ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/233584468

An Effective Simulated Annealing Algorithm for Solving the Traveling Salesman Problem

Article *in* Journal of Computational and Theoretical Nanoscience · July 2009

DOI: 10.1166/jctn.2009.1230

CITATIONS
34

READS
8,186

3 authors:

Zicheng Wang
Harbin Institute of Technology
92 PUBLICATIONS **760** CITATIONS
SEE PROFILE

Xiutang Geng
Huazhong University of Science and Technology
13 PUBLICATIONS **356** CITATIONS
SEE PROFILE

Zehui Shao
Guangzhou University
163 PUBLICATIONS **1,899** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

rainbow domination in graphs View project

Electromagnetic protection material View project

All content following this page was uploaded by Zehui Shao on 31 May 2017.

The user has requested enhancement of the downloaded file.

Copyright © 2009 American Scientific Publishers
All rights reserved
Printed in the United States of America

Journal of
Computational and Theoretical Nanoscience
Vol. 6, 1680–1686, 2009

An Effective Simulated Annealing Algorithm for Solving the Traveling Salesman Problem

Zicheng Wang[1], Xiutang Geng[1], and Zehui Shao[2,*]

[1] Department of Control Science and Engineering, Huazhong University of Science and Technology, 430074, China
[2] School of Information Science and Technology, Chengdu University, Chengdu, 610106, China

It is well known that the traveling salesman problem (TSP) is one of the most studied NP-complete problems, and evolutionary technique such as simulated annealing has mostly been used to solve various NP-complete problems. In this paper, a two-stage simulated annealing (two-stage SA) algorithm is proposed to solve the TSP, and the two-stage SA algorithm is made up of two stages. In the first stage, a simple simulated annealing (simple SA) algorithm is proposed to obtain some appropriate solutions or closed tours. In the second stage, an effective simulated annealing (effective SA) algorithm is proposed to obtain solutions with good quality based on the solutions or closed tours obtained by the simple SA algorithm. To assess the effectiveness of the two-stage SA algorithm, simulations were carried out on 23 benchmark TSP instances. The simulation results show that the two-stage SA algorithm can obtain solutions with better quality than four recent algorithms such as ant colony optimization algorithm, self-organizing maps algorithm, particle swarm optimization algorithm and constructive optimizer neural network algorithm.

Keywords: Simulated Annealing Algorithm, NP-Complete Problem, Traveling Salesman Problem, Greedy Search.

1. INTRODUCTION

The TSP is undoubtedly one of the most important members of the large set of combinatorial optimization problems. At the same time, the TSP is an NP-complete problem,[14] whose computational complexity rises exponentially with the input size. Hence, it is advantageous to find suboptimal solutions with a reasonable cost in many of TSP applications such as vehicle routing,[9] data transmission in computer networks,[13] image processing and pattern recognition,[5] robot navigation,[6] and drilling problem.[8] The TSP can be understood as a search for the shortest closed tour that visits each city once and only once. Until now, many methods based on deterministic or probabilistic heuristics have been proposed to solve the TSP, and the methods include simulated annealing,[7] tabu search,[1] genetic algorithms,[4,11] ant colony optimization,[2,3] self-organizing maps,[22] particle swarm optimization[18] and constructive optimizer neural networks.[15]

Since each of the above methods has strong points as well as weak points, it is nontrivial to determine a superior approach. For example, good quality solutions can be obtained by evolutionary methods such as genetic algorithm, while neural networks approaches have competitive CPU time. Method such as simulated annealing provides good compromise between CPU time and quality of solution. As we know, the simulated annealing algorithm can give good quality solutions at cost long CPU time. Therefore, it is of interest to develop a two-stage SA algorithm, which can give good quality solutions with short CPU time.

In this paper, a two-stage SA algorithm is proposed to solve the TSP. The two-stage SA algorithm is made up of two stages. For the first stage, a simple SA algorithm is proposed to obtain some appropriate solutions or closed tours. For the second stage, an effective SA algorithm is proposed to obtain solutions with good quality by using the closed tours obtained by the simple SA algorithm. To assess the effectiveness of the two-stage SA algorithm, simulations were carried out on 23 benchmark TSP instances, and the sizes of instances range from 51 to 442. The simulation results show that the two-stage SA algorithm works well. The solutions obtained by the two-stage SA algorithm have better average accuracy and lower average percent difference for the benchmark TSP instances than those obtained by four recent algorithms such as ant colony optimization algorithm,[3] self-organizing maps algorithm,[22] particle swarm optimization algorithm[18] and constructive optimizer neural network algorithm.[15]

*Author to whom correspondence should be addressed.

1546-1955/2009/6/1680/007 doi:10.1166/jctn.2009.1230

The rest of the paper is organized as follows. In the next section, the simulated annealing technique is briefly described. The first stage of the two-stage SA algorithm (i.e., a simple SA algorithm) for the TSP is presented in Section 3. The second stage of the two-stage SA algorithm (i.e., an effective SA algorithm) for the TSP is presented in Section 4. Experimental results and comparisons are given in Section 5. The paper is concluded in Section 6.

2. SIMULATED ANNEALING TECHNIQUE

Kirkpatrick et al.[16] first proposed simulated annealing as a powerful stochastic search technique. Simulated annealing, as its name implies, exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure and the search for a minimum in a more general system. In combinatorial optimization, simulated annealing technique uses the analogous cooling operation for transforming a poor, unordered solution into an ordered, desirable solution, so as to optimize the objective function.

The major advantage of simulated annealing is the ability to avoid trapped at local minima. The simulated annealing algorithm employs a random search, which not only accepts changes that decrease objective function f, but also accepts some changes that increase f. The latter are accepted with probability ρ, see formula (1).

$$\rho = e^{-\Delta F/t} \tag{1}$$

Where ΔF denotes the increase in the objective function f, and t denotes the current state temperature.

The basic components of the simulated annealing algorithm are as follows:

Configuration: Current best solution.

Cost function: An objective function to measure how well the system performs when a certain configuration is given.

Move set: A generator of random changes in the configuration.

Cooling schedule: An initial temperature and rules for cooling it as the search progresses.

The quality of the simulated annealing algorithm largely depends on the design of the move set and cooling schedule. The results are the trade-off between the convergence speed and the quality of solution, i.e., a fast cooling schedule often ends up in a local optimum whereas a slow cooling schedule requires a tremendous computation time. In the past twenties years, simulated annealing technique had mostly been used to solve various NP-hard optimization problems, e.g., Refs. [12, 17, 19–21].

3. A SIMPLE SA ALGORITHM FOR THE TSP

Let $P = \{c_1, c_2, \ldots, c_n\}$ be a random closed tour among n cities, where $c_i \in \{1, 2, \ldots, n\}$, and if $i \neq k$, then $c_i \neq c_k$. Let $f(P)$ be the function of the closed tour P, see formula (2),

$$f(P) = \sum_{i=1}^{n-1} d_{c_i, c_{i+1}} + d_{c_n, c_1} \tag{2}$$

Where d_{c_i, c_k} denotes the distance between cities c_i and c_k. Then, the objective of the TSP is to minimize the formula (2) by trying random variation of the order of closed tour P.

The detailed procedures of the simple SA algorithm for the TSP are given as follows:

Step 1: Initialize parameters: Set initial temperature t_{i-1}, end temperature t_{e-1} and cooling coefficient α_1 of temperature.

Step 2: Generate an initial solution: Randomly select a city x as the first city c_1^0 ($c_1^0 = x$) of the closed tour, select the nearest city from the remainder cities as the second city c_2^0 of the closed tour till the last city of the closed tour is selected as the last city c_n^0 of the closed tour. Finally, the initial solution $P_0 = \{c_1^0, c_2^0, \ldots, c_n^0\}$ is generated. Then compute the objective value $S_0 = f(P_0)$ according to formula (2).

Step 3: Generate a new solution: Randomly select three different positive integers x_1, x_2 and x_3, where $x_1, x_2, x_3 \in \{1, 2, \ldots, n\}$. Let

$$d_{c_{city1}, c_{city1 \bmod (n)+1}} = \max\{d_{c_{x1}, c_{x1 \bmod (n)+1}}, d_{c_{x2}, c_{x2 \bmod (n)+1}}, d_{c_{x3}, c_{x3 \bmod (n)+1}}\} \tag{3}$$

Randomly select three different positive integer y_1, y_2 and y_3, where $y_1, y_2, y_3 \neq city1$, $y_1, y_2, y_3 \in \{1, 2, \ldots, n\}$. Let

$$d_{c_{city1}, c_{city2}} = \min\{d_{c_{city1}, c_{y1}}, d_{c_{city1}, c_{y2}}, d_{c_{city1}, c_{y3}}\} \tag{4}$$

Where $city2 \in \{1, 2, \cdots, n\}$, $city2 \neq city1$, $city2 \neq (city1 - 2 + n) \bmod (n) + 1$, $city2 \neq city1 \bmod (n) + 1$. If $city1 < city2$, then

$$c_i' = \begin{cases} c_i & 1 \leq i \leq city1 \\ c_{city2} & i = city1 + 1 \\ c_{i-1} & city1 + 1 < i \leq city2 \\ c_i & city2 < i \leq n \end{cases} \tag{5}$$

Otherwise

$$c_i' = \begin{cases} c_i & 1 \leq i < city2 \\ c_{i+1} & city2 \leq i < city1 \\ c_{city2} & i = city1 \\ c_i & city1 < i \leq n \end{cases} \tag{6}$$

Where closed tour $P' = \{c_1', c_2', \ldots, c_n'\}$ is a new solution or a new closed tour.

Step 4: Update the objective value: Compute the new objective value $f(P')$. If $f(P') < S_0$, then $S_0 = f(P')$ and $P_0 = P'$.

Step 5: Accept or reject the new solution: Let $\Delta F = f(P') - f(P_0)$. If $\Delta F \leq 0$, $P_0 = P'$, i.e., P' is accepted. Otherwise, if $\Delta F > 0$ continuously for more than $g_{\text{score}-1}$ times (i.e., greedy search times), compute the probability of accepting the new solution P' according to formula (1), i.e., P' is accepted with probability ρ, but if P' is rejected continuously for more than $s_{\text{score}-1}$ times (i.e., satisfying search times), $P_0 = P'$, i.e., P' is accepted.

Step 6: Continue or end the simple SA algorithm: Compute the current temperature t, where $t = \alpha_1 t$. If $t < t_{e-1}$, the simple SA algorithm ends. Otherwise, return to step 3.

4. AN EFFECTIVE SA ALGORITHM FOR THE TSP

In this section, we describe the second stage of the two-stage SA algorithm (i.e., an effective SA algorithm) for solving the TSP. The main idea of the effective SA algorithm is base on the information between two different cities, and the information comes from the solutions or closed tours obtained by the simple SA algorithm in the first stage. Firstly, some appropriate solutions or closed tours are obtained by the simple SA algorithm in the first stage. Secondly, according to the information between two different cities, a restriction local search strategy is used in the effective SA algorithm to obtain good quality solutions with short CPU time in the second stage. The allocation of information between two cities is described as follows.

Let the m solutions or closed tours obtained by the simple SA algorithm be $P_1, P_2, \ldots, P_m$, where $P_i = \{c_1^i, c_2^i, \ldots, c_n^i\}$. Let $\tau_{i,k}$ denote the information between cities i and k. Then $\tau_{i,k}$, which restricts the local search in the effective SA algorithm, can be described in formula (7),

$$\tau_{c_i^k, c_{i \bmod (n)+1}^k} = \sum_{k=1}^{m} \sum_{i=1}^{n} \frac{l_{\text{opt}}}{l_k} \tag{7}$$

Where l_k denotes the length of the closed tour P_k, l_{opt} denotes the optimal closed tour length of the TSP instance.

The detailed procedures of the effective SA algorithm for the TSP are briefly illustrated as follows:

Step 1: Initialize parameters: Set initial temperature t_{i-2}, end temperature t_{e-2} and cooling coefficient α_2 of temperature.

Step 2: Generate an initial solution: Please see the step 2 in the simple SA algorithm.

Step 3: Generate a new solution: Randomly select a positive integer x_1, where $x_1 \in \{1, 2, \ldots, n\}$. Select another positive integer y_1 with probability η according to formula (8),

$$\eta = \frac{\pi_{C_{x_1}, C_{y_1}}}{\sum_{i=1}^{i \neq x_1}} \tag{8}$$

Where the second positive integer $y_1 \in \{1, 2, \ldots, n\}$, $y_1 \neq x_1$, $y_1 \neq (x_1 - 2 + n) \bmod (n) + 1$, $y_1 \neq x_1 \bmod (n) + 1$. Let $city1 = x_1$ and $city2 = y_1$. If $city1 < city2$, P' is set by formula (5). Otherwise P' is set by formula (6). Where the new solution is $P' = \{c_1', c_2', \ldots, c_n'\}$.

Step 4: Update the objective value: Please see the step 4 in the simple SA algorithm.

Step 5: Accept or reject the new solution: Let $\Delta F = f(P') - f(P_0)$. If $\Delta F \leq 0$, $P_0 = P'$, i.e., P' is accepted. Otherwise, if $\Delta F > 0$ continuously for more than $g_{\text{score}-2}$ times (i.e., greedy search times), compute the probability of accepting the new solution P' according to formula (1), i.e., P' is accepted with probability ρ, but if P' is rejected continuously for more than $s_{\text{score}-2}$ times (i.e., satisfying search times), $P_0 = P'$, i.e., P' is accepted.

Step 6: Continue or end the effective SA algorithm: Compute the current temperature t, where $t = \alpha_2 t$. If $t < t_{e-2}$, the effective SA algorithm ends. Otherwise, return to step 3.

5. COMPUTATIONAL EXPERIMENTAL DETAILS

To verify the validity of the two-stage SA algorithm, some benchmark TSP instances from the TSPLIB library[10] are selected for simulations. The experiments have been performed on a PC with CPU 1.86 GHz processor and 512 M memory. Each instance is run for ten times. The parameters used in the two-stage SA algorithm are listed in Table I, and the percent difference δ has been computed using

$$\delta = \frac{l_{\text{aver}} - l_{\text{opt}}}{l_{\text{opt}}} \times 100 \tag{9}$$

Where l_{aver} denotes the average length of closed tours obtained by the two-stage SA algorithm.

5.1. The Set of the Parameters in the Two-Stage SA Algorithm

In Table I, for the two-stage SA algorithm, the cooling coefficient of temperature is slow. As a result,

Table I. Parameters used in the two-stage SA algorithm

n	The size of the benchmark TSP instance
t_{i-1}	Initial temperature in the simple SA algorithm
t_{e-1}	End temperature in the simple SA algorithm
α_1	Cooling coefficient of temperature in the simple SA algorithm
$g_{\text{score}-1}$	The times of the greed search in the simple SA algorithm
$s_{\text{score}-1}$	The times of the satisfying search in the simple SA algorithm
m	The numbers of the solutions or closed tours obtained by the simple SA algorithm
t_{i-2}	Initial temperature in the effective SA algorithm
t_{e-2}	End temperature in the effective SA algorithm
α_2	Cooling coefficient of temperature in the effective SA algorithm
$g_{\text{score}-2}$	The times of the greed search in the effective SA algorithm
$s_{\text{score}-2}$	The times of the satisfying search in the effective SA algorithm

the difference between the initial temperature and the end temperature should be small ($t_{i-1} = t_{i-2} = 200$ and $t_{e-1} = t_{e-2} = 0.1$). The numbers of the solutions or closed tours obtained by the simple SA algorithm should be vary following the size of the TSP instance to obtain good solution. In this paper, for ranges $51 \leq n \leq 99$, $100 \leq n \leq 399$ and $400 \leq n \leq 783$, m should be 25, 50 and 100, respectively. At the same time, for the three ranges of n, the cooling coefficient of temperature in the simple SA algorithm would be 0.99998, 0.999993 and 0.999998, respectively, while the cooling coefficient of temperature in the effective SA algorithm would be 0.99998, 0.999999 and 0.9999995, respectively. The greed search times in the first stage and the second stage of the two-stage SA algorithm are same. In order to obtain solutions with good quality, the greed search times would be eight ($g_{\text{score}-1} = g_{\text{score}-2} = 8$). The satisfying search times in the first stage should be bigger ($s_{\text{score}-1} = 3n$). But, in order to attain faster convergence speed, the satisfying search times in the second stage should be smaller because of the strong restriction, which is used in the effective SA algorithm, see step 3 in Section 4 ($s_{\text{score}-2} = n/4$).

5.2. Comparing with the CONN Algorithm

In Table II, for comparing the two-stage SA algorithm with the constructive optimizer neural network (CONN) algorithm[15] in terms of CPU time, we scale the CPU time of the CONN algorithm by an appropriate scaling coefficient 0.7527 related its processing system. The simulation results of the two-stage SA algorithm were compared with those obtained by the CONN algorithm on 23 benchmark TSP instances. The numbers of cities range from 51 to 783. It may be observed from Table II and Figure 1 that the CONN algorithm takes shorter CPU time with average time 0.4215 second than the two-stage SA algorithm with average CPU time 782.696 seconds. As a matter of fact, the CONN algorithm is the best CPU time algorithm for the TSP. However, the latter gives lower average percent difference within reasonable CPU time than the former. As is shown in Table II and Figure 2 for 21 benchmark TSP instances out of the 23 benchmark TSP instances, the two-stage SA algorithm can find better solutions than the CONN algorithm. In addition, for benchmark TSP instance pr76, the two-stage SA algorithm can find the best known solution **108159**; and for benchmark TSP instance pr107, the old best known solution 44303 is replaced by the new best known solutions **44301.7**.

5.3. Comparing with the Simple SA Algorithm

In order to show the effectiveness of the two-stage SA algorithm, the results obtained by the two-stage SA algorithm are compared with the results obtained by the simple

Table II. The simulation results of the two-stage SA algorithm were compared with those obtained by a constructive optimizer neural network (CONN) algorithm[15] for the 23 benchmark TSP instances.

				CONN		Two-Stage SA				
ID	Instances	Best known solution	n	δ	Time (s)	Best	Worst	Average	δ	Time (s)
1	eil51	426	51	2.58	0.0452	428.872	430.244	429.045	0.71	4.87
2	berlin52	7542	52	6.03	0.0527	7544.37	7544.37	7544.37	0.03	8.32
3	st70	675	70	2.96	0.0677	677.11	685.83	680.95	0.88	6.50
4	eil76	538	76	5.02	0.0753	544.369	555.513	548.159	1.89	7.11
5	pr76	108159	76	4.34	0.0828	**108159**	113282	109679	1.40	10.00
6	rat99	1211	99	0.33	0.1054	1219.24	1251.79	1230.99	1.65	10.34
7	kroA100	21282	100	2.57	0.1204	21285.4	21285.4	21285.4	0.02	248.845
8	rd100	7910	100	3.59	0.1204	7910.4	7944.32	7917.27	0.09	221.386
9	lin105	14379	105	0.38	0.1129	14383	14535.3	14408.2	0.20	251.26
10	pr107	44303	107	2.77	0.0903	**44301.7**	44301.7	44301.7	—	205.153
11	bier127	118282	127	2.45	0.1505	118705	119720	119171	0.75	321.48
12	ch130	6110	130	4.66	0.1656	6114.01	6163.49	6139.84	0.49	273.02
13	pr136	96772	136	2.27	0.1430	97074	99339.6	98133.8	1.41	342.16
14	kroA150	26524	150	4.78	0.1806	26600.5	27622.8	26830.5	1.16	350.88
15	pr152	73682	152	0.79	0.1882	73821.3	76064.5	74861.4	1.60	397.06
16	rat195	2323	195	5.64	0.2409	2354.1	2407.93	2378.83	2.40	387.70
17	d198	15780	198	4.16	0.2710	15930	16048.7	15986.7	1.31	410.83
18	kroA200	29368	200	4.40	0.2860	29687.6	30139.8	29975.2	2.07	451.43
19	a280	2579	280	3.57	0.4516	2611.79	2707.51	2659.52	3.12	486.26
20	pcb442	50778	442	5.56	0.9559	51345.4	52612.7	52213.9	2.83	1253.55
21	u574	36905	574	5.90	1.5355	38082.6	39034.7	38518.4	4.37	3472.08
22	d657	48912	657	7.58	1.8441	50271.2	51458.9	50746.7	3.75	4140.91
23	rat783	8806	783	7.59	2.4086	9127.4	9331.26	9240.02	4.93	4740.86
Average			3.91	0.4215					1.61	782.696

Solution quality of the CONN algorithm was quoted from Ref. [15], while its CPU time was scaled by 0.7527.

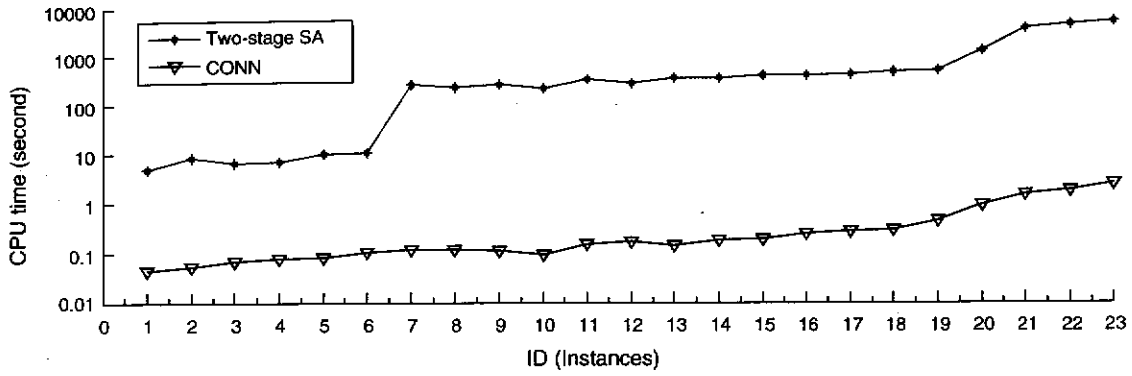

Fig. 1. Comparing the two-stage SA algorithm with the CONN algorithm for 23 benchmark TSPs from TSPLIB in terms of CPU time.

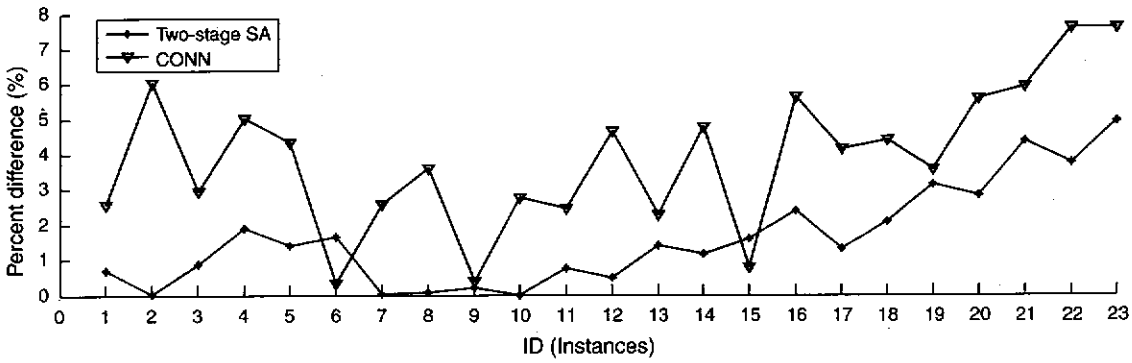

Fig. 2. Comparing the two-stage SA algorithm with the CONN algorithm for 23 benchmark TSPs from TSPLIB in terms of percent difference.

SA algorithm, slower cooling coefficient of temperature should be used in the simple SA algorithm. Cooling coefficient of temperature in the simple SA algorithm is 0.9999978 ($\alpha_1 = 0.9999978$), while cooling coefficient of temperature in the two-stage SA algorithm is 0.99998 ($\alpha_1 = \alpha_2 = 0.99998$). To compare the two-stage SA algorithm with the simple SA algorithm in terms of CPU time and percent difference, the simulations were carried on 5 benchmark TSP instances. The numbers of cities range from 51 to 76. It may be observed from Table III and Figure 3 that the two-stage SA algorithm takes shorter CPU time with average 7.36 seconds than the simple SA algorithm with average 9.656 seconds. At the same time, the former gives lower average percent difference with 0.982 than the latter with 2.57, see Table III and Figure 4.

Table III. The simulation results of the two-stage SA algorithm were compared with those obtained by the simple SA algorithm for the 5 benchmark TSP instances.

	Two-stage SA algorithm		Simple SA algorithm	
ID	δ	Time (s)	δ	Time (s)
1	0.71	4.87	1.50	8.53
2	0.03	8.32	1.90	9.50
3	0.88	6.50	1.99	9.44
4	1.89	7.11	4.85	9.19
5	1.40	10.00	2.61	11.62
Average	**0.982**	**7.36**	**2.57**	**9.656**

Each instance is run for ten times for the simple SA algorithm and the two-stage SA algorithm.

5.4. Comparing with Other Algorithms

In Table IV, the simulation results of the two-stage SA algorithm were compared with those obtained by three recent algorithms such as a self-organizing maps (MGSOM) algorithm,[22] an ant colony (ACOMAC+DNN) algorithm[3] and a particle swarm optimization (PSO) algorithm.[18]

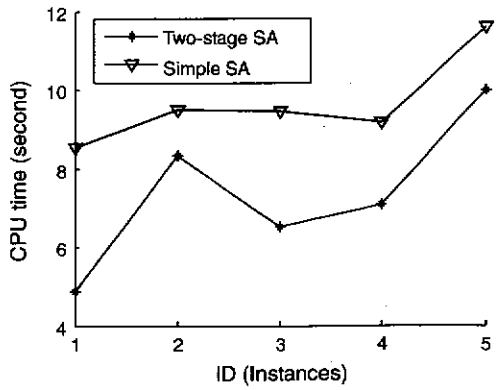

Fig. 3. Comparing the two-stage SA algorithm with the simple SA algorithm for 5 benchmark TSPs from TSPLIB in terms of CPU time.

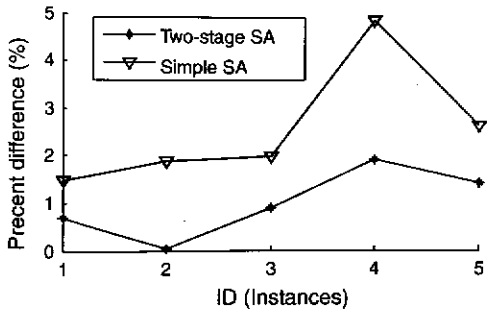

Fig. 4. Comparing the two-stage SA algorithm with the simple SA algorithm for 5 benchmark TSPs from TSPLIB in terms of percent difference.

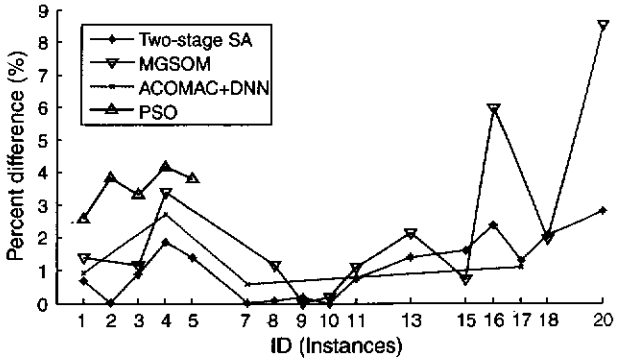

Fig. 5. Comparing the two-stage SA algorithm with the algorithms such as MGSOM, ACOMAC and PSO for total 16 benchmark TSPs from TSPLIB in terms of percent difference.

Firstly, the simulation results of the two-stage SA algorithm were compared with those obtained by the MGSOM algorithm[22] for the 12 benchmark TSP instances. The numbers of cities range from 51 to 442. The Table IV gives information that the two-stage SA algorithm can obtain lower average percent difference than the MGSOM algorithm. In addition, as illustrated in Figure 5, for 9 benchmark TSP instances out of the 12 benchmark TSP instances, the two-stage SA algorithm can find better solutions than the MGSOM algorithm.

Secondly, the simulation results of the two-stage SA algorithm were compared with those obtained by the ACOMAC+DNN algorithm[3] for the four benchmark TSP instances. The numbers of cities range from 51 to 198. The experimental results showed in Table IV demonstrate that the two-stage SA algorithm gives lower average percent difference with 0.9825 than the ACOMAC+DNN algorithm with 1.3407. For three benchmark TSP instances out of the four benchmark TSP instances, Figure 5 illustrates the two-stage SA algorithm can find better solutions than the ACOMAC+DNN algorithm.

Finally, the simulation results of the two-stage SA algorithm were compared with those obtained by the PSO algorithm[18] for the five benchmark TSP instances. The numbers of cities range from 51 to 76 (small scale). As we can see from Table IV, the two-stage SA algorithm can obtain lower average percent difference with 0.982 than the PSO algorithm with 3.5496. As shown in Figure 5, the two-stage SA algorithm can find better solutions than the PSO algorithm for the five instances.

6. CONCLUSION

In this paper, a two-stage SA algorithm for the TSP is proposed. To access the effectiveness of the two-stage SA algorithm, the two-stage SA algorithm is tested on 23 benchmark TSP instances. The simulation results show that the two-stage SA algorithm outperforms the four recent algorithms such as ant colony optimization

Table IV. The simulation results of the two-stage SA algorithm were compared with those obtained by the self-organizing maps algorithm (MGSOM),[22] an ant colony optimization algorithm (ACOMAC+DNN)[3] and a particle swarm optimization algorithm (PSO)[18] for the 16 benchmark TSP instances.

ID	MGSOM	Two-stage SA	ACOMAC+DNN	Two-stage SA	PSO	Two-stage SA
1	1.3983	0.71	0.9408	0.71	2.5751	0.71
2	—	—	—	—	3.8458	0.03
3	1.1835	0.88	—	—	3.3422	0.88
4	3.3840	1.89	2.7161	1.89	4.1673	1.89
5	—	—	—	—	3.8176	1.40
7	—	—	0.5931	0.02	—	—
8	1.1718	0.09	—	—	—	—
9	0.0278	0.20	—	—	—	—
10	0.1719	0.0	—	—	—	—
11	1.0936	0.75	—	—	—	—
13	2.1530	1.41	—	—	—	—
15	0.7412	1.60	—	—	—	—
16	5.9847	2.40	—	—	—	—
17	—	—	1.1128	1.31	—	—
18	1.9715	2.07	—	—	—	—
20	8.5770	2.83	—	—	—	—
Average δ	2.3215	1.2358	1.3407	0.9825	3.5496	0.982

Solutions quality of the algorithms such as MGSOM, ACOMAC+DNN and PSO were quoted from Refs. [22], [3], and [18], respectively.

View publication stats

algorithm,[3] self-organizing maps algorithm,[22] particle swarm optimization algorithm[18] and constructive optimizer neural network algorithm.[15]

However, from Table II, for benchmark instance rat783, it can be seen that the two-stage SA algorithm takes more than an hour (4740.86 seconds) for each run. Compared with the constructive optimizer neural network algorithm,[15] slow convergence speed is the disadvantage of the two-stage SA algorithm. Hence, the two-stage SA algorithm is deserved to be further investigated to improve its convergence speed. In addition, as we known, the adjustment of some parameters such as initial temperature, end temperature, the cooling coefficient of the temperature, greed search times, satisfying search times and the numbers of the solutions or closed tours obtained by the simple SA algorithm in the first stage of the two-stage SA algorithm could be useful in obtaining good solutions. So the adjustment of the parameters may be an interesting subject in the future work.

Acknowledgments: The research was supported in part by the National Natural Science Foundation of China (Grant Nos. 60373089, 60674106, and 60533010), the Program for New Century Excellent Talents in University (NCET-05-0612), the Ph.D., Programs Foundation of Ministry of Education of China (20060487014), the Chenguang Program of Wuhan (200750731262), and the National High Technology Research and Development 863 Program (2006AA01Z104).

References

1. A. Misevioius, *Informacines Technologijos ir Valdymas* 3, 29 (**2004**).
2. C. B. Cheng and C. P. Mao, *Mathematical and Computer Modeling* 46, 1225 (**2007**).
3. C. F. Tsai, C. W. Tsai, and C. C. Tseng, *Information Sciences* 166, 67 (**2004**).
4. C. Moon, J. Kim, G. Choi, and Y. Seo, *European Journal of Operational Research* 140, 606 (**2002**).
5. D. Banaszak, G. A. Dale, A. N. Watkins, and J. D. Jordan, An optical technique for detecting fatigue cracks in aerospace structures. *Proc. 18th ICIASF 1999* (**1999**), Vol. 27, pp. 1–7.
6. D. Barrel, J. P. Perrin, E. Dombre, and A. Liengeois, An evolutionary simulated annealing algorithm for optimizing robotic task ordering. *Proc. IEEE ISATP 1999* (**1999**), pp. 157–162.
7. F. Tian and L. Wang, Chaotic simulated annealing with augmented lagrange for solving combinatorial optimization problems. *Proc. 26th Annual IECON* (**2000**), Vol. 4, 2722–2725.
8. G. C. Onwubolu and M. Clerc, *International Journal of Production Research* 42, 473 (**2004**).
9. G. Laporte, *European Journal of Operational Research* 59, 345 (**1992**).
10. G. Reinelt, *ORSA Journal on Computing* 3, 376 (**1991**).
11. L. Jiao and L. Wang, *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans* 30, 552 (**2000**).
12. M. J. Ji, Z. H. Jin, and H. W. Tang, *Applied Mathematics and Computation* 183, 251 (**2006**).
13. M. K. M. Ali and F. Kamoun, *IEEE Transactions on Neural Networks* 4, 941 (**1993**).
14. P. Crescenzi and V. Kann, A Compendium of NP Optimization Problems, October (**1995**). [Online]. Available: http://www.zvne.fer.hr/~zmija/resources/science_resources/nn_for_optimization/index.html.
15. M. Saadatmand-Tarzjan, M. Khademi, M. R. Akbarzadeh-T, and H. A. Mghaddam, *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics* 37, 754 (**2007**).
16. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Science* 220, 671 (**1983**).
17. T. Poranen, *Information Sciences* 172, 155 (**2005**).
18. X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and Q. X. Wang, *Information Processing Letters* 103, 169 (**2007**).
19. X. R. Chen, N. C. Pahalawaththa, U. D. Annakkage, and C. S. Kumble, *Electric Power System Research* 39, 67 (**1996**).
20. X. S. Xu and J. Ma, *Neurocomputing* 69, 913 (**2006**).
21. X. T. Geng, J. Xu, J. H. Xiao, and L. Q. Pan, *Information Sciences* 177, 5064 (**2007**).
22. Y. Bai, W. Zhang, and Z. Jin, A New Self-Organizing Maps Strategy for Solving the Traveling Salesman Problem, Chaos, Solitons and Fractals (**2006**), Vol. 28, pp. 1082–1089.

Received: 20 January 2009. Accepted: 25 February 2009.