Comparative Study of Inhomogeneous Simulated Annealing Algorithms for Quadratic Assignment Problem

Zuoling Ma*, Lijin Wang*†, Song Lin*† and Yiwen Zhong*†
*Fujian Agriculture and Forestry University, Fuzhou, 350002, PR China
†Key Laboratory of Smart Agriculture and Forestry
(Fujian Agriculture and Forestry University), Fujian Province University, Fuzhou, 350002, PR China
Email: yiwzhong@fafu.edu.cn

***Abstract*—Simulated annealing (SA) algorithm is a popular metaheuristic which has been widely used to solve many continuous and combinatorial optimization problems. Temperature cooling scheme is one of the key factors for its promising performance. Several studies have shown that reannealing can improve SA's performance on quadratic assignment problem (QAP). Using the benchmark instances from QAPLIB, this paper carried systematic experiments to compare the effect of reannealing on performance of inhomogeneous SA (ISA). Simulation results show that: (1) although reannealing can improve ISA's performance in general, it may has no effect on some type of QAP instances; (2) parameters for cooling schedule are not only sensitive to the type of QAP instance, but sensitive to the specific instance also; (3) the timing to activate the reannealing is sensitive to the type of QAP instances also. The source code is made publicly available as a benchmark for future comparisons.**

I. Introduction

Quadratic assignment problem (QAP) is a fundamental NP-hard combinatorial optimization problems from the category of the facilities location problems. On the one hand, QAP is a generalization of a large number of theoretical issues[1]. On the other hand, QAP can be used to model many real world applications[2]. A QAP can be described as follows. Suppose there are a set of n facilities and a set of n locations, each pair of facilities have some flow to be exchanged, and each pair of locations have a distance. An $n \times n$ matrix A is used to store the distances among locations, where a_{ij} represents the distance between the pair of locations i and j. An $n \times n$ matrix B is used to store the flows among facilities, where b_{kl} represent the amount of flow exchanged between facilities k and l. Assigning facilities k and l to locations i and j results in a cost of $a_{ij} \times b_{kl}$. We can use a vector X to represent a candidate solution for the QAP. Each element x_i in X represent a facility assigned to location i. The goal of QAP is to find a solution X that minimizes the sum of costs:

$$f(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} \times b_{x_i,x_j} \tag{1}$$

Because QAP is NP-hard, there is no known algorithm which can make sure to obtain optimal solution in polynomial time. For many real-world QAP applications, it is preferred to find a workable solution with a reasonable cost, and therefore many researchers have been putting their efforts on studying how to design efficient metaheuristics to solve the QAP. In the recent years, many metaheuristics have been proposed for solving the QAP, such as tabu search (TS) [3], genetic algorithm [4], [5], [6], ant colony system [7], [8], particle swarm optimization algorithm [9], [10], migrating birds optimization [11], teaching learning-based optimization algorithms [12], memetic search [13], [14], honey bee algorithm [15], self organising migrating algorithm [16], multi-agent based optimization method [17], bat algorithm [18], [19], breakout local search [20], [21], and great deluge [22] etc.

Simulated annealing (SA) algorithm is a popular iterative metaheuristic widely used to address NP-hard optimization problems. Several versions of SA algorithm [23], [24], [25], [26], [27], [28], [29], [30], [31], [32] have been proposed for solving QAP. SA algorithm also can be combined with other meta-heuristic [4], [7], [33] and used in hyper-heuristic [2]. Several studies have shown that SA outperforms TS when computation resource is enough (or when high quality solution is required). SA is one of the state-of-the-art metaheuristics which have good performance even for large QAP instances[2]. Furthermore, SA algorithm is an important local search method which is often been combined into other population-based meta-heuristics to improve the intensification ability [4], [7].SA algorithm can also be combined into other metaheuristics, such as tabu search [35], [36], [37], genetic algorithm [38], [39], [40], [41], [42], ant colony optimization [43], [44], [45], [46], and particle swarm optimization [47], [48], [49] etc., to improve those algorithms' performance in a wide range of applications.

Several studies have shown that more sampling in promising temperatures can improve the performance of SA algorithm. In the field of QAP, several reannealing strategies have been proposed for inhomogeneous SA (ISA) [30], [31], [32] and homogeneous SA [23], [27] to find the promising temperatures. Misevičius compared three different reannealing strategies for ISAs and found that reannealing with Lundy Mees (LM) oscillation [32] is better than reannealing with cosine oscillation [31] and reannealing to fixed optimal temperature [30].

Misevičius also claimed that the initial and final temperatures are data independent and not very difficult to determine. But the comparison may not be fair, because: (1) both reannealing with LM oscillation and reannealing with cosine oscillation used local search method to further improve the quality of solution; and (2) the parameters used may not be optimal for the used QAP instances. To compare those reannealing strategies fairly, this paper carried an empirical study of those three reannealing strategies and tries to answer following questions: (a) do reannealing strategies always outperform basic ISA? (b) are those parameters robust on different type of QAP instances? (c) are those parameters robust on different QAP instances? and (d) is the parameter used to activate reannealing robust?

The remainder of this paper is organized as follows. Section 2 presents those inhomogeneous SA algorithms to be compared. Section 3 describes the experimental settings. The results obtained from the comparing experiments are presented and discussed in Section 4. Section 5 concludes our comparative study.

II. Methods

SA algorithm has explicit strategies to avoid being strapped into local minima. The fundamental idea of its strategy is to accept worse solutions in some probability. The probability of accepting a worse solution is decreased during the search through a control parameter temperature. Suppose current solution is x and current temperature is t, SA algorithm randomly selects a candidate solution x' from the neighbourhood of x. Then SA algorithm will replace x with x' based on the following acceptance probability:

$$p = \begin{cases} 1, & if\ f(x') \leq f(x) \\ e^{-(f(x')-f(x))/t}, & else \end{cases} \tag{2}$$

where parameter temperature t is greater than 0.

Neighbourhood structure and annealing schedule are the most important factors that determine the performance of SA algorithm. In QAP, the most used neighbour operator is swap where two facilities are swapped. There are two different strategies to search the neighbourhood defined by swap operator. One strategy is to choose the next potential solution at random. The other strategy is to arrange all the possible swap elements in order and explore the neighbourhood in a systematic way. For QAP, several studies have shown that searching the neighbour solutions in a fixed order has better performance. A cooling schedule may include the initial value of the temperature, the final value of the temperature, the updating function used to control the temperature, and the strategy used to guarantee to reach equilibrium in each temperature. SA algorithm may be homogeneous or inhomogeneous. In homogeneous SA algorithm, each temperature will be used by a number of trials so the state can reach equilibrium. The most widely used cooling schedule for homogeneous SA is geometric cooling:

$$t_{k+1} = \alpha t_k \tag{3}$$

where $0 < \alpha < 1$.

TABLE I
Pseudo code of basic ISA algorithm

Algorithm 1. Pseudo code of basic ISA algorithm
Input: Data about the QAP instance
Output: The best solution found
1: Set up parameter t_0, t_f, and L
2: Use Eq. 5 to calculate cooling coefficient β
3: Create initial solution x randomly
4: Let $t = t_0$, $k = 0$, and $best = x$
5: for $k = 1$ to L
6: Find a candidate solution y in x's neighborhood
7: Produce a random number p in range [0,1)
8: if $f(y) < f(x)$ or $p < exp(-(f(y) - f(x))/t)$ then
9: $x = y$
10: endif
11: if $f(x) < f(best)$ then
12: $best = x$
13: endif
14: Use Eq. 4 to update temperature t
15: end for
16: return $best$

In ISA algorithm, each temperature will be used only once. The most widely used cooling schedule for ISA is LM schedule:

$$t_{k+1} = \frac{t_k}{1 + \beta t_k} \tag{4}$$

In LM schedule, if the initial temperature t_0, final temperature t_f, and total schedule length L are given, the coefficient β can be calculated by:

$$\beta = \frac{t_0 - t_f}{L t_0 t_f} \tag{5}$$

The pseudo code of basic ISA algorithm can be described as Algo. I. In algo. I, parameter t_0, t_f, and L represents initial temperature, final temperature, and total schedule length respectively.

Three reannealing strategies have been proposed for ISA [30], [31], [32]. In Connolly's algorithm[30], a simple reannealing strategy is applied as follows. In the search process, a variable rc is used to count the number of times where consecutive worse solutions are rejected, and a variable t_b is used to record the temperature where the best so far solution is found. If the value of rc reaches a value specified by a parameter, then the next worse solution will be accepted, the temperature is assigned to t_b, and the search is carried out at constant temperature t_b until the end of the algorithm. The pseudo code of Connolly's ISA (CISA) algorithm can be described as Algo.II. In Algo. II, parameter MRN is the maximum consecutive rejected number, variable rc is used to record then number of the consecutive rejected uphill moves, variable t_b is used to record the temperature where the best solution is found, and variable rh is used to indicate whether the algorithm is in reheated state. After algorithm reaches reheated state, t is set to t_b which will be used until the end of the algorithm.

TABLE II
PSEUDO CODE OF CONNOLLY'S ISA ALGORITHM

Algorithm 1. Pseudo code of Connolly's ISA algorithm

Input: Data about the QAP instance
Output: The best solution found
1: Set up parameter t_0, t_f, L, and MRN
2: Use Eq. 5 to calculate cooling coefficient β
3: Create initial solution x randomly
4: Let $t = t_0, k = 0, rh = 0, t_b = t_0, rc = 0$, and $best = x$
5: for $k = 1$ to L
6: Find a candidate solution y in x's neighborhood
7: Produce a random number p in range [0,1)
8: if $f(y) < f(x)$ or $p < exp(-(f(y) - f(x))/t)$ then
9: $x = y$
10: $rc = 0$
11: else
12: $rc = rc + 1$
13: if $rc > MRN$ then
14: $rh = true$
15: $t = t_b$
16: endif
17: endif
18: if $f(x) < f(best)$ then
19: $best = x$
20: $t_b = t$
21: endif
22: if not rh then
23: Use Eq. 4 to update temperature t
24: endif
25: end for
26: return $best$

The ISA with cosine-based oscillation is proposed by Bölte and Thonemann [31]. The basic framwork of Bölte and Thonemann's ISA (BISA) algorithm is similar to Algo. II, but BISA uses a much more intelligent reannealing technique, which is referred to as cosine-based oscillation. Suppose, when the rc reaches MRN, the temperature is t_r and the iteration is k_r, then the temperature t in iteration k is calculated by equation (6):

$$t_k = t_r + 0.5 \times t_r \times cos(\omega(k - k_r)) \tag{6}$$

where

$$\omega = \frac{16\pi}{25n(n-1)} \tag{7}$$

The ISA with LM-function-based oscillation is proposed by Misevičius [32]. The basic framwork of Misevičius's ISA (MISA) algorithm is similar to Algo. II also, MISA uses LM-function-based oscillation after the rc reaches MRN. If the remaining iteration is less than n, then t_r will be used until the end of algorithm. Otherwise, let $t_0 = (1 + 1/3 \times t_r)$ and $t_f = (1 + 1/3 \times t_r)$, LM schedule is used in the following k_r iterations over and over until the remaining iteration is less than n. The features of those cooling schedule can be exhibited in Figure 1. The data in Figure 1 are from the simulation on tai25a instance when total iteration number L is equal to $25n(n-1)/2$.

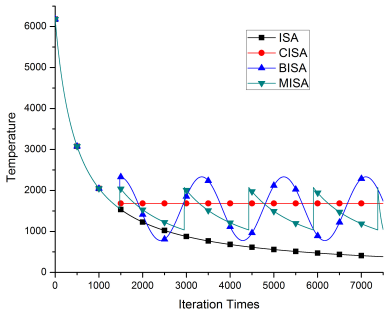

Fig. 1. Compare the temperature change process of different cooling schedules

III. EXPERIMENTAL SETTINGS

Misevičius [32] compared CISA, BISA, and MISA on 32 QAP instances and found that MISA is better than CISA and BISA. But the comparison may not be fair, because:

1) Both MISA and BISA used local search method to further improve the quality of solution, but CISA did not use any local search method. In MISA, each time the temperature is heated up, a local search method is used to improve the best solution found so far. In the end of search, MISA used a simplified tabu search to further improve the best solution. BISA also used a local search method to improve the best solution in the end of the search.
2) The used parameters may not be optimal for CISA and BISA on the used QAP instances. In [32], the parameters of CISA and BISA are based on that used by [30] and [31] respectively. This strategy seems fair. But considering that the QAP instances used by [30] and [31] are different from those used in [32], those parameters, which are optimal in [30] and [31], may not be optimal in [32] if the parameters are sensitive to QAP instances.

Considering the above mentioned two aspect, in our experiments, all those algorithms will not use any local search method; and parameter tuning is carried for each situation so all algorithms can run with their optimal parameters.

A. Parameter Settings

In their original implementations, CISA, BISA, and MISA used different way to produce initial temperature t_0 and final temperature t_f. To find suitable t_0 and t_f for those algorithms, we use MISA's strategy to produce t_0 and t_f first, and then the t_f is tuned to a suitable value for each algorithm. In MISA, the t_0 and t_f is calculated as Eq. 8 and Eq. 9 respectively.

$$t_0 = (1 - \lambda_1)\delta_{min} + \lambda_1\delta_{avr} \tag{8}$$

$$t_f = (1 - \lambda_2)\delta_{min} + \lambda_2\delta_{avr} \tag{9}$$

where δ_{min} and δ_{avr} are the minimum difference and average difference by performing $n(n-1)/2$ random swaps, $\lambda_1 = 0.5$ and $\lambda_2 = 0.05$. To find a suitable t_f for each algorithm, we let $t_f = t_f/scale$ where $scale$ is an adjustable parameter. As in [32], in the experiments of section 4, $L = 50n(n-1)/2$ and $MRN = n(n-1)/4$.

B. Benchmark Problems

In the QAPLIB, there are four types of QAP instances. Type 1 instances are unstructured instances whose distance and flow values are randomly generated by uniform distribution. Type 2 instances are grid-based distances whose distance values are the Manhattan distances between points on a grid, and the flow values are randomly generated. The data of type 3 instances is produced from real-life practical QAP applications. The data of type 4 instances is generated to simulate real-life QAP applications. The 32 instances used in MISA include type 1, type 2, and type 3 instances. Besides those 32 instances used in MISA, we select some other instances from QAPLIB to study the parameter sensitiveness of ISAs.

IV. Experimental Results and Discussion

A. Experiments on 32 hybrid instances

We use the 32 QAP instances used in MISA to find suitable final temperature t_f for each ISA algorithm. For each algorithm, we test 16 different $scale$ from 1.0 to 2.5 with a step 0.1. All algorithm use identical initial assignments. Each instance is run 100 times, and percentage error of average solution is used to compare the performance. Figure 2 is the simulation results. From Figure 2 we have: (1) Both CISA, BISA, and MISA have better performance than basic ISA; (2) there is not significant difference among CISA, BISA, and MISA; (3) the optimal $scale$ is 1.8 for CISA, BISA, and MISA, 1.3 and 1.8 can both be the optimal $scale$ for basic ISA. To compare those algorithms in detail, the simulation results on 32 instances with optimal $scale$ used are list in Table III where the best solution is highlighted in bold. Wilcoxon signed ranks test is used to compare basic ISA, CISA, BISA, and MISA. The results show that basic ISA is significantly worse than CISA, BISA, and MISA. There is no significant difference between BISA and MISA. There is no significant difference between CISA and MISA. CISA is significantly worse than BISA.

Although wilcoxon signed ranks test shows that CISA is significantly worse than BISA, but the data in Table III clearly show that there is difference phenomena on different types of instances. For example, CISA is apparently better than other algorithms on type 1 instances. To show those difference, we compare the performance of different algorithms on different type of instances on Figure 3. Figure 3 shows that the best algorithm on type 1, type 2, and type 3 instances is CISA, BISA, and MISA respectively.

B. Experiments on different types of instances

To verify whether the optimal t_f is robust on different type of instances. Using type 1 and type 4 instances (the used type

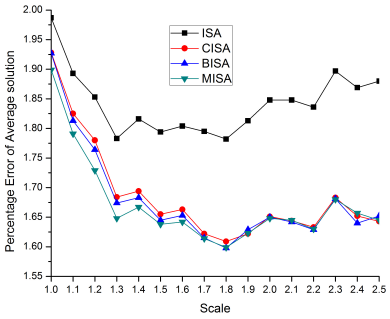

Fig. 2. Compare the performance with different parameter $scale$

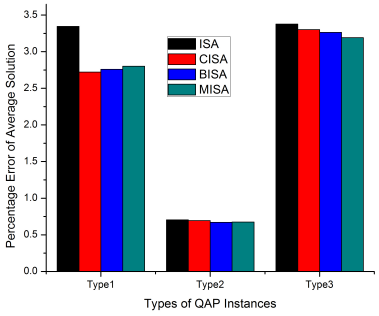

Fig. 3. Performance comparison on different types of the 32 instance

4 instances are listed in Table IV) independently, we try to find suitable final temperature t_f for each ISA algorithm. For each algorithm, we test 20 different $scale$ from 0.3 to 2.0 with a step 0.1. Figure 4 is the simulation results on type 1 instances. From Figure 4 we have: (1) the optimal $scale$s for CISA, BISA, and MISA on type 1 instances are different from that on 32 hybrid instances, for example, the optimal $scale$ on type 1 instances is 0.9 for MISA, which is different from optimal value 1.8 on 32 hybrid instances; (2) CISA, BISA, and MISA have better parameter t_f robust than basic ISA; (3) although Table III shows that BISA has best performance on type 1 instances, Figure 4 shows that MISA has best performance when optimal t_f is used. Figure 5 is the simulation results on type 4 instances. Figure 5 shows that all ISAs has same behaviour. It means reannealing strategy is not activated on type 4 instances. All algorithms have best performance when $scale$ is equal to 0.7, this is quite different from the value 1.8 as showed in Figure 2. It means that the parameter t_f is data dependent and has bad generalization ability.

C. Experiments on different instances of same type

To verify whether the optimal t_f is robust on different instances belong to same type. Using the 5 instances of type 3 in Table III as benchmark instances, we try to find suitable final temperature t_f for MISA algorithm. For each instance, we test 20 different $scale$ from 1.0 to 3.0 with a step 0.1. Figure 6 is the simulation results. Figure 6 shows that the optimal $scale$

TABLE III
PERFORMANCE COMPARISON WHEN OPTIMAL *scale* IS USED

No.	Instances	Type	Optimal	ISA	CISA	BISA	MISA
1	kra30a	3	88900	3.279	2.991	2.959	**2.854**
2	kra30b	3	91420	2.153	1.996	1.975	**1.852**
3	nug30	2	6124	1.082	1.049	**0.862**	0.933
4	sko42	2	15812	0.919	0.887	0.891	**0.880**
5	sko49	2	23386	0.773	0.758	0.728	**0.719**
6	sko56	2	34458	**0.759**	0.801	0.764	0.786
7	sko64	2	48498	**0.710**	0.726	**0.710**	0.716
8	sko72	2	66256	**0.605**	0.636	0.616	0.611
9	sko81	2	90998	0.570	0.584	0.577	**0.567**
10	sko90	2	115534	0.571	0.586	0.574	**0.569**
11	sko100a	2	152002	**0.427**	0.432	0.429	**0.427**
12	sko100b	2	153890	0.487	0.494	**0.486**	**0.486**
13	sko100c	2	147862	0.547	0.551	0.549	**0.544**
14	sko100d	2	149576	**0.520**	0.528	0.523	0.521
15	sko100e	2	149150	**0.579**	0.583	0.582	0.580
16	sko100f	2	149036	0.582	0.588	0.583	**0.580**
17	ste36a	3	9526	2.961	3.025	2.886	**2.794**
18	ste36b	3	15852	6.504	6.504	6.504	**6.475**
19	ste36c	3	8239110	1.988	1.995	1.992	**1.981**
20	tai25a	1	1167256	3.684	**2.763**	2.914	2.833
21	tai30a	1	1818146	3.344	**2.558**	2.629	2.593
22	tai35a	1	2422002	3.417	2.739	**2.643**	2.748
23	tai40a	1	3139370	3.475	**2.749**	2.898	2.842
24	tai50a	1	4938796	3.607	**2.922**	2.974	3.090
25	tai60a	1	7205962	3.434	**2.928**	2.932	3.084
26	tai80a	1	13557864	3.016	2.640	**2.606**	2.672
27	tai100a	1	21125314	2.781	**2.471**	2.487	2.545
28	tho30	2	149936	1.496	1.394	**1.298**	1.343
29	tho40	2	240516	1.586	1.427	1.409	**1.398**
30	tho150	2	8133398	**0.581**	**0.581**	**0.581**	**0.581**
31	wil50	2	48816	0.288	0.299	0.294	**0.284**
32	wil100	2	273038	**0.289**	0.293	0.291	0.290
	Average			1.782	1.609	**1.598**	1.599

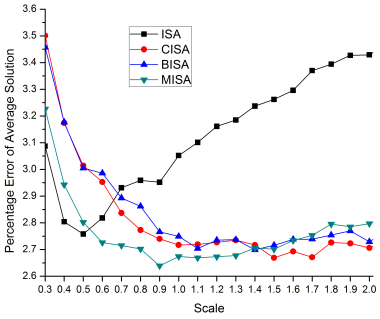

Fig. 4. Performance comparison with different parameter *scale* on type 1 instances

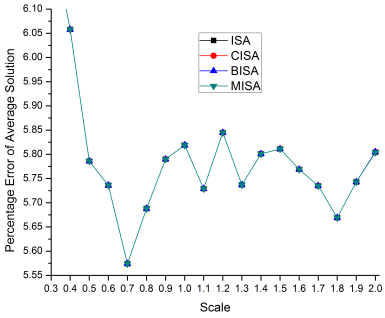

Fig. 5. Performance comparison with different parameter *scale* on type 4 instances

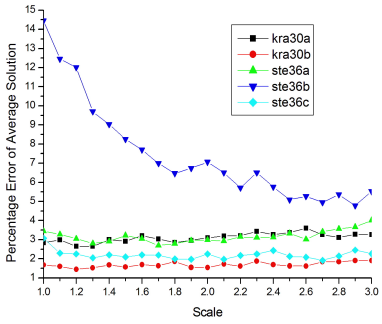

Fig. 6. Performance comparison on instances of type 3 with different *scale*

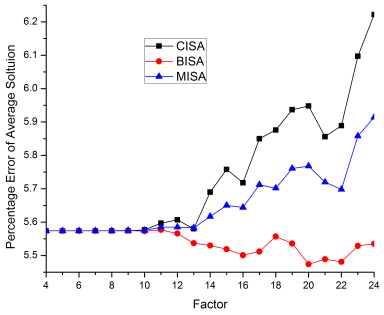

Fig. 7. Performance comparison on type 4 instances with different $factor$

for each instance is quite different, for example, the optimal $scale$s for kra30b and ste36b are 1.2 and 2.9 respectively.

D. Experiments on the timing of reannealing

In CISA, BISA, and MISA, there is another important parameter MRN which determine when the reannealing scheme will be activated. The results showed in Figure 5 mean that $MRN = n(n-1)/4$ is not suitable for type 4 instances. To find suitable MRN for each ISA algorithm on type 4 instances, we set $MRN = n(n-1)/factor$, and test 21 different $factor$ from 4 to 24 with a step 1. Figure 7 is the performance comparison of different $factor$ when $scale$ is equal to 0.7. Figure 7 shows that reannealing cannot improve the performance of CISA and MISA. For BISA, the optimal $factor$ is 20. To compare those algorithms in detail on type 4 instances, the simulation results on 9 instances are list in Table IV where the best solution is highlighted in bold. Table IV clearly shows that although BISA has best performance in terms of average solution, there are still 3 instanes which ISA has better performance than BISA. It means that the parameter MRN, which determines the activation of reannealing, is data dependent also.

TABLE IV
PERFORMANCE COMPARISON ON TYPE 4 INSTANCES WHEN $scale$ IS 0.7 AND $factor$ IS 20

No.	Instances	Optimal	ISA	CISA	BISA	MISA
1	tai20b	122455319	12.035	12.369	**11.418**	12.25
2	tai25b	344355646	8.059	8.913	**7.614**	8.636
3	tai30b	637117113	6.330	6.244	**6.125**	6.352
4	tai35b	283315445	**5.987**	7.202	6.327	6.636
5	tai40b	637250948	**5.765**	6.39	5.852	5.988
6	tai50b	458821517	**3.193**	3.418	3.204	3.210
7	tai60b	608215054	2.811	2.907	**2.790**	2.842
8	tai80b	818415043	2.896	2.956	**2.868**	2.901
9	tai100b	1185996137	3.092	3.136	**3.069**	3.096
Average			5.574	5.948	**5.474**	5.768

V. CONCLUSION

Suitable cooling scheme is one of the most important factors to guarantee SA's promising performance. In order to analyze the effect of reannealing on the performance of ISA algorithm, this paper carried a systematic experiment study on the four types of QAP instances. Experiment study uncovered several facts different from those claimed in [32]. First, although reannealing can improve ISA's performance in general, it may has no effect on some type of QAP instances. For example, reannealing cannot improve the performance of CISA and MISA on type 4 QAP instances. Second, both parameter for cooling scheme and parameter for activating the reannealing are data dependent. Using BISA as an example, the optimal parameter $scale$, which is used to produce final temperature, is 0.7 for type 4 QAP instances, but is 1.4 for type 1 QAP instances. Considering the parameter MRN, which determine the timing to activate reannealing, $n(n-1)/4$ is suitable for type 1 QAP instances, but should be $n(n-1)/20$ for type 4 QAP instance. Those results mean that it is not a easy task to find suitable cooling parameters for ISA with reannealing strategy. Those shortages leads to the following research issues: (1) can we find more robust parameters setting method for ISAs with reannealing strategy; (2) do other cooling schemes, such as geometry cooling or list-based cooling [34], have better parameters' robustness? We will study those issues later.

ACKNOWLEDGMENT

This work was supported by Nature Science Foundation of Fujian Province of P. R. China (No. 2015J01233, 2016J01280), major projects of regional development of Fujian Province of P. R. China (No. 2015N3011), and the special fund for scientific and technological innovation of Fujian Agriculture and Forestry University (No. CXZX2016026, No. CXZX2016031).

REFERENCES

[1] W. Chmiel, P. Kadłuczka, J. Kwiecień, and B. Filipowicz, "A comparison of nature inspired algorithms for the quadratic assignment problem," *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 65, no. 4, 2017.

[2] T. Dokeroglu and A. Cosar, "A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 10–25, 2016.

[3] M. Czapiński, "An effective parallel multistart tabu search for quadratic assignment problem on cuda platform," *Journal of Parallel and Distributed Computing*, vol. 73, no. 11, pp. 1461–1468, 2013.

[4] P. P. C. Yip and Y. H. Pao, "A guided evolutionary simulated annealing approach to the quadratic assignment problem," *IEEE Transactions on Systems Man and Cybernetics*, vol. 24, no. 9, pp. 1383–1386, 2002.

[5] U. Tosun, T. Dokeroglu, and A. Cosar, "A robust island parallel genetic algorithm for the quadratic assignment problem," *International Journal of Production Research*, vol. 51, no. 14, pp. 4117–4133, 2013.

[6] E. Lalla-Ruiz, C. Expósito-Izquierdo, B. Melián-Batista, and J. M. Moreno-Vega, "A hybrid biased random key genetic algorithm for the quadratic assignment problem," *Information Processing Letters*, vol. 116, no. 8, pp. 513–520, 2016.

[7] N. e. Demirel and M. D. Toksari, "Optimization of the quadratic assignment problem using an ant colony algorithm," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 427–435, 2006.

[8] E. N. Caceres, H. Fingler, H. Mongelli, and S. W. Song, "Ant colony system based solutions to the quadratic assignment problem on gpgpu," in *International Conference on Parallel Processing Workshops*, pp. 314–322, 2012.

[9] C. Lv, "An improved particle swarm optimization algorithm for quadratic assignment problem," in *International Conference on Multimedia Information NETWORKING and Security*, pp. 1728–1731, 2012.

[10] A. M. Helal and A. M. Abdelbar, "Incorporating domain-specific heuristics in a particle swarm optimization approach to the quadratic assignment problem," *Memetic Computing*, vol. 6, no. 4, pp. 241–254, 2014.

[11] E. Duman, M. Uysal, and A. F. Alkaya, "Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem," *Information Sciences*, vol. 217, no. 24, pp. 65–77, 2012.

[12] T. Dokeroglu, "Hybrid teachingClearning-based optimization algorithms for the quadratic assignment problem," *Computers and Industrial Engineering*, vol. 85, no. C, pp. 86–101, 2015.

[13] U. Benlic and J. K. Hao, "Memetic search for the quadratic assignment problem," *Expert Systems with Applications*, vol. 42, no. 1, pp. 584–595, 2015.

[14] M. Harris, R. Berretta, M. Inostroza-Ponta, and P. Moscato, "A memetic algorithm for the quadratic assignment problem with parallel local search," in *Evolutionary Computation*, pp. 838–845, 2015.

[15] M. Mirzazadeh, G. Shirdel, and B. Masoumi, "A honey bee algorithm to solve quadratic assignment problem," *Journal of Optimization in Industrial Engineering*, vol. 4, no. 9, pp. 27–36, 2011.

[16] D. Davendra and I. Zelinka, "Optimization of quadratic assignment problem using self organising migrating algorithm," *Computing and Informatics*, vol. 28, no. 2, pp. 169–180, 2009.

[17] I. Sghir, J. K. Hao, I. B. Jaafar, and K. Ghdira, "A multi-agent based optimization method applied to the quadratic assignment problem," *Expert Systems with Applications*, vol. 42, no. 23, pp. 9252–9262, 2015.

[18] A. Shukla, "A modified bat algorithm for the quadratic assignment problem," in *Evolutionary Computation*, pp. 486–490, 2015.

[19] M. E. Riffi, Y. Saji, and M. Barkatou, "Incorporating a modified uniform crossover and 2-exchange neighborhood mechanism in a discrete bat algorithm to solve the quadratic assignment problem," *Egyptian Informatics Journal*, vol. 18, no. 3, 2017.

[20] U. Benlic and J. K. Hao, "Breakout local search for the quadratic assignment problem," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4800–4815, 2013.

[21] Y. Aksan, T. Dokeroglu, and A. Cosar, "A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem," *Computers and Industrial Engineering*, vol. 103, 2017.

[22] A. Acan and A. Ünveren, "A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem," *Applied Soft Computing*, vol. 36, no. C, pp. 185–203, 2015.

[23] J. C. Wang, "A multistart simulated annealing algorithm for the quadratic assignment problem," in *International Conference on Innovations in Bio-Inspired Computing and Applications*, pp. 19–23, 2012.

[24] T. Peng, H. Wang, and D. Zhang, "Simulated annealing for the quadratic assignment problem: A further study," *Computers and Industrial Engineering*, vol. 31, no. 3, pp. 925–928, 1996.

[25] K. Shojaee Ghandeshtani, N. Mollai, S. M. H. Seyedkashi, and M. M. Neshati, "New simulated annealing algorithm for quadratic assignment problem," *Ind.eng.chem.res*, vol. 49, no. 24, pp. 12333–12338, 2010.

[26] G. Paul, "Comparative performance of tabu search and simulated annealing heuristics for the quadratic assignment problem," *Operations Research Letters*, vol. 38, no. 6, pp. 577–581, 2010.

[27] M. S. Hussin and T. Sttzle, "Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances," *Computers and Operations Research*, vol. 43, no. 43, pp. 286–291, 2014.
[28] R. E. Burkard and F. Rendl, "A thermodynamically motivated simulation procedure for combinatorial optimization problems," *European Journal of Operational Research*, vol. 17, no. 2, pp. 169–174, 1984.
[29] P. E. Mickeyr. Wilhelm and P. E. Thomasl. Ward, "Solving quadratic assignment problems by simulated annealing," *A I I E Transactions*, vol. 19, no. 1, pp. 107–119, 1987.
[30] D. T. Connolly, "An improved annealing scheme for the qap," *European Journal of Operational Research*, vol. 46, no. 1, pp. 93–100, 1990.
[31] A. Bölte and U. W. Thonemann, "Optimizing simulated annealing schedules with genetic programming," *European Journal of Operational Research*, vol. 92, no. 2, pp. 402–416, 1996.
[32] A. Misevičius, *A Modified Simulated Annealing Algorithm for the Quadratic Assignment Problem*. IOS Press, 2003.
[33] M. A. Kaviani, "A hybrid tabu search-simulated annealing method to solve quadratic assignment problem," *Decision Science Letters*, vol. 3, no. 3, 2014.
[34] S. Zhan, J. Lin, Z. Zhang, and Y. Zhong, "List-based simulated annealing algorithm for traveling salesman problem," *Computational Intelligence and Neuroscience,2016,(2016-3-13)*, vol. 2016, no. 5, p. 8, 2016.
[35] F. Larumbe and B. Sanso, "A tabu search algorithm for the location of data centers and software components in green cloud computing networks," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 22–35, 2013.
[36] Y. Peng, B. H. Soong, and L. Wang, "Broadcast scheduling in packet radio networks using mixed tabu-greedy algorithm," *Electronics Letters*, vol. 40, no. 6, pp. 375–376, 2004.
[37] G. Cavuslar, B. Catay, and M. S. Apayd?n, "A tabu search approach for the nmr protein structure-based assignment problem," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 6, pp. 1621–1628, 2012.
[38] J. Mccall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205–222, 2005.
[39] L. Wang, S. Arunkumaar, and W. Gu, "Genetic algorithms for optimal channel assignment in mobile communications," in *International Conference on Neural Information Processing*, pp. 1221–1225 vol.3, 2002.
[40] X. Fu and L. Wang, "Rule extraction by genetic algorithms based on a simplified rbf neural network," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, pp. 753–758 vol. 2, 2001.
[41] L. Wang, L. Zhou, and W. Liu, "Fpga segmented channel routing using genetic algorithms," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, pp. 2161–2165 Vol. 3, 2005.
[42] S. Li, S. C. Lay, W. H. Yu, and L. Wang, "Minimizing interference in mobile communications using genetic algorithms," in *Computational Science - Iccs 2002, International Conference, Amsterdam, the Netherlands, April 21-24, 2002. Proceedings*, pp. 960–969, 2002.
[43] M. Dorigo, M. Birattari, and T. Sttzle, "Ant colony optimization artificial," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.
[44] W. Pan and L. Wang, "An ant colony optimization algorithm based on the experience model," in *International Conference on Natural Computation*, pp. 13–18, 2009.
[45] B. Li, L. Wang, and W. Song, "Ant colony optimization for the traveling salesman problem based on ants with memory," in *International Conference on Natural Computation*, pp. 496–501, 2008.
[46] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, vol. 344, no. 2-3, pp. 243–278, 2005.
[47] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, 2002.
[48] L. Wang and G. Si, "Optimal location management in mobile computing with hybrid genetic algorithm and particle swarm optimization (ga-pso)," in *IEEE International Conference on Electronics, Circuits, and Systems*, pp. 1160–1163, 2010.
[49] X. Fu, S. Lim, L. Wang, and G. Lee, "Key node selection for containing infectious disease spread using particle swarm optimization," in *Swarm Intelligence Symposium, 2009. SIS '09. IEEE*, pp. 109–113, 2009.