Diversification-driven Memetic Algorithm for the Maximum Diversity Problem

Xiangjing Lai[1], JinKao Hao[2,3], Dong Yue[1], Hao Gao[1]

[1]Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[2]LERIA, University of Angers, 2 bd Lavoisier, 49045 Angers, France
[3]Institut Universitaire de France, 1 Rue Descartes, 75231 Paris, France

laixiangjing@gmail.com (X. Lai), jin-kao.hao@univ-angers.fr (J. K. Hao), medongy@vip.163.com (D. Yue)

Abstract: The maximum diversity problem (MDP) is a classic NP-hard optimization problem with a number of applications. We propose in this work an effective hybrid evolutionary algorithm for MDP called the diversification-driven memetic algorithm by introducing a diversification mechanism into an existing memetic algorithm. Computational results on 20 representative benchmark instances show that the proposed algorithm outperforms the state-of-the-art MDP algorithms in the literature, indicating the interest of the diversification mechanism within the proposed memetic algorithm.

Keywords: Maximum diversity problem; tabu search; memetic algorithm; diversification.

1 Introduction

Given a set $N = \{1,2,\cdots, n\}$ of n elements, the distances $d_{ij}(i < j)$ between elements, and a positive integer m, the maximum diversity problem (MDP) aims at selecting a subset M of N with m elements such that the sum of distances between the selected elements is maximized.

Formally, MDP can be written as follows [1].

$$Max\ f(x) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_i x_j \quad (1)$$

$$s.t.\ \sum_{i=1}^{n} x_i = m \quad (2)$$

where $x = (x_1, x_2, \cdots, x_n)$, and the binary variable $x_i = 1$ if the element i is selected, and 0 otherwise.

Despite its simplicity in form, MDP is proved to be NP-hard, and thus computationally challenging [2]. Given the relevance of the problem, many solution approaches have been proposed in the literature, including both exact algorithms [3] and heuristic algorithms. Moreover, most existing algorithms in the literature belong to the category of heuristic algorithms, such as tabu search algorithms [4-6], iterated greedy algorithm [7], memetic algorithms [8-9], and some other algorithms [10-12].

Previous studies show that memetic algorithms are a very useful framework for solving many optimization problems [13,14] and the diversity management of the population of a memetic algorithm is crucial to ensure its effectiveness [9,15-17]. Inspired by these studies, we propose in this work a diversification-driven memetic algorithm (DMAMDP) for MDP by integrating an additional diversification mechanism with a previous memetic algorithm [8]. Computational results on a set of representative MDP benchmark instances show the proposed algorithm competes very favorably with the state-of-the-art MDP algorithms in the literature.

The rest of paper is organized as follows. In Section 2, we describe in detail the proposed DMAMDP algorithm. In Section 3, we assess the proposed algorithm by providing computational results and making a comparison between the DMAMDP algorithm and two best performing algorithms in the literature. In the last section, we summarize the present work and provide some possible research directions.

2 Method

2.1 Search Space, Evaluation Function, and Solution Representation

Given an instance of MDP, a solution of the problem can be indicated by a subset M of N with m elements. Thus, the search space Ω explored by the algorithm is composed of all subsets M with m elements, i.e.,

$$\Omega = \{M : M \subset N, |M| = m\} \quad (3)$$

Clearly, the size of Ω equals to C_n^m. In addition, for any candidate solution in the search space, its quality can be directly evaluated by its objective function value by Eq. (1).

2.2 General Procedure of the DMAMDP

The algorithm proposed in this work can be regarded as

978-1-5386-6005-8/18/$31.00 ©2018 IEEE

an improved version of the state-of-the-art MAMDP algorithm [8], and its pseudo-code is given in Algorithm 1.

Algorithm 1 Diversification-driven memetic algorithm for the maximum diversity problem

1: **Input**: The instance I, the population size np, the time limit t_{max}, the parameter p.
2: **Output**: the best solution s^* found
3: **for** $i \leftarrow 1$ to np **do**
4: generate randomly the initial population $POP = \{s^1, s^2, \ldots, s^{np}\}$
5: $s^i \leftarrow TabuSearch(s^i)$
6: **end for**
7: $s^* \leftarrow arg\ max\{f(s^i) : i = 1, \ldots, np\}$
8: **while** $time() < t_{max}$ **do**
9: $r \leftarrow rand[0,1]$ /* $rand[0,1]$ denotes a random number in [0,1] */
10: **if** $r > p$ **then**
11: Randomly select two solutions s^a and s^b from POP
12: $s^o \leftarrow Crossover(s^a, s^b)$ /* Algorithm 3 */
13: $s^o \leftarrow TabuSearch(s^o)$ /* Algorithm 2 */
14: **else**
15: Randomly generate an initial solution s /* Increase the diversity of population */
16: $s \leftarrow TabuSearch(s)$ /* Algorithm 2 */
17: **if** $f(s) > f(s^*)$ **then**
18: $s^* \leftarrow s$
19: **end if**
20: $POP \leftarrow PoolUpdating(POP, s)$
21: randomly select a solution s' from POP
22: $s^o \leftarrow Crossover(s', s)$ /* Algorithm 3 */
23: $s^o \leftarrow TabuSearch(s^o)$ /* Algorithm 2 */
24: **end if**
25: **if** $f(s^o) > f(s^*)$ **then**
26: $s^* \leftarrow s^o$
27: **end if**
28: $POP \leftarrow PoolUpdating(POP, s^o)$ /* Algorithm 4 */
29: **end while**

The DMAMDP algorithm starts from an initial population composed of np individuals that are generated by selecting uniformly and randomly m elements from N and then improved by the tabu search procedure described in Section 2.3 (lines 3 to 6). After that, the algorithm performs a "while" loop until the time limit ($t_{\max}$) is reached (lines 8 to 29).

At each iteration of the "while" loop, the algorithm first generate a random number r in the interval [0,1], then performs the corresponding operations according to one of the following two situations: (1) $r > p$ and (2) $r \leq p$, where p is a predetermined parameter (line 9). If $r > p$, the algorithm randomly selects two parent solutions s^i and s^j from the population and applies the crossover operator (Section 2.4) to them to generate an offspring solution s^o, then the generated offspring solution s^o is further improved by the tabu search procedure (lines 11 to 13). On the contrary, if $r \leq p$, the algorithm randomly generates an initial solution s and then improves it by the tabu search procedure (lines 15 to 16). Subsequently, the resulting solution is combined with a parent solution s' randomly selected from the population to generate a new offspring solution s^o (line 22). Subsequently, the resulting solution s^o is improved by the tabu search procedure (line 23).

Note that in both cases, the new improved solution from the tabu search procedure is used to update the population (lines 20 and 28).

2.3 Tabu Search

Algorithm 2 $TabuSearch$

1: **Input**: Input solution s_0, the tabu depth α
2: **Output**: The best solution s_b found
3: $s \leftarrow s_0$ /* s denotes the current solution */
4: $s_b \leftarrow s$ /* s_b denotes the best solution found so far */
5: $NoImprove \leftarrow 0$ /* $NoImprove$ denotes the maximum number of iterations during which s_b has not been improved */
6: **repeat**
7: Choose randomly a best eligible neighbor solution s' from the current neighborhood $N(s)$
/* s' is identified to be eligible if it is not forbidden by the tabu list or $f(s') > f(s_b)$ */
8: $s \leftarrow s'$
9: Update tabu list with s
10: **if** $f(s) > f(s_b)$ **then**
11: $s_b \leftarrow s$, $NoImprove \leftarrow 0$
12: **else**
13: $NoImprove \leftarrow NoImprove + 1$
14: **end if**
15: **until** $NoImprove = \alpha$
16: **return** s_b

Given an input solution and a neighborhood structure, the tabu search method performs a number of iterations to improve the input solution until a stopping condition is satisfied. Specifically, at each iteration, the tabu search method selects a best eligible neighbor solution from the current neighborhood to replace the current solution, where a neighbor solution is regarded to be eligible if it is not forbidden by the tabu list or it is better than the best solution found so far in terms of the fitness value. We provide the general procedure of the tabu search method in Algorithm 2, where α is a parameter which is called the depth of tabu search.

Our tabu search method adopts the same neighborhood structure and the tabu list management strategy as the OBMA algorithm in [9]. The interested readers are referred to [9] for more details.

2.4 Crossover Operator

Given two solutions M_1 and M_2, the DMAMDP algorithm employs the backbone-based crossover operator [8,9] to generate the offspring solution, and the

pseudo-code of the crossover operator is given in Algorithm 3. The previous studies show that this crossover operator is very suitable for MDP [8,9].

Algorithm 3 Crossover operator

Input: Input solutions M_1 and M_2
Output: An offspring solution M_o
$M_o \leftarrow M_1 \cap M_2$
while $|M_o| < m$ **do**
  $i^* \leftarrow argmax\{p(i) : i \in M_1 \setminus M_o, p(i) = \sum_{j \in M_o} d_{ij}\}$
  $M_o \leftarrow M_o \cup \{i^*\}$
  if $|M_o| = m$ **then**
    break
  end if
  $i^* \leftarrow argmax\{p(i) : i \in M_2 \setminus M_o, p(i) = \sum_{j \in M_o} d_{ij}\}$
  $M_o \leftarrow M_o \cup \{i^*\}$
end while

2.5 Population Updating Rule

Algorithm 4 PoolUpdating

Input: Input population $POP = \{s^1, s^2, \ldots, s^{np}\}$, offspring solution s^o, parameter λ
Output: An updated population POP
$POP \leftarrow POP \cup \{s^o\}$
$d_{min} \leftarrow min\{distance(s^i, s^j) : 1 \le i < j \le np+1\}$
$d_{max} \leftarrow max\{distance(s^i, s^j) : 1 \le i < j \le np+1\}$
$f_{min} \leftarrow min\{f(s^i) : 1 \le i \le np+1\}$
$f_{max} \leftarrow max\{f(s^i) : 1 \le i \le np+1\}$
for $i \leftarrow 1$ to $np+1$ **do**
  $score(s^i) \leftarrow \lambda \times \frac{f(s^i)-f_{min}}{f_{max}-f_{min}+0.1} + (1-\lambda) \times \frac{d(s^i,POP)-d_{min}}{d_{max}-d_{min}+0.1}$
  /* $d(s^i, POP) = min\{distance(s^i, s^j) : j \ne i, s^j \in POP\}$, and $distance(s^i, s^j)$ represents the distance between s^i and s^j */
  $s^{\#} \leftarrow argmin\{score(s^i) : s^i \in POP\}$
  $POP \leftarrow POP \setminus \{s^{\#}\}$
end for

In our DMAMDP algorithm, we use the distance-based population updating strategy whose pseudo-code is shown in Algorithm 4. Given the current population POP and an offspring s^o, the population is updated as follows. First, the offspring s^o is added into the population POP (line 3). Then, each individual in the new population is scored by a scoring function which is defined as follows:

$$score(s^i) = \lambda \times \frac{f(s^i) - f_{\min}}{f_{\max} - f_{\min} + 0.01} + (1-\lambda) \times \frac{d(s^i, POP) - d_{\min}}{d_{\max} - d_{\min} + 0.01}$$

where $f_{\min}$ and $f_{\max}$ represent the minimum and maximum objective values for the individuals in the population, and the $d_{\min}$ and $d_{\max}$ represent the minimum and maximum distances between an individual and other individuals of the population, $d(s^i, POP)$ represent the distance between the solution s^i and the population POP, and λ is a predetermined parameter. Finally, the individual having the minimum score is removed from the population.

3 Computational Experiments

In this section, we assess our DMAMDP algorithm by performing a computational experiment and making a fair comparison between our DMAMDP algorithm and two best performing algorithms in the literature.

3.1 Benchmark Instances

In our experiments, we used a popular benchmark set (denoted by MDG-b) which is composed of 20 large-scale benchmark instances with n=2000 and m=200 as our test bed. For each instance tested, the distances between elements are a real number that is randomly generated in the interval [0,1000] by a uniform distribution. Note that these instances were widely used to assess the performance of the MDP algorithms in the literature [8,9]; and they are online available at http://www/optsocom.es/mdp/.

3.2 Parameter Settings and Experimental Protocol

Our DMAMDP algorithm adopts four main parameters, including the size of population (*np*), the depth of tabu search (α), the coefficient λ used in the population updating rule, and the parameter p, whose values were empirically set to 10, 10000, 0.7, 0.5, respectively.

DMAMDP was implemented in C++ and compiled by g++ complier with -O3 option. All the experiments and comparisons were based on the same computing platform, i.e., a cluster node with an Intel E5-2670 processor and 2 GB RAM, running Linux operating system. In addition, due to the stochastic behavior of the algorithms, each instance was independently solved 15 times by our DMAMDP algorithm, where the time limit was set to 600 seconds for each run, as in [9].

3.3 Computational Results and Comparison

The goal of this section is to compare the DMAMDP algorithm with two state-of-the-art MDP algorithms in the literature, i.e. the MAMDP algorithm [8] and the opposition-based memetic algorithm (OBMA) [9]. The computational experiments were carried out according to the experimental protocol in Section 3.2, and the experimental results are summarized in Tables 1 and 2, along with those of two reference algorithms.

In Table 1, the first two columns give the name of instances and the best-known results in the literature (BKR) that are compiled according to the results reported in [9]. Columns 3 and 4 show the results of MAMDP [8], including the gap (Δf_{best}) between the best results obtained in the experiment and BKR, i.e., $\Delta f_{best} = f_{best}$ – BKR, and the gap (Δf_{avg}) between the average results obtained and BKR, i.e., $\Delta f_{avg} = f_{avg}$ – BKR, where f_{best} and f_{avg} denotes respectively the best and average objective values obtained over 15 runs. The results of our DMAMDP algorithm are summarized in the last two columns. The last three rows indicate the

number of cases where an algorithm performs better, equally and worse compared to the other algorithm.

Table 1 Comparison between our DMAMDP algorithm and MAMDP in [8]. Better results between the compared algorithms are indicated in bold

Instance	BKR	MAMDP [8]		DMAMDP (this work)	
		Δf_{best}	Δf_{avg}	Δf_{best}	Δf_{avg}
MDG-b_21	11299894.8	0.0	-225.6	0.0	**0.0**
MDG-b_22	11292398.2	0.0	-2150.1	0.0	**0.0**
MDG-b_23	11299940.5	0.0	0.0	0.0	0.0
MDG-b_24	11291119.5	0.0	-471.5	0.0	**0.0**
MDG-b_25	11298026.9	0.0	-71.0	0.0	**0.0**
MDG-b_26	11298430.4	0.0	-3603.8	0.0	**0.0**
MDG-b_27	11305676.8	0.0	0.0	0.0	0.0
MDG-b_28	11282910.9	-0.9	-360.3	**0.0**	**0.0**
MDG-b_29	11297339.5	0.0	-603.3	0.0	**0.0**
MDG-b_30	11298064.6	0.0	0.0	0.0	0.0
MDG-b_31	11288901.2	0.0	-375.9	0.0	**0.0**
MDG-b_32	11283539.3	0.0	-87.0	0.0	**0.0**
MDG-b_33	11298038.1	0.0	-861.4	0.0	**0.0**
MDG-b_34	11290482.6	0.0	-1046.3	0.0	**0.0**
MDG-b_35	11307424.1	0.0	0.0	0.0	0.0
MDG-b_36	11302892.0	0.0	-2630.5	0.0	**0.0**
MDG-b_37	11295773.8	0.0	-856.7	0.0	**0.0**
MDG-b_38	11296535.5	0.0	-2068.5	0.0	**0.0**
MDG-b_39	11295054.2	0.0	-472.6	0.0	**0.0**
MDG-b_40	11309162.6	0.0	-1540.1	0.0	**0.0**
#Better		0	0	1	16
#Equal		19	4	19	4
#Worse		1	16	0	0

Table 1 shows that our DMAMDP algorithm outperforms significantly the MAMDP algorithm for the tested instances. In terms of Δf_{best}, our DMAMDP algorithm obtained a better result on 1 instance, while matching the result of MAMDP for the remaining instances. Moreover, in terms of Δf_{avg}, DMAMDP obtained a better result for 16 out of 20 instances than MAMDP in [8,9], which clearly shows the superiority of our DMAMDP algorithm.

To further compare the DMAMDP algorithm with the best performing algorithm in the literature, i.e., the OBMA algorithm [9], we show the results of both algorithms in Table 2, including Δf_{best}, Δf_{avg}, the standard deviation of objective values obtained (Std), the average run time (t_{avg}) in seconds to reach its final objective value, and the success rate (*SR*) to hit the best known result (BKR).

From Table 2, we observe that DMAMDP outperforms also the OBMA algorithm. Specifically, our DMAMDP algorithm reached the best-known result for each instance with a success rate of 100%, while the OBMA algorithm obtained a worse result in terms of Δf_{avg}, Std, and *SR* on 4 instances, respectively.

In summary, these outcomes indicate that the proposed DMAMDP algorithm has a better performance than the state-of-the-art MDP algorithms in the literature for the tested instances.

Table 2 Computational comparison of the DMAMDP algorithm with the best performing algorithm in the literature (i.e., OBMA) [9]. The better results between two compared algorithms are indicated in bold in terms of Δf_{avg}, Std, and *SR*

Instance	BKR	OBMA [9]					DMAMDP (this work)				
		Δf_{best}	Δf_{avg}	Std	t_{avg}	*SR*	Δf_{best}	Δf_{avg}	Std	t_{avg}	*SR*
MDG-b_21	11299894.8	0.0	0.0	0.0	325.1	15/15	0.0	0.0	0.0	344.4	15/15
MDG-b_22	11292398.2	0.0	0.0	0.0	380.8	15/15	0.0	0.0	0.0	362.2	15/15
MDG-b_23	11299940.5	0.0	0.0	0.0	283.4	15/15	0.0	0.0	0.0	233.7	15/15
MDG-b_24	11291119.5	0.0	-25	67.2	323.4	13/15	0.0	**0.0**	**0.0**	296.8	**15/15**
MDG-b_25	11298026.9	0.0	0.0	0.0	313.9	15/15	0.0	0.0	0.0	284.3	15/15
MDG-b_26	11298430.4	0.0	0.0	0.0	336.0	15/15	0.0	0.0	0.0	280.1	15/15
MDG-b_27	11305676.8	0.0	0.0	0.0	256.0	15/15	0.0	0.0	0.0	285.8	15/15
MDG-b_28	11282910.9	0.0	-0.2	0.4	351.1	12/15	0.0	**0.0**	**0.0**	365.0	**15/15**
MDG-b_29	11297339.5	0.0	0.0	0.0	288.3	15/15	0.0	0.0	0.0	347.3	15/15
MDG-b_30	11298064.6	0.0	0.0	0.0	274.9	15/15	0.0	0.0	0.0	312.1	15/15
MDG-b_31	11288901.2	0.0	0.0	0.0	308.0	15/15	0.0	0.0	0.0	330.0	15/15
MDG-b_32	11283539.3	0.0	-25.0	30.6	283.0	9/15	0.0	**0.0**	**0.0**	289.8	**15/15**
MDG-b_33	11298038.1	0.0	-65.1	166.1	277.5	13/15	0.0	**0.0**	**0.0**	224.2	**15/15**
MDG-b_34	11290482.6	0.0	0.0	0.0	355.4	15/15	0.0	0.0	0.0	300.3	15/15
MDG-b_35	11307424.1	0.0	0.0	0.0	331.0	15/15	0.0	0.0	0.0	303.6	15/15
MDG-b_36	11302892.0	0.0	0.0	0.0	329.4	15/15	0.0	0.0	0.0	366.2	15/15
MDG-b_37	11295773.8	0.0	0.0	0.0	291.2	15/15	0.0	0.0	0.0	216.5	15/15
MDG-b_38	11296535.5	0.0	0.0	0.0	297.6	15/15	0.0	0.0	0.0	311.5	15/15
MDG-b_39	11295054.2	0.0	0.0	0.0	289.7	15/15	0.0	0.0	0.0	297.0	15/15
MDG-b_40	11309162.6	0.0	0.0	0.0	266.5	15/15	0.0	0.0	0.0	369.1	15/15
#Better		0	0	0		0	0	4	4		4
#Equal		20	16	16		16	20	16	16		16
#Worse		0	4	4		4	0	0	0		0

4 Conclusions

In this paper, we proposed a diversification-driven memetic algorithm (denoted by DMAMDP) for solving the classic maximum diversity problem by integrating a diversification mechanism within a memetic algorithm. Experimental results on 20 large benchmark instances show that the proposed algorithm outperforms the best performing algorithms in the literature.

The present study further confirms that diversification is a crucial factor affecting the performance of a memetic algorithm. Given that the idea of our diversification mechanism, i.e., adding new local optimum solutions that are obtained by local search from random initial solutions into the population during the search process, is very general, it would be interesting to check its usefulness within memetic algorithms for other combinatorial search problems.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 61703213), the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20170904), and Nanjing University of Posts and Telecommunications Science Foundation (NUPTSF) (Grant no. NY217154).

References

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

[2] Ghosh J.B., Computational aspects of the maximum diversity problem. Operations Research Letters, 1996, 19(4), 175-181.

[3] MartíR., Gallego M., Duarte A., A branch and bound algorithm for the maximum diversity problem. European Journal of Operational Research, 2010, 200(1), 36-44.

[4] Duarte A., MartíR., Tabu search and GRASP for the maximum diversity problem. European Journal of Operational Research, 2007, 178(1), 71-84.

[5] Aringhieri R., Cordone R., Melzani Y., Tabu search versus GRASP for the maximum diversity problem. A Quarterly Journal of Operations Research, 2008, 6(1), 45-60.

[6] Palubeckis G., Iterated tabu search for the maximum diversity problem. Applied Mathematics and Computation, 2007, 189(1), 371-383.

[7] Lozano M., Molina D., García-Martínez C., Iterated greedy for the maximum diversity problem. European Journal of Operational Research, 2011, 214(1), 31-38.

[8] Wu Q., Hao J.K., A hybrid metaheuristic method for the maximum diversity problem, European Journal of Operational Research, 2013, 231(2), 452-464.

[9] Zhou Y., Hao J.K., Duval B., Opposition-based memetic search for the maximum diversity problem. IEEE Transactions on Evolutionary Computation, 2017, 21(5), 731-745.

[10] Martí R., Gallego M., Duarte A., Pardo E.G., Heuristics and metaheuristics for the maximum diversity problem. Journal of Heuristics, 2013, 19(4), 591-615.

[11] Aringhieri R., Cordone R., Comparing local search metaheuristics for the maximum diversity problem. Journal of the Operational Research Society, 2011, 62, 266-280.

[12] Glover F., Kuo C.C., Dhir K.S., 1998, Heuristic algorithms for the maximum diversity problem. Journal of Information and Optimization Sciences, 19(1), 109-132.

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

[14] Ong Y.S., Lim M. H., Chen X., Memetic computation-past, present& future. IEEE Computational Intelligence Magazine, 2010, 5(2): 24-31.

[15] Segura C., Aguirre A., Luna F., Alba E., Improving diversity in evolutionary algorithms: new best solutions for frequency assignment. IEEE Transactions on Evolutionary Computation, 2017, 21(4), 539-553.

[16] Segura C., Coello Coello C., Segredo E., Aguirre A., A novel diversity-based replacement strategy for evolutionary algorithms. IEEE Transactions on Cybernetics, 2016, 46(12), 3233-3246.

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