



# **A study of two evolutionary/tabu search approaches for the generalized max-mean dispersion problem**

Xiangjing Lai, Jin-Kao Hao, Fred Glover

## **► To cite this version:**

Xiangjing Lai, Jin-Kao Hao, Fred Glover. A study of two evolutionary/tabu search approaches for the generalized max-mean dispersion problem. Expert Systems with Applications, 2020, 139, pp.112856. <10.1016/j.eswa.2019.112856>. <hal-02309954>

**HAL Id: hal-02309954**

**<https://hal.science/hal-02309954v1>**

Submitted on 20 Jul 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY-NC 4.0 - Attribution - Non-commercial use - International License

# A study of two evolutionary/tabu search approaches for the generalized max-mean dispersion problem

Xiangjing Lai<sup>a</sup>, Jin-Kao Hao<sup>b,c,\*</sup>, Fred Glover<sup>d</sup>

<sup>a</sup>*Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

<sup>b</sup>*LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers, Cedex 01, France*

<sup>c</sup>*Institut Universitaire de France, 1 Rue Descartes, 75231 Paris, France*

<sup>d</sup>*College of Engineering & Applied Science, University of Colorado, Boulder, Colorado 80309 USA*

*Submission Oct 2018, first revision June 2019, second revision July 2019*

---

## Abstract

Evolutionary computing is a general and powerful framework for solving difficult optimization problems, including those arising in expert and intelligent systems. In this work, we investigate for the first time two hybrid evolutionary algorithms incorporating tabu search for solving the generalized max-mean dispersion problem (GMaxMeanDP) which has a variety of practical applications such as web page ranking, community mining, and trust networks. The proposed algorithms integrate innovative search strategies that help the search to explore the search space effectively. We report extensive computational results of the proposed algorithms on six types of 160 benchmark instances, demonstrating their effectiveness and usefulness. In addition to the GMaxMeanDP, the proposed algorithms can help to better solve other problems that can be formulated as the GMaxMeanDP.

**Keywords:** Combinatorial optimization; Dispersion problems; Tabu search; Evolutionary search; Heuristics.

---

## 1. Introduction

Many decision-making problems including those arising in expert and intelligent systems require finding a best subset of elements in a way that the selected objects optimize a dispersion or diversity criterion. Formally, given a set  $V = \{1, 2, \dots, n\}$  of  $n$  elements and the distances  $d_{ij}$  ( $i < j$ ) between elements, a dispersion or diversity problem involves selecting a subset  $M$  of  $V$  such that an objective function defined over the distances between the elements in  $M$  is optimized. According to whether a cardinality constraint is imposed

---

\*Corresponding author.

Email addresses: [laixiangjing@gmail.com](mailto:laixiangjing@gmail.com) (Xiangjing Lai),  
[jin-kao.hao@univ-angers.fr](mailto:jin-kao.hao@univ-angers.fr) (Jin-Kao Hao), [glover@opttek.com](mailto:glover@opttek.com) (Fred Glover)

on the subset  $M$ , the dispersion problems can be divided into two categories. The first category where the cardinality of  $M$  is fixed to a given positive number  $m$  includes the maximum diversity problem (Aringhieri et al., 2011; Glover et al., 1998; Palubeckis, 2007; Saboonchi et al., 2014; Wu and Hao, 2013), the max-min diversity problem (Della Croce et al., 2009; Porumbel et al., 2011; Resende et al., 2010), the minimum differential dispersion problem (Lai et al., 2019; Mladenović et al., 2016; Wang et al., 2017; Zhou and Hao, 2017), and the maximum min-sum dispersion problem (Amirgaliyeva et al., 2017; Aringhieri et al., 2015; Lai et al., 2018; Prokopyev et al., 2009). The second category where the cardinality of  $M$  is not fixed includes the Max-Mean dispersion problem (MaxMeanDP) (Brimberg et al., 2017; Della Croce et al., 2016; Lai and Hao, 2016; Martí and Sandoya, 2013) and the generalized Max-Mean dispersion problem (GMaxMeanDP) (Prokopyev et al., 2009).

This work addresses the GMaxMeanDP that is one of four dispersion problems introduced in (Prokopyev et al., 2009) and can be described by means of a weighted graph. Given a weighted complete graph  $G = (V, E, D, W)$ , where  $V$  is the set of  $n$  vertices,  $E$  is the set of  $\frac{n \times (n-1)}{2}$  edges,  $D$  represents the set of positive, negative or zero edge weights  $d_{ij}$  ( $i \neq j$ ), and  $W$  represents the set of positive vertex weights  $w_i$  ( $i = 1, 2, \dots, n$ ), the GMaxMeanDP is to select a subset  $M$  from  $V$  such that the weighted mean dispersion of the (complete) subgraph induced by  $M$  is maximized. In related literature, the vertices are also called the elements and the edge weights between vertices are called the distances between the elements.

Formally, the GMaxMeanDP can be formulated as an unconstrained fractional 0–1 combinatorial optimization problem with binary variables  $x_i$  that equal 1 if the element  $i$  is selected, and 0 otherwise (Prokopyev et al., 2009).

$$\text{Maximize } f(s) = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} x_i x_j}{\sum_{i=1}^n w_i x_i} \quad (1)$$

$$\text{Subject to } \sum_{i=1}^n x_i \geq 2 \quad (2)$$

$$x_i \in \{0, 1\}, i = 1, 2, \dots, n \quad (3)$$

The Max-Mean dispersion problem that has recently received substantial attention in the literature (Brimberg et al., 2017; Carrasco et al., 2015; Della Croce et al., 2016; Lai and Hao, 2016; Martí and Sandoya, 2013) is a special case of the GMaxMeanDP with  $w_i = 1$  for  $\forall i \in \{1, 2, \dots, n\}$ . As a result, any algorithm for the GMaxMeanDP can be directly applied to the Max-Mean dispersion problem, while the reverse is not true.

In addition to its theoretical significance as an NP-hard problem (Prokopyev et al., 2009), the GMaxMeanDP has a variety of potential applications, such as web page ranking (Kerchove and Dooren, 2008), [community mining in a signed social network](#) (Yang et al., 2007), and trust networks (Carrasco et al., 2015), among others. [For example, the community mining problem in a signed and weighted social network can be addressed by solving a series of](#)

GMaxMeanDP problems with smaller and smaller sizes (Yang et al., 2007). Given a signed social network  $G = (V, E, D, W)$ , where  $D$  represents the set of positive or negative edge weights  $d_{ij}$  ( $i \neq j$ ), and a positive (or negative)  $d_{ij}$  means that there exists an attractive (or repulsive) relationship between the vertices  $i$  and  $j$ , and  $W$  represents the set of vertex weights  $w_i$  ( $1 \leq i \leq n$ ), then a community corresponds to a high-quality solution of the corresponding GMaxMeanDP (i.e., a subset of  $V$ ) in  $G$ .

In spite of its importance and close relationship to other dispersion problems, the GMaxMeanDP has surprisingly received little attention in the literature. To the best of our knowledge, no heuristic or exact algorithm has ever been proposed for solving the GMaxMeanDP, even though existing heuristic or exact algorithms for the MaxMeanDP like those in (Brimberg et al., 2017; Della Croce et al., 2016; Garraffa et al., 2017) could be adapted to the GMaxMeanDP. On the other hand, previous studies (Benlic and Hao, 2015; Ghosh et al., 2019; Morra et al., 2018; Ismkhan, 2017; Silva et al., 2017; Zhao et al., 2015) showed that evolutionary computing is a particularly relevant approach for solving a number of difficult combinatorial optimization problems. Given the NP-hard nature of the GMaxMeanDP, evolutionary computing can be considered as a natural approach to be investigated for solving the GMaxMeanDP. We enhance this approach by forming two hybrid algorithms with tabu search, drawing on the adaptive memory features of the latter to uncover superior solutions. Our work is thus motivated by these observations with the purpose of proposing effective solution methods for the considered problem. We summarize the contributions of this work as follows.

- First, in terms of solution methods, we investigate the first perturbation-based evolutionary algorithm dedicated to the GMaxMeanDP, which integrates a multi-neighborhood tabu search procedure and a perturbation operator into the population-based framework. Additionally, we adapt the state-of-the-art MaxMeanDP algorithm introduced in (Lai and Hao, 2016) to the GMaxMeanDP, where a crossover operator is used to generate offspring solutions and a tabu search procedure is employed for local optimization. Given that solution method for solving the GMaxMeanDP does not currently exist, this work fills an important gap in the literature.
- Second, we assess the computational performance of the proposed algorithms on a set of 80 MaxMeanDP benchmark instances as well as on a set of additional 80 GMaxMeanDP instances that we introduce in this work and make publicly available. Our results provide a reference for performance assessment of other solution methods for the GMaxMeanDP in the future.
- Third, we analyze the effectiveness and time complexity of several key components such as the neighborhood structures used by the tabu search procedure and provide insights concerning their the impact on the behavior of the algorithm.

- Fourth, given that the GMaxMeanDP is a general model able to formulate a variety of real-world applications, the proposed algorithms can be advantageously applied to solve such practical problems.

The remainder of the paper is organized as follows. In the next section, we describe the proposed algorithms. In Section 3, we assess and compare the performance of the proposed algorithms based on the 160 benchmark instances. We analyse in Section 4 the influence of a key parameter on the performance of the perturbation-based evolutionary algorithm, and discuss the influence of the neighborhood size on the performance of the tabu search methods. Finally, Section 5 gives conclusions and provides some perspectives.

## 2. Two Hybrid Evolutionary Approaches for the GMaxMeanDP

In this section, we describe two hybrid evolutionary algorithms for solving the GMaxMeanDP. We first introduce the perturbation-based evolutionary algorithm (PBEA) that employs a perturbation operator to generate new solutions, and then describe briefly the memetic algorithm (denoted by MAMMDP\*) which is adapted from one of the state-of-the-art MaxMeanDP algorithms (called the MAMMDP algorithm (Lai and Hao, 2016)).

### 2.1. Perturbation Based Evolutionary Algorithm for the GMaxMeanDP

To reach a suitable tradeoff between the intensification and diversification of the search process, the perturbation-based evolutionary algorithm (PBEA) uses an effective tabu search procedure to intensify the search, a random perturbation operator to diversify the search, and a population updating strategy to manage the pool of elite solutions.

#### 2.1.1. General Procedure

As indicated in Algorithm 1, the proposed PBEA algorithm starts with an initial population of  $p$  individuals (solutions) that are generated according to the procedure described in Section 2.1.3 (line 3), and then performs a number of iterations (lines 5–16) to improve the initial population. At each iteration, the algorithm first selects randomly a solution  $s$  from the population, then slightly changes the solution with the perturbation operator (Section 2.1.4), and finally improves the perturbed solution by the tabu search procedure (Section 2.1.5). After that, the improved solution  $s^o$  is used to update the population by using a simple updating rule – the worst individual  $s^w$  in the population is replaced by  $s^o$  if  $s^o$  is distinct from any solution of the population and is better than  $s^w$ ; otherwise  $s^o$  is discarded. The algorithm stops and the solution  $s^*$  is returned when the timeout limit ( $t_{max}$ ) is reached.

---

**Algorithm 1** Perturbation based evolutionary algorithm (PBEA) for the GMaxMeanDP

---

```

1: Input: The set  $V = \{v_1, v_2, \dots, v_n\}$  of  $n$  elements, the associated distance matrix
    $D = [d_{ij}]_{n \times n}$ , the set  $W = \{w_1, w_2, \dots, w_n\}$  of vertex weights, the population size
    $p$ , the timeout limit  $t_{max}$ .
2: Output: the best solution  $s^*$  found
3:  $POP = \{s^1, \dots, s^p\} \leftarrow \text{PopInitialization}(G, p)$  /* Section 2.1.3 */
4:  $s^* \leftarrow \arg \max \{f(s^i) : i = 1, \dots, p\}$  /*  $s^*$  denotes the best solution found */
5: while  $\text{time}() < t_{max}$  do
6:   Randomly select a solution  $s$  from  $POP$ 
7:    $s^o \leftarrow \text{Perturbation}(s)$  /* Section 2.1.4 */
8:    $s^o \leftarrow \text{TabuSearch}(s^o)$  /* Section 2.1.5 */
9:   if  $f(s^o) > f(s^*)$  then
10:     $s^* \leftarrow s^o$ 
11:   end if
12:    $s^w \leftarrow \arg \min \{f(s^i) : i = 1, \dots, p\}$ 
13:   if  $s^o$  does not exist in  $POP$  and  $f(s^o) > f(s^w)$  then
14:      $POP \leftarrow POP \cup \{s^o\} \setminus \{s^w\}$ 
15:   end if
16: end while

```

---

2.1.2. Search Space and Evaluation Function

Since the GMaxMeanDP is an unconstrained binary optimization problem, any  $n$ -dimensional binary vector is a feasible solution. Thus, the search space to be explored by the proposed algorithm is given by

$$\Omega = \{(x_1, x_2, \dots, x_n) : x_i \in \{0, 1\}, 1 \leq i \leq n\} \quad (4)$$

Thus, the size of search space is equal to  $2^n$ , where  $n$  is the number of elements in the problem. Additionally, the quality of a candidate solution  $s = (x_1, x_2, \dots, x_n) \in \Omega$  is given by its objective value  $f(s)$  in Eq. (1).

2.1.3. Population Initialization

---

**Algorithm 2** Initial solution procedure

---

```

1: Input: An input instance  $G$ 
2: Output: A random initial solution  $s = (x_1, x_2, \dots, x_n)$ 
3: for  $i \leftarrow 1$  to  $n$  do
4:    $s.x_i \leftarrow \text{Rand}() \bmod 2$  /* Assign to  $x_i$  of  $s$  a random value in  $\{0, 1\}$  */
5: end for
6:  $s \leftarrow \text{TabuSearch}(s)$  /* Section 2.1.5 */
7: return  $s$ 

```

---

An initial solution  $s$  is generated by randomly assigning each of its components the value 0 or 1. Then, this random solution is improved by the tabu search procedure (Section 2.1.5). We repeat this generation procedure  $p$  times to obtain the initial population. The pseudo-code of this initialization procedure is given in Algorithm 2.

**Algorithm 3** Perturbation operator

---

```

1: Input: Input solution  $s = (x_1, x_2, \dots, x_n)$ , the perturbation strength  $\eta \times n$ 
2: Output: a perturbed solution  $s$ 
3: for  $l \leftarrow 1$  to  $\eta \times n$  do
4:    $i \leftarrow \text{Rand}() \bmod n$  /* Randomly pick a variable  $x_i$  */
5:    $s.x_i \leftarrow \text{Rand}() \bmod 2$  /* Assign to  $x_i$  of  $s$  a random value from  $\{0, 1\}$  */
6: end for
7: return  $s$ 

```

---

140 In order to diversify the search, the proposed algorithm uses a perturbation  
 141 operator to modify a parent solution that is randomly selected from the pop-  
 142 ulation. Specifically, we perform  $\eta \times n$  random changes to the parent solution  
 143 and then return the resulting solution as the perturbed solution, where  $\eta$  is a  
 144 parameter and  $\eta \times n$  is called the perturbation strength. Each random change  
 145 involves first selecting a variable  $x_i$  randomly and then assigning a random value  
 146 0 or 1 to the variable. As such, a large (small) value of  $\eta$  leads to more (fewer)  
 147 changes in the parent solution, thus inducing a strong (weak) diversification  
 148 effect. In practice, our experiments show that  $\eta = 0.4$  is a suitable perturbation  
 149 strength for solving the instances studied in this work (see Section 4.1 for the  
 150 details). Equivalently, this perturbation operator changes the values of about  
 151  $0.2 \times n$  randomly selected variables.

**Algorithm 4** *TabuSearch*( $s_0, N(s), f, Iter_{max}$ )

---

```

1: Input: Input solution  $s_0$ , neighborhood structure  $N(s)$ , evaluation function  $f(s)$ ,  

   maximum number of iterations  $Iter_{max}$ 
2: Output: The best solution  $s_b$  found in the current TS run
3:  $s \leftarrow s_0$  /*  $s$  denotes the current solution */
4:  $s_b \leftarrow s$  /*  $s_b$  denotes the best solution found so far in the current TS run */
5:  $iter \leftarrow 0$  /*  $iter$  denotes the current number of iterations */
6: repeat
7:   Choose randomly a best eligible neighbor solution  $s' \in N(s)$  /* Section 2.1.6 */
   /*  $s'$  is identified to be eligible if it is not forbidden by the tabu list or better than  

    $s_b$  */
8:    $s \leftarrow s'$ 
9:   Update tabu list  $TabuTenure[n]$  with  $s$   

   /*  $TabuTenure[n]$  is a  $n$ -dimensional vector, Section 2.1.8 */
10:  if  $f(s) > f(s_b)$  then
11:     $s_b \leftarrow s$ ,
12:  end if
13:   $iter \leftarrow iter + 1$ 
14: until  $iter = Iter_{max}$ 
15: return  $s_b$ 

```

---

153 The tabu search (TS) method is a popular metaheuristic for combinatorial  
 154 optimization (Glover and Laguna, 1997). Given a neighborhood structure ( $N$ )  
 155 (see Section 2.1.6) and the evaluation function  $f$ , our tabu search procedure per-  
 156 forms a number of iterations to improve the current solution. At each iteration,  
 157 the algorithm replaces the current solution  $s$  by a best eligible neighbor solution  
 158 ( $s' \in N(s)$ ), and meanwhile records the underlying move (see Section 2.1.6) in  
 159 the tabu list to prevent the reverse move from being performed for the next  $tt$   
 160 iterations, where  $tt$  is called the tabu tenure and is adjusted according to the  
 161 tabu list management strategy described in Section 2.1.8. In our TS method,  
 162 a neighbor solution is eligible if it is not forbidden by the tabu list or if it is  
 163 better than the best solution ( $s_b$ ) found so far in the current TS run. Finally,  
 164 the tabu search method stops when a maximum number ( $Iter_{max}$ ) of iterations  
 165 is reached. The general template of the TS method is provided in Algorithm 4,  
 166 and its components are explained in the next sections.

#### 167 2.1.6. Neighborhood Structures

168 In this work, we investigate the following four neighborhood structures.

169 **1) 1-flip neighborhood.** With this basic 1-flip neighborhood (denoted  
 170 by  $N_1$ ), a neighbor solution can be obtained by changing the value of a single  
 171 variable  $x_i$  to its complementary value  $1 - x_i$ . Clearly, this neighborhood  $N_1$   
 172 has a size of  $n$ , where  $n$  is the number of variables.

173 **2) 2-flip neighborhood.** The 2-flip neighborhood (denoted by  $N_2$ ) simul-  
 174 taneously changes the values of two variables  $x_i$  and  $x_j$  to their complementary  
 175 values to generate a neighbor solution. The neighborhood size of  $N_2$  is thus  
 176 equal to  $n(n - 1)/2$ .

177 **3) Union neighborhood.** The third neighborhood  $N_3$  is a combined neigh-  
 178 borhood that is the union of neighborhoods  $N_1$  and  $N_2$ , i.e.,  $N_3 = N_1 \cup N_2$ .  
 179 Thus, the size of  $N_3$  is equal to  $n + n(n - 1)/2$ .

180 **4) Reduced union neighborhood.** The fourth neighborhood (denoted by  
 181  $N_4$ ) is the union of the neighborhood  $N_1$  and a high-quality subset  $N_2^*$  of  $N_2$ ,  
 182 i.e.,  $N_4 = N_1 \cup N_2^*$ . Specifically, given a solution  $s$ , the neighborhood  $N_2^*(s)$  is  
 183 defined by:

$$184 \quad N_2^*(s) = \{s \oplus Flip < i, j > : i \neq j, \{\Delta_i, \Delta_j\} > \Delta_{max} - 0.05(\Delta_{max} - \Delta_{min})\}$$

185 where  $\Delta_{max} = \max_{l \leq n} \Delta_l$ ,  $\Delta_{min} = \min_{l \leq n} \Delta_l$ ,  $\Delta_l$  represents the move value  
 186 (i.e., the change of the objective value) of flipping a single variable  $x_l$  to its  
 187 complementary value, and  $Flip < i, j >$  represents a 2-flip move that simulta-  
 188 neously changes the values of variables  $x_i$  and  $x_j$  to their complementary values.  
 189 Clearly, a neighbor solution  $s' \in N_2^*$  can be obtained by consecutively perform-  
 190 ing two high-quality 1-flip moves from  $s'$ . As a result, the size of  $N_4$  is given by  
 191  $n + |N_2^*|$  and varies dynamically during the search process.

192 In the proposed PBEA algorithm, we select  $N_4$  as the neighborhood struc-  
 193 ture of the tabu search procedure, since  $N_4$  is able to reach a desirable tradeoff  
 194 between computing efficiency and solution quality according to our computa-  
 195 tional experiments (see Section 4.2 for the details).



### 2.1.7. Fast Neighborhood Evaluation Method

To rapidly examine the neighborhood, we employ a fast incremental evaluation method that ensures a high computational efficiency of the tabu search procedure.

Following (Lai and Hao, 2016), our neighborhood evaluation method maintains an  $n$ -dimensional vector  $P = (p_1, p_2, \dots, p_n)$  to rapidly calculate the move value of the possible moves applicable to the solution  $s$  by means of 1-flip or 2-flip operators, where the entry  $p_i$  is defined as the sum of distances between the element  $i$  and the selected elements in the current solution, i.e.,  $p_i = \sum_{j \in M; j \neq i} d_{ij}$ , where  $M$  is the set of selected elements.

If a 1-flip move is performed, then the corresponding move value  $\Delta_i$  can be easily calculated as follows:

$$\Delta_i = \begin{cases} \frac{-f(s)w_i}{SM + w_i} + \frac{p_i}{SM + w_i}, & \text{for } x_i = 0; \\ \frac{f(s)w_i}{SM - w_i} - \frac{p_i}{SM - w_i}, & \text{for } x_i = 1; \end{cases} \quad (5)$$

where  $f(s)$  is the objective value of the solution  $s$  and  $SM$  is the sum of vertex weights of selected elements in  $s$ , i.e.,  $SM = \sum_{i \in M} w_i$ . Subsequently, the vector  $P$  can be updated as follows:

$$p_j = \begin{cases} p_j + d_{ij}, & \text{for } x_i = 0, j \neq i; \\ p_j - d_{ij}, & \text{for } x_i = 1, j \neq i; \\ p_j, & \text{for } j = i; \end{cases} \quad (7)$$

$$p_j = \begin{cases} p_j - d_{ij}, & \text{for } x_i = 1, j \neq i; \\ p_j, & \text{for } j = i; \end{cases} \quad (8)$$

$$p_j = \begin{cases} p_j, & \text{for } j = i; \end{cases} \quad (9)$$

If a 2-flip move is performed by simultaneously flipping variables  $x_i$  and  $x_j$ , then the corresponding move value  $\Delta_{ij}$  can be conveniently obtained by:

$$\Delta_{ij} = \begin{cases} \frac{-f(s)(w_i + w_j) + p_i + p_j + d_{ij}}{SM + w_i + w_j}, & \text{for } x_i = 0, x_j = 0; \\ \frac{f(s)(w_i + w_j) - p_i - p_j + d_{ij}}{SM - w_i - w_j}, & \text{for } x_i = 1, x_j = 1; \\ \frac{f(s)(w_i - w_j) + p_j - p_i + 2d_{ij}}{SM - w_i + w_j}, & \text{for } x_i = 1, x_j = 0; \\ \frac{f(s)(w_j - w_i) + p_i - p_j + 2d_{ij}}{SM - w_j + w_i}, & \text{for } x_i = 0, x_j = 1; \end{cases} \quad (10)$$

$$\Delta_{ij} = \begin{cases} \frac{f(s)(w_i + w_j) - p_i - p_j + d_{ij}}{SM - w_i - w_j}, & \text{for } x_i = 1, x_j = 1; \end{cases} \quad (11)$$

$$\Delta_{ij} = \begin{cases} \frac{f(s)(w_i - w_j) + p_j - p_i + 2d_{ij}}{SM - w_i + w_j}, & \text{for } x_i = 1, x_j = 0; \end{cases} \quad (12)$$

$$\Delta_{ij} = \begin{cases} \frac{f(s)(w_j - w_i) + p_i - p_j + 2d_{ij}}{SM - w_j + w_i}, & \text{for } x_i = 0, x_j = 1; \end{cases} \quad (13)$$

where  $f(s)$  is the objective value of the solution  $s$ ,  $SM = \sum_{i \in M} w_i$ , and  $d_{ij}$  is the distance between elements  $i$  and  $j$ . Subsequently, the vector  $P$  is consecutively updated two times by formula (6-8), since one 2-flip move is composed of two consecutively performed 1-flip moves.

As in (Lai and Hao, 2016), the vector  $P$  can be initialized in  $O(n^2)$  time at the beginning of the tabu search procedure, and updated in  $O(n)$  time after each neighborhood transition.

### 2.1.8. Tabu List Management Strategy

The tabu list management strategy plays a key role in the performance of a tabu search algorithm. In our case, we adopt a popular strategy in the literature to periodically tune the tabu tenure  $tt$ .

In this strategy, the tabu tenure is given by a periodic step function defined on the number of iterations. We denote the current iteration by  $iter$ , and denote the tabu tenure of the current move by  $tt(iter)$ . For each period, the tabu tenure function is defined by a sequence of values  $(a_1, a_2, \dots, a_q)$  and a sequence of interval margins  $(b_1, b_2, \dots, b_{q+1})$ , such that for any  $iter$  in  $[b_i, b_{i+1} - 1]$  we define  $tt(iter) = a_i + rand(C)$ , where  $rand(C)$  denotes a random integer between 0 to  $C - 1$ , and  $C$  is a constant that is set to 3 in this work. The value of  $q$  is set to 15, and  $(a)_{i=1, \dots, 15} = \frac{T_{max}}{8} \times (1, 2, 1, 4, 1, 2, 1, 8, 1, 2, 1, 4, 1, 2, 1)$ , where  $T_{max}$  is a parameter that is used to control the maximum tabu tenure. The interval margins are then defined by  $b_1 = 1$ ,  $b_{i+1} = b_i + 5a_i$  ( $i \leq 15$ ).

For the 1-flip operator and the current number  $iter$  of iterations, if a variable  $x_i$  is flipped by setting  $x_i \leftarrow (1 - x_i)$ , then the variable  $x_i$  is forbidden to change in the following  $tt(iter)$  iterations. For a 2-flip move, if two variables  $x_i$  and  $x_j$  are simultaneously flipped to their complementary values  $1 - x_i$  and  $1 - x_j$ , then both of these variables are forbidden to change in the following  $tt(iter)$  iterations. On the other hand, a 2-flip move  $Flip < i, j >$  is considered to be forbidden if and only if at least one variable is forbidden among the variables  $x_i$  and  $x_j$ .

This tabu list management strategy is adapted from a method proposed in (Galinier et al., 2011), whose effectiveness has been demonstrated for several hard optimization problems, such as the graph partitioning problem (Galinier et al., 2011), the maximum diversity problem (Wu and Hao, 2013), and the Max-Mean dispersion problem (Lai and Hao, 2016). In principle, a small tabu tenure leads usually to a strong search intensification while a large tabu tenure favors search diversification. As such, the periodical change of the tabu tenure among several small and large values provides a strategy to reach a desirable balance between the intensification and diversification of the search.

### 2.2. Memetic Approach for the GMaxMeanDP

The memetic algorithm MAMMDP presented in (Lai and Hao, 2016) is a state-of-the-art algorithm for solving the MaxMeanDP, which is a special case of the GMaxMeanDP studied in this work. In order to verify the potential merit of the MAMMDP approach for the GMaxMeanDP, we adapt MAMMDP to the GMaxMeanDP by basically replacing its local search component with the tabu search method in Section 2.1.5 in which the fast neighborhood  $N_1$  is adopted while keeping its other ingredients (e.g., crossover and pool updating) unchanged. We use MAMMDP\* to denote this adapted algorithm for the GMaxMeanDP. Thus, the main difference between the MAMMDP\* and MAMMDP algorithms lies at their local search methods. In the local search method of MAMMDP\*, in order to consider the weights of vertices, we employ an extended incremental neighborhood evaluation technique that uses Eqs.(5) and (6) to calculate quickly the move values of neighborhood moves.

---

**Algorithm 5** Memetic Algorithm for the GMaxMeanDP (MAMMDP\*)
 

---

```

1: Input: The set  $V = \{v_1, v_2, \dots, v_n\}$  of  $n$  elements and the associated distance matrix
    $D = [d_{ij}]_{n \times n}$ , the set  $W = \{w_1, w_2, \dots, w_n\}$  of vertex weights, the population size
    $p$ , the timeout limit  $t_{max}$ .
2: Output: the best solution  $s^*$  found
3: repeat
4:    $POP = \{s^1, \dots, s^p\} \leftarrow \text{PopInitialization}(V, p)$  /* Section 2.1.3 */
5:   if the repeat loop is not performing its first execution then
6:      $s^w \leftarrow \arg \min\{f(s^i) : i = 1, \dots, p\}$ 
7:      $POP \leftarrow POP \cup \{s^*\} \setminus \{s^w\}$ 
8:   end if
9:    $s^* \leftarrow \arg \max\{f(s^i) : i = 1, \dots, p\}$  /*  $s^*$  keeps the best solution found */
10:   $PairSet \leftarrow \{(s^i, s^j) : 1 \leq i < j \leq p\}$ 
11:  while  $PairSet \neq \emptyset$  and  $time() < t_{max}$  do
12:    Randomly pick a solution pair  $(s^i, s^j) \in PairSet$ 
13:     $s^o \leftarrow \text{CrossoverOperator}(s^i, s^j)$  /* uniformly random crossover operator */
14:     $s^o \leftarrow \text{TabuSearch}(s^o)$  /* Section 2.1.5 */
15:    if  $f(s^o) > f(s^*)$  then
16:       $s^* \leftarrow s^o$ 
17:    end if
18:     $PairSet \leftarrow PairSet \setminus \{(s^i, s^j)\}$ 
19:     $s^w \leftarrow \arg \min\{f(s^i) : i = 1, \dots, p\}$ 
20:    if  $s^o$  dose not exist in  $POP$  and  $f(s^o) > f(s^w)$  then
21:       $POP \leftarrow POP \cup \{s^o\} \setminus \{s^w\}$ 
22:       $PairSet \leftarrow PairSet \setminus \{(s^w, s^k) : s^k \in POP\}$ 
23:       $PairSet \leftarrow PairSet \cup \{(s^o, s^k) : s^k \in POP\}$ 
24:    end if
25:  end while
26: until  $time() \geq t_{max}$ 

```

---

263 MAMMDP\* is composed of four components: a population initialization pro-  
 264 cedure, a tabu search based optimization procedure, a crossover operator, and a  
 265 population updating rule. For the sake of completeness, the pseudo-code of the  
 266 MAMMDP\* algorithm, which closely follows the MAMMDP algorithm in (Lai  
 267 and Hao, 2016), is shown in Algorithm 5, where  $POP = \{s^1, \dots, s^p\}$  denotes  
 268 the current population,  $s^o$  denotes the new solution generated by the crossover  
 269 operator or by the tabu search procedure,  $s^*$  and  $s^w$  denote respectively the best  
 270 solution found so far and the worst solution in  $POP$ , and  $PairSet$  represents  
 271 the set of solution pairs that have not been used by the crossover operator in  
 272  $POP$ .

273 MAMMDP\* starts with the initial population generated by the initialization  
 274 procedure in Section 2.1.3 and then performs a number of generations until the  
 275 timeout limit  $t_{max}$  is reached, i.e.,  $time() \geq t_{max}$ . At each generation, a solution  
 276 pair  $(s^i, s^j)$  is randomly chosen from  $PairSet$  (line 12), and then used to gen-  
 277 erate a new solution  $s^o$  by the standard uniform crossover operator (Syswerda,  
 278 1989) (line 13). The quality of  $s^o$  is improved by the tabu search procedure  
 279 (line 14). Subsequently,  $s^*$ ,  $POP$  and  $PairSet$  are accordingly updated (lines  
 280 15–24). Finally, to diversify the search, the population  $POP$  and the associated  
 281  $PairSet$  are re-initialized each time  $PairSet$  becomes empty, while keeping  $s^*$   
 282 in the new population (lines 4–10).

### 283 3. Computational Experiments

284 We perform computational experiments on six types of 160 benchmark in-  
 285 stances to assess the proposed algorithms. The benchmark instances, the exper-  
 286 imental protocol, and the computational results are presented in the following  
 287 subsections.

#### 288 3.1. Benchmark Instances

289 For the GMaxMeanDP, no vertex-weighted benchmark instance is available  
 290 in the literature. To evaluate the performance of our algorithms, we generated  
 291 four types of instances, each containing 20 vertex-weighted instances<sup>1</sup>. For each  
 292 type, we generated 10 instances with  $n = 3000$  and 10 instances with  $n = 5000$ ,  
 293 where the distances between elements and the vertex weights were randomly  
 294 selected from a given set with the uniform probability distribution. Given that  
 295 any MaxMeanDP instance can be viewed as a special GMaxMeanDP instance  
 296 in which all vertex weights take the value of 1, we additionally used two types  
 297 of 80 MaxMeanDP instances<sup>2</sup>, which were used in (Brimberg et al., 2017) or  
 298 (Lai and Hao, 2016) to assess the MaxMeanDP algorithms. The characteristics  
 299 of these 160 instances are as follows:

<sup>1</sup>Available at <http://www.info.univ-angers.fr/pub/hao/gmaxmeandp.html>

<sup>2</sup>Available at <http://www.info.univ-angers.fr/pub/hao/maxmeandp.html> and <http://www.mi.sanu.ac.rs/~nenad/edp/>

- Type I (20 instances): The distances  $d_{ij}$  between elements were randomly generated in the interval  $[-10, 10]$ , and the vertex weights  $w_i$  ( $i = 1, 2, \dots, n$ ) were randomly generated in the interval  $[1, 5]$ .
- Type II (20 instances): The distances  $d_{ij}$  between elements were randomly taken in the interval  $[-10, -5] \cup [5, 10]$ , and the vertex weights  $w_i$  ( $i = 1, 2, \dots, n$ ) were randomly generated in the interval  $[1, 6]$ .
- Type III (20 instances): The distances  $d_{ij}$  between elements were randomly selected from the set  $\{-1, 0, 1\}$ , and the vertex weights  $w_i$  ( $i = 1, 2, \dots, n$ ) were randomly generated in the interval  $[0.9, 1.1]$ .
- Type IV (20 instances): The distances  $d_{ij}$  between elements were randomly taken from the set  $\{-10, 0, 10\}$ , and the vertex weights  $w_i$  ( $i = 1, 2, \dots, n$ ) were uniformly set to 1.
- Type MDPI (40 instances): This set of MaxMeanDP instances includes 10 instances for each  $n \in \{1500, 2000, 3000, 5000\}$ . The distances between elements were uniformly randomly generated in the interval  $[-10, 10]$ , and the vertex weights  $w_i$  ( $i = 1, 2, \dots, n$ ) were uniformly set to 1.
- Type MDPII (40 instances): This set of MaxMeanDP instances includes 10 instances for each  $n \in \{1500, 2000, 3000, 5000\}$ . The distances between elements were randomly generated in the interval  $[-10, -5] \cup [5, 10]$ , and the vertex weights  $w_i$  ( $i = 1, 2, \dots, n$ ) were uniformly set to 1.

### 3.2. Experimental Protocol

Table 1: Settings of parameters

Parameters	Section	Description	Values
$p$	2.1.1	size of population	20
$Iter_{max}$	2.1.5	maximum number of iterations for the tabu search	$5 \times 10^4$
$T_{max}$	2.1.8	the maximum tabu tenure	$80 + Rand(100)$
$\eta$	2.1.4	strength of the perturbation operator	0.4

The PBEA algorithm adopts four parameters, including the population size  $p$ , the maximum number  $Iter_{max}$  of iterations and the maximum tabu tenure  $T_{max}$  for the tabu search procedure, and the coefficient  $\eta$  used to control the perturbation strength, whose values are empirically set as in Table 1. The MAMMDP\* algorithm has three parameters: the population size  $p$  which was set to 10 following the setting of original MAMMDP algorithm in (Lai and Hao, 2016),  $Iter_{max}$  and  $T_{max}$  whose values were set as in Table 1. In addition, both MAMMDP\* and PBEA were implemented in C and compiled by the g++ compiler with the -O3 option, and the corresponding experiments were carried out on a computing platform with an Intel E5-2670 processor (2.5 GHz and 2G RAM), running the Linux operating system. The source codes of the proposed MAMMDP\* and PBEA algorithms will be available at <http://www.info.univ-angers.fr/pub/hao/gmaxmeandp.html>.

In addition, due to the stochastic feature of both algorithms, PBEA and MAMMDP\* were independently run 20 times to solve each instance based on the same time limit  $t_{max}$  for each run, where  $t_{max}$  was set to 100, 500 and 1000 seconds for the instances with  $n \leq 2000$ ,  $n = 3000$  and  $n = 5000$ , respectively. Finally, we employed a commercial software called LocalSolver (<https://www.localsolver.com/>) as our reference algorithm, since no direct reference algorithm is available in the literature for the GMaxMeanDP. In our experiment, we ran LocalSolver once for each instance with the same time limit  $t_{max}$  as our proposed algorithms on a computer with a Intel i7-6700 processor (3.4 GHz CPU and 4G RAM), running Windows 10 operating system, since we only obtained an academic license of LocalSolver on this computer.

### 3.3. Computational Results and Comparisons on the MaxMeanDP Instances

Table 2: Computational results and comparisons on the 40 MaxMeanDP instances with  $n = 1500$  or 2000 from the literature. The best  $f_{best}$  values among all the results are indicated in boldface.

Instance	VNS	LocalSolver	MAMMDP* (this work)				PBEA (this work)			
	$f_{best}$	$f$	$f_{best}$	$f_{avg}$	SR	$t(s)$	$f_{best}$	$f_{avg}$	SR	$t(s)$
MDPI1_1500	136.26	66.6568	<b>136.535222</b>	136.535222	20/20	14.21	<b>136.535222</b>	136.535222	20/20	18.75
MDPI2_1500	138.00	70.7226	<b>138.341482</b>	138.341482	20/20	5.38	<b>138.341482</b>	138.341482	20/20	8.04
MDPI3_1500	138.91	66.8269	<b>139.200599</b>	139.200599	20/20	3.17	<b>139.200599</b>	139.200599	20/20	3.28
MDPI4_1500	139.81	68.0931	<b>140.166920</b>	140.166920	20/20	5.65	<b>140.166920</b>	140.166920	20/20	4.67
MDPI5_1500	136.47	66.8041	<b>137.129630</b>	137.129630	20/20	7.73	<b>137.129630</b>	137.129630	20/20	12.65
MDPI6_1500	136.22	65.6676	<b>136.508768</b>	136.508768	20/20	7.05	<b>136.508768</b>	136.508768	20/20	10.13
MDPI7_1500	137.65	63.4105	<b>137.971032</b>	137.971032	20/20	2.49	<b>137.971032</b>	137.971032	20/20	3.20
MDPI8_1500	138.02	67.9306	<b>138.728444</b>	138.728444	20/20	13.56	<b>138.728444</b>	138.728444	20/20	13.95
MDPI9_1500	136.30	66.9695	<b>136.495674</b>	136.495674	20/20	21.39	<b>136.495674</b>	136.495674	20/20	28.95
MDPI10_1500	140.33	66.0519	140.333159	140.333159	20/20	3.47	140.333159	140.333159	20/20	3.90
MDPI1_2000	158.03	55.3813	<b>158.588217</b>	158.588217	20/20	10.40	<b>158.588217</b>	158.588217	20/20	11.79
MDPI2_2000	162.91	54.2658	<b>163.939616</b>	163.939616	20/20	19.11	<b>163.939616</b>	163.939616	20/20	31.71
MDPI3_2000	158.98	51.9819	<b>159.570786</b>	159.545090	13/20	39.86	<b>159.570786</b>	159.528479	6/20	38.94
MDPI4_2000	159.14	52.6407	<b>160.185217</b>	160.185217	20/20	28.46	<b>160.185217</b>	160.184761	17/20	54.41
MDPI5_2000	156.11	53.8956	<b>156.805331</b>	156.758147	10/20	41.25	<b>156.805331</b>	156.776147	13/20	55.30
MDPI6_2000	161.61	52.1516	<b>161.839100</b>	161.839100	20/20	11.30	<b>161.839100</b>	161.839100	20/20	13.72
MDPI7_2000	157.58	53.8223	<b>158.336131</b>	158.336131	20/20	9.79	<b>158.336131</b>	158.336131	20/20	7.93
MDPI8_2000	161.43	53.6872	<b>161.446931</b>	161.446931	20/20	20.03	<b>161.446931</b>	161.446931	20/20	22.30
MDPI9_2000	159.15	54.9125	<b>160.190374</b>	160.190374	20/20	29.21	<b>160.190374</b>	160.187769	17/20	44.28
MDPI10_2000	160.90	53.6239	<b>161.638099</b>	161.638099	20/20	7.60	<b>161.638099</b>	161.638099	20/20	4.99
MDPI1_1500	181.67	94.7889	<b>182.089413</b>	182.089413	20/20	6.33	<b>182.089413</b>	182.089413	20/20	8.23
MDPI2_1500	185.48	98.7439	<b>186.243869</b>	186.243869	20/20	6.78	<b>186.243869</b>	186.243869	20/20	4.66
MDPI3_1500	181.55	93.3692	<b>182.142902</b>	182.142902	20/20	3.13	<b>182.142902</b>	182.142902	20/20	4.76
MDPI4_1500	184.91	92.6379	<b>185.557302</b>	185.500190	8/20	42.93	<b>185.557302</b>	185.514675	9/20	35.93
MDPI5_1500	190.15	101.379	<b>190.860529</b>	190.860529	20/20	2.25	<b>190.860529</b>	190.860529	20/20	1.65
MDPI6_1500	183.14	99.3436	<b>183.575336</b>	183.575336	20/20	3.05	<b>183.575336</b>	183.575336	20/20	1.90
MDPI7_1500	179.34	93.6409	<b>179.820242</b>	179.820242	20/20	13.93	<b>179.820242</b>	179.820242	20/20	18.43
MDPI8_1500	186.60	96.7090	186.602804	186.602804	20/20	2.74	186.602804	186.602804	20/20	3.30
MDPI9_1500	181.43	97.7207	<b>181.918814</b>	181.918814	20/20	17.85	<b>181.918814</b>	181.918814	20/20	14.75
MDPI10_1500	182.70	99.0640	<b>183.384692</b>	183.384692	20/20	32.37	<b>183.384692</b>	183.384692	20/20	26.01
MDPI1_2000	208.85	75.3906	<b>209.845273</b>	209.845273	20/20	8.13	<b>209.845273</b>	209.845273	20/20	11.24
MDPI2_2000	218.19	81.7475	<b>218.404860</b>	218.404860	20/20	16.03	<b>218.404860</b>	218.404860	20/20	22.40
MDPI3_2000	209.57	69.9621	<b>210.819147</b>	210.807415	19/20	15.52	<b>210.819147</b>	210.819147	20/20	18.05
MDPI4_2000	211.99	74.7847	<b>212.424859</b>	212.424859	20/20	16.15	<b>212.424859</b>	212.424859	20/20	25.81
MDPI5_2000	215.33	75.5558	<b>216.088722</b>	216.088722	20/20	9.90	<b>216.088722</b>	216.088722	20/20	8.96
MDPI6_2000	210.61	73.9974	<b>211.769151</b>	211.769151	20/20	10.88	<b>211.769151</b>	211.769151	20/20	6.88
MDPI7_2000	209.65	77.1172	<b>209.780651</b>	209.780651	20/20	19.95	<b>209.780651</b>	209.780651	20/20	25.74
MDPI8_2000	212.43	80.1608	<b>212.575432</b>	212.575432	20/20	17.03	<b>212.575432</b>	212.575432	20/20	27.75
MDPI9_2000	214.61	72.2590	<b>215.007759</b>	215.007759	20/20	15.87	<b>215.007759</b>	215.007759	20/20	12.92
MDPI10_2000	210.06	74.5694	<b>210.735749</b>	210.735749	15/20	28.22	<b>210.735749</b>	210.735561	17/20	28.19
Avg.	173.30	73.2110	173.839956	173.836405		16.44	173.839956	173.837022		19.25
#Best	2	0	40				40			
p-value	3.569e-8	3.569e-8	1.0	0.6121						

The first experiment aims to assess and compare the proposed PBEA algorithm and the adapted MAMMDP\* algorithm on the MaxMeanDP instances (i.e., the unweighted GMaxMeanDP instances), since the MaxMeanDP is a special case of the GMaxMeanDP in which all vertex weights take the value of 1

Table 3: Computational results and comparisons on the 40 large MaxMeanDP instances with  $n = 3000$  or  $5000$  from the literature. The best  $f_{best}$  and  $f_{avg}$  values among the compared results are indicated in boldface.

Instance	LocalSolver	MAMMDP* (this work)					PBEA (this work)				
	$f$	$f_{best}$	$f_{avg}$	SR	$t(s)$		$f_{best}$	$f_{avg}$	SR	$t(s)$	
MDPI1_3000	72.8274	189.048965	189.048965	20/20	54.08		189.048965	189.048965	20/20	75.99	
MDPI2_3000	72.8196	187.387292	187.387292	20/20	50.59		187.387292	187.387292	20/20	81.62	
MDPI3_3000	71.1284	185.666806	<b>185.642604</b>	5/20	310.42		185.666806	185.640815	4/20	173.32	
MDPI4_3000	67.3049	186.163727	<b>186.159939</b>	19/20	165.94		186.163727	186.156150	18/20	121.87	
MDPI5_3000	68.5859	187.545515	187.545515	20/20	56.64		187.545515	187.545515	20/20	124.46	
MDPI6_3000	71.5833	189.431257	189.431257	20/20	36.28		189.431257	189.431257	20/20	71.08	
MDPI7_3000	65.0592	188.242583	188.242583	20/20	90.13		188.242583	188.242583	20/20	76.43	
MDPI8_3000	68.5892	186.796814	186.796814	20/20	36.91		186.796814	186.796814	20/20	75.72	
MDPI9_3000	70.9764	188.231264	188.228646	19/20	65.43		188.231264	<b>188.231264</b>	20/20	84.02	
MDPI10_3000	69.1644	185.682511	185.572559	4/20	105.14		185.682511	<b>185.632187</b>	11/20	197.56	
MDPI11_3000	97.6705	252.320433	252.320433	20/20	46.18		252.320433	252.320433	20/20	90.08	
MDPI12_3000	101.229	250.062137	<b>250.062137</b>	20/20	127.57		250.062137	250.060127	16/20	248.03	
MDPI13_3000	104.731	251.906270	251.906270	20/20	99.94		251.906270	251.906270	20/20	142.28	
MDPI14_3000	99.7977	253.941007	253.936173	14/20	187.38		253.941007	<b>253.939366</b>	16/20	208.28	
MDPI15_3000	103.008	253.260423	<b>253.260302</b>	15/20	190.57		253.260423	253.260278	14/20	256.84	
MDPI16_3000	104.409	250.677750	250.677750	20/20	49.99		250.677750	250.677750	20/20	58.46	
MDPI17_3000	100.621	251.134413	251.134413	20/20	55.07		251.134413	251.134413	20/20	99.94	
MDPI18_3000	105.536	252.999648	252.999648	20/20	74.56		252.999648	252.999648	20/20	83.54	
MDPI19_3000	100.811	252.425770	252.425770	20/20	45.77		252.425770	252.425770	20/20	114.67	
MDPI110_3000	99.4736	252.396590	252.396590	20/20	16.30		252.396590	252.396590	20/20	15.39	
MDPI1_5000	NA	240.141212	<b>240.070982</b>	9/20	464.66		<b>240.162535</b>	240.015046	1/20	644.69	
MDPI2_5000	NA	241.827401	<b>241.744421</b>	5/20	360.20		241.827401	241.735443	2/20	495.52	
MDPI3_5000	NA	240.890819	<b>240.865427</b>	15/20	410.53		240.890819	240.812439	11/20	466.86	
MDPI4_5000	NA	240.997186	240.951055	4/20	592.65		240.997186	<b>240.955450</b>	4/20	656.19	
MDPI5_5000	NA	242.480129	<b>242.471643</b>	18/20	269.86		242.480129	242.454732	14/20	612.06	
MDPI6_5000	NA	240.322850	<b>240.304443</b>	14/20	33.30		<b>240.376038</b>	240.281210	1/20	585.48	
MDPI7_5000	NA	242.820139	<b>242.771514</b>	4/20	490.60		242.820139	242.771003	1/20	604.73	
MDPI8_5000	NA	241.194990	<b>241.154430</b>	13/20	111.35		241.194990	241.138956	5/20	568.30	
MDPI9_5000	NA	<b>239.760560</b>	<b>239.566397</b>	7/20	139.82		239.681094	239.498462	3/20	536.47	
MDPI10_5000	NA	243.385487	<b>243.345183</b>	8/20	548.48		<b>243.473734</b>	243.334446	1/20	521.23	
MDPI11_5000	NA	322.235897	<b>322.177715</b>	4/20	298.40		322.235897	322.148548	2/20	581.82	
MDPI12_5000	NA	327.301910	<b>326.996573</b>	5/20	729.93		327.301910	326.970214	4/20	551.71	
MDPI13_5000	NA	324.813456	<b>324.792109</b>	9/20	290.15		324.813456	324.785177	3/20	482.32	
MDPI14_5000	NA	322.227657	<b>322.182679</b>	6/20	422.89		<b>322.237586</b>	322.126451	2/20	705.12	
MDPI15_5000	NA	322.491211	322.355484	3/20	506.35		322.491211	<b>322.365463</b>	4/20	556.09	
MDPI16_5000	NA	322.728902	<b>322.638339</b>	3/20	101.37		322.950488	322.629351	2/20	678.45	
MDPI17_5000	NA	322.850438	322.773052	8/20	606.48		322.850438	<b>322.787011</b>	9/20	415.61	
MDPI18_5000	NA	323.112120	<b>323.009085</b>	6/20	285.51		323.112120	322.948455	2/20	555.26	
MDPI19_5000	NA	323.543775	<b>323.299190</b>	5/20	774.36		323.543775	323.182444	1/20	574.49	
MDPI110_5000	NA	324.519908	<b>324.456763</b>	17/20	440.56		324.519908	324.335221	12/20	500.37	
Avg.	NA	251.124181	251.077554		243.56		251.132051	251.062725		342.31	
#Best	0	36					39				
p-value	NA	0.173	3.649e-3								

and any algorithm for the GMaxMeanDP problem can be directly applied to the MaxMeanDP problem as well. The experimental results on the 40 medium-sized instances with  $n = 1500, 2000$  and the 40 large instances with  $n = 3000, 5000$  from the MDPI and MDPII sets are summarized in Tables 2 and 3 respectively. For this experiment, in addition to LocalSolver, we also adopted as another reference method the VNS algorithm designed, which is one of the state of the art MaxMeanDP algorithms (Brimberg et al., 2017). Please note that when it is applied to the MaxMeanDP, the MAMMDP\* algorithm becomes MAMMDP presented in (Lai and Hao, 2016).

In Table 2 (for the 40 medium-sized instances with  $n = 1500$  or  $2000$ ), the first column gives the names of instances, columns 2-3 report respectively the best results from the VNS algorithm and the results of LocalSolver. Columns 4-7 report the results of the MAMMDP\* algorithm over 20 runs, including the best objective value ( $f_{best}$ ), the average objective value ( $f_{avg}$ ), the success rate (SR) to reach the associated  $f_{best}$  value, and the average run time ( $t(s)$ ) in seconds to obtain its final result. Columns 8-11 report the results of the PBEA algorithm with the same information as in the columns 4-7. The row *Avg.* shows the average result for each associated column. The row *#Best* shows the number of instances for which an algorithm finds the best results in terms of  $f_{best}$  among the compared algorithms. Finally, to verify the statistical difference between the dedicated PBEA algorithm and other algorithms in terms of  $f_{best}$  and  $f_{avg}$ , the *p-values* from the Wilcoxon signed-rank tests are given in the last row of the tables, where a *p-value* less than 0.05 means that there exists a significant difference between the compared results. Moreover, the results of LocalSolver are compared with the average results of PBEA algorithm, since LocalSolver was run once for each instance.

Table 3 reports the results on the 40 large instances with  $n = 3000$  and  $5000$  in the same way as in Table 2, where 'NA' indicates that LocalSolver failed to provide a result due to the memory limitation of the computer used. We ignore the VNS algorithm in Table 3 since the results on these large instances are not reported in (Brimberg et al., 2017) for this method.

From Table 2, we observe that both the proposed PBEA algorithm and the adapted MAMMDP\* algorithm dominate the VNS algorithm and the general-purpose LocalSolver software on the medium-sized MaxMeanDP instances. Compared with the dedicated VNS algorithm designed for MaxMeanDP in (Brimberg et al., 2017), MAMMDP\* and PBEA obtain better results in terms of  $f_{best}$  for 38 out of 40 instances and the same results for the two remaining instances. It is worth noting that for these instances the results of VNS algorithms were obtained in (Brimberg et al., 2017) by using a time limit of  $t_{max} = n$  that is much longer than the time used in this work ( $t_{max} = 100$ ). Compared with LocalSolver, the dominance of MAMMDP\* and PBEA is even more evident for all tested instances. The small *p-value* confirms that there is a significant difference between the proposed PBEA algorithm and these two reference algorithms in terms of  $f_{best}$ . On the other hand, MAMMDP\* and PBEA perform similarly on these instances. First, both algorithms obtain the same  $f_{best}$  values for all 40 instances. Second, both algorithms have a high success rate (SR = 100%) for



most instances, while the computation time to obtain their final results is less than 1.0 minute for any instance. Moreover, the large  $p$ -values indicate that there does not exist a significant difference between the results of MAMMDP\* and PBEA in terms of  $f_{best}$  and  $f_{avg}$ . These outcomes imply that MAMMDP\* and PBEA are both highly efficient for solving the medium-sized MaxMeanDP instances, and the crossover operator of the MAMMDP\* algorithm and the perturbation operator of the PBEA algorithm have a similar diversification ability.

Table 3 shows that the PBEA algorithm and the adapted MAMMDP\* algorithm significantly outperform the LocalSolver software on these large-scale instances with  $n = 3000, 5000$ . Between MAMMDP\* and PBEA, one observes that they obtain the same result in  $f_{best}$  for 35 out of the 40 instances. Even if PBEA performs marginally better in terms of  $f_{best}$  with four better  $f_{best}$  results for PBEA against one better  $f_{best}$  result for MAMMDP\* ( $p$ -value  $> 0.05$ ), MAMMDP\* is better in terms of  $f_{avg}$  with 21 better  $f_{avg}$  results against six better  $f_{avg}$  results for PBEA with a  $p$ -value  $< 0.05$ . Finally, the success rates decrease significantly for both MAMMDP\* and PBEA as the size of instance increases, indicating the high difficulty of these largest instances.

In summary, this experiment indicates that when they are applied to the MaxMeanDP which is a special case of GMaxMeanDP, both the PBEA algorithm and the adapted MAMMDP\* algorithm perform very competitively compared to the general-purpose software LocalSolver and the dedicated VNS algorithm. In the next section, we assess the MAMMDP\* and PBEA algorithm for solving the GMaxMeanDP for which they were designed.

#### 3.4. Computational Results and Comparisons on the Weighted Instances

We now turn our attention to the assessment of MAMMDP\* and PBEA on the set of 40 large GMaxMeanDP for which these algorithms are designed. We report in Tables 4 and 5 the computational results of MAMMDP\* and PBEA on the instances with  $n = 3000, 5000$  respectively. Table 4 also includes the results of LocalSolver while the instances with  $n = 5000$  are too large for LocalSolver on our computer. In these tables, the same information as in the last section is reported.

We observe from Table 4 that both the MAMMDP\* and PBEA algorithms largely dominate the general-purpose LocalSolver software in terms of solution quality. For each instance, MAMMDP\* and PBEA obtain a much better solution than LocalSolver. On the other hand, the MAMMDP\* and PBEA algorithms have a similar performance for these instances with  $n = 3000$ . First, the two algorithms obtain the same result in term of  $f_{best}$  for 39 out of 40 instances ( $p$ -value  $> 0.05$ ). In terms of  $f_{avg}$ , PBEA has a slightly better result than MAMMDP\* (121.959884 vs. 121.958056) ( $p$ -value  $> 0.05$ ). Furthermore, both algorithms report the same  $f_{best}$  value with a success rate of 100% for the 28 instances, indicating that they are highly robust for these instances. These outcomes indicate that the PBEA and MAMMDP\* algorithms perform similarly on the GMaxMeanDP instances with  $n = 3000$ .

Table 5 shows that the overall performances of both algorithms are globally quite similar: 157.856608 for MAMMDP\* vs 157.853553 for PBEA in terms of

Table 4: Computational results and comparisons on the 40 large GMaxMeanDP instances (weighted instances) with  $n = 3000$ . The best  $f_{best}$  and  $f_{avg}$  values among the compared results are indicated in boldface.

Instance	LocalSolver	MAMMDP* (this work)				PBEA (this work)			
	$f$	$f_{best}$	$f_{avg}$	SR	$t(s)$	$f_{best}$	$f_{avg}$	SR	$t(s)$
I_3000_1	22.7528	80.743467	80.743467	20/20	44.45	80.743467	80.743467	20/20	92.53
I_3000_2	25.1306	84.201027	84.201027	20/20	17.82	84.201027	84.201027	20/20	46.71
I_3000_3	24.4089	81.630082	81.630082	20/20	6.62	81.630082	81.630082	20/20	9.34
I_3000_4	25.8106	80.234334	80.234334	20/20	29.61	80.234334	80.234334	20/20	28.07
I_3000_5	24.5990	81.218062	81.218043	19/20	108.59	81.218062	<b>81.218062</b>	20/20	138.28
I_3000_6	23.5651	83.197618	83.197618	20/20	37.99	83.197618	83.197618	20/20	64.05
I_3000_7	24.0235	81.732080	81.732080	20/20	2.73	81.732080	81.732080	20/20	4.50
I_3000_8	22.5924	80.624273	80.624273	20/20	79.61	80.624273	80.624273	20/20	83.53
I_3000_9	25.3955	80.574438	80.574438	20/20	7.59	80.574438	80.574438	20/20	10.76
I_3000_10	25.0323	83.397670	83.397670	20/20	48.63	83.397670	83.397670	20/20	138.84
II_3000_1	31.2938	99.055143	99.055143	20/20	15.04	99.055143	99.055143	20/20	12.66
II_3000_2	32.1219	105.574146	105.574146	20/20	29.08	105.574146	105.574146	20/20	63.27
II_3000_3	29.8576	101.299271	101.299271	20/20	3.31	101.299271	101.299271	20/20	6.71
II_3000_4	28.9800	101.079824	101.079824	20/20	8.41	101.079824	101.079824	20/20	8.03
II_3000_5	32.9165	100.029225	<b>100.029225</b>	20/20	84.01	100.029225	100.028322	18/20	241.64
II_3000_6	29.1903	101.978783	101.978783	20/20	5.80	101.978783	101.978783	20/20	4.56
II_3000_7	31.5154	100.189718	100.189718	20/20	6.43	100.189718	100.189718	20/20	17.36
II_3000_8	32.0808	101.160428	101.160428	20/20	3.36	101.160428	101.160428	20/20	4.52
II_3000_9	30.5477	98.665034	98.665034	20/20	39.15	98.665034	98.665034	20/20	59.96
II_3000_10	31.4593	104.896612	104.896612	20/20	4.40	104.896612	104.896612	20/20	11.86
III_3000_1	10.8747	27.847334	27.847334	20/20	102.65	27.847334	27.847334	20/20	108.70
III_3000_2	10.9677	27.776796	<b>27.774430</b>	7/20	214.29	27.776796	27.774272	4/20	120.08
III_3000_3	11.8823	27.946519	27.944592	17/20	147.23	27.946519	<b>27.946519</b>	20/20	157.85
III_3000_4	10.6279	27.816272	27.816272	20/20	81.41	27.816272	27.816272	20/20	70.66
III_3000_5	11.3929	27.727167	27.727167	20/20	115.40	27.727167	27.727167	20/20	160.51
III_3000_6	10.9057	27.686986	27.677719	8/20	136.73	<b>27.691682</b>	<b>27.686631</b>	4/20	131.25
III_3000_7	11.3789	27.642060	27.642060	20/20	74.29	27.642060	27.642060	20/20	158.84
III_3000_8	11.0592	27.736643	27.733842	5/20	287.29	27.736643	<b>27.734079</b>	6/20	184.58
III_3000_9	11.4658	27.745820	27.744637	19/20	139.88	27.745820	<b>27.745820</b>	20/20	77.88
III_3000_10	10.7100	27.561083	27.560295	19/20	157.43	27.561083	<b>27.561083</b>	20/20	92.74
IV_3000_1	136.7020	278.039443	<b>278.037117</b>	19/20	137.79	278.039443	278.027811	15/20	151.80
IV_3000_2	131.0830	276.539877	276.530847	18/20	216.82	276.539877	<b>276.539691</b>	18/20	238.37
IV_3000_3	127.1120	277.334878	277.334878	20/20	31.02	277.334878	277.334878	20/20	40.40
IV_3000_4	131.6190	278.956422	278.956422	20/20	42.08	278.956422	278.956422	20/20	61.36
IV_3000_5	130.2750	276.595238	276.595238	20/20	152.28	276.595238	276.595238	20/20	108.00
IV_3000_6	127.5350	280.721533	280.721533	20/20	55.32	280.721533	280.721533	20/20	60.47
IV_3000_7	132.0830	273.653396	273.653396	20/20	84.47	273.653396	273.653396	20/20	169.85
IV_3000_8	128.6810	276.358447	276.358447	20/20	70.96	276.358447	276.358447	20/20	81.56
IV_3000_9	133.6610	274.864865	274.821773	17/20	159.03	274.864865	<b>274.838571</b>	18/20	241.92
IV_3000_10	132.8980	276.428571	<b>276.411918</b>	17/20	220.16	276.428571	276.407810	16/20	151.95
Avg.	49.4047	121.961632	121.958056		92.85	121.961632	121.959884		90.40
#Best	0	39				40			
p-value	3.569e-8	0.3173	0.3078						

Table 5: Computational results and comparisons on the 40 large GMaxMeanDP instances (weighted instances) with  $n = 5000$ . The best  $f_{best}$  and  $f_{avg}$  values among the compared results are indicated in boldface.

Instance	MAMMDP* (this work)				PBEA (this work)			
	$f_{best}$	$f_{avg}$	SR	$t(s)$	$f_{best}$	$f_{avg}$	SR	$t(s)$
I_5000_1	<b>104.827798</b>	<b>104.803953</b>	7/20	480.14	104.818572	104.779182	1/20	603.16
I_5000_2	104.053704	<b>104.053704</b>	20/20	154.85	104.053704	104.045180	10/20	433.77
I_5000_3	<b>104.803139</b>	<b>104.794625</b>	12/20	469.36	104.796184	104.794027	14/20	577.26
I_5000_4	107.326793	107.300838	10/20	500.22	107.326793	<b>107.302907</b>	1/20	550.42
I_5000_5	105.195058	<b>105.191547</b>	16/20	494.22	105.195058	105.188447	3/20	617.42
I_5000_6	103.651929	103.635670	11/20	397.00	103.651929	<b>103.637288</b>	2/20	572.61
I_5000_7	105.452981	105.427086	12/20	258.28	105.452981	<b>105.452647</b>	17/20	614.86
I_5000_8	104.686123	<b>104.686123</b>	20/20	212.67	104.686123	104.682843	4/20	610.63
I_5000_9	102.894130	<b>102.891559</b>	19/20	336.93	102.894130	102.869843	8/20	517.80
I_5000_10	108.205395	<b>108.205395</b>	20/20	123.79	108.205395	108.205205	19/20	269.50
II_5000_1	<b>130.041711</b>	<b>129.903022</b>	15/20	30.82	129.988730	129.890200	1/20	574.84
II_5000_2	127.790529	<b>127.790529</b>	20/20	195.76	127.790529	127.785800	6/20	418.03
II_5000_3	129.223564	<b>129.223564</b>	20/20	88.77	129.223564	129.220797	18/20	412.31
II_5000_4	132.381785	132.381785	20/20	46.93	132.381785	132.381785	20/20	121.94
II_5000_5	<b>131.291478</b>	<b>131.273801</b>	11/20	445.48	131.262016	131.262016	20/20	201.13
II_5000_6	128.199403	<b>128.199403</b>	20/20	56.60	128.199403	128.198547	16/20	421.87
II_5000_7	128.901011	<b>128.901011</b>	20/20	241.46	128.901011	128.869417	3/20	450.94
II_5000_8	129.742428	<b>129.742428</b>	20/20	245.07	129.742428	129.741596	18/20	511.32
II_5000_9	127.593892	<b>127.585685</b>	18/20	388.89	127.593892	127.543106	4/20	505.00
II_5000_10	134.691155	134.691155	20/20	22.95	134.691155	134.691155	20/20	234.79
III_5000_1	35.820098	35.809506	6/20	279.75	35.820098	<b>35.813498</b>	4/20	554.74
III_5000_2	36.231529	36.214299	6/20	271.56	36.231529	<b>36.216595</b>	3/20	718.32
III_5000_3	<b>36.036199</b>	36.030249	2/20	165.20	36.034200	<b>36.032858</b>	5/20	605.56
III_5000_4	36.480238	36.462380	12/20	391.82	36.480238	<b>36.477088</b>	16/20	685.53
III_5000_5	36.150412	36.141578	3/20	436.69	36.150412	<b>36.145352</b>	4/20	549.78
III_5000_6	36.031067	36.025122	12/20	367.06	36.031067	<b>36.029319</b>	18/20	531.48
III_5000_7	35.945148	35.932945	6/20	323.86	<b>35.945224</b>	<b>35.941077</b>	2/20	651.21
III_5000_8	35.977378	35.958775	1/20	1397.13	35.977378	<b>35.964061</b>	3/20	646.35
III_5000_9	36.174472	<b>36.147119</b>	5/20	407.26	36.174472	36.146802	1/20	616.11
III_5000_10	36.450138	<b>36.449973</b>	18/20	174.13	36.450138	36.449407	13/20	457.63
IV_5000_1	357.412342	357.299214	8/20	495.33	357.412342	<b>357.355749</b>	11/20	667.19
IV_5000_2	363.733876	363.653599	7/20	241.50	363.733876	<b>363.703214</b>	5/20	641.44
IV_5000_3	361.401490	361.233101	10/20	141.84	361.401490	<b>361.316492</b>	7/20	657.13
IV_5000_4	365.320648	365.221758	7/20	379.34	365.320648	<b>365.271635</b>	12/20	607.23
IV_5000_5	361.628709	361.619700	15/20	251.79	361.628709	<b>361.627548</b>	11/20	699.02
IV_5000_6	<b>358.013986</b>	<b>357.931943</b>	5/20	370.20	357.976519	357.924349	6/20	740.73
IV_5000_7	353.071271	352.935036	2/20	956.15	353.071271	<b>352.952883</b>	2/20	690.22
IV_5000_8	359.201624	359.159182	17/20	377.93	359.201624	<b>359.177581</b>	14/20	521.58
IV_5000_9	361.105769	361.016744	5/20	475.21	<b>361.121622</b>	<b>361.088689</b>	6/20	621.83
IV_5000_10	361.123900	361.085726	6/20	179.52	361.123900	<b>361.099469</b>	5/20	718.69
Avg.	157.856608	157.825271		331.84	157.853553	157.831891		545.03
#Best	38				34			
p-value	9.289e-2	2.204e-1						

the average of the  $f_{best}$  values and 157.825271 for MAMMDP\* vs 157.831891 for PBEA in terms of the average of the  $f_{avg}$  values ( $p$ -values  $> 0.05$ ). Meanwhile, we observe that the success rates of both algorithms are below 50% for more than 15 instances, which shows that these instances are much harder than the instances with  $n = 3000$ . Interestingly, we observe that MAMMDP\* performs better than PBEA for the instances of Types I and II, while the reverse is true for the Type III and IV instances. This indicates that these two algorithms are complementary for solving these hard instances.

#### 4. Analysis and Discussion

In this section, we perform additional experiments to analyze the influence of two key ingredients of the PBEA algorithm (i.e., the perturbation strength and the neighborhood structure of the tabu search procedure), while for MAMMDP\*, an analysis of its underlying MAMMDP algorithm can be found in (Lai and Hao, 2016).

##### 4.1. Sensitivity Analysis of an Important Parameter of the PBEA algorithm

Table 6: Influence of the parameter  $\eta$  on the performance of the PBEA algorithm.

Instance/ $\eta$	$f_{avg}$									
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
I_5000_1	104.80	104.81	104.80	104.80	104.80	104.79	104.81	104.78	104.80	104.80
I_5000_2	104.05	104.05	104.05	104.05	104.05	104.05	104.05	104.05	104.05	104.05
I_5000_3	104.80	104.80	104.80	104.79	104.80	104.80	104.80	104.79	104.80	104.80
I_5000_4	107.31	107.30	107.32	107.31	107.31	107.30	107.31	107.30	107.32	107.31
I_5000_5	105.19	105.19	105.19	105.19	105.19	105.20	105.20	105.19	105.20	105.19
I_5000_6	103.64	103.64	103.64	103.64	103.64	103.64	103.65	103.64	103.64	103.64
I_5000_7	105.45	105.45	105.44	105.45	105.44	105.45	105.44	105.45	105.45	105.45
I_5000_8	104.69	104.69	104.69	104.69	104.69	104.69	104.69	104.68	104.69	104.69
I_5000_9	102.89	102.89	102.89	102.89	102.89	102.89	102.89	102.87	102.89	102.89
I_5000_10	108.21	108.21	108.21	108.21	108.21	108.21	108.21	108.21	108.21	108.21
II_5000_1	129.94	129.96	129.95	129.92	129.93	129.96	129.91	129.89	129.93	129.92
II_5000_2	127.79	127.79	127.79	127.79	127.79	127.79	127.79	127.79	127.79	127.79
II_5000_3	129.22	129.22	129.22	129.22	129.22	129.22	129.22	129.22	129.22	129.22
II_5000_4	132.38	132.38	132.38	132.38	132.38	132.38	132.38	132.38	132.38	132.38
II_5000_5	131.28	131.28	131.27	131.27	131.27	131.28	131.27	131.26	131.27	131.26
II_5000_6	128.20	128.20	128.20	128.20	128.20	128.20	128.20	128.20	128.20	128.20
II_5000_7	128.90	128.90	128.90	128.90	128.90	128.90	128.90	128.87	128.90	128.90
II_5000_8	129.74	129.74	129.74	129.74	129.74	129.74	129.74	129.74	129.74	129.74
II_5000_9	127.59	127.57	127.57	127.57	127.58	127.58	127.58	127.54	127.57	127.59
II_5000_10	134.69	134.69	134.69	134.69	134.69	134.69	134.69	134.69	134.69	134.69
III_5000_1	35.81	35.81	35.81	35.81	35.81	35.81	35.81	35.81	35.81	35.81
III_5000_2	36.21	36.21	36.21	36.21	36.20	36.21	36.21	36.22	36.21	36.21
III_5000_3	36.03	36.03	36.03	36.03	36.03	36.03	36.03	36.03	36.03	36.03
III_5000_4	36.46	36.46	36.45	36.45	36.45	36.46	36.46	36.48	36.46	36.47
III_5000_5	36.14	36.14	36.14	36.14	36.14	36.14	36.14	36.15	36.14	36.14
III_5000_6	36.02	36.02	36.02	36.02	36.02	36.03	36.02	36.03	36.02	36.02
III_5000_7	35.94	35.94	35.94	35.94	35.94	35.94	35.94	35.94	35.94	35.94
III_5000_8	35.96	35.96	35.96	35.96	35.96	35.96	35.96	35.96	35.96	35.96
III_5000_9	36.14	36.15	36.14	36.14	36.13	36.13	36.15	36.15	36.14	36.14
III_5000_10	36.45	36.45	36.45	36.45	36.45	36.45	36.45	36.45	36.45	36.45
IV_5000_1	357.27	357.33	357.29	357.28	357.28	357.26	357.29	357.36	357.28	357.23
IV_5000_2	363.67	363.65	363.66	363.66	363.62	363.66	363.65	363.70	363.63	363.65
IV_5000_3	361.21	361.21	361.20	361.21	361.23	361.20	361.21	361.32	361.19	361.27
IV_5000_4	365.20	365.26	365.18	365.18	365.17	365.25	365.24	365.27	365.19	365.18
IV_5000_5	361.61	361.62	361.62	361.62	361.62	361.62	361.62	361.63	361.62	361.61
IV_5000_6	357.88	357.88	357.87	357.87	357.87	357.91	357.93	357.92	357.87	357.91
IV_5000_7	352.86	352.91	352.91	352.86	352.77	352.92	352.95	352.95	352.91	352.92
IV_5000_8	359.15	359.14	359.15	359.14	359.15	359.20	359.16	359.18	359.14	359.15
IV_5000_9	360.99	360.99	361.01	361.00	361.02	361.02	361.00	361.09	361.04	361.03
IV_5000_10	361.05	361.08	361.05	361.06	361.04	361.06	361.05	361.10	361.04	361.04
Avg	157.82	<b>157.83</b>	157.82	157.82	157.82	<b>157.83</b>	157.82	<b>157.83</b>	157.82	157.82

The perturbation operator is an essential ingredient of the PBEA algorithm. To understand the influence of its perturbation strength (i.e.,  $\eta \times n$ ) on the

performance of the algorithm, we carried out an experiment based on the 40 large GMaxMeanDP instances with  $n = 5000$ , where the algorithm was run 20 times with each value of  $\eta \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$  on each instance. Recall that given a solution composed of  $n$  components the perturbation operator assigns randomly a value from  $\{0, 1\}$  to  $\eta \times n$  variables, thus there are about  $0.5 \times \eta \times n$  variables whose values are changed by the perturbation operator. The results are reported in Table 6, where column 1 and row 2 give respectively the name of instances and the setting of parameter  $\eta$ , columns 2–11 report the average objective values ( $f_{avg}$ ) over 20 runs for each  $\eta$  value, and the row 'Avg' indicates the average results for each column.

Table 6 shows that the different settings of  $\eta$  yielded very similar results in terms of  $f_{avg}$  for each instance tested, which means that the performance of PBEA algorithm is not sensitive to the setting of  $\eta$  due to the strong local search ability of its underlying tabu search procedure as well as the features of the GMaxMeanDP problem. Moreover, we observe that the settings  $\eta = 0.1, 0.3$ , and  $0.4$  lead to slightly better results in terms of  $Avg$  than other settings. Hence, the default value of  $\eta$  is set to  $0.4$  for the PBEA algorithm.

#### 4.2. Influence of the Neighborhoods on the Performance of Tabu Search

As described in Algorithm 4, at each iteration of the tabu search algorithms, a best eligible neighbor solution is selected to replace the current solution by examining the whole neighborhood. As such, for each iteration, a larger neighborhood usually offers a greater chance to encounter a neighbor solution of high quality, but requires a larger computational effort. Hence, we face the challenge of identifying an appropriate neighborhood structure to enable the resulting algorithm to reach a good tradeoff between solution quality and computing speed.

To check the influence of the neighborhoods on the tabu search algorithm and select a proper neighborhood for our tabu search algorithm, we carried out an experiment based on the 40 instances with  $n = 3000$ . Using the neighborhoods  $N_1, N_2, N_3 (= N_1 \cup N_2), N_4 (= N_1 \cup N_2^*)$  described in Section 2.1.6 as the neighborhood structure and setting the parameter  $T_{max}$  to 100, we obtain four tabu search algorithms. Given the stochastic nature of these algorithms, we solved each instance 20 times by each of these algorithms, and recorded the average computing times and average objective values. The stopping condition was given by the maximum number  $Iter_{max}$  of iterations, which was set to  $5 \times 10^4$  in this experiment. The results of this experiment are summarized in Table 7. The first column of the table gives the names of instances. Columns 2–5 report the average objective values over 20 runs for the four tabu search algorithms, and columns 6–9 report the average computing times consumed for each algorithm. The rows '#Better', '#Equal' and '#Worse' show the number of instances for which the associated neighborhood obtains a better, equal, or worse result compared to the neighborhood  $N_1$ .

Table 7 shows that the four tabu search algorithms obtained similar results in terms of the average objective value, implying that the four neighborhoods have a similar search ability when the same number of iterations is used. Nevertheless, compared to the neighborhood  $N_1$ , the three other neighborhoods  $N_2, N_3$  and

Table 7: Comparative results of tabu search procedures with different neighborhoods under the same maximum number of iterations. Each instance was solved 20 times by each tabu search variant, and the average objective values and the run times are recorded.

Instance	Average objective value				Average computing time (s)			
	$N_1$	$N_4$	$N_2$	$N_3$	$N_1$	$N_4$	$N_2$	$N_3$
I_3000_1	80.370223	80.331709	80.419228	80.397729	1.70	2.48	1196.98	1211.05
I_3000_2	84.160923	84.175919	84.132061	84.156005	1.70	2.62	1234.33	1181.27
I_3000_3	81.543666	81.563254	81.493745	81.567133	1.71	2.36	1178.35	1281.49
I_3000_4	80.009048	80.044790	80.074550	80.099239	1.70	2.40	1219.60	1294.26
I_3000_5	81.105796	81.086351	81.028486	81.129050	1.70	2.26	1289.05	1177.42
I_3000_6	83.122833	83.139438	83.053114	83.136528	1.77	2.25	1178.04	1190.21
I_3000_7	81.68306	81.727730	81.683873	81.724791	1.69	2.25	1167.27	1173.33
I_3000_8	80.525859	80.526347	80.477204	80.528973	1.70	2.37	1253.14	1192.05
I_3000_9	80.556881	80.560794	80.502281	80.556804	1.78	2.51	1179.42	1193.18
I_3000_10	83.326823	83.320350	83.315442	83.320493	1.70	2.37	1200.32	1194.71
II_3000_1	98.980639	99.000333	98.969911	98.980702	1.74	2.30	1177.97	1193.19
II_3000_2	105.448977	105.488372	105.387800	105.468175	1.74	2.36	1280.10	1159.60
II_3000_3	101.099149	101.134966	101.043572	101.198633	1.70	2.59	1172.63	1163.89
II_3000_4	101.074346	101.078024	101.064486	101.075638	1.77	2.41	1202.44	1156.11
II_3000_5	99.876333	99.893006	99.789128	99.869741	1.71	2.42	1256.77	1243.89
II_3000_6	101.970764	101.968581	101.869966	101.961902	1.71	2.30	1163.45	1155.65
II_3000_7	100.132899	100.123789	100.103272	100.158365	1.70	2.33	1171.30	1194.30
II_3000_8	101.105064	101.027986	100.970914	101.154872	1.69	2.26	1167.71	1241.09
II_3000_9	98.601737	98.596904	98.508047	98.598148	1.70	2.27	1273.98	1172.29
II_3000_10	104.862917	104.887103	104.795615	104.874885	1.68	2.30	1183.17	1164.99
III_3000_1	27.754071	27.774755	27.818733	27.781418	1.69	2.53	1298.58	1355.77
III_3000_2	27.702657	27.719219	27.757757	27.716593	1.73	2.48	1397.21	1349.85
III_3000_3	27.881157	27.885209	27.920852	27.897084	1.69	2.36	1317.78	1367.02
III_3000_4	27.723692	27.732678	27.766265	27.744262	1.72	2.27	1293.15	1320.68
III_3000_5	27.66126	27.643878	27.699754	27.671768	1.69	2.37	1416.65	1370.78
III_3000_6	27.612341	27.627724	27.638232	27.631653	1.68	2.34	1431.19	1302.33
III_3000_7	27.580934	27.582994	27.605799	27.574759	1.74	2.34	1303.56	1354.12
III_3000_8	27.671951	27.680622	27.712513	27.693571	1.69	2.55	1297.23	1365.41
III_3000_9	27.650248	27.651881	27.685241	27.665465	1.70	2.52	1419.31	1372.90
III_3000_10	27.462855	27.471166	27.503728	27.480884	1.78	2.61	1306.16	1311.41
IV_3000_1	277.297445	277.456078	277.707822	277.411484	1.70	2.32	1309.75	1369.06
IV_3000_2	275.569955	275.505513	275.964659	275.865817	1.70	2.33	1303.08	1368.02
IV_3000_3	276.736029	277.062846	277.089373	276.904170	1.70	2.24	1319.46	1379.99
IV_3000_4	278.244348	278.299395	278.576349	278.460311	1.75	2.37	1391.38	1382.73
IV_3000_5	275.912872	275.856499	276.179065	276.011616	1.69	2.36	1319.15	1386.32
IV_3000_6	279.908391	280.232102	280.348491	280.295156	1.68	2.31	1437.05	1357.70
IV_3000_7	272.736927	272.878628	273.156411	272.960018	1.73	2.48	1312.67	1373.88
IV_3000_8	275.456754	275.566789	275.983281	275.651022	1.70	2.56	1427.62	1403.40
IV_3000_9	273.611056	273.878003	274.185110	273.944759	1.75	2.43	1306.62	1492.63
IV_3000_10	275.58143	275.532812	275.907729	275.621588	1.69	2.31	1315.50	1356.68
#Better	28	23	33		0	0	0	
#Equal	0	0	0		0	0	0	
#Worse	12	17	7		40	40	40	

Table 8: Comparative results of tabu search procedures with different neighborhoods under the same time limit. Each instance was solved 20 times by each tabu search variant, and the average objective values are recorded. The best  $f_{best}$  values among the compared results are indicated in boldface.

Instance	$t_{max}(s)$	$f_{avg}$			
		$N_1$	$N_4$	$N_2$	$N_3$
I_3000.1	2.5	80.249099	<b>80.311231</b>	11.721149	11.721149
I_3000.2	2.5	84.162707	<b>84.162944</b>	12.370702	12.370702
I_3000.3	2.5	<b>81.543361</b>	81.536433	14.916785	14.916785
I_3000.4	2.5	<b>80.106844</b>	80.014370	12.227146	12.227146
I_3000.5	2.5	<b>81.116207</b>	81.090353	15.303211	15.303211
I_3000.6	2.5	83.081651	<b>83.098734</b>	11.820082	11.820082
I_3000.7	2.5	81.698899	<b>81.715287</b>	13.997985	13.997985
I_3000.8	2.5	80.509067	<b>80.523652</b>	14.891707	14.891707
I_3000.9	2.5	80.557968	<b>80.564241</b>	14.067301	14.067301
I_3000.10	2.5	83.325346	<b>83.339849</b>	10.676445	10.676445
II_3000.1	2.5	98.977606	<b>98.997774</b>	12.915757	12.915757
II_3000.2	2.5	105.442076	<b>105.446369</b>	12.575737	12.575737
II_3000.3	2.5	<b>101.196340</b>	101.099364	12.540325	12.540325
II_3000.4	2.5	<b>101.078750</b>	101.073721	17.414740	17.414740
II_3000.5	2.5	<b>99.898393</b>	99.865010	14.157223	14.157223
II_3000.6	2.5	<b>101.974963</b>	101.955398	13.611427	13.611427
II_3000.7	2.5	<b>100.132530</b>	100.131676	12.576563	12.576563
II_3000.8	2.5	<b>101.081766</b>	100.999474	12.955339	12.955339
II_3000.9	2.5	<b>98.566209</b>	98.565474	12.533706	12.533706
II_3000.10	2.5	104.882068	<b>104.883430</b>	11.974704	11.974704
III_3000.1	2.5	<b>27.760076</b>	27.747686	6.015289	6.015289
III_3000.2	2.5	27.690291	<b>27.713241</b>	4.333629	4.333629
III_3000.3	2.5	27.882708	<b>27.892154</b>	4.663547	4.663547
III_3000.4	2.5	27.717461	<b>27.721817</b>	4.703571	4.703571
III_3000.5	2.5	27.650975	<b>27.660587</b>	4.752437	4.752437
III_3000.6	2.5	<b>27.626892</b>	27.619878	4.267233	4.267233
III_3000.7	2.5	<b>27.572748</b>	27.558271	4.498055	4.498055
III_3000.8	2.5	<b>27.682396</b>	27.662442	4.441804	4.441804
III_3000.9	2.5	27.641488	<b>27.658272</b>	4.584740	4.584740
III_3000.10	2.5	27.439402	<b>27.464877</b>	4.591576	4.591576
IV_3000.1	2.5	<b>277.438126</b>	277.103842	47.790044	47.790044
IV_3000.2	2.5	275.520943	<b>275.601493</b>	44.924906	44.924906
IV_3000.3	2.5	<b>276.956653</b>	276.880232	48.493625	48.493625
IV_3000.4	2.5	<b>278.319177</b>	278.030400	46.031313	46.031313
IV_3000.5	2.5	275.669572	<b>275.722372</b>	44.559275	44.559275
IV_3000.6	2.5	280.009358	<b>280.102139</b>	50.117318	50.117318
IV_3000.7	2.5	<b>272.933195</b>	272.763686	44.694693	44.694693
IV_3000.8	2.5	275.364275	<b>275.436477</b>	58.228575	58.228575
IV_3000.9	2.5	273.757565	<b>273.852615</b>	45.895074	45.895074
IV_3000.10	2.5	<b>275.490623</b>	275.465413	61.333437	61.333437
#Best		19	21	0	0

Table 9: Comparison between the PBEA algorithms with the neighborhoods  $N_1$  and  $N_4$  on the set of 40 GMaxMeanDP instances with  $n = 3000$ . The better results between two algorithms are indicated in bold both in terms of  $f_{best}$  and  $f_{avg}$

Instance	PBEA*( $N_1$ )				PBEA( $N_4$ )			
	$f_{best}$	$f_{avg}$	SR	$t(s)$	$f_{best}$	$f_{avg}$	SR	$t(s)$
L_3000_1	80.743467	80.743467	20/20	107.21	80.743467	80.743467	20/20	92.53
L_3000_2	84.201027	84.201027	20/20	35.87	84.201027	84.201027	20/20	46.71
L_3000_3	81.630082	81.630082	20/20	15.60	81.630082	81.630082	20/20	9.34
L_3000_4	80.234334	80.234334	20/20	43.26	80.234334	80.234334	20/20	28.07
L_3000_5	81.218062	81.218004	17/20	182.29	81.218062	<b>81.218062</b>	20/20	138.28
L_3000_6	83.197618	83.197618	20/20	78.91	83.197618	83.197618	20/20	64.05
L_3000_7	81.732080	81.732080	20/20	5.31	81.732080	81.732080	20/20	4.50
L_3000_8	80.624273	80.623660	19/20	130.81	80.624273	<b>80.624273</b>	20/20	83.53
L_3000_9	80.574438	80.574438	20/20	10.83	80.574438	80.574438	20/20	10.76
L_3000_10	83.397670	83.397670	20/20	132.95	83.397670	83.397670	20/20	138.84
IL_3000_1	99.055143	99.055143	20/20	19.81	99.055143	99.055143	20/20	12.66
IL_3000_2	105.574146	105.574146	20/20	73.35	105.574146	105.574146	20/20	63.27
IL_3000_3	101.299271	101.299271	20/20	5.97	101.299271	101.299271	20/20	6.71
IL_3000_4	101.079824	101.079824	20/20	9.76	101.079824	101.079824	20/20	8.03
IL_3000_5	100.029225	100.028216	15/20	257.77	100.029225	<b>100.028322</b>	18/20	241.64
IL_3000_6	101.978783	101.978783	20/20	6.25	101.978783	101.978783	20/20	4.56
IL_3000_7	100.189718	100.189718	20/20	20.30	100.189718	100.189718	20/20	17.36
IL_3000_8	101.160428	101.160428	20/20	5.97	101.160428	101.160428	20/20	4.52
IL_3000_9	98.665034	98.665034	20/20	45.85	98.665034	98.665034	20/20	59.96
IL_3000_10	104.896612	104.896612	20/20	9.17	104.896612	104.896612	20/20	11.86
III_3000_1	27.847334	27.846822	19/20	92.23	27.847334	<b>27.847334</b>	20/20	108.70
III_3000_2	27.776796	<b>27.774430</b>	5/20	99.81	27.776796	27.774272	4/20	120.08
III_3000_3	27.946519	27.946519	20/20	146.48	27.946519	27.946519	20/20	157.85
III_3000_4	27.816272	27.816272	20/20	82.58	27.816272	27.816272	20/20	70.66
III_3000_5	27.727167	27.726868	18/20	150.04	27.727167	<b>27.727167</b>	20/20	160.51
III_3000_6	27.691682	27.683984	4/20	144.90	27.691682	<b>27.686631</b>	4/20	131.25
III_3000_7	27.642060	27.642060	20/20	137.72	27.642060	27.642060	20/20	158.84
III_3000_8	27.736643	27.733845	5/20	151.96	27.736643	<b>27.734079</b>	6/20	184.58
III_3000_9	27.745820	27.742271	17/20	88.43	27.745820	<b>27.745820</b>	20/20	77.88
III_3000_10	27.561083	27.561083	20/20	101.94	27.561083	27.561083	20/20	92.74
IV_3000_1	278.039443	278.023159	13/20	159.32	278.039443	<b>278.027811</b>	15/20	151.80
IV_3000_2	276.539877	<b>276.539784</b>	19/20	228.20	276.539877	276.539691	18/20	238.37
IV_3000_3	277.334878	277.334878	20/20	61.11	277.334878	277.334878	20/20	40.40
IV_3000_4	278.956422	278.956422	20/20	70.76	278.956422	278.956422	20/20	61.36
IV_3000_5	276.595238	276.592466	19/20	123.03	276.595238	<b>276.595238</b>	20/20	108.00
IV_3000_6	280.721533	280.721533	20/20	66.67	280.721533	280.721533	20/20	60.47
IV_3000_7	273.653396	273.653396	20/20	143.06	273.653396	273.653396	20/20	169.85
IV_3000_8	276.358447	276.358447	20/20	102.80	276.358447	276.358447	20/20	81.56
IV_3000_9	274.864865	274.807248	15/20	201.57	274.864865	<b>274.838571</b>	18/20	241.92
IV_3000_10	276.428571	276.381189	11/20	164.19	276.428571	<b>276.407810</b>	16/20	151.95
Avg	121.961632	121.958056		92.85	121.961632	<b>121.959884</b>		90.40
#Better	0	2			0	12		
#Equal	40	26			40	26		
#Worse	0	12			0	2		
p-value					1.0	3.51e-3		



$N_4$  yielded a slightly better result for 28, 23, and 33 instances, respectively. In addition, the multi-neighborhood tabu search methods (with  $N_3$  or  $N_4$ ) yielded a better result than those using a single basic neighborhood (i.e.,  $N_1$  or  $N_2$ ) in terms of  $\#Better$ , and hence the combined use of multiple complementary neighborhoods enhanced the search ability of our methods in the case that the tabu search procedures employ the same  $Iter_{max}$  as the stopping condition. On the other hand, Table 7 indicates a significant difference among the four neighborhoods in terms of the computing time. First, the times required to examine the neighborhoods  $N_1$  and  $N_4$  are much smaller than those required to examine other two neighborhoods, since  $N_1$  and  $N_4$  are much smaller than  $N_2$  and  $N_3$  and the move values (i.e., the change of objective value) of a flip or swap move can be calculated in  $O(1)$  (see Section 2.1.7). In addition, we observe that the examination of the neighborhoods  $N_2$  and  $N_3$  is very time-consuming due to their large sizes. Finally, the speed of examining the neighborhood  $N_4$  is slightly slower to that of examining  $N_1$  but is much faster than that of examining  $N_2$  and  $N_3$ .

To assess and compare the effectiveness of the above four neighborhoods based on the same time limit, we carried out another experiment on the 40 instances mentioned above, where each instance was solved 20 times by each tabu search algorithm, and the stopping criterion was a time limit  $t_{max} = 2.5$  seconds. The experimental results are reported in Table 8, where the first two columns give the names of instances and the time limit used ( $t_{max}$ ), columns 3–6 report the average objective value ( $f_{avg}$ ) over 20 runs for the four algorithms, respectively, and the row ' $\#Best$ ' shows the number of instances for which the associated algorithm yields the best result in  $f_{avg}$ .

Table 8 shows that the algorithms with the neighborhood  $N_1$  or  $N_4$  performs much better than those with the neighborhood  $N_2$  or  $N_3$ . When comparing  $N_1$  and  $N_4$ , we observe that the two corresponding tabu search algorithms obtain similar results in ' $\#Best$ ', i.e., with the best results in  $f_{avg}$  for 19 and 21 instances, respectively. This finding further shows the merit of small neighborhoods for the tabu search algorithms.

To further compare the effectiveness of the neighborhoods  $N_1$  and  $N_4$  within the proposed PBEA algorithm, we first created a variant of PBEA (called PBEA\*) by replacing the neighborhood  $N_4$  with the neighborhood  $N_1$  and keeping other ingredients unchanged. Then, we carried out an experiment with PBEA and PBEA\* on the 40 GMaxMeanDP instances with  $n = 3000$ , where both algorithms were performed 20 times on each instance according to the experimental protocol in Section 3.2. The results are summarized in Table 9, where the rows ' $\#Better$ ', ' $\#Equal$ ' and ' $\#Worse$ ' show the number of instances for which the associated algorithm obtains a better, equal, or worse result compared to the other algorithm.

Table 9 shows that the PBEA and PBEA\* algorithms have a similar performance both in  $f_{best}$  and the success rate. Specifically, both algorithms reached the best known result for all instances tested. However, regarding the average objective value ( $f_{avg}$ ) over 20 runs, PBEA slightly outperformed PBEA\*. For 12 and 2 out of 40 instances, PBEA obtained a better and worse result in terms

549 of  $f_{avg}$  compared to PBEA\*, respectively, while matching the results of PBEA\*  
550 for the remaining instances. This outcome indicates that the neighborhood  $N_4$   
551 is superior to the neighborhood  $N_1$  on the tested instances. On this basis we  
552 have selected the neighborhood  $N_4$  as the neighborhood structure of the tabu  
553 search procedure for the proposed PBEA algorithm.

## 554 5. Conclusions

555 The generalized max-mean dispersion problem (GMaxMeanDP) is a gener-  
556 alization of the popular NP-hard max-mean dispersion problem (MaxMeanDP).  
557 Contrary to the MaxMeanDP which has been studied intensively in the past,  
558 the GMaxMeanDP has received little research effort until now and no practical  
559 solution method has been ever proposed for it. To fill the gap in the literature  
560 produced by the absence of a solution method for this important problem, we in-  
561 vestigate for the first time two population-based heuristic algorithms for solving  
562 the GMaxMeanDP. The dedicated perturbation based evolutionary algorithm  
563 (PBEA) combines a tabu search procedure for solution improvement, a simple  
564 perturbation operator to diversify the search process and a population to record  
565 the elite solutions found during the search. The other algorithm (MAMMDP\*)  
566 is a simple adaptation of the state-of-the-art memetic algorithm called MAM-  
567 MDP for the MaxMeanDP, which uses a crossover operator to generate new  
568 starting solutions for its tabu search improvement procedure.

569 We performed extensive experiments of our two algorithms on six types of  
570 160 instances with  $n \in \{1500, 2000, 3000, 5000\}$ , leading to the following obser-  
571 vations. First, an effective algorithm such as MAMMDP for the MaxMeanDP  
572 can be easily converted to an effective algorithm for the GMaxMeanDP. Second,  
573 for the GMaxMeanDP, the simple perturbation operator used in PBEA plays a  
574 similar role with respect to the crossover operator used in MAMMDP\*. Third,  
575 the two proposed algorithms are complementary since there are instances that  
576 are better solved either by MAMMDP\* or by PBEA. Fourth, these algorithms  
577 designed for the GMaxMeanDP also perform very well on the special MaxMe-  
578 anDP. Fifth, for the tabu search method designed for the GMaxMeanDP, a  
579 small and cost-effective neighborhood proves to be highly efficient.

580 Since the GMaxMeanDP can formulate various real-world applications (e.g.,  
581 web page ranking (Kerchov and Dooren, 2008), community mining in a signed  
582 social network (Yang et al., 2007) and trust networks (Carrasco et al., 2015)),  
583 the proposed algorithms can be used to handle such practical problems as well.  
584 The availability of the source codes of our algorithms will certainly facilitate  
585 such applications. More generally, the approach of using an effective tabu  
586 search procedure combined with the evolutionary computing framework can  
587 be applied to solve other dispersion problems such as the max-mean disper-  
588 sion problem that has recently received widespread attention. Our design for  
589 PBEA and MAMMDP\* can be adapted to other binary optimization prob-  
590 lems like max-cut/max-bisection (Benlic and Hao, 2013; Ma et al., 2017; Wu  
591 et al., 2015). Finally, combining the present algorithms with other approaches  
592 like path relinking (Glover, 1998) and learning strategies like opposition-based

learning (Mahdavi et al., 2018), and diversification-based learning (Glover and Hao, 2018) provides other interesting possibilities for future research.

## Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions which helped us to improve the paper. This work was partially supported by the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20170904), the National Natural Science Foundation of China (Grant No. 61703213), six talent peaks project in Jiangsu Province (Grant No. RJFW-011), and NUPTSF (Grant Nos. NY217154 and RK043YZZ18004).

## References

## References

- Amirgaliyeva, Z., Mladenović, N., Todosijević, R. & Urošević, D. (2017). Solving the maximum min-sum dispersion by alternating formulations of two different problems. *European Journal of Operational Research* 260, 444–459.
- Aringhieri, R. & Cordone, R. (2011). Comparing local search metaheuristics for the maximum diversity problem. *Journal of the Operational Research Society* 62, 266–280.
- Aringhieri, R., Cordone, R. & Grosso A. (2015). Construction and improvement algorithms for dispersion problems. *European Journal of Operational Research* 242(1), 21–33.
- Benlic, U. & Hao, J.K. (2013). Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence* 26(3), 1162–1173.
- Benlic, U. & Hao, J.K. (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications* 42, 584–595.
- Brimberg, J., Mladenović, N., Todosijević, R. & Urošević D. (2017). Less is more: Solving the max-mean diversity problem with variable neighborhood search. *Information Sciences* 382, 179–200.
- Carrasco, R., Anthanh, P.T., Gallego, M., Gortázar, F., Duarte, A. & Martí, R. (2015). Tabu search for the max-mean dispersion problem. *Knowledge Based System* 85, 256–264.
- Della Croce, F., Grosso, A. & Locatelli, M. (2009). A heuristic approach for the max-min diversity problem based on max-clique. *Computers & Operations Research* 36(8), 2429–2433.
- Della Croce, F., Garraffa, M. & Salassa, F. (2016). A hybrid three-phase approach for the max-mean dispersion problem. *Computers & Operations Research* 71, 16–22.

- 629 Galinier, P., Boujbel, Z., Fernandes & M.C., 2011, An efficient memetic al-  
630 gorithm for the graph partitioning problem. *Annals of Operations Research*  
631 191(1), 1–22.
- 632 Garraffa, M., Della Croce, F. & Salassa, F. (2017). An exact semidefinite pro-  
633 gramming approach for the max-mean dispersion problem. *Journal of Com-*  
634 *binatorial Optimization* 34, 1–23.
- 635 Ghosh, M., Begum, S., Sarkar, R., Chakraborty, D. & Maulik, U. (2019). Recur-  
636 sive Memetic Algorithm for gene selection in microarray data. *Expert Systems*  
637 *with Applications* 116, 172–185.
- 638 Glover, F. (1998). A template for scatter search and path relinking. *Lecture*  
639 *Notes in Computer Science* 1363, 13–54.
- 640 Glover, F. & Hao, J.K. (2018). Diversification-based learning in computing and  
641 optimization. *Journal of Heuristics* DOI: [https://doi.org/10.1007/s10732-](https://doi.org/10.1007/s10732-018-9384-y)  
642 [018-9384-y](https://doi.org/10.1007/s10732-018-9384-y).
- 643 Glover, F. & Laguna, M. (1997). Tabu search. *Kluwer Academic Publishers*  
644 Boston.
- 645 Glover, F., Kuo, C.C. & Dhir, K.S. (1998). Heuristic algorithms for the max-  
646 imum diversity problem. *Journal of Information and Optimization Sciences*  
647 19(1), 109–132.
- 648 Ismkhan, H. (2017). Effective three-phase evolutionary algorithm to handle the  
649 large-scale colorful traveling salesman problem. *Expert Systems with Applica-*  
650 *tions* 67, 148–162.
- 651 Kerchove, C. & Dooren, P.V. (2008). The page trust algorithm: how to rank  
652 web pages when negative links are allowed? *Proceedings SIAM International*  
653 *Conference on Data Mining*, pp. 346–352.
- 654 Lai, X.J. & Hao, J.K. (2016) A tabu search based memetic search algorithm  
655 for the max-mean dispersion problem. *Computers & Operations Research* 72,  
656 118–127.
- 657 Lai, X.J., Yue, D., Hao, J.K. & Glover, F. (2018). Solution-based tabu search for  
658 the maximum min-sum dispersion problem. *Information Sciences* 441, 79–94.
- 659 Lai, X.J., Hao, J.K. Glover, F. & Yue, D. (2019). Intensification-driven tabu  
660 search for the minimum differential dispersion problem. *Knowledge-Based Sys-*  
661 *tems* 167, 68–86.
- 662 Ma, F., Hao, J.K. & Wang, Y. (2017). An effective iterated tabu search for the  
663 maximum bisection problem. *Computers & Operations Research* 81, 78–89.
- 664 Mahdavi, S., Rahnamayan, S. & Deb, K. (2018). Opposition based learning: A  
665 literature review. *Swarm and Evolutionary Computation* 39, 1–23.

- 666 Martí, R., & Sandoya, F. (2013). GRASP and path relinking for the equitable  
667 dispersion problem. *Computers & Operations Research* 40(12), 3091–3099.
- 668 Mladenović, N., Todosijević, R. & Urošević, D. (2016). Less is more: Basic  
669 variable neighborhood search for minimum differential dispersion problem.  
670 *Information Sciences* 326, 160–171.
- 671 Morra, L., Coccia, N. & Cerquitelli T. (2018). Optimization of computer aided  
672 detection systems: An evolutionary approach. *Expert Systems with Applica-*  
673 *tions* 100, 145–156.
- 674 Palubeckis, G. (2007). Iterated tabu search for the maximum diversity problem.  
675 *Applied Mathematics and Computation* 189(1), 371–383.
- 676 Porumbel, D.C., Hao, J.K. & Glover, F. (2011). A simple and effective algorithm  
677 for the MaxMin diversity problem. *Annals of Operations Research* 186(1),  
678 275–293.
- 679 Prokopyev, O.A., Kong, N. & Martinez-Torres, D.L. (2009). The equitable dis-  
680 persion problem. *European Journal of Operational Research* 197(1), 59–67.
- 681 Resende, M.G.C., Martí, R., Gallego, M. & Duarte, A. (2010). GRASP and  
682 path relinking for the max–min diversity problem. *Computers & Operations*  
683 *Research* 37(3), 498–508.
- 684 Saboonchi, B., Hansen, P. & Perron, S. (2014). MaxMinMin  $p$ -dispersion prob-  
685 lem: A variable neighborhood search approach. *Computers & Operations Re-*  
686 *search* 52, 251–259.
- 687 Syswerda, G. (1989). Uniform crossover in genetic algorithms. *Proceedings of*  
688 *the 3rd International Conference on Genetic Algorithms*, pp. 2–9, Morgan  
689 Kaufmann Publishers Inc., San Francisco, CA, USA.
- 690 Silva, J.D.A., Hruschka, E.R. & Gama, J. (2017). An evolutionary algorithm for  
691 clustering data streams with a variable number of clusters. *Expert Systems*  
692 *with Applications* 67, 228–238.
- 693 Wang, Y., Wu, Q. & Glover, F. (2017). Effective metaheuristic algorithms for the  
694 minimum differential dispersion problem. *European Journal of Operational*  
695 *Research* 258, 829–843.
- 696 Wu, Q. & Hao, J.K. (2013). A hybrid metaheuristic method for the maximum  
697 diversity problem. *European Journal of Operational Research* 231(2), 452–464.
- 698 Wu, Q., Wang, Y. & Lü, Z. (2015). A tabu search based hybrid evolutionary  
699 algorithm for the max-cut problem. *Applied Soft Computing* 34, 827–837.
- 700 Yang, B., Cheung, W. & Liu J. (2007). Community mining from signed so-  
701 cial networks. *IEEE Transactions on Knowledge & Data Engineering* 19(10),  
702 1333–1348.

- 703 Zhao, H., Xu, W. & Jiang, R. (2015). The Memetic algorithm for the optimiza-  
704 tion of urban transit network. *Expert Systems with Applications* 42, 3760–  
705 3773.
- 706 Zhou, Y. & Hao, J.K. (2017). An iterated local search algorithm for the mini-  
707 mum differential dispersion problem. *Knowledge-Based Systems* 125, 26–38.