Computers & Operations Research 72 (2016) 118–127

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A tabu search based memetic algorithm for the max-mean dispersion problem

Xiangjing Lai [a], Jin-Kao Hao [a,b,*]

[a] LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers, France
[b] Institut Universitaire de France, Paris, France

ARTICLE INFO

Available online 4 March 2016

Keywords:
Dispersion problem
Tabu search
Memetic algorithm
Heuristics

ABSTRACT

Given a set V of n elements and a distance matrix $[d_{ij}]_{n\times n}$ among elements, the max-mean dispersion problem (MaxMeanDP) consists in selecting a subset M from V such that the mean dispersion (or distance) among the selected elements is maximized. Being a useful model to formulate several relevant applications, MaxMeanDP is known to be NP-hard and thus computationally difficult. In this paper, we present a tabu search based memetic algorithm for MaxMeanDP which relies on solution recombination and local optimization to find high quality solutions. One key contribution is the identification of the fast neighborhood induced by the one-flip operator which takes linear time. Computational experiments on the set of 160 benchmark instances with up to 1000 elements commonly used in the literature show that the proposed algorithm improves or matches the published best known results for all instances in a short computing time, with only one exception, while achieving a high success rate of 100%. In particular, we improve 53 previous best results (new lower bounds) out of the 60 most challenging instances. Results on a set of 40 new large instances with 3000 and 5000 elements are also presented. The key ingredients of the proposed algorithm are investigated to shed light on how they affect the performance of the algorithm.

1. Introduction

Given a weighted complete graph $G=(V,E,D)$, where V is the set of n vertices, E is the set of $\frac{n\times(n-1)}{2}$ edges, and D represents the set of edge weights d_{ij} $(i\neq j)$, the generic equitable dispersion problem consists in selecting a subset M from V such that some objective function f defined on the subgraph induced by M is optimized [22]. In the related literature, a vertex $v\in V$ is also called an element, and the edge weight $d_{ij}\in D$ is called the distance (or diversity) between elements i and j.

According to the objective function to be optimized as well as the constraints on the cardinality of subset M, several specific equitable dispersion problems can be defined. At first, if the cardinality of M is fixed to a given number m, the related equitable dispersion problems include the following four classic variants: (1) the max-sum diversity problem, also known as the maximum diversity problem (MDP), which is to maximize the sum of distances among the selected elements [1,3,8,13,16,20,25]; (2) the max-min diversity problem that aims to maximize the minimum distance among the selected elements [7,21,23,24]; (3) the maximum minsum dispersion problem (MaxMinsumDP) that aims to maximize the minimum aggregate dispersion among the selected elements [2,22]; (4) the minimum differential dispersion problem (MinDiffDP) whose goal is to minimize the difference between the maximum and minimum aggregate dispersion among the selected elements to guarantee that each selected element has the approximately same total distance from the other selected elements [2,9,22]. In addition, when the cardinality of subset M is not fixed, i.e., the size of M is allowed to vary from 2 to n, the related equitable dispersion problems include the max-mean dispersion problem (MaxMeanDP) and the weighted MaxMeanDP [4,5,17,22].

In this study, we focus on MaxMeanDP which can be described as follows [22]. Given a set V of n elements and a distance matrix $[d_{ij}]_{n\times n}$ where d_{ij} represents the distance between elements i and j and can take a positive or negative value, the max-mean dispersion problem consists in selecting a subset M ($|M|$ is not fixed) from V such that the mean dispersion among the selected elements, i.e., $\frac{\sum_{i,j\in M; i<j} d_{ij}}{|M|}$, is maximized.

* Corresponding author at: LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers, France.
E-mail addresses: laixiangjing@gmail.com (X. Lai), hao@info.univ-angers.fr (J.-K. Hao).

http://dx.doi.org/10.1016/j.cor.2016.02.016
0305-0548/

MaxMeanDP can be naturally expressed as a fractional 0–1 programming problem with binary variables x_i that takes 1 if element i is selected and 0 otherwise [17,22], i.e.,

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

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

$$x_i \in \{0,1\}, \quad i = 1,2,\ldots,n; \tag{3}$$

where the constraint (2) guarantees that at least two elements are selected.

In addition to its theoretical significance as a NP-hard problem [22], MaxMeanDP has a variety of real-world applications, such as web pages ranks [15], community mining [26], and others mentioned in [4].

Given the interest of MaxMeanDP, several solution approaches have been proposed in the literature to deal with this hard combinatorial optimization problem. In 2009, Prokopyev et al. [22] presented a linear mixed 0–1 programming formulation for MaxMeanDP. In the same work, the authors also presented a Greedy Randomized Adaptive Search Procedure (GRASP) for generic equitable dispersion problems. In 2013, Martí and Sandoya [17] proposed a GRASP with the path relinking method (GRASP-PR), and the computational results show that GRASP-PR outperforms the previously reported methods. In 2014, Della Croce et al. [5] developed a two-phase hybrid heuristic approach combining a mixed integer non linear solver and a local branching procedure, and showed competitive results compared to the GRASP-PR method. In 2015, Carrasco et al. [4] introduced a highly effective two-phase tabu search algorithm which was also compared to the GRASP-PR method. Very recently (in 2016), Della Croce et al. [6] extended their previous two-phase hybrid heuristic approach of [5] by adding a third phase based on path-relinking, and presented competitive results. Among these reviewed heuristics, the four most recent methods of [4–6,17] can be considered to represent the current state of the art and thus are used as reference algorithms for our computational studies in Section 3.

In this paper, we propose the first population-based memetic algorithm MAMMDP for solving MaxMeanDP. The proposed algorithm combines a random crossover operator to generate new offspring solutions and a tabu search method [11] to find good local optima. For local optimization, MAMMDP critically relies on its tabu search procedure which explores a very effective one-flip neighborhood. The performance of our algorithm is assessed on a set of 160 benchmark instances ($20 \leq n \leq 1000$) commonly used in the literature and a set of additional 40 large-sized instances that we generate (n=3000,5000). For the first set of existing benchmarks, the experimental results show that the proposed algorithm is able to attain, in a short or very short computing time, all current best known results established by any existing algorithms, except for one instance. Furthermore, it can even improve the previous best known result for a number of these instances. The effectiveness of the proposed algorithm is also verified on much larger instances of the second set with 3000 and 5000 elements.

In Section 2, we describe the general scheme and the components of the proposed algorithm. Section 3 is dedicated to computational results based on the 200 benchmark instances and comparisons with state-of-the-art algorithms from the literature. In Section 4, we analyze some important components of the proposed algorithm. Finally, we conclude the paper in the last section.

2. Memetic algorithm for the max-mean dispersion problem

2.1. General procedure

Memetic algorithms are a general framework which aims to provide the search with a desirable trade-off between intensification and diversification through the combined use of a crossover operator (to generate new promising solutions) and a local optimization procedure (to locally improve the generated solutions) [18,19]. The proposed memetic algorithm (denoted by MAMMDP) adopts the principles and guidelines of designing effective memetic algorithm for discrete combinatorial problems [14]. The general procedure of our MAMMDP algorithm is shown in Algorithm 1, where s^* and s^w respectively represent the best solution found so far and the worst solution in the population in terms of the objective value, and *PairSet* is the set of solution pairs (s^i, s^j), which is initially composed of all the possible solution pairs (s^i, s^j) in the population and is dynamically updated as the search progresses.

Our MAMMDP algorithm starts with an initial population P (line 4) which includes p different solutions, where each of them is randomly generated and then improved by the tabu search procedure. After the initialization of population (Section 2.3), the algorithm enters a while loop (lines 11 to 25) to make a number of generations. At each generation, a solution pair (s^i, s^j) is randomly selected from *PairSet* and then the crossover operator (line 14) is applied to the selected solution pair (s^i, s^j) to generate a new solution s^o (Section 2.5). Subsequently, s^o is improved by the tabu search procedure (line 15) (Section 2.4). After that, a population updating rule is used to update the population (lines 20 to 24) (Section 2.6). Meanwhile, the *PairSet* is accordingly updated as follows: First, the solution pair (s^i, s^j) is removed from *PairSet* (line 13); Then, if an offspring solution s^o replaces the worst solution s^w in the population, all the solution pairs containing s^w are removed from *PairSet* and all the solution pairs that can be generated by combining s^o with other solutions in the population are added into *PairSet* (lines 22 to 23). The while loop ends when *PairSet* becomes empty, then the population is recreated, while preserving the best solution (s^*) found so far in the new population (lines 4 to 8), and the above while loop is repeated if the timeout limit is not reached.

It is worth noting that compared with the traditional random selection scheme, the proposed MAMMDP algorithm uses the set *PairSet* to contain the solution pairs of the population for crossover operations. This strategy, inspired by the path relinking method [12], ensures that every pair of solutions in the population is combined exactly once, favoring a more intensified search.

Algorithm 1. Memetic algorithm for max-mean dispersion problem.

1: **Input**: The set $V = \{v_1, v_2, \ldots, v_n\}$ of n elements and the distance matrix $D = [d_{ij}]_{n \times n}$, the population size p, the timeout limit t_{out}.
2: **Output**: the best solution s^* found
3: **repeat**
4: $P = \{s^1, \ldots, s^p\} \leftarrow$ Population_Initialization(V) /* Section 2.3 */
5: **if** it is not in the first loop **then**
6: $s^w \leftarrow \arg\min \{f(s^i) : i = 1, \ldots, p\}$
7: $P \leftarrow P \cup \{s^*\} \setminus \{s^w\}$
8: **end if**
9: $s^* \leftarrow \arg\max \{f(s^i) : i = 1, \ldots, p\}$ /* s^* keeps the best solution found */
10: $PairSet \leftarrow \{(s^i, s^j) : 1 \leq i < j \leq p\}$
11: **while** $PairSet \neq \varnothing$ and $time < t_{out}$ **do**

```
Randomly pick a solution pair (s^i, s^j) ∈ PairSet
PairSet ← PairSet \ {(s^i, s^j)}
s^o ← CrossoverOperator(s^i, s^j)      /* Section 2.5 */
s^o ← TabuSearch(s^o)                  /* Section 2.4 */
if f(s^o) > f(s*) then
    s* ← s^o
end if
s^w ← arg min {f(s^i) : i = 1, ..., p}
if s^o dose not exist in P and f(s^o) > f(s^w) then
    P ← P ∪ {s^o} \ {s^w}
    PairSet ← PairSet \ {(s^w, s^k) : s^k ∈ P}
    PairSet ← PairSet ∪ {(s^o, s^k) : s^k ∈ P}
end if                                 /* Section 2.6 */
end while
until time ≥ t_out
```

2.2. Search space and solution representation

Given a MaxMeanDP instance with a set V of n elements as well as its distance matrix $D=[d_{ij}]_{n\times n}$, the search space Ω explored by our MAMMDP algorithm is composed of all possible subsets of V, i.e., $\Omega=\{M: M\subseteq V\}$. Formally, a subset M of V can be expressed by a n-dimensional binary vector, $(x_1,x_2,\ldots,x_n)$, where x_i takes 1 if element i belongs to M, and 0 otherwise. The search space Ω is thus given by:

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

Clearly, the size of the search space Ω is bounded by $O(2^n)$.

For any candidate solution $s=(x_1,x_2,\ldots,x_n)\in\Omega$, its quality is determined by the objective value ($f(s)$, Formula (1)) of the max-mean dispersion problem.

2.3. Population initialization

In our memetic algorithm, the initial population of p solutions is generated as follows. First, we generate p random solutions, where each component x_i $(i=1,2,\ldots,n)$ of solution $(x_1,x_2,\ldots,x_n)$ is randomly assigned a value from $\{0,1\}$ using a uniform probability distribution. Then, we apply the tabu search procedure (see Section 2.4) to these solutions to reach p local optima which form the initial population.

2.4. Local optimization using tabu search

Local optimization is a key component of a memetic algorithm and ensures generally the role of an intensified search to locate high quality local optimum. In this study, we devise a tabu search (TS) method as the local optimization procedure which proves to be highly effective when it is applied alone. Given a neighborhood structure ($N(s)$) and a starting solution (s_0), our tabu search procedure iteratively replaces the incumbent solution s by a best eligible neighboring solution ($s'\in N(s)$) until the stopping condition is met, i.e., the best solution (s_b) is not improved for α consecutive iterations (called the depth of TS). At each iteration of TS, the performed move is recorded in the tabu list to prevent the reverse move from being performed for the next tt iterations. Here, tt is called the tabu tenure and controlled by a special tabu list management strategy. A move is identified to be eligible if it is not forbidden by the tabu list or it leads to a solution better than the best solution found so far in terms of the objective function value (aspiration criterion). The general scheme of our TS method is described in Algorithm 2, and the neighborhood structure employed by our TS method and the tabu list management strategy are described in the following subsections.

Algorithm 2. *TabuSearch*$(s_0, N(s), \alpha)$.

```
Input: Input solution s_0, neighborhood N(s), search depth α
Output: The best solution s_b found during the tabu search process
s ← s_0    /*s is the current solution */
s_b ← s    /* s_b is the best solution found so far */
d = 0      /* d counts the consecutive iterations where s_b is not updated */
repeat
    Choose a best eligible neighboring solution s' ∈ N(s)
    /*s' is called eligible if it is not forbidden by the tabu list or better than s_b */
    s ← s'
    Update tabu list
    if f(s) > f(s_b) then
        s_b ← s,
        d = 0
    else
        d = d + 1
    end if
until d = α
return s_b
```

2.4.1. Move and neighborhood

The neighborhood N_1 of our tabu search algorithm is defined by the one-flip move operator which consists of changing the value of a single variable x_i to its complementary value $1-x_i$. As such, given a solution s, the one-flip neighborhood $N_1(s)$ of s is composed of all possible solutions that can be obtained by applying the one-flip move to s. The size of the neighborhood $N_1(s)$ is thus bounded by $O(n)$, where n is the number of elements in V. To efficiently examine the neighborhood N_1, we devise a fast neighborhood evaluation technique which contributes greatly to the computational effectiveness of the tabu search method.

2.4.2. Fast neighborhood evaluation technique

Our fast neighborhood evaluation technique maintains a n-dimensional vector $W=(p_1,p_2,\ldots,p_n)$ to effectively calculate the move value (i.e., the change of objective value) of each possible move applicable to the current solution s, where the entry p_i represents the sum of distances between the element i and the selected elements for 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 an one-flip move is performed by flipping variable x_i as $x_i\leftarrow(1-x_i)$, then the move value Δ_i can be rapidly computed as follows:

$$\Delta_i=\begin{cases}\dfrac{-f(s)}{|M|+1}+\dfrac{p_i}{|M|+1}, & \text{for } x_i=0;\\ \dfrac{f(s)}{|M|-1}-\dfrac{p_i}{|M|-1}, & \text{for } x_i=1;\end{cases}\tag{4}$$

where $f(s)$ is the objective value of the current solution s and $|M|$ is the number of selected elements in s. Subsequently, the vector W is accordingly updated as:

$$p_j=\begin{cases}p_j+d_{ij}, & \text{for } x_i=0, j\neq i;\\ p_j-d_{ij}, & \text{for } x_i=1, j\neq i;\\ p_j, & \text{for } j=i;\end{cases}\tag{5}$$

The vector W is initialized at the beginning of each call of TS with the complexity of $O(n^2)$, and is updated in $O(n)$ after each move. With the fast evaluation technique, the best move can be identified in $O(n)$. Therefore, the total complexity of each iteration of the TS method is bounded by $O(n)$.

2.4.3. Tabu list management strategy

In our TS procedure, we use a tabu list management strategy to dynamically tune the tabu tenure tt, which is adapted according to a technique proposed in [10] where the tabu tenure is given by a periodic step function. If the current iteration is y, then the tabu tenure of a move is denoted by $tt(y)$.

Precisely, our tabu tenure function is defined, for each period, by a sequence of values $(a_1, a_2, \ldots, a_{q+1})$ and a sequence of interval margins $(y_1, y_2, \ldots, y_{q+1})$ such that for each y in $[y_i, y_{i+1}-1]$, $tt(y) = a_i + rand(2)$, where $rand(2)$ denotes a random integer between 0 and 2. Here, q is fixed to 15, $(a)_{i=1,\ldots,15} = \frac{T_{max}}{8} \times (1,2,1,4,1,2,1,8,1,2,1,4,1,2,1)$, where T_{max} is a parameter which represents the maximum tabu tenure. Finally, the interval margins are defined by $y_1 = 1$, $y_{i+1} = y_i + 5a_i$ $(i \leq 15)$.

As such, the tabu tenure varies periodically (with cycles of 15 periods) and for each period the tabu tenure takes one of four possible values: $\frac{T_{max}}{8} \times 1$ (the smallest), $\frac{T_{max}}{8} \times 2$, $\frac{T_{max}}{8} \times 4$, and $\frac{T_{max}}{8} \times 8$ (the largest). In our case, T_{max} is set to 120 (See Section 4.2 for an analysis of this parameter), leading to the values of 15, 30, 60, 120. Each tabu tenure is kept for a number of consecutive iterations. In principle, this function helps the TS method reach a good tradeoff between intensification and diversification during its search.

2.5. Crossover operator

Within a memetic algorithm, the crossover operator is another essential ingredient whose main goal is to bring the search process to new promising search regions to diversify the search. In this work, we adopt the standard uniform crossover operator. Given two parent solutions $s^1 = (x_1^1, x_2^1, \ldots, x_n^1)$ and $s^2 = (x_1^2, x_2^2, \ldots, x_n^2)$, the value of each component x_i^o $(i = 1, 2, \ldots, n)$ of the offspring solution s^o is randomly chosen from the set $\{x_i^1, x_i^2\}$ with an equal probability of 0.5. In spite of its simplicity, this crossover operator has shown to be quite robust and effective in many settings.

2.6. Population updating rule

When a new offspring solution is generated by the crossover operator, it is first improved the tabu search procedure and then used to update the population according to the following rule. If the offspring solution is distinct from any existing solution in the population and is better than the worst solution in the population in terms of objective value, then the offspring solution replaces the worst solution of the population. Otherwise, the population is kept unchanged.

3. Experimental results and comparisons

In this section, we present extensive computational experiments to assess the performance of our memetic algorithm.

3.1. Benchmark instances

Our computational experiments are carried out on two types of instances, namely Type I and Type II. The distances of Type I instances are randomly generated in the interval $[-10, 10]$ with a uniform probability distribution, while the distances of Type II instances are generated from $[-10, -5] \cup [5, 10]$ with the same probability distribution.

Additionally, the set of benchmark instances used is composed of two subsets. The first subset consists of 80 Type I instances and 80 Type II instances with the number of elements n ranging from 20 to 1000. These 160 instances were extensively adopted by the previous studies [4,5,17] and are available online at http://www.optsicom.es/edp/. We also use a second subset of 20 Type I and 20 Type II large instances with $n = 3000$ or 5000 that we generated in the same way as the previous instances.[1]

3.2. Parameter settings and experimental protocol

Our memetic algorithm relies on only three parameters: the population size p, the depth of tabu search α and the maximum tabu tenure T_{max}. For p and α, we follow [25] and set $p = 10, \alpha = 50\,000$ while setting $T_{max} = 120$ empirically (See Section 4.2 for an analysis of this parameter). This parameter setting is used for all the experiments reported in Section 3. Even if fine-tuning these parameters would lead to better results, as we show below, our algorithm with this fixed setting is able to attain a high performance with respect to the state of the art results.

Our memetic algorithm is programmed in C++ and compiled using g++ compiler with the '−O2' flag.[2] All experiments are carried out on a computer with an Intel Xeon E5440 processor (2.83 GHz CPU and 2 Gb RAM), running the Linux operating system. Following the DIMACS machine benchmark procedure,[3] our machine requires respectively 0.23, 1.42, and 5.42 s for the graphs r300.5, r400.5, r500.5.

Given the stochastic nature of our algorithm, we solve each tested problem instance 20 times, where the stopping condition is given by a cutoff time limit which depends on the size of the instances. Specifically, the cutoff limit t_{out} is set to be 10 s for $n \leq 150$, 100 s for $n \in [500, 1000]$, 1000 s for $n = 3000$, and 2000 s for $n = 5000$. As we discuss in Section 3.3, these time limits are significantly shorter than those used by the reference algorithms of the literature.

3.3. Results and comparisons on small and medium sized instances

Our first experiment aims to evaluate the performance of our MAMMDP algorithm on the set of 160 popular instances with up to 1000 elements. The computational results of MAMMDP on the 60 medium sized instances are summarized in Table 1, whereas the results of the 100 small instances with $n \leq 150$ are available at our web-page (see Section 3.1, footnote 1).

In addition to the instance name and size (columns 1 and 2), column 3 of Table 1 indicates the best objective values (f_{pre}) of the literature which are compiled from the best results yielded by four recent and best performing algorithms, namely GRASP-PR [17], a two-phased hybrid heuristic approach [5], a three-phase hybrid approach [6], and a two-phase tabu search (TP-TS) method [4] (from http://www.optsicom.es/edp/). Note that the previous best known results (f_{pre}) are given with two decimal in the literature. Columns 4 to 7 respectively give the best objective values obtained by each of these four reference algorithms, where the mark '–' means that the corresponding result is not available. In [4,17], the cutoff time limits were set to 90, 600, and 1800 s for instances with $n = 500, 750, 1000$, respectively while in [5] the cutoff limits were set to 60 and 600 s for the instances of size 150 and 500, and prolonged to be 120 and 1200 s for instances of 150 and 500 items in [6]. The GRASP-PR method was performed on a computer with an Intel Core Solo 1.4 GHz CPU with 3 GB RAM [17]. The two-phase hybrid heuristic in [5] and the three-phase hybrid approach in [6] were run on a computer with an Intel Core i5-3550 3.30 GHz CPU with 4 GB RAM [5] and the TP-TS method was run on a computer with an Intel Core 2 Quad CPU and 6GB RAM [4].

[1] The source code of generating these instances is available from our website: http://www.info.univ-angers.fr/pub/hao/maxmeandp.html.

[2] Our best results are available at our web-page (see Section 3.1, footnote 1). The source code of our algorithm will also be available.

[3] dmclique, ftp://dimacs.rutgers.edu/pub/dsj/clique, the benchmark procedure is complied by gcc compiler with the '−O2' flag.

Table 1
Computational results of the proposed MAMMDP algorithm on the set of 60 representative instances with $500 \leq n \leq 1000$. Each instance is independently solved 20 times, and improved results are indicated in bold compared to the previous best known results f_{pre} of the literature reported in [4–6,17].

Instance	n	f_{pre}	[17] (2013)	[5] (2014)	[6] (2016)	[4] (2014)	MAMMDP			
							f_{best}	f_{avg}	SR	$t(s)$
MDPI1_500	500	81.28	78.6050000	81.25	81.28	81.28	81.277044	81.277044	20/20	0.69
MDPI2_500	500	77.79	76.8734667	77.45	77.79	77.60	**78.610216**	**78.610216**	20/20	1.43
MDPI3_500	500	76.30	75.6914063	75.31	76.30	75.65	76.300787	76.300787	20/20	2.71
MDPI4_500	500	82.33	81.8058434	82.28	82.33	81.47	82.332081	82.332081	20/20	0.95
MDPI5_500	500	80.08	78.5695714	80.01	80.08	79.92	**80.354029**	**80.354029**	20/20	2.80
MDPI6_500	500	81.25	79.6426282	81.12	81.25	79.93	81.248553	81.248553	20/20	0.78
MDPI7_500	500	78.16	75.4989726	78.09	78.16	77.71	78.164511	78.164511	20/20	0.92
MDPI8_500	500	79.06	76.9836424	79.01	79.06	78.70	**79.139881**	**79.139881**	20/20	1.27
MDPI9_500	500	77.36	75.7209449	76.98	77.36	77.15	**77.421000**	**77.421000**	20/20	2.37
MDPI10_500	500	81.25	80.3789051	81.24	81.25	81.02	**81.309871**	**81.309871**	20/20	0.91
MDPII1_500	500	109.38	108.152545	109.16	109.38	109.33	**109.610136**	**109.610136**	20/20	0.75
MDPII2_500	500	105.33	103.287851	105.06	105.33	104.81	**105.717536**	**105.717536**	20/20	0.88
MDPII3_500	500	107.79	106.301714	107.64	107.79	107.18	**107.821739**	**107.821739**	20/20	0.89
MDPII4_500	500	106.10	104.618442	105.37	106.10	105.69	106.100071	106.100071	20/20	0.56
MDPII5_500	500	106.59	103.608188	106.37	106.55	106.59	**106.857162**	**106.857162**	20/20	0.99
MDPII6_500	500	106.17	104.813987	105.52	105.77	106.17	**106.297958**	**106.297958**	20/20	0.98
MDPII7_500	500	107.06	104.503378	106.61	107.06	106.92	**107.149379**	**107.149379**	20/20	0.88
MDPII8_500	500	103.78	100.021407	103.41	103.78	103.49	103.779195	103.779195	20/20	0.59
MDPII9_500	500	106.24	104.927769	106.20	106.24	105.97	**106.619793**	**106.619793**	20/20	1.11
MDPII10_500	500	104.15	103.497014	103.79	104.15	103.56	**104.651507**	**104.651507**	20/20	1.01
MDPI1_750	750	95.86	–	–	–	95.86	**96.650699**	**96.650699**	20/20	4.31
MDPI2_750	750	97.42	–	–	–	97.42	**97.564880**	**97.564880**	20/20	3.82
MDPI3_750	750	96.97	–	–	–	96.97	**97.798864**	**97.798864**	20/20	1.81
MDPI4_750	750	95.21	–	–	–	95.21	**96.041364**	**96.041364**	20/20	4.38
MDPI5_750	750	96.65	–	–	–	96.65	**96.761928**	**96.761928**	20/20	0.65
MDPI6_750	750	99.25	–	–	–	99.25	**99.861250**	**99.861250**	20/20	5.55
MDPI7_750	750	96.26	–	–	–	96.26	**96.545413**	**96.545413**	20/20	1.01
MDPI8_750	750	96.46	–	–	–	96.46	**96.726976**	**96.726976**	20/20	1.73
MDPI9_750	750	96.78	–	–	–	96.78	**98.058377**	**98.058377**	20/20	2.18
MDPI10_750	750	99.85	–	–	–	99.85	**100.064185**	**100.064185**	20/20	3.42
MDPII1_750	750	127.69	–	–	–	127.69	**128.863707**	**128.863707**	20/20	5.66
MDPII2_750	750	130.79	–	–	–	130.79	**130.954426**	**130.954426**	20/20	2.31
MDPII3_750	750	129.40	–	–	–	129.40	**129.782453**	**129.782453**	20/20	11.64
MDPII4_750	750	125.68	–	–	–	125.68	**126.582271**	**126.582271**	20/20	1.48
MDPII5_750	750	128.13	–	–	–	128.13	**129.122878**	**129.122878**	20/20	1.32
MDPII6_750	750	128.55	–	–	–	128.55	**129.025215**	**129.025215**	20/20	7.98
MDPII7_750	750	124.91	–	–	–	124.91	**125.646682**	**125.646682**	20/20	3.38
MDPII8_750	750	130.66	–	–	–	130.66	**130.940548**	**130.940548**	20/20	1.91
MDPII9_750	750	128.89	–	–	–	128.89	**128.889908**	**128.889908**	20/20	1.30
MDPII10_750	750	132.99	–	–	–	132.99	**133.265300**	**133.265300**	20/20	1.81
MDPI1_1000	1000	118.76	–	–	–	118.76	**119.174112**	**119.174112**	20/20	8.25
MDPI2_1000	1000	113.22	–	–	–	113.22	**113.524795**	**113.524795**	20/20	3.52
MDPI3_1000	1000	114.51	–	–	–	114.51	**115.138638**	**115.138638**	20/20	2.32
MDPI4_1000	1000	110.53	–	–	–	110.53	**111.150397**	**111.150397**	20/20	3.58
MDPI5_1000	1000	111.24	–	–	–	111.24	**112.723188**	**112.723188**	20/20	1.61
MDPI6_1000	1000	112.08	–	–	–	112.08	**113.198718**	**113.198718**	20/20	7.72
MDPI7_1000	1000	110.94	–	–	–	110.94	**111.555536**	**111.555536**	20/20	1.88
MDPI8_1000	1000	110.29	–	–	–	110.29	**111.263194**	**111.263194**	20/20	3.55
MDPI9_1000	1000	115.78	–	–	–	115.78	**115.958833**	**115.958833**	20/20	2.38
MDPI10_1000	1000	114.29	–	–	–	114.29	**114.731644**	**114.731644**	20/20	2.16
MDPII1_1000	1000	145.46	–	–	–	145.46	**147.936175**	**147.936175**	20/20	1.60
MDPII2_1000	1000	150.49	–	–	–	150.49	**151.380035**	**151.380035**	20/20	1.78
MDPII3_1000	1000	149.36	–	–	–	149.36	**150.788178**	**150.788178**	20/20	4.92
MDPII4_1000	1000	147.91	–	–	–	147.91	**149.178006**	**149.178006**	20/20	3.80
MDPII5_1000	1000	150.23	–	–	–	150.23	**151.520847**	**151.520847**	20/20	3.28
MDPII6_1000	1000	147.29	–	–	–	147.29	**148.343378**	**148.343378**	20/20	3.22
MDPII7_1000	1000	148.41	–	–	–	148.41	**148.742375**	**148.742375**	20/20	6.30
MDPII8_1000	1000	145.87	–	–	–	145.87	**147.826804**	**147.826804**	20/20	13.52
MDPII9_1000	1000	145.67	–	–	–	145.67	**147.083880**	**147.083880**	20/20	3.83
MDPII10_1000	1000	148.40	–	–	–	148.40	**150.046137**	**150.046137**	20/20	2.13
#Better							53	53		
#Equal							7	7		
#Worse							0	0		
p-value			7.74e−06	7.74e−06	3.12e−4	4.85e−13				

The results of our MAMMDP algorithm are given in columns 8 to 11, including the best objective value (f_{best}) over 20 independent runs, the average objective value (f_{avg}), the success rate (SR) to reach f_{best}, and the average computing time in seconds ($t(s)$) to reach f_{best}. The rows *Better, Equal, Worse* respectively indicate the number of instances for which our result is better, equal to and worse than f_{pre}. The improved results compared to f_{pre} are indicated in bold. In addition, to verify whether there exists a

significant difference between the best results of MAMMDP and those of four reference algorithms, the p-values from the non-parametric Friedman tests are reported in the last row of Table 1.

First, Table 1 shows that MAMMDP improves the previous best known result for all instances except for 7 cases for which our result matches the previous best known result. These results clearly indicate the superiority of MAMMDP compared to the previous MaxMeanDP algorithms. Second, when examining the success rate of the algorithm, one can find that the MAMMDP algorithm achieves a success rate of 100% for all tested instances, which means a good robustness of the MAMMDP algorithm. Third, in terms of average computing time, it can be seen that for all instances, MAMMDP obtains its best result with an average time of less than 14 s, which are much shorter than those of the previous algorithms in the literature. Moreover, all p-values are smaller than 0.05, confirming the statistical significance of the observed differences.

3.4. Computational results and comparison on large-scale instances

In order to further assess the performance of the proposed MAMMDP algorithm on large-scale instances, our second experiment was carried out based on the set of 40 instances with $n=3000$, 5000. In this experiment, one of the best performing algorithms in the literature (i.e., the recent two-phase tabu search (TP-TS) algorithm of [4]) is run once with a long computational time of one week, and the proposed MAMMDP algorithm is run according to the experimental protocol in Section 3.2. Note that for the TP-TS algorithm only one run is conducted due to the fact that the random seed for generating initial solutions is fixed in the executable code provided by its authors. Experimental results are reported in Table 2, where the rows *Better, Equal, Worse* of the table respectively show the number of instances for which the corresponding result of our MAMMDP algorithm is better, equal to and worse than the result of the TP-TS algorithm, and other entries have the same meanings as those of Table 1.

Table 2 shows that for the instances with 3000 elements, MAMMDP reaches a success rate of at least 10/20, which is an interesting indicator as to its good performance for these instances. However, for the still larger instances with $n=5000$, the success rate of the algorithm significantly varies between 4/20 and 19/20, which means that these large instances are clearly more difficult. Moreover, the difference between the best and average

Table 2
Computational results and comparison on the set of 40 new large instances with $n=3000$, 5000. Bold values indicate better results compared to those obtained by the TP-TS algorithm of [4], one of the best performing methods in the literature.

Instance	n	TP-TS [4]	MAMMDP			
			f_{best}	f_{avg}	SR	$t(s)$
MDPI1_3000	3000	188.095317	**189.048965**	**189.048965**	20/20	88.36
MDPI2_3000	3000	186.473026	**187.387292**	**187.387292**	20/20	60.71
MDPI3_3000	3000	184.341415	**185.666806**	**185.655084**	13/20	352.85
MDPI4_3000	3000	185.588182	**186.163727**	**186.153631**	16/20	300.37
MDPI5_3000	3000	186.234859	**187.545515**	**187.545515**	20/20	61.29
MDPI6_3000	3000	189.093513	**189.431257**	**189.431257**	20/20	51.99
MDPI7_3000	3000	187.451175	**188.242583**	**188.242583**	20/20	86.57
MDPI8_3000	3000	185.735801	**186.796814**	**186.796814**	20/20	48.04
MDPI9_3000	3000	187.107609	**188.231264**	**188.231264**	20/20	151.78
MDPI10_3000	3000	184.686569	**185.682511**	**185.623778**	10/20	228.72
MDPII1_3000	3000	252.181753	**252.320433**	**252.320433**	20/20	59.70
MDPII2_3000	3000	248.697168	**250.062137**	**250.062137**	20/20	220.10
MDPII3_3000	3000	250.530306	**251.906270**	**251.906270**	20/20	146.32
MDPII4_3000	3000	253.096329	**253.941007**	**253.940596**	19/20	370.76
MDPII5_3000	3000	252.562146	**253.260423**	**253.260350**	17/20	374.00
MDPII6_3000	3000	249.715999	**250.677750**	**250.677750**	20/20	55.35
MDPII7_3000	3000	249.593867	**251.134413**	**251.134413**	20/20	74.72
MDPII8_3000	3000	252.056539	**252.999648**	**252.999648**	20/20	79.82
MDPII9_3000	3000	251.362462	**252.425770**	**252.425770**	20/20	90.27
MDPII10_3000	3000	251.116925	**252.396590**	**252.396590**	20/20	13.18
MDPI1_5000	5000	236.333205	**240.162535**	**240.102875**	7/20	312.13
MDPI2_5000	5000	239.014282	**241.827401**	**241.792978**	6/20	1244.36
MDPI3_5000	5000	238.474238	**240.890819**	**240.888162**	19/20	810.48
MDPI4_5000	5000	237.397159	**240.997186**	**240.976789**	6/20	653.64
MDPI5_5000	5000	240.043931	**242.480129**	**242.475885**	19/20	735.16
MDPI6_5000	5000	238.001498	**240.322850**	**240.306326**	8/20	976.02
MDPI7_5000	5000	239.744358	**242.814943**	**242.774982**	5/20	259.50
MDPI8_5000	5000	237.915045	**241.194990**	**241.161763**	8/20	1148.60
MDPI9_5000	5000	235.910266	**239.760560**	**239.667613**	4/20	1219.71
MDPI10_5000	5000	241.804289	**243.473734**	**243.373015**	4/20	457.28
MDPII1_5000	5000	316.747833	**322.235897**	**322.181291**	5/20	1519.05
MDPII2_5000	5000	323.682866	**327.301910**	**327.006342**	5/20	1103.13
MDPII3_5000	5000	321.929067	**324.813456**	**324.801590**	10/20	955.81
MDPII4_5000	5000	317.676681	**322.237586**	**322.197276**	5/20	664.10
MDPII5_5000	5000	317.747934	**322.491211**	**322.380726**	7/20	1014.90
MDPII6_5000	5000	319.388979	**322.950488**	**322.703887**	4/20	352.88
MDPII7_5000	5000	319.980558	**322.850438**	**322.793125**	10/20	714.31
MDPII8_5000	5000	318.854528	**323.112120**	**323.053268**	11/20	879.48
MDPII9_5000	5000	320.437562	**323.543775**	**323.339842**	7/20	569.73
MDPII10_5000	5000	320.853036	**324.519908**	**324.414458**	15/20	752.95
#Better			40	40		
#Equal			0	0		
#Worse			0	0		
p-value			2.54e−10	2.54e−10		

objective values obtained by the MAMMDP algorithm is very small for all instances, implying a good robustness of the proposed MAMMDP algorithm.

When comparing with the TP-TS method in [4], one finds that the proposed MAMMDP algorithm largely outperforms this reference method. Specifically, for each instance tested, both the average and best objective values obtained by the MAMMDP algorithm with a short time limit (1000 and 2000 s respectively for $n=3000$ and 5000) are better than that obtained by the TP-TS method with a long running time of one week. Furthermore, the small p-values (<0.05) imply that the improvement of the MAMMDP algorithm over the TP-TS method is statistically significant.

4. Analysis and discussions

In this section, we study some essential ingredients of the proposed MAMMDP algorithm to understand their impacts on the performance of the algorithm, including the fast one-flip neighborhood, a sensitivity analysis of the main parameter, and the role of the memetic framework.

4.1. Importance of the fast one-flip neighborhood

In MAMMDP, local optimization is based on the one-flip neighborhood which can be rapidly examined in linear time. To highlight the key role of this fast neighborhood, we carried out an experiment with two simple methods, i.e., a multi-start steepest descent (MSD) method and our tabu search method described in Section 2.4. The steepest descent method is a special case of our tabu search method in which both the tabu tenure $tt(y)$ and the depth of tabu search α are set to 0, and the MSD method restarts the steepest descent method with randomly generated initial solutions until the given cutoff time is reached. For this experiment, the MSD method and our tabu search method were independently run 20 times to solve each of 30 representative instances with a time limit of 10 s per run.

The computational results are summarized in Table 3, where the symbols f_{pre}, f_{best}, f_{avg} and SR have the same meanings as those of the previous tables, and the rows *Better, Equal, Worse* respectively show the number of instances for which the corresponding result of the associated algorithm is better, equal to and worse than the best known value f_{pre} reported in the literature. In addition, to verify whether there exists a significant difference between the corresponding results of our tabu search algorithm (as well as the MSD algorithm) and f_{pre}, the p-values from the non-parametric Friedman tests are given in the last row of the table.

Table 3 discloses that the simple MSD method which only uses the one-flip neighborhood is able to obtain very competitive results compared to f_{pre}. Indeed, the best objective value f_{best} of the MSD method is better than the best known value of f_{pre} for 18 out of 30 representative instances. However, the p-value ($0.0833>0.05$) does not indicate a significant difference between the best objective values f_{best} of the MSD method and the values of f_{pre}. On the other hand, one finds that the average objective value f_{avg} of the MSD method is

Table 3
Comparison of results of our tabu search method and the multi-start steepest descent (MSD) method (using the fast one-flip neighborhood) with the previous best known results (f_{pre}) on 30 representative instances. Bold values indicate better results compared to f_{pre} in terms of both f_{best} and f_{avg}. Note that each instance was independently solved 20 times by the two algorithms.

Instance	n	f_{pre}	MSD			Tabu search		
			f_{best}	f_{avg}	SR	f_{best}	f_{avg}	SR
MDPI1_500	500	81.28	81.277044	81.106549	6/20	81.277044	81.246582	19/20
MDPI2_500	500	77.79	**78.546377**	**78.099058**	2/20	**78.610216**	**78.607906**	19/20
MDPI3_500	500	76.30	76.132727	75.801544	4/20	76.300787	76.245534	16/20
MDPI4_500	500	82.33	82.332081	82.193750	4/20	82.332081	82.326721	18/20
MDPI5_500	500	80.08	**80.335310**	**80.119814**	2/20	**80.354029**	**80.339680**	8/20
MDPII1_500	500	109.38	**109.610136**	**109.589862**	12/20	**109.610136**	**109.600254**	17/20
MDPII2_500	500	105.33	**105.627817**	105.239588	4/20	**105.717536**	**105.702056**	18/20
MDPII3_500	500	107.79	**107.821739**	107.619251	4/20	**107.821739**	107.763377	19/20
MDPII4_500	500	106.10	106.100071	105.790225	3/20	106.100071	106.082241	18/20
MDPII5_500	500	106.59	**106.817718**	**106.602249**	3/20	**106.857162**	**106.843908**	18/20
MDPI1_750	750	95.86	**96.366463**	**95.941724**	5/20	**96.650699**	**96.645990**	19/20
MDPI2_750	750	97.42	**97.459545**	97.070551	3/20	**97.564880**	**97.562612**	14/20
MDPI3_750	750	96.97	**97.362054**	**97.023236**	4/20	**97.798864**	**97.797455**	16/20
MDPI4_750	750	95.21	**95.368811**	94.924359	1/20	**96.041364**	**96.010351**	14/20
MDPI5_750	750	96.65	**96.667671**	**95.684861**	2/20	**96.761928**	**96.758349**	17/20
MDPII1_750	750	127.69	**128.068348**	127.539988	2/20	**128.863707**	**128.765676**	17/20
MDPII2_750	750	130.79	130.464095	130.068083	4/20	**130.954426**	**130.934415**	17/20
MDPII3_750	750	129.40	**129.53194**	128.967954	5/20	**129.782453**	**129.744375**	10/20
MDPII4_750	750	125.68	**126.506605**	**125.928465**	2/20	**126.582271**	**126.568047**	15/20
MDPII5_750	750	128.13	**128.580648**	127.904501	2/20	**129.122878**	**129.041765**	16/20
MDPI1_1000	1000	118.76	118.329488	117.986893	1/20	**119.174112**	**119.149807**	15/20
MDPI2_1000	1000	113.22	**113.249248**	112.646028	5/20	**113.524795**	**113.517910**	17/20
MDPI3_1000	1000	114.51	114.497181	113.977224	5/20	**115.138638**	**114.968788**	18/20
MDPI4_1000	1000	110.53	110.373096	109.833438	3/20	**111.150397**	**110.934387**	15/20
MDPI5_1000	1000	111.24	**112.073920**	111.07192	4/20	**112.723188**	**112.557951**	16/20
MDPII1_1000	1000	145.46	**147.099271**	**145.863805**	2/20	**147.936175**	**147.936175**	20/20
MDPII2_1000	1000	150.49	150.368746	149.588345	5/20	**151.380035**	**151.329955**	18/20
MDPII3_1000	1000	149.36	149.119968	148.337575	4/20	**150.788178**	**150.782873**	18/20
MDPII4_1000	1000	147.91	147.835705	147.155971	3/20	**149.178006**	**149.140618**	17/20
MDPII5_1000	1000	150.23	150.083205	149.466598	3/20	**151.520847**	**151.519357**	19/20
#Better			18	9		26	25	
#Equal			3	0		4	0	
#Worse			9	21		0	5	
p-value			8.33e−2	1.1e−2		3.41e−7	2.61e−4	

better than the value of f_{pre} for 9 out of 30 tested instances, even though the reverse is true for 21 instances. These outcomes clearly show that the fast one-flip neighborhood is very effective for the MaxMeanDP problem compared to those employed by other heuristic algorithms in the literature.

Furthermore, when comparing our tabu search method with the MSD method, one finds that the tabu search method which, like MSD, explores only the one-flip neighborhood, performs even better. It obtains improved best known results for 26 instances (and the same result for the remaining cases) and dominates the MSD method in terms of both the best and average objective values.

This experiment confirms that the fast one-flip neighborhood is very appropriate for local optimization applied to the MaxMeanDP problem and constitutes one key element of the proposed MAMMDP algorithm.

4.2. Sensitivity analysis of parameter T_{max}

Now, we turn our attention to a sensitivity analysis of the important parameter T_{max} which is used to control the tabu tenures. To analyze the influence of parameter T_{max}, we carried out an experiment on a set of 34 representative instances, where we varied the value of T_{max} within a reasonable range, i.e., $T_{max} \in \{60, 80, 100, 120, 140, 160, 180, 200\}$, and then run the resulting tabu search procedure 20 times for each value of T_{max} and each instance. Finally, the average objective values obtained over 20 runs were recorded for this study.

The experimental results are summarized in Table 4, where the second row gives the value of T_{max}. Columns 2 to 9 show the average objective values obtained by the tested values of T_{max} for each instance, respectively.

Table 4 discloses that the average objective values obtained by different values of T_{max} are in most cases very close, indicating that the performance of the tabu search method is not sensitive to the setting of parameter T_{max}. Moreover, it can be seen that $T_{max} = 120$ yields a desirable result in most cases, which is the reason why the default value of T_{max} is set to 120 in this study.

4.3. Role of the memetic framework

As shown in Section 4.1, our tabu search procedure is very competitive compared to the existing algorithms in the literature. So it is interesting to know whether our MAMMDP algorithm has a significant improvement over this efficient TS procedure. For this purpose, we show a comparison between MAMMDP and a multi-start version of the tabu search procedure (MTS). For this experiment, we used 40 large instances with $n=3000$ or 5000, and run both MTS and MAMMDP 20 times to solve each instance under the time limits given in Section 3.2 (1000 s for $n=3000$ and 2000 s for $n=5000$). Notice that for MTS, the TS procedure was run in a multi-start way with a randomly generated initial solution for each re-start until the timeout limit was reached, the TS procedure being re-started once the depth of tabu search α (which is set to 5×10^4) is reached. The computational results of both algorithms are summarized in Table 5 which is composed of two parts, where

Table 4
Influence of the parameter T_{max} on the performance of tabu search procedure. Each instance was independently solved 20 times using the tabu search procedure for each parameter value in the range {60, 80, 100, 120, 140, 160, 180, 200}, and the average objective values (f_{avg}) over 20 runs are respectively reported.

Instance/T_{max}	f_{avg}							
	60	80	100	120	140	160	180	200
MDPI1_500	81.22	81.21	81.28	81.28	81.28	81.26	81.26	81.19
MDPI2_500	78.28	78.31	78.58	78.59	78.53	78.58	78.55	78.54
MDPI3_500	75.99	76.14	76.17	76.25	76.26	76.14	76.21	76.21
MDPI4_500	82.33	82.32	82.33	82.33	82.32	82.31	82.32	82.31
MDPI5_500	80.28	80.27	80.35	80.35	80.33	80.34	80.32	80.30
MDPII1_500	109.55	109.60	109.60	109.47	109.60	109.58	109.61	109.61
MDPII2_500	105.49	105.65	105.70	105.68	105.70	105.68	105.66	105.66
MDPII3_500	107.65	107.80	107.80	107.81	107.82	107.80	107.78	107.64
MDPII4_500	106.01	105.84	106.08	106.09	106.09	106.07	105.92	106.10
MDPII5_500	106.56	106.76	106.74	106.83	106.83	106.83	106.76	106.79
MDPI1_750	96.31	96.58	96.61	96.47	96.47	96.59	96.59	96.57
MDPI2_750	97.42	97.49	97.56	97.56	97.54	97.53	97.52	97.49
MDPI3_750	97.59	97.44	97.79	97.78	97.76	97.79	97.76	97.71
MDPI4_750	95.65	95.77	95.95	95.88	95.88	95.94	95.91	95.85
MDPI5_750	96.53	96.72	96.74	96.73	96.74	96.73	96.72	96.72
MDPII1_750	128.42	128.71	128.75	128.40	128.74	128.72	128.49	128.67
MDPII2_750	130.63	130.88	130.95	130.92	130.71	130.92	130.91	130.84
MDPII3_750	129.26	129.71	129.64	129.56	129.71	129.69	129.67	129.65
MDPII4_750	126.31	126.40	126.51	126.44	126.55	126.56	126.24	126.54
MDPII5_750	128.72	128.82	129.10	129.03	129.03	129.06	129.04	129.03
MDPI1_1000	118.81	119.01	119.15	118.84	119.12	119.15	119.15	119.12
MDPI2_1000	113.27	113.32	113.50	113.36	113.49	113.48	113.27	113.46
MDPI3_1000	114.58	114.99	115.10	115.11	115.10	115.05	114.96	114.99
MDPI4_1000	110.81	110.89	111.03	110.92	111.06	111.02	110.95	111.01
MDPI5_1000	111.48	111.82	112.62	112.72	112.71	112.57	112.25	112.61
MDPII1_1000	147.25	147.81	147.91	147.92	147.91	147.91	147.86	147.85
MDPII2_1000	150.80	151.12	151.36	151.34	151.31	151.31	151.28	151.22
MDPII3_1000	149.81	150.71	150.56	150.48	150.63	150.60	150.63	150.51
MDPII4_1000	148.84	149.06	149.14	149.09	149.10	149.03	149.05	149.00
MDPII5_1000	151.09	151.44	151.43	151.50	151.35	151.48	151.46	151.33
MDPI1_3000	186.63	188.07	188.51	188.67	188.69	188.82	188.90	188.70
MDPII1_3000	248.85	250.58	250.73	251.74	251.90	251.84	251.94	251.94
MDPI1_5000	236.42	237.83	237.86	238.74	239.33	238.94	239.49	239.44
MDPII1_5000	317.17	318.14	319.46	320.76	321.24	320.83	321.41	321.62
Average	128.41	128.74	128.90	128.96	129.02	129.00	129.00	129.01

Table 5
Comparison between the multi-start tabu search method (MTS) and the proposed memetic algorithm on the set of 40 large instances with $n \geq 3000$. Each instance was independently solved 20 times by both algorithms respectively, and better results between the two compared algorithms are indicated in bold in terms of the best and average objective values.

Instance	MAMMDP				MTS			
	f_{best}	f_{avg}	SR	$t(s)$	f_{best}	f_{avg}	SR	$t(s)$
MDPI1_3000	189.048965	189.048965	20/20	88.36	189.048965	189.048965	20/20	120.05
MDPI2_3000	187.387292	187.387292	20/20	60.71	187.387292	187.387292	20/20	101.07
MDPI3_3000	185.666806	**185.655084**	13/20	352.85	185.666806	185.651588	10/20	526.05
MDPI4_3000	186.163727	186.153631	16/20	300.37	186.163727	**186.163727**	20/20	136.64
MDPI5_3000	187.545515	187.545515	20/20	61.29	187.545515	187.545515	20/20	133.70
MDPI6_3000	189.431257	189.431257	20/20	51.99	189.431257	189.431257	20/20	35.59
MDPI7_3000	188.242583	188.242583	20/20	86.57	188.242583	188.242583	20/20	137.56
MDPI8_3000	186.796814	186.796814	20/20	48.04	186.796814	186.796814	20/20	66.76
MDPI9_3000	188.231264	188.231264	20/20	151.78	188.231264	188.231264	20/20	101.11
MDPI10_3000	185.682511	185.623778	10/20	228.72	185.682511	**185.672371**	18/20	352.70
MDPII1_3000	252.320433	252.320433	20/20	59.70	252.320433	252.320433	20/20	72.49
MDPII2_3000	250.062137	**250.062137**	20/20	220.10	250.062137	250.054744	7/20	513.80
MDPII3_3000	251.906270	251.906270	20/20	146.32	251.906270	251.906270	20/20	127.32
MDPII4_3000	253.941007	**253.940596**	19/20	370.76	253.941007	253.939680	18/20	352.64
MDPII5_3000	253.260423	**253.260350**	17/20	374.00	253.260423	253.260164	14/20	349.19
MDPII6_3000	250.677750	250.677750	20/20	55.35	250.677750	250.677750	20/20	69.78
MDPII7_3000	251.134413	251.134413	20/20	74.72	251.134413	251.134413	20/20	97.74
MDPII8_3000	252.999648	252.999648	20/20	79.82	252.999648	252.999648	20/20	115.84
MDPII9_3000	252.425770	252.425770	20/20	90.27	252.425770	252.425770	20/20	106.79
MDPII10_3000	252.396590	252.396590	20/20	13.18	252.396590	252.396590	20/20	16.06
#Better	0	4	4	14	0	2	2	6
#Equal	20	14	14	0	20	14	14	0
#Worse	0	2	2	6	0	4	4	14
p-value	1.0	4.142e−1						
MDPI1_5000	**240.162535**	**240.102875**	7/20	312.13	240.141212	240.021201	2/20	163.67
MDPI2_5000	**241.827401**	**241.792978**	6/20	1244.36	241.817543	241.753546	2/20	734.33
MDPI3_5000	240.890819	**240.888162**	19/20	810.48	240.890819	240.825167	4/20	531.94
MDPI4_5000	**240.997186**	**240.976789**	6/20	653.64	240.973489	240.915459	3/20	560.43
MDPI5_5000	242.480129	**242.475885**	19/20	735.16	242.480129	242.430474	5/20	515.62
MDPI6_5000	240.322850	**240.306326**	8/20	976.02	**240.328684**	240.266264	5/20	290.72
MDPI7_5000	242.814943	**242.774982**	5/20	259.50	**242.820139**	242.759895	3/20	819.01
MDPI8_5000	**241.194990**	**241.161763**	8/20	1148.60	241.144781	241.113453	3/20	721.59
MDPI9_5000	239.760560	**239.667613**	4/20	1219.71	239.760560	239.514958	4/20	360.05
MDPI10_5000	**243.473734**	**243.373015**	4/20	457.28	243.385487	243.348149	9/20	939.59
MDPII1_5000	**322.235897**	**322.181291**	5/20	1519.05	322.223220	322.131204	3/20	197.42
MDPII2_5000	327.301910	327.006342	5/20	1103.13	327.301910	**327.075247**	5/20	18.52
MDPII3_5000	**324.813456**	**324.801590**	10/20	955.81	324.810826	324.790223	4/20	27.82
MDPII4_5000	**322.237586**	**322.197276**	5/20	664.10	322.212289	322.126605	4/20	338.02
MDPII5_5000	**322.491211**	**322.380726**	7/20	1014.90	322.420806	322.301249	5/20	105.12
MDPII6_5000	322.950488	**322.703887**	4/20	352.88	322.950488	322.615227	5/20	212.55
MDPII7_5000	322.850438	**322.793125**	10/20	714.31	322.850438	322.778396	8/20	756.14
MDPII8_5000	**323.112120**	**323.053268**	11/20	879.48	323.033840	322.873156	5/20	161.94
MDPII9_5000	**323.543775**	**323.339842**	7/20	569.73	323.522709	323.278556	3/20	148.94
MDPII10_5000	324.519908	**324.414458**	15/20	752.95	324.519908	324.294790	10/20	753.14
#Better	11	19			2	1		
#Equal	7	0			7	0		
#Worse	2	1			11	19		
p-value	1.26e−2	5.699e−5						

the rows *Better, Equal, Worse* respectively indicate the number of instances for which the result of an algorithm is better, equal to and worse than that of another one, and other symbols have same meanings as those of Table 1. Moreover, to verify whether there exists a significant difference between the MAMMDP and MTS algorithms in terms of f_{best} and f_{avg}, the p-values from the non-parametric Friedman tests are also reported in the table.

Table 5 discloses that for the 20 instances with $n=3000$ the MAMMDP algorithm performs slightly better than the MTS algorithm, but the difference is small. However, for the 20 larger instances with $n=5000$, the MAMMDP algorithm significantly outperforms the MTS algorithm. First, compared with the MTS algorithm, the MAMMDP algorithm obtains better and worse results in terms of the best objective value on 11 and 2 instances respectively. Second, in terms of the average objective value, the MAMMDP algorithm yields better results on 19 out of 20 instances. In addition, from the Friedman tests, it can be seen that the obtained p-values are 1.26e−2 (<0.05) and 5.699e−5 (<0.05) respectively for the best and average objective values, implying there exists a significant difference between these two methods. These outcomes indicate that although the memetic part of the proposed MAMMDP algorithm is not so critical for small and easy instances (i.e., local optimization with tabu search equipped with the fast one-flip neighborhood suffices), it is quite useful to better solve large and difficult instances.

5. Conclusions

In this paper, we propose the first population-based memetic algorithm (MAMMDP) for solving the NP-hard max-mean dispersion problem (MaxMeanDP). MAMMDP integrates an effective

tabu search procedure and a random crossover operator while adopting an original scheme for parent selection. The computational results on a large number of 200 benchmark instances show that the proposed algorithm is very competitive compared with the state-of-the-art algorithms in the literature. Specifically, it improves or matches the previous best known results for all tested instances with $n \leq 1000$ with an average computing time of less than 14 s and a success rate of 100%, with only one exception. In particular, we found improved best results (new lower bounds) for 53 out of the 60 most challenging instances. We also show computational results on 40 large instances with 3000 or 5000 elements which can serve as reference lower bounds for evaluating new MaxMeanDP algorithms.

The investigations of several essential components of the proposed algorithm shed light on the following points. First, the high performance of the proposed algorithm is largely attributed to the fast liner-time neighborhood induced by the one-flip operator. Second, the adopted technique for tuning the tabu list is robust and is not so sensitive to its parameter T_{max}. Third, the population-based memetic framework is particularly suitable to solve large and difficult problem instances.

The proposed algorithm could be adapted to the weighted version of the max-mean dispersion problem with several small modifications. Some ideas of the proposed algorithm could be applied to other binary optimization problems (including some dispersion problems) where no constraint is imposed on the number of variables taking the value of one.

Acknowledgements

We are grateful to the anonymous referees for their valuable comments and suggestions. Our special thanks to Dr. R. Martí, Dr. M. Gallego, and Dr. A. Duarte for providing us with the executable code of their TS algorithm [4]. This work was partially supported by a post-doc grant (for X.J. Lai) from the Region of Pays de la Loire (France) and the PGMO project (2013–2015, Jacques Hadamard Mathematical Foundation, Paris).

References

[1] Aringhieri R, Cordone R. Comparing local search metaheuristics for the maximum diversity problem. J Oper Res Soc 2011;62:266–80.
[2] Aringhieri R, Cordone R, Grosso A. Construction and improvement algorithms for dispersion problems. Eur J Oper Res 2015;242(1):21–33.
[3] Aringhieri R, Cordone R, Melzani Y. Tabu search versus GRASP for the maximum diversity problem. 4OR: A Q J Oper Res 2008;6(1):45–60.
[4] Carrasco R, Pham A, Gallego M, Gortázar F, Martí R, Duarte A. Tabu search for the max-mean dispersion problem. Knowl Based Syst 2015;85:256–64.
[5] Della Croce F, Garraffa M, Salassa F. A hybrid heuristic approach based on a quadratic knapsack formulation for the max-mean dispersion problem. Lecture Notes in Computer Science; 2014. p. 186–97.
[6] DellaCroce F, Garraffa M, Salassa F. A hybrid three-phase approach for the max-mean dispersion problem. Comput Oper Res 2016;71:16–22.
[7] Della Croce F, Grosso A, Locatelli M. A heuristic approach for the max-min diversity problem based on max-clique. Comput Oper Res 2009;36(8):2429–2433.
[8] Duarte A, Martí R. Tabu search and grasp for the maximum diversity problem. Eur J Oper Res 2007;178(1):71–84.
[9] Duarte A, Sánchez-Oro J, Resende MGC, Glover F, Martí R. Greedy randomized search procedure with exterior path relinking for differential dispersion minimization. Inf Sci 2015;296:46–60.
[10] Galinier P, Boujbel Z, Fernandes MC. An efficient memetic algorithm for the graph partitioning problem. Ann Oper Res 2011;191(1):1–22.
[11] Glover F, Laguna M. Tabu search. Boston: Kluwer Academic Publishers; 1997.
[12] Glover F, Laguna M, Martí R. Fundamentals of scatter search and path relinking. Control Cybern 2000;39:653–84.
[13] Glover F, Kuo CC, Dhir KS. Heuristic algorithms for the maximum diversity problem. J Inf Optim Sci 1998;19(1):109–32.
[14] Hao JK. Memetic algorithms in discrete optimization. In: Neri F, Cotta C, Moscato P, editors. Handbook of memetic algorithms. Studies in computational intelligence, vol. 379. Springer; 2012. p. 73–94 Chapter 6.
[15] Kerchove C, Dooren PV. The page trust algorithm: how to rank web pages when negative links are allowed? Proceedings SIAM international conference on data mining; 2008. p. 346–52.
[16] Martí R, Gallego M, Duarte A. A branch and bound algorithm for the maximum diversity problem. Eur J Oper Res 2010;200(1):36–44.
[17] Martí R, Sandoya F. GRASP and path relinking for the equitable dispersion problem. Comput Oper Res 2013;40(12):3091–9.
[18] Moscato P, Cotta C. A gentle introduction to memetic algorithms. In: Glover F, Kochenberger G, editors. Handbook of metaheuristics. Norwell, Massachusetts, USA: Kluwer; 2003.
[19] Neri F, Cotta C, Moscato P, editors. Handbook of memetic algorithms. Studies in computational intelligence, vol. 379. Springer; 2011.
[20] Palubeckis G. Iterated tabu search for the maximum diversity problem. Appl Math Comput 2007;189(1):371–83.
[21] Porumbel DC, Hao JK, Glover F. A simple and effective algorithm for the MaxMin diversity problem. Ann Oper Res 2011;186(1):275–93.
[22] Prokopyev OA, Kong N, Martinez-Torres DL. The equitable dispersion problem. Eur J Oper Res 2009;197(1):59–67.
[23] Resende MGC, Martí R, Gallego M, Duarte A. GRASP and path relinking for the max-min diversity problem. Comput Oper Res 2010;37(3):498–508.
[24] Saboonchi B, Hansen P, Perron S. MaxMinMin p-dispersion problem: A variable neighborhood search approach. Comput Oper Res 2014;52(Part B):251–9.
[25] Wu QH, Hao JK. A hybrid metaheuristic method for the maximum diversity problem. Eur J Oper Res 2013;231(2):452–64.
[26] Yang B, Cheung W, Liu J. Community mining from signed social networks. IEEE Trans Knowl Data Eng 2007;19(10):1333–48.