International Journal of Innovative Computing, Information and Control
Volume **13**, Number **5**, October **2017**

ICIC International ©2017 ISSN 1349-4198
pp. 1769–1776

SOLVING THE MAXIMUM DIVERSITY PROBLEM USING SIMULATED ANNEALING BASED EVOLUTIONARY ALGORITHM

Geng Lin
Department of Mathematics
Minjiang University
No. 200, Xiyuangong Road, Fuzhou 350108, P. R. China
lingeng413@163.com

Received December 2016; revised April 2017

Abstract. *The maximum diversity problem is an NP-hard combinatorial optimization problem with lots of applications. In this paper, we propose a simulated annealing based evolutionary algorithm for the maximum diversity problem, which integrates a crossover operator, a simulated annealing based local search procedure, and a population updating procedure. These strategies achieve a good compromise between intensification and diversification in the search process. The proposed algorithm is tested on two sets of benchmark instances from the literature. Experimental results and comparisons show that the proposed algorithm is efficient. Finaly, an experiment is done to disclose the benefit of integrating evolutionary strategies and simulated annealing.*

Keywords: Maximum diversity problem, Simulated annealing, Evolutionary algorithm, Combinatorial optimization

1. Introduction.

The maximum diversity problem (MDP) consists in selecting a subset of m elements from a set of n elements in such a way that the sum of the distances between the chosen elements is maximized. More formally, let $N = \{1, \ldots, n\}$ be a set of elements and d_{ij} be the distance between elements i and j. Let $x_i = 1$ if element i is selected, $x_i = 0$ otherwise. The MDP can be formulated as follows [1]:

$$(MDP)\begin{cases} \max & f(x) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_ix_j, \\ \text{s.t.} & \sum_{i=1}^{n} x_i = m, \\ & x_i \in \{0,1\}, \quad \text{for } i \in \{1,\ldots,n\}. \end{cases}$$

The MDP is known to be NP-hard [1], and has applications in several fields, including location, medical treatment, genetics, and so on. Since the wide practical applications of the MDP, it has received a great deal of attention in the past two decades. Several exact solution approaches to MDP have been proposed, but since the problem is NP-hard, the practical usefulness of these approaches is limited to fairly small problem instances. Metaheuristic algorithms have a crucial role for the solution of large scale instances in acceptable computing times. Over the last decade, various metaheuristic algorithms have been proposed to solve the MDP. This includes GRASP [2, 3], variable neighborhood search [4], scatter search [4], iterated greedy algorithm [5], tabu search [6, 7], hybridizing hopfield network and variable neighborhood search [8]. A comprehensive review concerning the MDP can be found in [9].

Simulated annealing (SA) [10] is a stochastic heuristic algorithm. Because of its ease of use, SA has been applied successfully to a number of optimization problems [11, 12]. However, SA has two drawbacks [13]: being trapped by local minima or taking too long to find a reasonable solution. Hybridizing SA with other heuristics such as the genetic algorithms is one of efficient ways to overcome these drawbacks. Several versions of SA based genetic algorithms [14, 15, 16] have been proposed for solving hard optimization problems. In this work, based on the structure of the MDP, we propose a simulated annealing based evolutionary algorithm (SAEA) for solving the MDP. First, SAEA employs an SA based local search procedure, which is based on the characteristic of MDP, as an intensification strategy. The SA based local search procedure selects two elements from two restricted candidate lists instead of the whole set to speed up the search. Second, to diversify the search, the SA based local search procedure randomly selects two elements to swap, and a distance based population updating procedure is used. These strategies prevent the algorithm from premature convergence. Two sets of 28 benchmark instances from the literature are used to test the proposed SAEA. Compared with some existing heuristics, the SAEA is able to find high quality solutions. Furthermore, the SAEA can find the best known solutions on 17 out of 28 tested instances. In addition, we also analyze the influence of the evolutionary strategies (crossover operator and population updating procedure).

This paper is organized as follows. Section 2 describes the proposed SAEA to the MDP. Experimental results and comparisons are provided in Section 3. Finally, we give our conclusions in Section 4.

2. The Proposed Algorithm.

In this section, a simulated annealing based evolutionary algorithm (SAEA) is presented. Firstly, we give the general scheme of the SAEA for MDP. Then, the main components of SAEA are described.

2.1. General scheme.

The general scheme of the SAEA for MDP is summarized in Algorithm 1. During the search process, we use the objective function $f(x)$ as the fitness function. The SAEA starts with an initial feasible solution x^c and a random generated

Algorithm 1 The general scheme of the SAEA

```
Input: An instance.
Output: The found best solution $x^*$.
 1: Randomly generate $x^c$ by selecting $m$ elements from $\{1, \dots, n\}$ at random.
 2: for $k = 1$ to $s$ do
 3:     Randomly generate $x^k$ by selecting $m$ elements from $\{1, \dots, n\}$ at random.
 4: end for
 5: Let $P = \{x^1, \dots, x^s\}$, and $x^* = \text{argmax}\{f(x^k), k = 1, \dots, s\}$.
 6: for $G = 1$ to $G_{\max}$ do
 7:     Randomly select a solution $y$ from $P$.
 8:     $z \leftarrow crossover(x^c, y)$.
 9:     $z$ is further improved by the SA based local search procedure.
10:     if $f(z) > f(x^*)$ then
11:         Let $x^* = z$.
12:     end if
13:     if $f(z) > f(x^c)$ then
14:         Let $x^c = z$.
15:     else
16:         Apply the population updating procedure to update $P$.
17:     end if
18: end for
19: Return $x^*$.
```

population P. The population P contains s different feasible solutions, i.e., $P = \{x^1, \ldots, x^s\}$. Let x^* be the current best solution. The SAEA repeats an iterative process for a fixed number of generations. At each generation, a solution, say y, is randomly selected from P. Then, the crossover operator (Section 2.2) is applied to x^c and y to generating a new offspring solution z. The obtained feasible solution is further improved by the SA based local search procedure (Section 2.3). Finally, we use z to update x^c, or the population P. If z is better than x^c, we let $x^c = z$; otherwise, we apply a population updating procedure (Section 2.4) to update P. When the predetermined maximum number of generations ($G_{\max}$) is reached, we stop the algorithm and return x^*. In the following subsections, we give more details on the components of SAEA.

2.2. **Crossover operator.** Crossover is a key component of SAEA. The proposed crossover operator consists of two steps. We firstly create a partial solution z by selecting the common selected elements with respect to the selected parents x^c and y. More precisely, if $x_i^c = y_i$, we set $z_i = y_i$; otherwise, let $z_i = 0$. Then, we use a random strategy to complete the partial solution z. Let $|z|$ be the number of the selected elements. If $|z| < m$, we randomly select an unselected element k, and let $z_k = 1$. The above operation repeats until $|z| = m$.

2.3. **SA based local search procedure.** The SA based local search procedure is the most important component of SAEA. It affects both the solution quality and the solution time.

Let x^* and x^{best} be the current best solution found so far, and the best solution obtained by the SA based local search procedure, respectively. We define S and U as the set of selected elements and the set of unselected elements, respectively. During the search process, the SA based local search procedure selects an element from S, and an element from U, and swaps them. In this way, the newly obtained solution is feasible. The SA based local search procedure selects elements by the concept of the move gain. The move gain of element i is defined by

$$\triangle_i = \begin{cases} \sum_{j\in S} -d_{ij}, & \text{if } i \in S; \\ \sum_{j\in S} d_{ij}, & \text{if } i \in U. \end{cases} \tag{1}$$

Our SA based local search procedure starts from an initial solution z. The pseudo code of this procedure is given in Algorithm 2. Initially, the current temperature T is set to T_0 (line 1). Line 4 calculates the initial move gains of each element according to (1). Let $S_{\max}$ ($U_{\max}$) and $S_{\min}$ ($U_{\min}$) be the highest and the lowest move gains of the elements in S (U), respectively. At each iteration, we firstly identify the values $S_{\max}$, $U_{\max}$, $S_{\min}$, and $U_{\min}$ (line 5). In order to diversify the search, two restricted candidate lists (RCL_0 and RCL_1) are constructed by the elements with move gains larger than a threshold value. Specifically, let

$$RCL_1 = \{i \in S : \triangle_i > S_{\max} - \alpha \times (S_{\max} - S_{\min})\}, \tag{2}$$

$$RCL_0 = \{j \in S : \triangle_j > U_{\max} - \alpha \times (U_{\max} - U_{\min})\}, \tag{3}$$

where $\alpha \in [0, 1]$ is a parameter. We randomly select an element k from RCL_1 and an element l from RCL_0 (line 7). Afterward, k and l are swapped to obtain a new solution z' (line 8).

Next, if z' is not worse than z, then it replaces z as the new current solution (lines 12-13). Otherwise, z' is accepted with a small probability p (lines 14-18), which is given as follows: $p = e^{\frac{-(\triangle_k + \triangle_l - d_{kl})}{T}}$, where T is the current temperature.

Finally, once a move is performed, we just need to update a subset of move gains affected by the move. The following rule [7, 17] is applied to updating the move gains:

$$\triangle_i = \begin{cases} -\triangle_k + d_{kl}, & \text{if } i \in k; \\ -\triangle_l + d_{kl}, & \text{if } i \in l; \\ \triangle_i + d_{ki} - d_{li}, & \text{if } i \in S - \{k, l\}; \\ \triangle_i - d_{ki} + d_{li}, & \text{if } i \in U - \{k, l\}. \end{cases} \tag{4}$$

The above swapping operator performs L iterations. After that, line 22 decreases the current temperature T according to the rule $T = \gamma T$, $\gamma \in (0, 1)$. When the above process repeats $I_{\max}$ times, the SA based local search procedure stops, and the best solution obtained is returned.

Algorithm 2 SA based local search procedure

```
Input: an initial solution $z$, $L$, $I_{\max}$.
Output: an improved solution $x^{best}$.
1: Initial $T = T_0$, and $x^{best} = z$, $G = 0$.
2: for $G = 1$ to $I_{\max}$ do
3:   for $Iteration = 1$ to $L$ do
4:     Initialize $z' = z$. Calculate the move gains $\triangle_i$, $i = 1, \ldots, n$, according to (1).
5:     Identify $S_{\max}$, $S_{\min}$, $U_{\max}$, and $U_{\min}$.
6:     Construct $RCL_0$ and $RCL_1$ according to (2), and (3), respectively.
7:     Randomly select elements $k$ and $l$ from $RCL_1$ and $RCL_0$, respectively.
8:     Let $z'_k = 1 - z'_k$, $z'_l = 1 - z'_l$.
9:     if $f(z') > f(x^{best})$ then
10:      $x^{best} = z'$.
11:    end if
12:    if $f(z') > f(z)$ then
13:      $z = z'$.
14:    else
15:      Generate a number $p \in (0, 1)$ at random.
16:      if $p < e^{\frac{-(\triangle_k + \triangle_l - d_{kl})}{T}}$ then
17:        Let $z = z'$.
18:      end if
19:    end if
20:    Use (4) to update the move gains.
21:  end for
22:  Let $T = \gamma T$.
23: end for
24: Return $x^{best}$.
```

2.4. **Population updating procedure.** When the newly obtained solution z is not better than x^c, we insert z into the population P and decide which existing solution of P should be replaced. In order to maintain the diversification of the population, our population updating procedure is based on the distance of solutions. Given two solutions x and y, the distance between them is defined as:

$$dis(x, y) = \sum_{i=1}^{n} |x_i - y_i|. \tag{5}$$

The proposed population updating procedure firstly calculates the distances between z and x^i, $i \in P$. Then, the solution x^{small} with the smallest distance is identified, i.e., $x^{small} = \text{argmin}\{dis(z, x^i), i \in P\}$. Finally, we use z to replace x^{small}.

3. Computational Results.

3.1. Comparison with other algorithms.

The proposed algorithm SAEA was coded in the C programming language and the tests were carried out on an AMD processor with 3.4 GHz clockpulse and 2.0 GB RAM under Windows XP. Two sets of benchmark instances with the size ranging from 400 to 2000 are used to test the performance of SAEA. The characteristics of the instance sets are given as follows [6].

• Silva instances [20]: There are several instances with $n \in [100, 500]$. We take 8 instances with $n = 400, 500$ to test the proposed algorithm.

• Random Type 1 instances (Type1_22): matrices with real numbers generated from a $(0, 10)$ uniform distribution. 20 instances with $n = 2000$ and $m = 200$ are considered.

The parameter setting of SAEA used in our experiments is listed in Table 1. These parameter values were determined by a preliminary experiment.

Table 1. Settings of parameters

Parameters	Description	Values
s	number of solutions in P	10
$G_{\max}$	number of generations of SAEA	500
α	parameter in (2) and (3)	0.1
T_0	initial temperature	1.0
γ	temperature updating parameter	0.95
L	number of iterations in Algorithm 2	30
$I_{\max}$	parameter in Algorithm 2	100

We ran our proposed algorithm 10 times. The SAEA is compared with several existing heuristics, including iterated tabu search (ITS) [19], tabu search (MTS) [21], hybridizing hopfield network and variable neighborhood search (DCHNN-VNS) [8], iterated greedy algorithm (ITG) [5], learnable tabu search guided by estimation of distribution (LTS-EDA) [6]. Tables 2 and 3 list the name of the instances, the best known solution value, the results including the best and average solutions produced by some existing algorithms. In Tables 2 and 3, 'Best' means the deviation from the best known solution value of the best solution value found by the algorithms, and 'Av.' means the deviation from the best

Table 2. Experimental results on Silva instances

Instance	Best Known values	ITS		MTS		DCHNN-VNS		SAEA	
		Best	Av.	Best	Av.	Best	Av.	Best	Av.
400_40	4658	0	0	27	52.6	0	3	0	0
400_80	16956	0	0	24	78.7	0	10	0	0
400_120	36317	0	0	0	101.6	0	14	0	0
400_160	62487	0	7.0	10	88.1	0	19	0	0
500_50	7141	0	0	24	74.3	0	14	0	0
500_100	26258	0	0	15	96.8	0	7	0	0
500_150	56572	0	0	0	110.0	0	1	0	0
500_200	97344	0	0	18	108.7	0	23	0	0

TABLE 3. Experimental results on random Type 1 instances

Instance	Best Known values	ITS		ITG		LTS-EDA		SAEA	
		Best	Av.	Best	Av.	Best	Av.	Best	Av.
Type1_22.1	114271	65	209.87	48	101.57	5	60.73	0	29.90
Type1_22.2	114327	29	262.27	0	69.90	0	89.87	0	27.50
Type1_22.3	114195	69	201.40	5	117.77	0	98.97	4	77.80
Type1_22.4	114093	22	200.53	58	141.93	0	79.87	2	46.60
Type1_22.5	114196	95	273.27	99	194.70	51	134.47	45	100.90
Type1_22.6	114265	41	168.17	9	96.20	0	40.17	15	53.10
Type1_22.7	114361	12	167.47	0	71.27	0	18.20	0	6.90
Type1_22.8	114327	25	256.40	0	193.60	0	159.10	0	70.70
Type1_22.9	114199	9	139.83	16	80.37	0	70.97	2	56.60
Type1_22.10	114229	24	204.93	35	121.43	0	56.20	7	53.30
Type1_22.11	114214	74	237.77	59	139.57	3	69.87	15	77.20
Type1_22.12	114214	55	249.53	88	156.00	15	84.93	60	107.70
Type1_22.13	114233	93	279.87	42	167.40	6	85.30	0	78.30
Type1_22.14	114216	92	248.50	64	202.80	0	81.00	0	23.80
Type1_22.15	114240	11	117.50	6	80.53	0	22.03	1	13.20
Type1_22.16	114335	11	225.40	35	167.90	0	36.47	8	27.90
Type1_22.17	114255	56	217.53	18	144.53	6	57.07	0	47.4
Type1_22.18	114408	46	169.97	2	117.37	2	22.83	0	6.40
Type1_22.19	114201	34	243.20	0	144.37	0	35.87	20	32.40
Type1_22.20	114349	151	270.67	45	187.23	0	95.40	0	72.20

known solution value of the average solution value found by the algorithms. The results shown in Tables 2 and 3 are from [8] and [6], respectively.

Results from Table 2 show that ITS, DCHNN-VNS, and SAEA can find the best known solutions on all tested Silva instances. MTS is able to find the best known solutions on 2 out of 8 instances. In terms of average solution quality, SAEA performs much better than MTS, and DCHNN-VNS. From Table 3, one can see that ITS, ITG, LTS-EDA and SAEA are able to find the best known solutions on 0, 4, 13, 9 instances, respectively. In terms of average solution quality, SAEA performs better than ITS, ITG, and LTS-EDA. These results provide evidence of the efficacy of our proposed algorithm.

3.2. **Influence of evolutionary strategies.** The SAEA combines evolutionary strategies (crossover operator, population updating procedure) with SA to improve the performance of the algorithm. To confirm the effectiveness of these evolutionary strategies, we conducted an experiment to compare SAEA with a variant of SAEA, which is denoted as MSSA. The MSSA is a multi-start SA based local search algorithm. At each iteration, MSSA randomly selects m elements to generate an initial solution, which is further improved by the SA based local search procedure. When the maximum number of iterations ($iter_{\max}$) is reached, we stop the MSSA and return the best solution found during the search.

We ran 10 times MSSA with $iter_{\max} = 400$ on the Type 1 instances. In order to show the advantage of SAEA over MSSA visually, we plot a figure based on the data of the Type 1 instances. Figure 1 compares the deviations of the best known solution value to the average solution value found by SAEA and MSSA, respectively.

From Figure 1, it can be seen that the deviation produced by SAEA is much smaller than that of MSSA. Based on the results in this figure, the conclusion is that the proposed

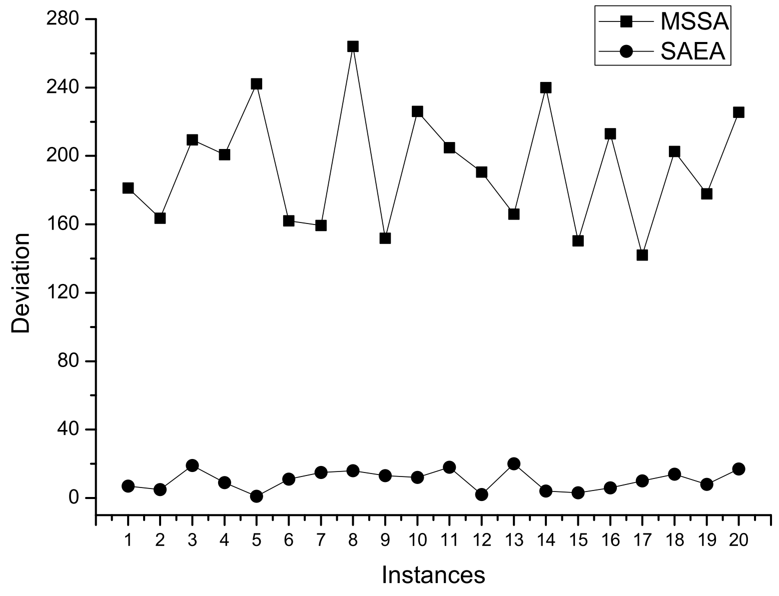

FIGURE 1. The deviation of the best known solution value to the average solution value found by SAEA and MSSA

evolutionary strategies are able to escape from the local optima produced by SA local search procedure.

4. Conclusions.

In this paper, we presented a simulated annealing based evolutionary algorithm (SAEA) for the maximum diversity problem. Firstly, a random population is generated to diversify the initial elite solutions. Afterward, the crossover operator is applied to generating good quality solutions. The newly generated solutions are enhanced by using the SA based local search procedure. It achieves a good compromise between intensification and diversification in the search process. Computational experiments on two sets of benchmark instances have demonstrated that our proposed algorithm is efficient. In future work, we try to solve other related NP hard problems with the proposed SAEA.

Acknowledgment. This research was supported partially by the Natural Science Foundation of Fujian Province of China under Grant 2016J01025, and the Program for New Century Excellent Talents in Fujian Province University.

REFERENCES

[1] C. C. Kuo, F. Glover and K. S. Dhir, Analyzing and modeling the maximum diversity problem by zero-one programming, *Decision Sciences*, vol.24, no.6, pp.1171-1185, 1993.
[2] G. C. Silva, M. R. Q. Andrade, L. S. Ochi et al., New heuristics for the maximum diversity problem, *Journal of Heuristics*, vol.13, no.4, pp.315-336, 2007.
[3] A. Duarte and R. Martí, Tabu search and GRASP for the maximum diversity problem, *European Journal of Operational Research*, vol.178, no.1, pp.71-84, 2007.
[4] R. Aringhieri and R. Cordone, Comparing local search metaheuristics for the maximum diversity problem, *Journal of the Operational Research Society*, vol.62, no.2, pp.266-280, 2011.
[5] M. Lozano, D. Molina and C. García-Martínez, Iterated greedy for the maximum diversity problem, *European Journal of Operational Research*, vol.214, no.1, pp.31-38, 2011.

[6] J. H. Wang, Y. Zhou, Y. Cai and J. Yin, Learnable tabu search guided by estimation of distribution for maximum diversity problems, *Soft Computing*, vol.16, no.4, pp.711-728, 2012.
[7] Y. Wang, J. K. Hao, F. Glover and Z. P. Lü, A tabu search based memetic algorithm for the maximum diversity problem, *Engineering Applications of Artificial Intelligence*, vol.27, pp.103-114, 2014.
[8] Y. L. Zhou, J. H. Wang, W. Bi, B. Mo and S. G. Li, Competitive hopfield network combined with variable neighborhood search for maximum diversity problems, *Computer Science*, vol.37, no.3, pp.208-211, 2010 (in Chinese).
[9] R. Martí, M. Gallego, A. Duarte and E. G. Pardo, Heuristics and metaheuristics for the maximum diversity problems, *Journal of Heuristics*, vol.19, no.4, pp.591-615, 2013.
[10] S. Kirkpatrick, C. Gelatt and M. Vecchi, Optimization by simulated, *Science*, vol.220, no.19, pp.671-680, 1983.
[11] H. Ahonen, A. G. de Alvarenga and A. R. S. Amaral, Simulated annealing and tabu search approaches for the corridor allocation problem, *European Journal of Operational Research*, vol.232, no.1, pp.221-233, 2014.
[12] Y. Y. Chen and C. Chen, Simulated annealing for interface-constrained channel assignment in wireless mesh networks, *Ad Hoc Networks*, vol.29, pp.32-44, 2015.
[13] M. E. Aydin and T. C. Fogarty, A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems, *Journal of Heuristics*, vol.10, no.3, pp.269-292, 2004.
[14] M. Shokouhifar and A. Jalali, An evolutionary-based methodology for symbolic simplification of analog circuits using genetic algorithm and simulated annealing, *Expert Systems with Applications*, vol.42, no.3, pp.1189-1201, 2015.
[15] M. Dai, D. B. Tang, A. Giret et al., Energy-efficient scheduling for a flexible flow shop using an improved generic-simulated annealing algorithm, *Robotics and Computer-Integrated Manufacturing*, vol.29, no.5, pp.418-429, 2013.
[16] S. Sakamoto, E. Kulla, T. Oda et al., A comparison study of simulated annealing and genetic algorithm for node placement problem in wireless mesh networks, *Journal of Mobile Multimedia*, vol.9, nos.1-2, pp.101-110, 2013.
[17] Z. Lü, F. Glover and J. K. Hao, Neighborhood combination for unconstrained binary quadratic problems, in *MIC-2009 Post-Conference Book*, M. Caserta and S. Voss (eds.), 2013.
[18] J. E. Beasley, *OR-Library: Unconstrained Binary Quadratic Programming*, 2000.
[19] G. Palubeckis, Iterated tabu search for the maximum diversity problem, *Applied Mathematics and Computation*, vol.189, no.1, pp.371-383, 2007.
[20] G. C. Silva, L. S. Ochi and S. L. Martins, Experimental comparison of greedy randomized adaptive search procedure for the maximum diversity problem, *Lecture Notes in Computer Science*, vol.3059, pp.498-512, 2004.
[21] E. M. Macambira, An application of tabu search heuristic for the maximum edge-weighted subgraph problem, *Annals of Operations Research*, vol.117, pp.175-190, 2002.