the MA framework and show the effectiveness of the approach within the context of solving the MDP.
2) From a computational perspective, we compare the proposed OBMA algorithm with state-of-the-art results on several sets of 80 large size MDP benchmark instances with 2000–5000 elements. Our results indicate that OBMA matches most of the best-known results and in particular finds improved best solutions (new lower bounds) for 22 instances. These new bounds are valuable for the assessment of new MDP algorithms. These computational results demonstrate the competitiveness of OBMA and the benefit of using OBL to enhance an MA.

The reminder of this paper is organized as follows. After a brief introduction of OBL and memetic search in Section II, we present the proposed OBMA in Section III. Sections IV and V show computational results and comparisons as well as an experimental study of key issues of the proposed algorithm. The conclusions and perspective are provided in Section VI.

II. Background

This section introduces the concept of OBL and the general memetic search framework, which are then combined in the proposed approach.

A. Opposition-Based Learning

OBL was originally proposed as a machine intelligence scheme for reinforcement learning [49]. The main idea behind OBL is the simultaneous consideration of a candidate solution and its corresponding opposite solution. To explain the concept of opposition, we consider a real number $x \in [a, b]$, then the opposite number $\bar{x}$ is defined as $\bar{x} = a + b - x$. For the case of MDP, we define the concept of opposite solution in Section III. OBL is a fast growing research field in which a variety of new theoretical models and technical methods have been studied to deal with complex and significant problems [1], [40], [50], [54]. Recently, the idea of OBL has also been used to reinforce several *global* optimization methods such as differential evolution, particle swarm optimization, biogeography-based optimization, artificial neural network, and bee and ant colony optimization [5], [54].

To apply OBL to solve an optimization problem, one needs to answer a fundamental question: given a solution from the search space, why is it more advantageous to consider an opposite solution of the current solution than a second pure random solution? For 1-D search space, a proof and an empirical evidence confirmed how much an opposite solution is better than a uniformly generated random solution [41]. This result was further generalized to the *N*-dimensional search spaces for black-box (continuous) problems in [42].

We observe that existing studies on OBL-based optimization concerns only global optimization with two exceptions. In 2008, Ventresca and Tizhoosh [50] proposed a diversity maintaining population-based incremental learning algorithm for solving the traveling salesman problem (TSP), where the concept of opposition was used to control the amount of diversity within a given sample population. In 2011, Ergezer and Simon [14] hybridized open-path opposition and circular opposition with biogeography-based optimization for solving the graph coloring problem and TSP. The main difficulty of these applications is how to define and evaluate opposite solutions in a discrete space. OBL being a generally applicable technique, its efficiency depends on the matching degree between the definition of OBL and the solution space of the considered problem, as well as the rationality justifying a combination of OBL with a search algorithm [54].

B. Memetic Algorithm

The MA framework [27], [36] is a well-known hybrid search approach combining population-based search and local optimization. MA has been successfully applied to tackle numerous classical NP-hard problems [9], [24], such as graph coloring [31], graph partitioning [7], [16] and generalized quadratic multiple knapsack [10] as well as the MDP [11], [53].

A typical MA algorithm (Algorithm 1) begins with a set of random or constructed solutions (initial population). At each generation, MA selects two or more parent solutions from the population, and performs a recombination or crossover operation to generate one or more offspring solutions. Then a local optimization procedure is invoked to improve the offspring solution(s). Finally, a population management strategy is applied to decide if each improved offspring solution is accepted to join the population. The process repeats until a stopping condition is satisfied. We show below how OBL and

Algorithm 1 General MA Framework

Input: a problem instance and population size p
Output: the best solution S^* found
// Build an initial population
$P = \{S^1, S^2, \ldots, S^p\} \leftarrow$ PoolInitialization ()
// We suppose it is a maximization problem, record the best solution found so far
$S^* = \arg\max\{f(S^i)|i = 1, 2, \ldots, p\}$
while a stopping condition is not reached **do**
  $(S^i, \ldots, S^j) \leftarrow$ ParentsSelection (P)
  // Generate an offspring solution
  $S^o \leftarrow$ CrossoverOperator$(S^i, \ldots, S^j)$
  // Improve the offspring solution
  $S^o \leftarrow$ LocalImprovement(S^o)
  // Accept or discard the improved solution
  $P \leftarrow$ UpdatePopulation (P, S^o)
  // Update the best solution found
  if $f(S^o) > f(S^*)$ **then**
    $S^* \leftarrow S^o$
  end if
end while

Algorithm 2 OBMA for MDP

Input: a $n \times n$ distance matrix (d_{ij}), and an integer $m < n$
Output: the best solution S^* found
// Build an initial population, Section III-C
$P = \{S^1, S^2, \ldots, S^p\} \leftarrow$ OppositionBasedPoolInitialize$()$
$S^* \leftarrow \arg\max\{f(S^i) : i = 1, 2, \ldots, p\}$
while a stopping condition is not reached **do**
  Randomly select two parent solutions S^i and S^j from P
  // Generate an offspring solution and its opposite solution by crossover, Section III-E
  $S^o, \overline{S^o} \leftarrow$ BackboneBasedCrossover(S^i, S^j)
  // Perform a double trajectory search, Section III-D
  $S^o \leftarrow$ TabuSearch(S^o)/*trajectory 1: search around S^o */
  if $f(S^o) > f(S^*)$ **then**
    $S^* = S^o$
  end if
  // Insert or discard the improved solution, Section III-F
  $P \leftarrow$ RankBasedPoolUpdating (P, S^o)
  $\overline{S^o} \leftarrow$ TabuSearch$(\overline{S^o})$/*trajectory 2: search around $\overline{S^o}$ */
  if $f(\overline{S^o}) > f(S^*)$ **then**
    $S^* = \overline{S^o}$
  end if
  // Insert or discard the improved solution, Section III-F
  $P \leftarrow$ RankBasedPoolUpdating $(P, \overline{S^o})$
end while

MA can be combined to obtain a powerful search algorithm for the highly combinatorial MDP.

III. Opposition-Based Memetic Algorithm for MDP

We describe in this section the proposed OBMA for MDP. We start with the issues of solution representation and search space, followed by a detailed presentation of the ingredients of the proposed approach.

A. Solution Representation and Search Space

Given an MDP instance with set $N = \{e_1, e_2, \ldots, e_n\}$ and integer m, any subset $S \subset N$ of size m is a feasible solution. A candidate solution S can then be represented by $S = \{e_{S(1)}, e_{S(2)}, \ldots, e_{S(m)}\}$ such that $S(i)$ is the index of element i in N or equivalently by a binary vector of size of n such that exactly m variables receive the value of 1 and the other $n - m$ variables receive the value of 0. The quality of a candidate solution S is assessed by the objective function f of (1).

Given an MDP instance, the search space Ω is composed of all the m-element subsets of N, i.e., $\Omega = \{S \subset N : |S| = m\}$. The size of Ω is given by $\binom{n}{m} = [n!/(m!(n-m)!)]$ and increases extremely fast with n and m.

B. Main Scheme

The proposed OBMA algorithm for MDP is based on opposition learning, which relies on the key concept of opposite solution in the context of MDP, which is defined as follows.

Definition 1 (Opposite Solution): Given an MDP instance with set N and integer m, let S be a feasible solution of Ω represented by its binary n-vector x, an opposite solution $\overline{S}$ corresponds to a feasible binary vector $\overline{x}$ whose components match the complement[1] of x as closely as possible.

[1]Let $x \in \{0, 1\}^n$, its complement $\overline{x}$ is an n-vector such that $\overline{x}[i] = 1$ if $x[i] = 0$; $\overline{x}[i] = 0$ if $x[i] = 1$.

According to this definition, if $m < (n/2)$, $\overline{S}$ corresponds to any subset of elements of size of m from $N \backslash S$. If $m = (n/2)$, the unique opposite solution is given by $\overline{S} = N \backslash S$. If $m > (n/2)$, $\overline{S}$ is any subset of $n - m$ elements from $N \backslash S$, completed by other $2m - n$ elements from S.

We observe that a diversification framework introduced in [21] also yields the type of opposite solution provided by our definition and applies to constraints more general than the constraint defined by (2). We make two related comments. First, as we observe in Section IV, the MDP benchmark instances in the literature correspond to the case $m \leq (n/2)$. Second, in practice, when an opposite solution is required while there are multiple opposite solutions, we can just select one solution at random among the candidate opposite solutions.

The proposed OBMA algorithm consists of four key components: an opposition-based population initialization procedure, a backbone-based crossover operator, an opposition-based double trajectory search (ODTS) procedure and an RBQD pool updating strategy. OBMA starts from a collection of diverse elite solutions which are obtained by the opposition-based initialization procedure (Section III-C). At each generation, two parent solutions are selected at random from the population, and then the backbone-based crossover operator (Section III-E) is applied to the selected parents to generate an offspring solution and a corresponding opposite solution. Subsequently, the ODTS procedure (Section III-D) is invoked to search simultaneously from the offspring solution and its opposite solution. Finally, the rank-based pool updating strategy (Section III-F) decides whether these two improved offspring solutions should be inserted into the population. This process repeats until a stopping condition (e.g., a time limit) is satisfied. The general framework of the OBMA algorithm is shown in Algorithm 2 while its components are described in the following sections.

Algorithm 3 Opposition-Based Population Initialization

Input: population size p
Output: an initial population $P = \{S^1, S^2, \ldots, S^p\}$
$count = 0$
while $count < p$ **do**
  /$*$ Generate a random solution S^r and its opposite solution $\overline{S^r}$ $*$/
  $S^r \leftarrow$ TabuSearch (S^r)
  $\overline{S^r} \leftarrow$ TabuSearch ($\overline{S^r}$)
  /$*$ Identify the better solution between S^r and $\overline{S^r}$ $*$/
  $S' \leftarrow \arg\max\{f(S^r), f(\overline{S^r})\}$
  /$*$ Insert S' into the population P or modify it $*$/
  if S' is different from any solutions in the P **then**
    Add S' into the population P directly
  else
    Modify S' and add it into the population P
    $count \leftarrow count + 1$
  end if
end while

Algorithm 4 Parametric Constrained Neighborhood TS

Input: a starting solution S, the maximum allowed iterations $MaxIter$
Output: the best solution S^* found
$S^* \leftarrow S$
$iter \leftarrow 0$
Initialize the tabu list
Calculate the $gain(e_i)$ for each element $e_i \in N$ according to Eq. (4)
while $iter < MaxIter$ **do**
  $minGain \leftarrow \min\{gain(e_i) : e_i \in S\}$
  Determine subset U_S^c according to Eq. (6)
  $maxGain \leftarrow \max\{gain(e_i) : e_i \in N \backslash S\}$
  Determine subset $U_{N \backslash S}^c$ according to Eq. (7)
  Choose a best eligible $swap(e_u, e_v)$ (see Sect. III-D3)
  $S \leftarrow S \setminus \{e_u\} \cup \{e_v\}$
  Update the tabu list and $gain(e_i)$ for each element $e_i \in N$ according to Eq. (8)
  if $f(S) > f(S^*)$ **then**
    $S^* \leftarrow S$
  end if
  $iter \leftarrow iter + 1$
end while

C. Opposition-Based Population Initialization

The initial population P is composed of p diverse and high quality solutions. Unlike traditional population initialization, our population initialization procedure integrates the concept of OBL. As shown in Algorithm 3, the OBL-based initialization procedure considers not only a random candidate solution but also a corresponding opposite solution. Specifically, we first generate a pair of solutions, i.e., a random solution $S_r \in \Omega$ and a corresponding opposite solution $\overline{S}_r$ according to Definition 1 (if multiple opposite solutions exist, one of them is taken at random). These two solutions are then improved by the TS procedure described in Section III-D. Finally, the better one of the two improved solutions S' is inserted into the population if S' is not the same as any existing individual of the population. Otherwise, we modify S' with the swap operation (see Section III-D1) until S' becomes different from all individuals in P before inserting it into the population. This procedure is repeated until the population is filled up with p solutions. With the help of this initialization procedure, the initial solutions of P are not only of good quality, but also of high diversity.

D. Opposition-Based Double Trajectory Search Procedure

In the proposed OBMA algorithm, we use an ODTS procedure for local optimization. ODTS simultaneously searches around an offspring solution S^o and one opposite solution $\overline{S^o}$. The local optimization procedure used here is an improved constrained neighborhood TS. TS is a well-known metaheuristic that guides a local search heuristic to explore the solution space beyond local optimality [20]. The original constrained neighborhood TS algorithm was proposed in [53], which is specifically designed for MDP by using a constrained neighborhood and a dynamic tabu tenure management mechanism. Compared with this TS algorithm, our improved TS procedure (see Algorithm 4) distinguishes itself by its parametric constrained neighborhood which allows the search process to explore more promising candidate solutions. In the following, we present the key ingredients of this local optimization procedure including the parametric constrained neighborhood, the fast neighborhood evaluation technique and the dynamic tabu tenure management scheme.

1) Parametric Constrained Neighborhood: In general, local search for MDP starts from an initial solution S and subsequently swaps an element of S and an element of $N \backslash S$ according to some specific transition rule (e.g., accepting the first or the best improving transition). Clearly, the size of this neighborhood is bound by $O(m(n - m))$ and an exhaustive exploration of all the possible swap moves is too time-consuming for the large values of n. To reduce the size of the swap neighborhood, we employ an extension of a candidate list strategy sometimes called a neighborhood decomposition strategy [19] or a successive filtration strategy [44], and which we refer to as a constrained swap strategy [53]. As it is shown in Section V-A, although this constrained swap strategy accelerates the search process, it imposes a too strong restriction and may exclude some promising swap moves for the TS procedure. In this paper, we introduce the *parametric constrained neighborhood* which adopts the idea of the constrained neighborhood, but weakens the imposed constraint by introducing a parameter ρ ($\rho \geqslant 1$) to control the size of the explored neighborhood. Both constrained neighborhoods rely on the notion of move gain of each element e_i with respect to the objective value of the current solution S defined as follows:

$$\mathrm{gain}(e_i) = \sum_{e_j \in S} d_{ij}, \; i = 1, 2, \ldots, m. \tag{4}$$

Let $\mathrm{swap}(e_u, e_v)$ denote the swap operation which exchanges an element $e_u \in S$ against an element $e_v \in N \backslash S$. Once a swap $S \xrightarrow{\mathrm{swap}(e_u, e_v)} S'$ is made, it provides a new solution $S' = S \backslash \{e_u\} \cup \{e_v\}$ and the move gain Δ_{uv} of this swap can be calculated according to the following formula:

$$\Delta_{uv} = f\left(S'\right) - f(S) = \mathrm{gain}(e_v) - \mathrm{gain}(e_u) - d_{uv}. \tag{5}$$

Equation (5) suggests that in order to maximize the move gain, it is a good strategy to consider swap moves that replaces in the current solution S an element e_u with a small gain by an element e_v out of S with a large gain. In other words, the search process can only consider swap moves that involve an element e_{u^*} from S with the minimal gain value and an element e_{v^*} in $N \backslash S$ with a maximal gain value. However, the move gain also depends on the distance $d_{u^*v^*}$ between e_{u^*} and e_{v^*}. These remarks lead to the following definition for the parametric constrained neighborhood.

For a current solution S, let minGain $= \min\{\text{gain}(e_i) : e_i \in S\}$ and maxGain $= \max\{\text{gain}(e_i) : e_i \in N \backslash S\}$ and $d_{\max} = \max\{d_{ij} : 1 \leq i < j \leq n\}$. The parametric constrained neighborhood relies on the two following sets:

$$U_S^c = \left\{ e_i \in S : \text{gain}(e_i) \leq \text{minGain} + \frac{\rho}{2} d_{\max} \right\} \tag{6}$$

and

$$U_{N \backslash S}^c = \left\{ e_i \in N \backslash S : \text{gain}(e_i) \geqslant \text{maxGain} - \frac{\rho}{2} d_{\max} \right\}. \tag{7}$$

Therefore, a constrained neighbor solution S' can be obtained from S by swapping one element $e_u \in U_S^c$ and another element $e_v \in U_{N \backslash S}^c$. Clearly, the evaluation of all constrained neighboring solutions can be achieved in $O(|U_S^c| \times |U_{N \backslash S}^c|)$. Conveniently, we can adjust the value of parameter ρ ($\rho \geqslant 1$) to control the size of the constrained neighborhood.

One notices that the neighborhood of [53] is a special case of the above neighborhood when $\rho = 2$. In general, a larger ρ value leads to a less constrained neighborhood compared to the neighborhood of [53], allowing thus additional promising candidate solutions to be considered by the TS procedure. Section V-A confirms the effectiveness of this parametric constrained neighborhood.

2) Fast Neighborhood Evaluation Technique: Once a swap(e_u, e_v) move is performed, we need to update the gains gain(e_i) affected by the move. To rapidly determine the gain of each element e_i, we resort to the fast neighborhood evaluation technique used in [2], [4], and [53]

$$\text{gain}(e_i) = \begin{cases} \text{gain}(e_i) + d_{iv} & \text{if} \quad e_i = e_u \\ \text{gain}(e_i) - d_{iu} & \text{if} \quad e_i = e_v \\ \text{gain}(e_i) + d_{iv} - d_{iu} & \text{if} \quad e_i \neq e_u \text{ and } e_i \neq e_v. \end{cases} \tag{8}$$

Updating the gains of n elements requires $O(n)$ time. Therefore, the time to update the parametric constrained neighborhood at each iteration is bounded by $O(n) + O(|U_S^c| \times |U_{N \backslash S}^c|)$.

3) Dynamic Tabu Tenure Management Scheme: Starting with a solution S, TS iteratively visits a series of neighboring solutions generated by the swap operator. At each iteration, a *best swap* (i.e., with the maximum move gain Δ_{uv}) is chosen among the eligible swap moves to transform the current solution even if the resulting solution is worse than the current solution. To prevent the search from cycling among visited solutions, TS typically incorporates a short-term history memory H, known as the tabu list [20].

Initially, all elements are eligible for a swap operation. Once a swap(e_u, e_v) is performed, we record it in the tabu list H to mark element e_u as tabu, meaning that element e_u is forbidden to join again solution S during the next T_u iterations (T_u is called the tabu tenure). Similarly, element e_v is also marked as tabu for the next T_v iterations and thus cannot be removed from S during this period. The tabu status of an element is disabled if the swap operation with this element leads to a solution better than any already visited solution (this rule is called the *aspiration criterion* in TS). An eligible swap move involves only elements that are not forbidden by the tabu list or satisfy the aspiration criterion.

It is important to determine a suitable tabu tenure for the elements of a swap. Yet, there does not exist a universally applicable tabu tenure management scheme. In this paper, we adopt a dynamic tabu list management technique which was proposed in [16] and proved to work well for MDP [53]. The tabu tenure T_x of an element e_x taking part in a swap operation is determined according to a periodic step function $T(\text{iter})$, where iter is the number of iterations. Specifically, $T(\text{iter})$ takes the value of α (a parameter set to 15 in this paper), $2 \times \alpha$, $4 \times \alpha$, and $8 \times \alpha$ according to the value of iter, and each $T(\text{iter})$ value is kept for 100 consecutive iterations (see [16], [53] for more details). Following [53], we set $T_u = T(\text{iter})$ for the element e_u dropped from the solution and $T_v = 0.7 * T(\text{iter})$ for the element e_v added to the solution.

To implement the tabu list, we use an integer vector H of size n whose components are initially set to 0 (i.e., $H[i] = 0, \forall i \in [1, \dots, n]$). After each swap$(e_u, e_v)$ operation, we set $H[u]$ (resp. $H[v]$) to iter $+ T_u$ (resp. iter $+ T_v$), where iter is the current number of iterations and T_u (resp. T_v) is the tabu tenure explained above. With this implementation, it is very easy to know whether an element e_i is forbidden by the tabu list as follows. If iter $\leq H[i]$, then e_i is forbidden by the tabu list; otherwise, e_i is not forbidden by the tabu list.

E. Backbone-Based Crossover Operator

The crossover operator plays a critical role in memetic search and defines the way information is transmitted from parents to offspring [24]. A meaningful crossover operator should preserve good properties of parent solutions through the recombination process. In our case, we adopt a backbone-based crossover operator which generates an offspring solution in the same way as in [53] while introducing additionally an opposite solution. For MDP, the backbone is a good property that is to be transmitted from parents to offspring, as shown in Definition 2. Specially, the backbone-based crossover operator not only produces an offspring solution, but also creates a corresponding opposite solution.

Definition 2 (Backbone [24], [53]): Let S^u and S^v be two solutions of MDP, the backbone of S^u and S^v is defined as the set of common elements shared by these two solutions, i.e., $S^u \cap S^v$.

Given a population $P = \{S^1, S^2, \dots, S^p\}$ of p individuals, an offspring solution is constructed in two phases. The first phase randomly selects two parents S^u and S^v in P and identifies the backbone which is used to form the partial offspring S^o, i.e., $S^o = S^u \cap S^v$. If $|S^o| < m$, then the second phase successively extends S^o with $m - |S^o|$ other elements in a greedy way.

Specifically, we alternatively consider each parent and select an unassigned element with maximum gain with respect to S^o until S^o reaches the size of m. Once the offspring solution S^o is obtained, we generate its corresponding opposite solution $\overline{S^o}$ according to Definition 1. Consequently, we obtain two different and distant offspring solutions S^o and $\overline{S^o}$ which are further improved by the TS procedure of Section III-D.

F. Rank-Based Pool Updating Strategy

To maintain a healthy diversity of the population, we use a rank-based pool updating strategy to decide whether the improved solutions (S^o and $\overline{S^o}$) should be inserted into the population or discarded. This pool updating strategy simultaneously considers the solution quality and the distance between individuals in the population to guarantee the population diversity. Similar quality-and-distance pool updating strategies have been used in MAs in [10], [31], [48], and [53].

For two solutions S^u and S^v, we use the well-known set-theoretic partition distance [23] to measure their distance

$$D\left(S^u, S^v\right) = m - \text{Sim}\left(S^u, S^v\right) \tag{9}$$

where $\text{Sim}(S^u, S^v) = |S^u \cap S^v|$ denotes the number of common elements shared by S^u and S^v.

Given a population $P = \{S^1, S^2, \ldots, S^p\}$ and one solution S^i in P, the average distance between S^i and the remaining individuals in P is computed by [10]

$$\text{AD}\left(S^i, P\right) = \frac{1}{p} \sum_{S^j \in P, j \neq i} D\left(S^i, S^j\right). \tag{10}$$

To update the population with an improved offspring solution (S^o or $\overline{S^o}$), let us consider the case of S^o (the same procedure is applied to $\overline{S^o}$). We first tentatively insert S^o into the population P, i.e., $P' \leftarrow P \cup \{S^o\}$. Then all individuals in P' are assessed based on the following function:

$$\text{Score}\left(S^i, P'\right) = \beta * \text{RF}\left(f\left(S^i\right)\right) + (1 - \beta) * \text{RF}\left(\text{AD}\left(S^i, P'\right)\right) \tag{11}$$

where $\text{RF}(f(S^i))$ and $\text{RF}(\text{AD}(S^i, P'))$ represent, respectively, the rank of solution S^i with respect to its objective value and the average distance to the population. Specifically, $\text{RF}(\cdot)$ ranks the solutions of P in decreasing order according to their objective values or their average distances to the population. In case of ties, the solution with the smallest index is preferred. β is the weighting coefficient between the objective value of the solution and its average distance to the population, which is empirically set to $\beta = 0.6$.

Based on this scoring function, we identify the worst solution S^w with the largest score value from the population P'. If the worst solution S^w is not the improved offspring S^o, then the population is updated by replacing S^w by S^o; otherwise, S^o is simply discarded.

G. Computational Complexity of OBMA

To analyze the computational complexity of the proposed OBMA algorithm, we consider the main steps in one generation in the main loop of Algorithm 2.

As shown in Algorithm 2, each generation of the OBMA algorithm is composed of four components: parents selection, backbone-based crossover, TS, and rank-based pool updating strategy. The step of parents selection is bounded by $O(1)$. The backbone-based crossover operation can be achieved in $O(nm^2)$. The computational complexity of the parametric constrained neighborhood search procedure is $O((n + |U_S^c| \times |U_{N\setminus S}^c|)\text{MaxIter})$, where $|U_S^c|$ is the number of elements that can be swapped out from S, $|U_{N\setminus S}^c|$ is the number of elements in $N\setminus S$ that can be swapped into S, and MaxIter is the allowable maximum number of iterations in TS. The computational complexity for the pool updating strategy is $O(p(m^2 + p))$, where p is the population size. To summarize, the total computational complexity of the proposed OBMA within one generation is $O(nm^2 + (n + |U_S^c| \times |U_{N\setminus S}^c|)\text{MaxIter})$.

IV. Computational Results

This section presents computational experiments to test the efficiency of our OBMA algorithm. We aim to: 1) demonstrate the added value of OBMA (with OBL) compared to the memetic search framework (without OBL) and 2) evaluate the performance of OBMA with respect to the best-known results ever reported by state-of-the-art algorithms in the literature.

A. Benchmark Instances

Our computational assessment were based on 80 large instances with 2000–5000 elements which are classified into the following sets.

Set I contains three data sets: 1) MDG-a (also known as Type1_22); 2) MDG-b; and 3) MDG-c. They are available at http://www.optsicom.es/mdp/.

1) *MDG-a:* This data set consists of 20 instances with $n = 2000$ and $m = 200$. The distance d_{ij} between any two elements i and j is an integer number which is randomly selected between 0 and 10 from a uniform distribution.
2) *MDG-b:* This data set includes 20 instances with $n = 2000$ and $m = 200$. The distance d_{ij} between any two elements i and j is a real number which is randomly selected between 0 and 1000 from a uniform distribution.
3) *MDG-c:* This data set is composed of 20 instances with $n = 2000$ and $m = 300$, 400, 500, and 600. The distance d_{ij} between any two elements i and j is an integer number which is randomly selected between 0 and 1000 from a uniform distribution.

Set II (b2500) contains ten instances with $n = 2500$ and $m = 1000$, where the distance d_{ij} between any two elements e_i and e_j is an integer randomly generated from $[-100, 100]$. This data set was originally derived from the unconstrained binary quadratic programming problem by ignoring the diagonal elements and is part of ORLIB: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/.

Set III (p3000 and p5000) contains five very large instances with $n = 3000$ and $m = 1500$, and five instances with $n = 5000$ and $m = 2500$, where d_{ij} are integers generated

TABLE I
EIGHTY LARGE BENCHMARK INSTANCES USED IN THE EXPERIMENTS

Data set	n	m	#instance	time limit (s)	#run
MDG-a	2000	200	20	20	30
MDG-b	2000	200	20	600	15
MDG-c	2000	300-600	20	600	15
b2500	2500	1000	10	300	30
p3000	3000	1500	5	600	15
p5000	5000	2500	5	1800	15

TABLE II
PARAMETER SETTING OF THE OBMA ALGORITHM

Parameters	Description	Value	Section
p	population size	10	III-C
$MaxIter$	allowable number of iterations of TS	50,000	III-D
α	tabu tenure management factor	15	III-D
ρ	scale coefficient	4	III-D
β	weighting coefficient	0.6	III-F

from a [0, 100] uniform distribution. The sources of the generator and input files to replicate this data set can be found at: http://www.proin.ktu.lt/~gintaras/mdgp.html.

B. Experimental Settings

Our algorithm[2] was implemented in C++, and complied using GNU gcc 4.1.2 with "−O3" option on an Intel E5-2670 with 2.5 GHz and 2-GB RAM under Linux. Without using any compiler flag, running the DIMACS machine benchmark program dfmax[3] on our machine requires 0.19, 1.17, and 4.54 s to solve graphs r300.5, r400.5, and r500.5, respectively. To obtain our experimental results, each instance was solved according to the settings (including time limit and number of runs) provided in Tables I and II. Notice that, like most reference algorithms of Section IV-D, we used a cutoff time limit (instead of fitness evaluations) as the stopping condition. This choice is suitable in the context of MDP given that its fitness evaluation is computationally cheap enough, contrary to expensive-to-evaluate problems like many engineering optimization problems where using fitness evaluations is a standard practice [25].

C. Benefit of OBL for Memetic Search

To verify the benefit of OBL for memetic search, we compare OBMA with its alternative algorithm OBMA_0 without OBL. To obtain OBMA_0, two modifications have been made on OBMA.

1) For the population initialization phase, we randomly generate two initial solutions at a time (instead of a random solution and an opposite solution).
2) For the crossover phase, we perform twice the crossover operation to generate two offspring solutions (instead of one offspring solution and an opposite solution).

[2]The best solution certificates and our program will be made available at http://www.info.univ-angers.fr/pub/hao/OBMA.html.

[3]dfmax: ftp://dimacs.rutgers.edu/pub/dsj/clique.

To make a fair comparison between OBMA and OBMA_0, we ran both algorithms under the same conditions, as shown in Tables I and II. The comparative results for the five data sets are summarized in Tables III–VII.

In these tables, columns 1 and 2, respectively, show for each instance its name (Instance) and the current best objective value (f_{prev}) jointly reported in recent studies [17], [35], [51], [53]. Columns 3–7 report the results of the OBMA_0 algorithm: the difference between f_{prev} and the best objective value f_{best} (i.e., $\Delta f_{\text{best}} = f_{\text{prev}} - f_{\text{best}}$), the difference between f_{prev} and average objective value f_{avg} (i.e., $\Delta f_{\text{avg}} = f_{\text{prev}} - f_{\text{avg}}$), the standard deviation of objective values (σ), the average CPU time to attain the best objective values (t_{best}) and the success rate (#succ) over 30 or 15 independent runs. Columns 8–12 present the same information of the OBMA algorithm. The best values among the results of the two compared algorithms are indicated in bold. At the last row, we also provide the average number of instances for which one algorithm outperforms the other algorithm. 0.5 is assigned to each compared algorithm in case of ties.

To analyze these results, we resort to a widely used statistical methodology known as *two-tailed sign test* [12]. This test is a popular way to compare the overall performance of algorithms by counting the number of winning instances of each compared algorithm and thus to identify the overall winner algorithm. The test makes the null hypothesis that the compared algorithms are equivalent. The null hypothesis is accepted if each algorithm wins on approximately $X/2$ out of X instances. Otherwise, the test rejects the null hypothesis, suggesting a difference between the compared algorithms. The critical values (CVs) for the two-tailed sign test at a significance level of 0.05 are, respectively, $\text{CV}_{0.05}^{20} = 15$ for $X = 20$ instances and $\text{CV}_{0.05}^{10} = 9$ for $X = 10$ instances. In other words, algorithm A is significantly better than algorithm B if A performs better than B for at least $\text{CV}_{0.05}^{X}$ instances for a data set of X instances.

From Table III which shows the results of OBMA_0 and OBMA for the 20 MDG-a instances, we first observe that both algorithms attain the best-known results reported in the literature. However, OBMA performs better than OBMA_0 in terms of the average objective value and success rate, and wins 14.5 and 13.5 instances, respectively. We also observe that the standard deviation of the best objective values is significantly smaller for OBMA, and OBMA wins 14.5 instances, which is very close to the critical value ($\text{CV}_{0.05}^{20} = 15$). Finally, compared to OBMA_0, OBMA needs less average CPU time to find the best-known solutions for all instances except MDG-a_26 and wins 13.5 instances in terms of the success rate.

Table IV shows the results of OBMA_0 and OBMA for the 20 MDG-b instances. The best-known objective values (f_{prev}) of this data set were obtained by a scatter search algorithm (G_SS) [17] with a time limit of 2 h on an Intel Core 2 Quad CPU 8300 with 6 GB of RAM running Ubuntu 9.04 [35]. This table indicates that both OBMA and OBMA_0 find improved best-known solutions for 14 out of 20 instances and attain the best objective values for the remaining six instances. On the other hand, compared to the OBMA_0 algorithm, OBMA

TABLE III
COMPARISON OF THE RESULTS OBTAINED BY $OBMA_0$ AND OBMA ON THE DATA SET MDG-A

		$OBMA_0$					OBMA				
Instance	f_{prev}	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ
MDG-a_21	114271	**0**	**4.5**	**10.6**	9.3	**22/30**	**0**	5.2	10.8	**7.4**	20/30
MDG-a_22	114327	**0**	4.2	22.6	9.2	**29/30**	**0**	**0.1**	**0.5**	**7.2**	**29/30**
MDG-a_23	114195	**0**	**10.9**	15.2	11.5	**15/30**	**0**	15.2	**15.1**	**7.0**	11/30
MDG-a_24	114093	**0**	25.3	21.0	11.3	4/30	**0**	**11.2**	**12.1**	**8.2**	**7/30**
MDG-a_25	114196	**0**	55.9	32.7	13.3	1/30	**0**	**41.5**	**30.7**	**8.8**	**5/30**
MDG-a_26	114265	**0**	**7.3**	**10.2**	**9.9**	**17/30**	**0**	**10.2**	11.5	11.6	14/30
MDG-a_27	114361	**0**	**0.0**	**0.0**	4.8	**30/30**	**0**	0.2	0.9	**4.1**	29/30
MDG-a_28	114327	**0**	57.1	53.8	15.2	12/30	**0**	**18.9**	**39.2**	**7.9**	**23/30**
MDG-a_29	114199	**0**	11.2	16.0	14.6	8/30	**0**	**4.4**	**8.4**	**9.0**	**14/30**
MDG-a_30	114229	**0**	12.8	16.3	10.5	**14/30**	**0**	**8.1**	**10.5**	**7.2**	**14/30**
MDG-a_31	114214	**0**	30.4	22.7	16.2	5/30	**0**	**16.7**	**13.6**	**11.9**	**9/30**
MDG-a_32	114214	**0**	28.5	19.9	10.4	**4/30**	**0**	**23.7**	**17.1**	**6.0**	3/30
MDG-a_33	114233	**0**	6.1	10.5	12.8	15/30	**0**	**2.0**	**5.6**	**8.8**	**23/30**
MDG-a_34	114216	**0**	25.4	43.8	11.6	15/30	**0**	**2.4**	**7.0**	**6.6**	**26/30**
MDG-a_35	114240	**0**	**1.6**	**2.2**	12.2	9/30	**0**	**1.6**	2.4	**10.7**	**11/30**
MDG-a_36	114335	**0**	7.5	11.7	12.4	19/30	**0**	**5.7**	**9.5**	**10.5**	**21/30**
MDG-a_37	114255	**0**	**4.2**	**8.2**	12.2	**18/30**	**0**	5.2	8.6	**7.8**	**18/30**
MDG-a_38	114408	**0**	1.2	3.1	10.6	19/30	**0**	**0.5**	**1.1**	**7.6**	**25/30**
MDG-a_39	114201	**0**	2.2	**6.0**	9.7	26/30	**0**	**2.0**	**6.0**	**4.9**	**27/30**
MDG-a_40	114349	**0**	28.1	37.7	9.9	18/30	**0**	**23.0**	**31.5**	**9.1**	**19/30**
	wins	10	5.5	5.5	1	6.5	10	14.5	14.5	19	13.5

The f_{prev} values were obtained by several algorithms including LTS-EDA [51] and MAMDP [53].

TABLE IV
COMPARISON OF THE RESULTS OBTAINED BY $OBMA_0$ AND OBMA ON THE DATA SET MDG-B

		$OBMA_0$					OBMA				
Instance	f_{prev}	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ
MDG-b_21	11299895	**0**	0.2	**0.0**	378.6	**15/15**	**0**	0.2	**0.0**	**325.1**	**15/15**
MDG-b_22	11286776	**-5622**	**-5622.2**	**0.0**	**336.5**	**15/15**	**-5622**	**-5622.2**	**0.0**	380.8	**15/15**
MDG-b_23	11299941	**0**	**0.5**	**0.0**	300.5	**15/15**	**0**	**0.5**	**0.0**	**283.4**	**15/15**
MDG-b_24	11290874	**-245**	**-229.1**	**61.2**	345.5	**14/15**	**-245**	-220.5	67.2	**323.4**	13/15
MDG-b_25	11296067	**-1960**	**-1959.9**	**0.0**	**271.2**	**15/15**	**-1960**	**-1959.9**	**0.0**	313.9	**15/15**
MDG-b_26	11292296	**-6134**	-5216.0	1836.9	**276.8**	12/15	**-6134**	**-6134.4**	**0.0**	336.0	**15/15**
MDG-b_27	11305677	**0**	**0.2**	**0.0**	330.0	**15/15**	**0**	**0.2**	**0.0**	**256.0**	**15/15**
MDG-b_28	11279916	**-2995**	-2994.6	0.5	**329.5**	10/15	**-2995**	**-2994.7**	**0.4**	351.1	**12/15**
MDG-b_29	11297188	**-151**	**-151.5**	**0.0**	323.0	**15/15**	**-151**	**-151.5**	**0.0**	**288.3**	**15/15**
MDG-b_30	11296415	**-1650**	**-1649.6**	**0.0**	311.2	**15/15**	**-1650**	**-1649.6**	**0.0**	**274.9**	**15/15**
MDG-b_31	11288901	**0**	**-0.2**	**0.0**	313.9	**15/15**	**0**	**-0.2**	**0.0**	**308.0**	**15/15**
MDG-b_32	11279820	**-3719**	-3669.3	**25.0**	**177.5**	3/15	**-3719**	**-3694.3**	30.6	283.0	**9/15**
MDG-b_33	11296298	**-1740**	-1381.7	216.1	**112.2**	4/15	**-1740**	**-1675.0**	**166.1**	277.5	**13/15**
MDG-b_34	11281245	**-9238**	-8881.8	435.8	**325.7**	9/15	**-9238**	**-9237.6**	**0.0**	355.4	**15/15**
MDG-b_35	11307424	**0**	**-0.1**	**0.0**	343.6	**15/15**	**0**	**-0.1**	**0.0**	**331.0**	**15/15**
MDG-b_36	11289469	**-13423**	-13174.5	929.8	**251.7**	14/15	**-13423**	**-13423.0**	**0.0**	329.4	**15/15**
MDG-b_37	11290545	**-5229**	-5099.5	329.7	**217.5**	13/15	**-5229**	**-5228.8**	**0.0**	291.2	**15/15**
MDG-b_38	11288571	**-7965**	**-7964.5**	**0.0**	**242.5**	**15/15**	**-7965**	**-7964.5**	**0.0**	297.6	**15/15**
MDG-b_39	11295054	**0**	**-0.2**	**0.0**	374.5	**15/15**	**0**	**-0.2**	**0.0**	**289.7**	**15/15**
MDG-b_40	11307105	**-2058**	**-2057.6**	**0.0**	301.1	**15/15**	**-2058**	**-2057.6**	**0.0**	**266.5**	**15/15**
	wins	10	6.5	8	10	7	10	13.5	12	10	13

The best-known values f_{prev} were obtained by a scatter search algorithm (G_SS) [17] with a time limit of 2 hours, which are available at http://www.optsicom.es/mdp/.

obtains a better average objective value and higher success rate for 13.5 and 13 instances. It is worth noting that OBMA has a steady performance, and achieves these results with a 100% success rate on almost all instances except for MDG-b_24, MDG-b_32, and MDG-b_33. To summarize, OBMA performs better than $OBMA_0$ for this data set, but the differences are not very significant at a significance level of 0.05.

Table V presents the results of $OBMA_0$ and OBMA for the 20 instances of the MDG-c instances. All best-known results (f_{prev}) were achieved by ITS [38] or VNS [8] under a time limit of 2 h on an Intel Core 2 Quad CPU 8300 with 6 GB of RAM running Ubuntu 9.04 [35]. We observe that both OBMA and $OBMA_0$ obtain improved best-known solutions for eight instances and match the best-known solutions for four instances. In fact, OBMA improves all best-known solutions obtained by VNS, but it fails to attain eight best-known solutions found by ITS. Compared to $OBMA_0$, OBMA obtains two improved best solutions for MDG-c_17 and MDG-c_19.

TABLE V
COMPARISON OF THE RESULTS OBTAINED BY OBMA_0 AND OBMA ON THE DATA SET MDG-C

		OBMA_0					OBMA				
Instance	f_{prev}	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ
MDG-c_1	24924685⋆	**-1659**	-493.5	852.7	**165.5**	5/15	**-1659**	**-1262.4**	**749.0**	251.3	**11/15**
MDG-c_2	24909199⋆	**-3347**	140.3	2514.6	**94.6**	4/15	**-3347**	**-3346.3**	**2.5**	286.6	**14/15**
MDG-c_3	24900820*	**-4398**	-299.2	4040.7	**22.5**	7/15	**-4398**	**-2805.2**	**2717.9**	239.4	**11/15**
MDG-c_4	24904964⋆	**-4746**	-1890.5	1999.4	**106.5**	4/15	**-4746**	**-3917.2**	1657.6	276.2	**12/15**
MDG-c_5	24899703*	**3999**	4767.8	1025.0	**8.7**	9/15	**3999**	**4047.3**	**180.6**	212.5	**14/15**
MDG-c_6	43465087*	**20139**	22534.4	2512.3	**77.4**	**6/15**	**20139**	**21054.5**	**1190.4**	290.8	**6/15**
MDG-c_7	43477267⋆	**0**	277.5	1038.2	**6.1**	**14/15**	**0**	**126.9**	**314.7**	111.7	12/15
MDG-c_8	43458007⋆	**-7565**	-4644.3	1833.2	**58.9**	3/15	**-7565**	**-7546.7**	**68.6**	163.6	**14/15**
MDG-c_9	43448137⋆	**0**	142.2	116.1	82.1	6/15	**0**	**0.0**	**0.0**	**72.5**	**15/15**
MDG-c_10	43476251*	**10690**	**10690.0**	**0.0**	**27.1**	**15/15**	**10690**	**10690.0**	**0.0**	115.1	**15/15**
MDG-c_11	67009114⋆	**-12018**	-11345.3	2110.0	**90.8**	**13/15**	**-12018**	**-11776.3**	**522.3**	335.0	10/15
MDG-c_12	67021888*	**7718**	12209.1	5502.4	**9.3**	7/15	**7718**	**10179.7**	**3250.5**	302.8	**9/15**
MDG-c_13	67024373⋆	**0**	2082.0	2944.8	**106.4**	10/15	**0**	**839.4**	**2140.1**	380.1	**13/15**
MDG-c_14	67024804⋆	**-5386**	-4667.9	1830.9	**11.3**	13/15	**-5386**	**-5118.7**	**1000.3**	276.3	**14/15**
MDG-c_15	67056334⋆	**0**	1846.5	1353.5	**31.0**	**5/15**	**0**	**1021.2**	**1122.0**	269.0	**5/15**
MDG-c_16	95637733⋆	**-1196**	5861.5	8193.7	318.8	2/15	**-1196**	**-1116.3**	**298.3**	**270.8**	14/15
MDG-c_17	95645826*	75241	86848.9	8727.7	**291.8**	2/15	**74713**	**74981.7**	**373.3**	312.8	**8/15**
MDG-c_18	95629207*	**97066**	100609.9	3526.8	**90.5**	7/15	**97066**	**99767.0**	**2972.8**	292.1	**8/15**
MDG-c_19	95633549*	35131	39027.5	5420.3	**236.5**	**7/15**	**34385**	**35121.3**	**816.2**	343.7	4/15
MDG-c_20	95643586*	**59104**	**59133.2**	**109.3**	**111.0**	**14/15**	**59104**	**59133.2**	**109.3**	299.9	**14/15**
	wins	9	1	1	18	5	11	19	19	2	15

* Results are obtained by ITS with 2 hours CPU time [35].
⋆ Results are obtained by VNS with 2 hours CPU time [35].

TABLE VI
COMPARISON OF THE RESULTS OBTAINED BY OBMA_0 AND OBMA ON THE DATA SET B2500

		OBMA_0					OBMA				
Instance	f_{prev}	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ
b2500-1	1153068	**0**	193.1	429.2	154.1	23/30	**0**	**0.0**	**0.0**	**100.3**	**30/30**
b2500-2	1129310	**0**	106.0	163.1	149.2	21/30	**0**	**37.9**	**73.2**	**147.4**	**22/30**
b2500-3	1115538	**0**	303.2	347.2	105.5	17/30	**0**	**0.4**	**2.2**	**95.3**	**29/30**
b2500-4	1147840	**0**	549.7	461.9	191.2	8/30	**0**	**65.8**	**118.5**	**98.9**	**20/30**
b2500-5	1144756	**0**	50.9	129.3	117.6	24/30	**0**	**5.3**	**28.4**	**86.3**	**29/30**
b2500-6	1133572	**0**	88.9	210.9	89.4	22/30	**0**	**0.0**	**0.0**	**66.8**	**30/30**
b2500-7	1149064	**0**	106.2	111.9	**114.8**	13/30	**0**	**14.1**	**31.0**	128.3	**23/30**
b2500-8	1142762	**0**	113.7	349.0	**98.4**	22/30	**0**	**1.5**	**5.5**	105.4	**28/30**
b2500-9	1138866	**0**	**0.2**	**1.1**	**135.7**	**29/30**	**0**	1.3	2.9	139.8	25/30
b2500-10	1153936	**0**	**0.0**	**0.0**	**81.4**	**30/30**	**0**	**0.0**	**0.0**	107.5	**30/30**
	wins	5	1.5	1.5	4	1.5	5	8.5	8.5	6	8.5

The f_{prev} values were compiled from the results reported by ITS [38], LTS-EDA [51] and MAMDP [53].

Moreover, OBMA performs significantly better than OBMA_0 in terms of the average best solution ($19 > \text{CV}^{20}_{0.05} = 15$), success rate ($15 >= \text{CV}^{20}_{0.05} = 15$) and standard deviation ($19 > \text{CV}^{20}_{0.05} = 15$) at a significance level of 0.05.

Table VI reports the results of OBMA_0 and OBMA for the 10 instances of the b2500 data set. From this table, we observe that both algorithms reach the best-known values for all the instances. Meanwhile, the average value of best objective values of OBMA is better than that of OBMA_0, and the difference of this measure between these two algorithms is weakly significant ($8.5 < \text{CV}^{10}_{0.05} = 9$). Even though there is no significant difference on the success rate, OBMA obtains a higher success rate for 8.5 instances, while the reverse is true only for 1.5 instances. In addition, OBMA achieves these results more steadily than OBMA_0, wining 8.5 out of 10 instances in terms of the standard deviation.

Table VII displays the results of OBMA_0 and OBMA for the ten largest instances (p3000 and p5000 instances). For these very large instances, OBMA matches all the best-known objective values without exception while OBMA_0 fails to do so for four instances. In addition, OBMA performs significantly better than OBMA_0, and wins 10, 9.5 instances in terms of the average best objective value and success rate, respectively. The performance of OBMA is also more stable than OBMA_0 by wining 8 out of 10 instances in term of the standard deviation.

Finally, Table VIII provides a summary of the comparative results for the five data sets between OBMA (OBL enhanced MA) and OBMA_0 (MA without OBL). As we observe from the table, OBMA achieves a better performance than OBMA_0, i.e., achieving improved solutions for six instances and matching the best solutions on the remaining 75 instances.

TABLE VII
COMPARISON OF THE RESULTS OBTAINED BY $OBMA_0$ AND OBMA ON THE DATA SETS p3000 AND p5000

		$OBMA_0$					OBMA				
Instance	f_{prev}	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ
p3000_1	6502330	**0**	84.1	**28.0**	**172.6**	1/15	**0**	**24.4**	35.8	275.6	**9/15**
p3000_2	18272568	**0**	152.8	151.1	151.5	7/15	**0**	**0.0**	**0.0**	**89.3**	**15/15**
p3000_3	29867138	**0**	544.5	344.2	244.0	4/15	**0**	**0.0**	**0.0**	**26.2**	**15/15**
p3000_4	46915044	**0**	715.0	531.0	**250.1**	2/15	**0**	**1.2**	**19.3**	336.9	**14/15**
p3000_5	58095467	**0**	209.9	198.2	180.0	6/15	**0**	**0.0**	**0.0**	**65.4**	**15/15**
p5000_1	17509369	**0**	168.9	**176.5**	**518.7**	7/15	**0**	**128.2**	181.8	1053.9	**13/15**
p5000_2	50103092	70	819.1	494.3	**242.5**	**1/15**	**0**	**22.8**	**8.0**	370.8	**1/15**
p5000_3	82040316	176	3450.3	1671.3	333.1	1/15	**0**	**209.3**	**141.3**	**217.1**	**2/15**
p5000_4	129413710	598	1460.1	661.6	1019.1	1/15	**0**	**97.8**	**122.1**	**625.7**	**7/15**
p5000_5	160598156	344	669.6	323.6	1348.7	1/15	**0**	**102.9**	**52.3**	**843.2**	**5/15**
	wins	3	0	2	4	0.5	7	10	8	6	9.5

The best-known values f_{prev} were extracted from [53].

TABLE VIII
SUMMARY OF WIN STATISTICAL RESULTS ($OBMA_0$ | OBMA) ON ALL DATA SETS

Data set	Δf_{best}	Δf_{avg}	σ	t_{best}	#succ
MDG-a	10 | 10	5.5 | 14.5	5.5 | 14.5	1 | 19	6.5 | 13.5
MDG-b	10 | 10	6.5 | 13.5	8 | 12	10 | 10	7 | 13
MDG-c	9 | 11	1 | 19	1 | 19	18 | 2	5 | 15
b2500	5 | 5	1.5 | 8.5	1.5 | 8.5	4 | 6	1.5 | 8.5
p3000-5000	3 | 7	0 | 10	2 | 8	4 | 6	0.5 | 9.5

In addition, OBMA also achieves a better performance in terms of the average best value, the success rate and the standard deviation, wining $OBMA_0$ on most benchmark instances. Therefore, we conclude that OBL can beneficially enhance the popular memetic search framework to achieve an improved performance.

D. Comparison With State-of-the-Art Algorithms

We turn now our attention to a comparison of our OBMA algorithm with state-of-the-art algorithms, including iterated TS (ITS) [38], scatter search (G_SS) [17], VNS [8], TIG [30], TS with estimation of distribution algorithm (LTS-EDA) [51], and MA (MAMDP) [53]. We omit the TS/MA (TS/MA) [52] and the memetic self-adaptive evolution strategies (MSES) [11] since TS/MA performs quite similar to MAMDP of [53] while MSES does not report detailed results. Among these reference algorithm, only the program of the MA (MAMDP) [53] is available. For our comparative study, we report the results of the MAMDP algorithm by running its code on our platform with its default parameter values reported in [53]. For the other reference algorithms, we use their results presented in the corresponding references. The detailed comparative results in terms of Δf_{best} and Δf_{avg} are reported in Tables IX and X.

Table IX presents the comparative results on the 40 instances of the data sets MDG-a, b2500, p3000, and p5000 for which the detailed results of reference algorithms are available. At the last row of the table, we also indicate the number of wining instances relative to our OBMA algorithm both in terms of the best objective value and average objective value (recall that a tied result counts 0.5 for each algorithm). From this table, we observe that OBMA dominates all the reference algorithms. Importantly, OBMA is the only algorithm which obtains the best-known values and the largest average objective values for all 40 instances.

Table X displays the comparative results on the data sets MDG-b and MDG-c. The best-known objective values f_{prev} for the MDG-b instances are obtained by G_SS [17] while the f_{prev} values of the MDG-c instances are obtained by ITS and VNS [35], both with a time limit of 2 h. No result is available for the TIG and LTS-EDA algorithms for these data sets. The results of our OBMA algorithm (and MAMDP) are obtained with a time limit of 10 min. Table X indicates that both OBMA and MAMDP improve the best-known results for the majority of the 40 instances. Moreover, compared to MAMDP, our OBMA algorithm obtains an improved best objective value for one MDG-b instance and three MDG-c instances, while matching the best objective values for the remaining instances. Finally, OBMA dominates MAMDP in terms of the average objective value, wining 18 out of the 20 MDG-b instances and all 20 MDG-c instances.

To summarize, compared to the state-of-the-art results, our OBMA algorithm finds improved best-known solutions (new lower bounds) for 22 out of the 80 benchmark instances, matches the best-known solutions for 50 instances, but fails to attain the best-known results for eight instances. Such a performance indicates that the proposed algorithm competes favorably with state-of-the-art MDP algorithms and enriches the existing solution arsenal for solving MDP.

V. EXPERIMENTAL ANALYSIS

In this section, we perform additional experiments to gain some understanding of the proposed algorithm including the parametric constrained neighborhood, the RBQD pool management and the benefit of OBL for population diversity.

A. Study of the Parametric Constrained Neighborhood

Our TS procedure relies on the parametric constrained neighborhood whose size is controlled by the parameter ρ.

TABLE IX
COMPARISON OF OBMA WITH OTHER ALGORITHMS ON THE DATA SETS MDG-A, B2500, P3000, AND P5000

		ITS [38]		VNS [8]		TIG [30]		LTS-EDA [51]		MAMDP [53]		OBMA	
Instance	f_{prev}	Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}
MDG-a_21	114271	65	209.9	48	150.6	48	101.6	5	60.7	**0**	8.1	**0**	**5.2**
MDG-a_22	114327	29	262.3	**0**	168.9	**0**	69.9	**0**	89.9	**0**	8.8	**0**	**0.1**
MDG-a_23	114195	69	201.4	19	110.8	5	117.8	**0**	99.0	**0**	**15.2**	**0**	**15.2**
MDG-a_24	114093	22	200.5	70	188.1	58	141.9	**0**	79.9	**0**	15.7	**0**	**11.2**
MDG-a_25	114196	95	273.3	87	184.1	99	194.7	51	134.5	**0**	42.1	**0**	**41.5**
MDG-a_26	114265	41	168.2	30	99.3	9	96.2	**0**	40.2	**0**	10.8	**0**	**10.**2
MDG-a_27	114361	12	167.5	**0**	56.3	**0**	71.3	**0**	18.2	**0**	**0.0**	**0**	0.2
MDG-a_28	114327	25	256.4	**0**	163.3	**0**	193.6	**0**	159.1	**0**	20.9	**0**	**18.9**
MDG-a_29	114199	9	139.8	16	78.5	16	80.4	**0**	71.0	**0**	7.6	**0**	**4.4**
MDG-a_30	114229	24	204.9	7	139.3	35	121.4	**0**	56.2	**0**	9.3	**0**	**8.1**
MDG-a_31	114214	74	237.8	42	145.1	59	139.6	3	69.9	**0**	17.8	**0**	**16.7**
MDG-a_32	114214	55	249.5	95	143.3	88	156.0	15	84.9	**0**	26.8	**0**	**23.7**
MDG-a_33	114233	93	279.9	22	168.1	42	167.4	6	85.3	**0**	3.6	**0**	**2.0**
MDG-a_34	114216	92	248.5	117	194.3	64	202.8	**0**	81.0	**0**	3.4	**0**	**2.4**
MDG-a_35	114240	11	117.5	1	62.9	6	80.5	**0**	22.0	**0**	**1.2**	**0**	1.6
MDG-a_36	114335	11	225.4	42	215.4	35	167.9	**0**	36.5	**0**	8.6	**0**	**5.7**
MDG-a_37	114255	56	217.5	**0**	170.0	18	144.5	6	57.1	**0**	6.5	**0**	**5.2**
MDG-a_38	114408	46	170.0	**0**	57.1	2	117.4	2	22.8	**0**	0.7	**0**	**0.5**
MDG-a_39	114201	34	243.2	**0**	124.6	**0**	144.4	**0**	35.9	**0**	3.4	**0**	**2.0**
MDG-a_40	114349	151	270.7	65	159.4	45	187.2	**0**	95.4	**0**	24.1	**0**	**23.0**
b2500-1	1153068	624	3677.3	96	1911.9	42	1960.3	**0**	369.2	**0**	72.1	**0**	**0.0**
b2500-2	1129310	128	1855.3	88	1034.3	1096	1958.5	154	454.5	**0**	143.7	**0**	**37.9**
b2500-3	1115538	316	3281.9	332	1503.7	34	2647.9	**0**	290.4	**0**	184.5	**0**	**0.4**
b2500-4	1147840	870	2547.9	436	1521.1	910	1937.1	**0**	461.7	**0**	152.3	**0**	**65.8**
b2500-5	1144756	356	1800.3	**0**	749.4	674	1655.9	**0**	286.1	**0**	10.5	**0**	**5.3**
b2500-6	1133572	250	2173.5	**0**	1283.5	964	1807.6	80	218.0	**0**	80.5	**0**	**0.0**
b2500-7	1149064	306	1512.6	116	775.5	76	1338.7	44	264.6	**0**	45.0	**0**	**14.1**
b2500-8	1142762	**0**	2467.7	96	862.5	588	1421.5	22	146.5	**0**	1.7	**0**	**1.5**
b2500-9	1138866	642	2944.7	54	837.1	658	1020.6	6	206.3	**0**	3.7	**0**	**1.3**
b2500-10	1153936	598	2024.6	278	1069.4	448	1808.7	94	305.3	**0**	**0.0**	**0**	**0.0**
p3000-1	6502330	466	1487.5	273	909.8	136	714.7	96	294.1	**0**	76.7	**0**	**24.4**
p3000-2	18272568	**0**	1321.6	**0**	924.2	**0**	991.1	140	387.0	**0**	146.1	**0**	**0.0**
p3000-3	29867138	1442	2214.7	328	963.5	820	1166.1	**0**	304.3	**0**	527.9	**0**	**0.0**
p3000-4	46915044	1311	2243.9	254	1068.5	426	2482.2	130	317.1	**0**	399.5	**0**	**1.2**
p3000-5	58095467	423	1521.6	**0**	663.0	278	1353.3	**0**	370.4	**0**	210.7	**0**	**0.0**
p5000-1	17509369	2200	3564.9	1002	1971.3	1154	2545.8	191	571.0	**0**	165.1	**0**	**128.2**
p5000-2	50103092	2931	4807.8	1499	2640.0	549	2532.7	547	913.8	21	475.5	**0**	**22.8**
p5000-3	82040316	5452	8242.3	1914	3694.4	2156	6007.1	704	1458.5	176	1419.0	**0**	**209.3**
p5000-4	129413710	1630	5076.9	1513	2965.9	1696	3874.8	858	1275.2	279	800.9	**0**	**97.8**
p5000-5	160598156	2057	4433.9	1191	2278.3	1289	2128.9	579	1017.9	136	411.9	**0**	**102.9**
wins		1	0	5	0	2.5	0	9.5	0	18	2		

The f_{prev} values were compiled from the results reported by the reference methods [38], [8], [30], [51], [53]. The results of MAMDP are those we obtained by running its program on our computer, which are slightly different from the results reported in [53] due to the stochastic nature of the algorithm.

To highlight the effect of this parameter and determine a proper value, we ran the TS procedure to solve the first ten instances of MDG-a (i.e., MDG-a_21–MDG-a_30) with $\rho \in [1, 10]$. Each instance was independently solved until the number of iterations reached MaxIter. Fig. 1 shows the average objective values achieved (left) and the average CPU times consumed (right) by TS on these ten instances.

As we see from Fig. 1(left), the average objective value has a drastic rise when we increase ρ from 1 to 3. Then, it slowly increases if we continue to increase ρ to 10. On Fig. 1(right), the average CPU time of TS needed to finish MaxIter iterations continuously increases when ρ increases from 1 to 10. As ρ increases, the size of the constrained neighborhood also increases, thus the algorithm needs more time to examine the candidate solutions. To make a compromise between neighborhood size and solution quality, we set the scale coefficient ρ to 4 in our experiments.

B. Effectiveness of the Pool Updating Strategy

To validate the effectiveness of the RBQD pool updating strategy, we compare it with the general quality-and-distance (GQD) pool updating strategy used in [53]. GQD evaluates each individual by a weighted sum of the quality and the distance to the population. In this experiment, we compared the performance of the OBMA algorithm under these two pool updating strategies (the two OBMA variants are called $\text{OBMA}_{\text{RBQD}}$ and OBMA_{GQD}). The experiment was performed on the largest data set, i.e., p3000 and p5000. We performed 20 runs of each algorithm to solve each instance, and recorded the best objective value (f_{best}), the difference between the average objective value and the best objective value (Δf_{avg}), the standard deviation of objective value over each run (σ), the average time of one run (t_{avg}), the average time over the runs which attained f_{best} (t_{best}), and the success rate (#succ).

Table XI shows the comparison of the results obtained by OBMA under the RBQD strategy ($\text{OBMA}_{\text{RBQD}}$) and the

TABLE X
COMPARISON OF OBMA WITH MAMDP [53] ON THE DATA SETS MDG-B AND MDG-C, THE BEST-KNOWN RESULTS ARE OBTAINED BY G_SS [17], ITS, AND VNS [35]

		MAMDP [53]		OBMA				MAMDP [53]		OBMA	
Instance	f_{prev}	Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}	Instance	f_{prev}	Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}
MDG-b_21	11299895	**0**	225.8	**0**	**0.2**	MDG-c_1	24924685	**-1659**	3481.7	**-1659**	**-1262.4**
MDG-b_22	11286776	**-5622**	-3472.1	**-5622**	**-5622.2**	MDG-c_2	24909199	0	4938.7	**-3347**	**-3346.3**
MDG-b_23	11299941	**0**	**0.5**	**0**	**0.5**	MDG-c_3	24900820	**-4398**	5206.1	**-4398**	**-2805.2**
MDG-b_24	11290874	**-245**	226.0	**-245**	**-220.5**	MDG-c_4	24904964	**-4746**	-411.2	**-4746**	**-3917.2**
MDG-b_25	11296067	**-1960**	-1888.9	**-1960**	**-1959.9**	MDG-c_5	24899703	**3999**	7500.3	**3999**	**4047.3**
MDG-b_26	11292296	**-6134**	-2530.6	**-6134**	**-6134.4**	MDG-c_6	43465087	**20139**	25023.7	**20139**	**21054.5**
MDG-b_27	11305677	**0**	**0.2**	**0**	**0.2**	MDG-c_7	43477267	**0**	1020.8	**0**	**126.9**
MDG-b_28	11279916	-2994	-2634.6	**-2995**	**-2994.7**	MDG-c_8	43458007	-4568	-1329.9	**-7565**	**-7546.7**
MDG-b_29	11297188	**-151**	451.8	**-151**	**-151.5**	MDG-c_9	43448137	237	1207.3	**0**	**0.0**
MDG-b_30	11296415	**-1650**	**-1649.6**	**-1650**	**-1649.6**	MDG-c_10	43476251	**10690**	11060.9	**10690**	**10690.0**
MDG-b_31	11288901	**0**	375.7	**0**	**-0.2**	MDG-c_11	67009114	**-12018**	-6942.7	**-12018**	**-11776.3**
MDG-b_32	11279820	**-3719**	-3632.3	**-3719**	**-3694.3**	MDG-c_12	67021888	**7718**	17470.0	**7718**	**10179.7**
MDG-b_33	11296298	**-1740**	-878.7	**-1740**	**-1675.0**	MDG-c_13	67024373	**0**	6673.1	**0**	**839.4**
MDG-b_34	11281245	**-9238**	-8191.3	**-9238**	**-9237.6**	MDG-c_14	67024804	**-5386**	-1050.9	**-5386**	**-5118.7**
MDG-b_35	11307424	**0**	**-0.1**	**0**	**-0.1**	MDG-c_15	67056334	**0**	3716.2	**0**	**1021.2**
MDG-b_36	11289469	**-13423**	-10792.5	**-13423**	**-13423.0**	MDG-c_16	95637733	**-1196**	1495.2	**-1196**	**-1116.3**
MDG-b_37	11290545	**-5229**	-4372.1	**-5229**	**-5228.8**	MDG-c_17	95645826	**74713**	79061.1	**74713**	**74981.7**
MDG-b_38	11288571	**-7965**	-5896.0	**-7965**	**-7964.5**	MDG-c_18	95629207	**97066**	106806.6	**97066**	**99767.0**
MDG-b_39	11295054	**0**	472.4	**0**	**-0.2**	MDG-c_19	95633549	**34385**	36189.1	**34385**	**35121.3**
MDG-b_40	11307105	**-2058**	-517.5	**-2058**	**-2057.6**	MDG-c_20	95643586	**59104**	61961.2	**59104**	**59133.2**
wins		9.5	2	10.5	18	wins		8.5	0	11.5	20

The f_{prev} values for the MDG-b instances are reported by G_SS [17], while the f_{prev} values for the MDG-c instances are from [35] with a time limit of 2 hours, all available at http://www.optsicom.es/mdp/. The results of MAMDP were obtained by running the program on our computer (results of MAMDP for these instances are not reported in [53]).

TABLE XI
COMPARISON OF THE RESULTS OBTAINED BY OBMA UNDER THE RBQD POOL UPDATING STRATEGY ($OBMA_{RBQD}$) AND THE GQD POOL UPDATING STRATEGY ($OBMA_{GQD}$)

	$OBMA_{RBQD}$						$OBMA_{GQD}$					
Instance	f_{best}	Δf_{avg}	σ	t_{avg}	t_{best}	#succ	f_{best}	Δf_{avg}	σ	t_{avg}	t_{best}	#succ
p3000_1	6502330	-23.0	35.3	176.9	118.8	14/20	6502330	-27.3	37.4	158.3	81.4	13/20
p3000_2	18272568	0.0	0.0	75.8	75.8	20/20	18272568	-10.5	45.8	54.7	57.1	19/20
p3000_3	29867138	0.0	0.0	37.7	37.7	20/20	29867138	0.0	0.0	55.4	55.4	20/20
p3000_4	46915044	0.0	0.0	113.7	113.7	20/20	46915044	-0.9	3.9	147.5	146.4	19/20
p3000_5	58095467	0.0	0.0	22.9	22.9	20/20	58095467	0.0	0.0	92.8	92.8	20/20
p5000_1	17509369	-13.8	60.4	621.9	624.3	19/20	17509369	-27.8	83.1	674.3	646.1	17/20
p5000_2	50103071	-23.4	4.8	561.1	594.3	16/20	50103071	-26.4	6.0	584.3	464.0	11/20
p5000_3	82040316	-305.8	304.2	791.1	527.5	01/20	82040316	-241.0	176.8	642.2	718.4	01/20
p5000_4	129413710	-116.4	143.9	756.5	802.8	12/20	129413710	-174.3	176.2	662.5	705.2	09/20
p5000_5	160598156	-161.8	99.4	511.7	471.6	02/20	160598156	-182.6	112.7	745.5	1081.3	02/20
wins	5	8	8	6	6	8	5	2	2	4	4	2

GQD strategy ($OBMA_{GQD}$). From the table, we observe that $OBMA_{RBQD}$ achieves the same best objective values for all tested instances compared with $OBMA_{GQD}$. However, for the five metrics, $OBMA_{RBQD}$ performs better than $OBMA_{GQD}$ for much more instances, and, respectively, winning 8, 8, 6, 6, and 8 out of 10 tested instances. These results confirm the effectiveness of our proposed RBQD pool updating strategy.

C. Opposition-Based Learning Over Population Diversity

In this section, we further verify the benefit brought by OBL in maintaining the population diversity of the OBMA algorithm. To assess the diversity of a population, a suitable metric is necessary. In this experiment, we resort to *minimum distance* and *average distance* of individuals in the population to measure the population diversity. The minimum distance is defined as the minimum distance between any two individuals in the population, i.e., MD $= \min_{i \neq j \in \{1,2,\ldots,p\}} D(S^i, S^j)$. Correspondingly, the AD is the average distance between all individuals in the population, as defined by (10).

Using the data sets MDG-a and b2500, we compared the diversity of the population with or without OBL. The population initialization (PI_0) procedure without OBL first generates two random solutions, which are then, respectively, improved by the TS procedure. The best of two improved solutions is inserted into the population if it does not duplicate any existing individual in the population. We repeat this process until p different solutions are generated. In contrast, the population initialization with OBL (PI_{OBL}) is the procedure described in Section III-C, which considers both a random solution and its corresponding opposite solution. We solved each instance 20 times and recorded the minimum distance and average distance

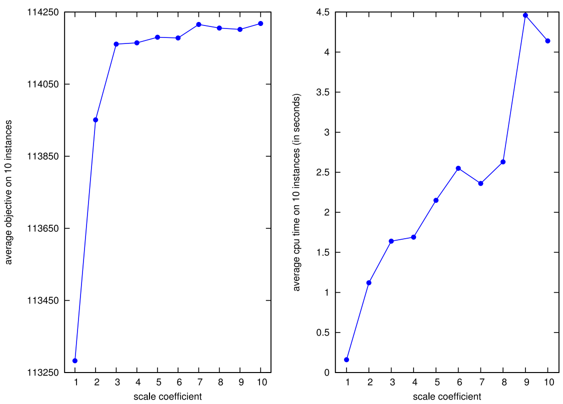

Fig. 1. Average objective values and average CPU times spent on ten MDG-a instances obtained by executing TS with different values of the scale coefficient ρ.

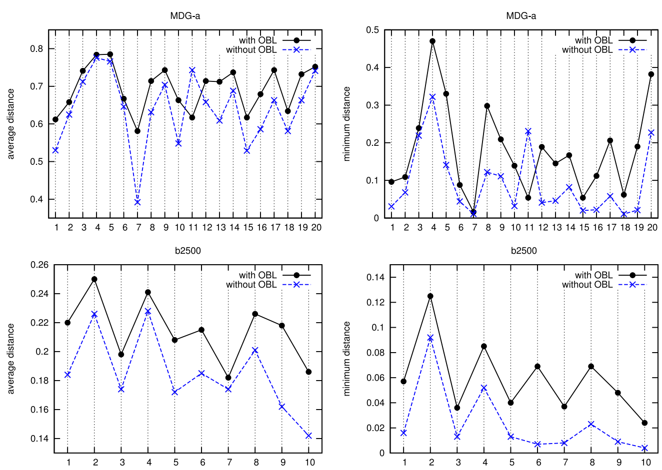

Fig. 2. Comparative results of the populations built by population initialization with OBL (PI_{OBL}) or without OBL (PI_0).

of each population initialization procedure on each instance. The comparative results of the population constructed with or without OBL are shown in Fig. 2, where the *x*-axis shows the instances in each benchmark and *y*-axis indicates the average distance and minimum distance.

From Fig. 2, we observe that the population built by PI_{OBL} has a relatively larger average distance and minimum distance. This is particularly true for all instances of the MDG-a data set except for MDG-a_31. Also, the population produced by PI_{OBL} has a larger minimum distance than that of PI_0 for 18 out of 20 instances of the MDG-a data set. Equal or better results are found for the b2500 data set, since the population generated by PI_{OBL} dominates the population produced by PI_0 in terms of the average and minimum distances. This experiment shows that OBL helps the OBMA algorithm to start its search with a population of high diversity, which is maintained by the RBQD strategy during the search.

VI. Conclusion

We have proposed an OBMA which uses OBL to improve an MA for solving MDP. The OBMA algorithm employs OBL to reinforce population diversity and improve evolutionary search. OBMA distinguishes itself from existing MAs by three aspects.

1) A double trajectory search procedure which simultaneously both a candidate solution and a corresponding opposite solution.
2) A parametric constrained neighborhood for effective local optimization.
3) An RBQD pool updating strategy.

Extensive comparative experiments on 80 large benchmark instances (with 2000–5000 items) from the literature have demonstrated the competitiveness of the OBMA algorithm. OBMA matches the best-known results for most of instances and in particular finds improved best results (new lower bounds) for 22 instances which are useful for the assessment of other MDP algorithms. Our experimental analysis has also confirmed that integrating OBL into the memetic search framework does improve the search efficiency of the classical memetic search. It would be interesting to study the behavior of the OBMA algorithm when it is applied to

As future work, several potential research lines can be followed. First, to further improve OBMA, it is worth studying alternative strategies for tuning tabu tenure, generating initial solutions, and managing population diversity. Second, it would be interesting to study the behavior of the OBMA algorithm on much larger instances (e.g., with tens of thousands items) and investigate whether techniques developed for large scale continuous optimization [37], [43] could be helpful in this setting. Third, OBL being a general technique, it is worth studying its usefulness within other heuristic algorithms. Finally, it would be interesting to investigate the opposition-based optimization approach for solving additional combinatorial problems including those with other diversity and dispersion criteria.

Acknowledgment

The authors would like to thank the anonymous referees for their insightful comments and suggestions which helped us to significantly improve this paper. They would also like to thank Dr. Q. Wu for kindly sharing the source code of the MAMDP algorithm described in [53] and Y. Zhou from the China Scholarship Council (2014–2018).

References

[1] F. S. Al-Qunaieer, H. R. Tizhoosh, and S. Rahnamayan, “Opposition based computing—A survey,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Barcelona, Spain, 2010, pp. 1–7.

[2] R. Aringhieri, R. Cordone, and Y. Melzani, “Tabu search versus GRASP for the maximum diversity problem,” *Quart. J. Oper. Res.*, vol. 6, no. 1, pp. 45–60, 2008.

[3] R. Aringhieri, M. Bruglieri, and R. Cordone, “Optimal results and tight bounds for the maximum diversity problem,” *Found. Comput. Decis. Sci.*, vol. 34, no. 2, pp. 73–85, 2009.

[4] R. Aringhieri and R. Cordone, “Comparing local search metaheuristics for the maximum diversity problem,” *J. Oper. Res. Soc.*, vol. 62, no. 2, pp. 266–280, 2011.

[5] M. Aziz and M.-H. Tayarani-N., “Opposition-based magnetic optimization algorithm with parameter adaptation strategy,” *Swarm Evol. Comput.*, vol. 26, pp. 97–119, Feb. 2016.

[6] R. Bellio, L. Di Gaspero, and A. Schaerf, “Design and statistical analysis of a hybrid local search algorithm for course timetabling,” *J. Sched.*, vol. 15, no. 1, pp. 49–61, 2012.

[7] U. Benlic and J.-K. Hao, “A multilevel memetic approach for improving graph k-partitions,” *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 624–642, Oct. 2011.

[8] J. Brimberg, N. Mladenović, D. Urošević, and E. Ngai, "Variable neighborhood search for the heaviest k-subgraph," *Comput. Oper. Res.*, vol. 36, no. 11, pp. 2885–2891, 2009.
[9] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, Oct. 2011.
[10] Y. Chen and J.-K. Hao, "Memetic search for the generalized quadratic multiple knapsack problem," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 908–923, Dec. 2016.
[11] A. R. R. de Freitas, F. G. Guimarães, R. C. P. Silva, and M. J. F. Souza, "Memetic self-adaptive evolution strategies applied to the maximum diversity problem," *Optim. Lett.*, vol. 8, no. 2, pp. 705–714, 2014.
[12] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[13] A. Duarte and R. Martí, "Tabu search and GRASP for the maximum diversity problem," *Eur. J. Oper. Res.*, vol. 178, no. 1, pp. 71–84, 2007.
[14] M. Ergezer and D. Simon, "Oppositional biogeography-based optimization for combinatorial problems," in *Proc. Congr. Evol. Comput. (CEC)*, New Orleans, LA, USA, 2011, pp. 1496–1503.
[15] U. Feige, D. Peleg, and G. Kortsarz, "The dense *k*-subgraph problem," *Algorithmica*, vol. 29, no. 3, pp. 410–421, 2001.
[16] P. Galinier, Z. Boujbel, and M. C. Fernandes, "An efficient memetic algorithm for the graph partitioning problem," *Ann. Oper. Res.*, vol. 191, no. 1, pp. 1–22, 2011.
[17] M. Gallego, A. Duarte, M. Laguna, and R. Martí, "Hybrid heuristics for the maximum diversity problem," *Comput. Optim. Appl.*, vol. 44, no. 3, pp. 411–426, 2009.
[18] J. B. Ghosh, "Computational aspects of the maximum diversity problem," *Oper. Res. Lett.*, vol. 19, no. 4, pp. 175–181, 1996.
[19] F. Glover, E. Taillard, and D. de Werra, "A user's guide to tabu search," *Ann. Oper. Res.*, vol. 41, nos. 1–4, pp. 3–28, 1993.
[20] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer, 1997.
[21] F. Glover, "A template for scatter search and path relinking," in *Artificial Evolution* (LNCS 1363), J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, Eds. Heidelberg, Germany: Springer, 1997, pp. 1–51.
[22] F. Glover, C.-C. Kuo, and K. S. Dhir, "Heuristic algorithms for the maximum diversity problem," *J. Inf. Optim. Sci.*, vol. 19, no. 1, pp. 109–132, 1998.
[23] D. Gusfield, "Partition-distance: A problem and class of perfect graphs arising in clustering," *Inf. Process. Lett.*, vol. 82, no. 3, pp. 159–164, 2002.
[24] J.-K. Hao, "Memetic algorithms in discrete optimization," in *Handbook of Memetic Algorithms* (Studies in Computational Intelligence), vol. 379, F. Neri, C. Cotta, and P. Moscato, Eds. Heidelberg, Germany: Springer, 2012, ch. 6, pp. 73–94.
[25] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 61–70, 2011.
[26] L. Kotthoff, I. P. Gent, and I. Miguel, "An evaluation of machine learning in algorithm selection for search problems," *AI Commun.*, vol. 25, no. 3, pp. 257–270, 2012.
[27] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy, and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474–488, Oct. 2005.
[28] C.-C. Kuo, F. Glover, and K. S. Dhir, "Analyzing and modeling the maximum diversity problem by zero-one programming," *Decis. Sci.*, vol. 24, no. 6, pp. 1171–1185, 1993.
[29] K. Leyton-Brown, H. H. Hoos, F. Hutter, and L. Xu, "Understanding the empirical hardness of NP-complete problems," *Commun. ACM*, vol. 57, no. 5, pp. 98–107, 2014.
[30] M. Lozano, D. Molina, and C. García-Martínez, "Iterated greedy for the maximum diversity problem," *Eur. J. Oper. Res.*, vol. 214, no. 1, pp. 31–38, 2011.
[31] Z. Lü and J.-K. Hao, "A memetic algorithm for graph coloring," *Eur. J. Oper. Res.*, vol. 203, no. 1, pp. 241–250, 2010.
[32] E. M. Macambira and C. C. De Souza, "The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations," *Eur. J. Oper. Res.*, vol. 123, no. 2, pp. 346–371, 2000.
[33] E. M. Macambira, "An application of tabu search heuristic for the maximum edge-weighted subgraph problem," *Ann. Oper. Res.*, vol. 117, nos. 1–4, pp. 175–190, 2002.
[34] R. Martí, M. Gallego, and A. Duarte, "A branch and bound algorithm for the maximum diversity problem," *Eur. J. Oper. Res.*, vol. 200, no. 1, pp. 36–44, 2010.
[35] R. Martí, M. Gallego, A. Duarte, and E. G. Pardo, "Heuristics and metaheuristics for the maximum diversity problem," *J. Heuristics*, vol. 19, no. 4, pp. 591–615, 2013.
[36] P. Moscato, "Memetic algorithms: A short introduction," in *New Ideas in Optimization*. London, U.K.: McGraw-Hill, 1999, pp. 219–234.
[37] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.
[38] G. Palubeckis, "Iterated tabu search for the maximum diversity problem," *Appl. Math. Comput.*, vol. 189, no. 1, pp. 371–383, 2007.
[39] O. A. Prokopyev, N. Kong, and D. L. Martinez-Torres, "The equitable dispersion problem," *Eur. J. Oper. Res.*, vol. 197, no. 1, pp. 59–67, 2009.
[40] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.
[41] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition versus randomness in soft computing techniques," *Appl. Soft Comput.*, vol. 8, no. 2, pp. 906–918, 2008.
[42] S. Rahnamayan, G. G. Wang, and M. Ventresca, "An intuitive distance-based explanation of opposition-based sampling," *Appl. Soft Comput.*, vol. 12, no. 9, pp. 2828–2839, 2012.
[43] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.
[44] B. Rangaswamy, A. S. Jain, and F. Glover, "Tabu search candidate list strategies in scheduling," in *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search, Interfaces in Computer Science and Operations Research*, D. L. Woodruff, Ed. New York, NY, USA: Springer, 1998, pp. 215–233.
[45] M. Raschip, C. Croitoru, and K. Stoffel, "Using association rules to guide evolutionary search in solving constraint satisfaction," in *Proc. Congr. Evol. Comput. (CEC)*, Sendai, Japan, 2015, pp. 744–750.
[46] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi, "Heuristic and special case algorithms for dispersion problems," *Oper. Res.*, vol. 42, no. 2, pp. 299–310, 1994.
[47] L. F. Santos, M. H. Ribeiro, A. Plastino, and S. L. Martins, "A hybrid GRASP with data mining for the maximum diversity problem," in *Hybrid Metaheuristics*. Heidelberg, Germany: Springer, 2005, pp. 116–127.
[48] K. Sörensen and M. Sevaux, "MA|PM: Memetic algorithms with population management," *Comput. Oper. Res.*, vol. 33, no. 5, pp. 1214–1225, 2006.
[49] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Model. Control Autom. Int. Conf. Intell. Agents Web Technol. Internet Commer. (CIMCA/IAWTIC)*, Vienna, Austria, 2005, pp. 695–701.
[50] M. Ventresca and H. R. Tizhoosh, "A diversity maintaining population-based incremental learning algorithm," *Inf. Sci.*, vol. 178, no. 21, pp. 4038–4056, 2008.
[51] J. Wang, Y. Zhou, Y. Cai, and J. Yin, "Learnable tabu search guided by estimation of distribution for maximum diversity problems," *Soft Comput.*, vol. 16, no. 4, pp. 711–728, 2012.
[52] Y. Wang, J.-K. Hao, F. Glover, and Z. Lü, "A tabu search based memetic algorithm for the maximum diversity problem," *Eng. Appl. Artif. Intell.*, vol. 27, pp. 103–114, Jan. 2014.
[53] Q. Wu and J.-K. Hao, "A hybrid metaheuristic method for the maximum diversity problem," *Eur. J. Oper. Res.*, vol. 231, no. 2, pp. 452–464, 2013.
[54] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, "A review of opposition-based learning from 2005 to 2012," *Eng. Appl. Artif. Intell.*, vol. 29, pp. 1–12, Mar. 2014.
[55] Y. Zhou, J.-K. Hao, and B. Duval, "Reinforcement learning based local search for grouping problems: A case study on graph coloring," *Expert Syst. Appl.*, vol. 64, pp. 412–422, Dec. 2016.

Yangming Zhou received the B.Sc. degree from Donghua University, Shanghai, China, in 2011, and the M.Sc. degree from Zhejiang University, Hangzhou, China, in 2014. He is currently pursuing the Ph.D. degree with the LERIA Laboratory, University of Angers, Angers, France.

His current research interests include evolutionary computation, memetic algorithms, metaheuristics, and machine learning for solving combinatorial optimization problems.

Jin-Kao Hao received the B.Sc. degree from the School of Computer Science, National University of Defense Technology, Changsha, China, in 1982, the M.Sc. degree from the National Institute of Applied Sciences, Lyon, France, in 1987, the Ph.D. degree from the University of Franche-Comté, Besançon, France, in 1991, and the Professorship Diploma HDR (Habilitation à Diriger des Recherches) degree from the University of Science and Technology of Montpellier, Montpellier, France, in 1998.

He is a Distinguished Professor (Professeur des Universités de classe exceptionnelle) of Computer Science with the University of Angers, Angers, France. He has authored or co-authored over 230 peer-reviewed publications and co-edited nine books in Springer's LNCS series. His current research interests include design of effective algorithms and intelligent computational methods for solving large-scale combinatorial search problems, bioinformatics, data science, telecommunication networks, and transportation.

Dr. Hao is a Senior Fellow of the Institut Universitaire de France. He is on the editorial board of seven international journals. He has served as an Invited Member of over 200 Program Committees of International Conferences.

Béatrice Duval received the master's degree from the Ecole Normale Supérieur de Fontenay-aux-Roses, Montrouge, France, in 1984, and the Ph.D. degree in machine learning from the University of Paris-Sud, Orsay, France, in 1991.

She is currently an Associate Professor of Computer Science with the University of Angers, Angers, France. Her current research interests include data mining, machine learning, applications in bioinformatics with hybrid methods combining machine learning techniques, and metaheuristics.