2008 International Symposium on Nonlinear Theory and its Applications
NOLTA'08, Budapest, Hungary, September 7-10, 2008

An Effective Chaotic Search by Using 2-opt and Or-opt Algorithms for Traveling Salesman Problem

Takafumi MATSUURA and Tohru IKEGUCHI

Graduate School of Science and Engineering, Saitama University
225 Shimo-Ohkubo, Sakura-ku, Saitama-city 338-8570, JAPAN
Email: takafumi@nls.ics.saitama-u.ac.jp, tohru@ics.saitama-u.ac.jp

Abstract—We have already proposed chaotic search methods to find near optimum solutions of TSPs. In these methods, local search method is driven by chaotic neurodynamics to avoid a local minimum. Due to effective control by the chaotic dynamics, the chaotic search methods shows good performance. In this paper, we propose a new method to solve TSP by using two different types of simple local searches, 2-opt algorithm and Or-opt algorithm. In the proposed method, the 2-opt and Or-opt algorithms are driven by the chaotic neurodynamics. As a result, the proposed method shows higher performance than the previous chaotic search methods.

1. Introduction

The traveling salesman problem (TSP) is one of the famous combinatorial optimization problems which is described as follows: given positions of cities, find the minimum length tour which visits each city exactly once. For an n-city TSP, the number of possible tours is $\frac{(n-1)!}{2}$. If n is increase, the number of all possible tours exponentially diverges. Then, the TSP generally belongs to a class of $\mathcal{NP}$-hard, and it is believed that it is almost impossible to obtain an optimal solution of the TSP in a reasonable time frame. Thus, it needs to develop effective algorithms for finding near optimum solutions or approximate solutions.

To find the near optimal solutions or approximate solutions for TSP, chaotic search methods for solving TSP [6, 7, 8] have been already proposed. In these methods, a chaotic dynamics is used to avoid local minima. To realize the chaotic dynamics, a chaotic neural network model constructed by chaotic neurons [5] is used. The chaotic neuron realizes an important biological feature of a real neuron—the refractory effect [5]. Due to the refractory effect, a neuron cannot fire for a certain period of time after it had fired or emitted a spike. Then, it has already been shown that good near optimal solutions can be found not only for TSP [6, 7, 8] but also for other $\mathcal{NP}$-hard problems such as the quadratic assignment problem [10] and the motif extraction problem [12, 13, 14, 15].

A simplest chaotic search method has already been proposed to solve TSP [6, 7, 8]. In this method, an execution of the 2-opt algorithm is driven by the chaotic neurodynamics. The 2-opt algorithm exchanges two paths for other two paths until no further improvement can be obtained (Fig. 1). However, a given tour by the 2-opt algorithm is not a global optimum but a local optimum. To jump from such a local optimum, we applied the chaotic neurodynamics to the 2-opt algorithm [6, 7, 8]. As a result, this method shows good results [6, 7, 8]. Next, to improve the performance of the chaotic search, adaptive k-opt algorithm driven by chaotic dynamics have been proposed. The adaptive k-opt algorithm is almost the same as the Lin-Kernighan algorithm [2] which is considered to be the best local search for the TSP. In this method, the value of k is not fixed but varied [7]: first, the 2-opt ($k = 2$) algorithm is applied to a current tour to improve the tour; second, if a positive gain value is obtained by the 2-opt algorithm, the 3-opt ($k = 3$) algorithm is applied to improve the tour by a deterministic rule [7]. While the positive gain value is obtained, the k-opt algorithm is applied by increasing the value of k. As a result, this chaotic search method [7] shows better solutions than the previous chaotic search [6, 8].

However, it is not so much effective to implement the adaptive k-opt algorithm because of its algorithmic complexity. In this sense, the chaotic search can be much improved by simpler algorithms or their combinations. In this paper, we proposed a new chaotic algorithm by introducing a simple local search, Or-opt [1]. The Or-opt improves the current tour by moving a partial tour of maximum three consecutive paths in a different location (Fig. 2). The Or-opt algorithm is considered to obtain solutions that are comparable to the 3-opt algorithm in terms of quality of solutions and an amount of time is closer to that of the 2-opt algorithm. As a result, the proposed method obtained better solution for previous chaotic search methods [8].

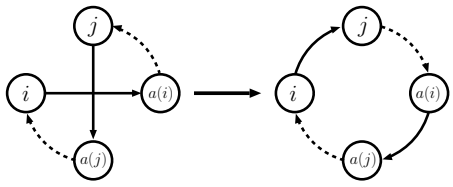

Figure 1: An example of the 2-opt algorithm. In this example, $a(i)$ is the next city to i. Two paths (i-$a(i)$ and j-$a(j)$) are deleted from the current tour, then new two paths, i-j and $a(j)$-$a(i)$, are added to obtain a shorter tour.

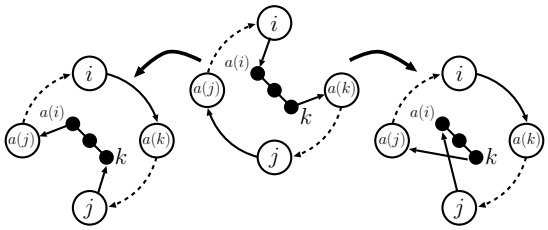

Figure 2: An example of the Or-opt algorithm. In this example, $a(i)$ is the next city to i. A partial tour $a(i)$-$\cdots$-k is inserted into another path (j-$a(j)$).

2. The proposed method

In the proposed method, two local search algorithms, the 2-opt and the Or-opt [1], are driven by the chaotic dynamics. To realize the method, a chaotic neuron [5] is assigned to each city. If a chaotic neuron fires, the local searches related to the corresponding city are carried out. The firing of the ith neuron $x_i(t)$ is defined by

$$x_i(t) = f(y_i(t)), \tag{1}$$

where $f(y) = 1/(1 + \exp(-y/\epsilon))$. $y_i(t)$ is the internal state of the ith chaotic neuron at time t. If $x_i(t) > \frac{1}{2}$, the ith neuron fires at the time t, otherwise, the neuron is resting. $y_i(t)$ is decomposed into two parts, $\zeta_i(t)$ and $\xi_i(t)$. Each component represents different factor to the dynamics of chaotic neurons, a gain effect and a refractory effect, respectively.

The gain effect is expressed as:

$$\xi_i(t+1) = \begin{cases} \max_j\{\beta_2 \Delta_{ij}(t) + \zeta_j(t)\} & \text{(2-opt)} \\ \max_{j,k}\{\beta_{\mathrm{Or}} \Delta_{ijk}(t) + \zeta_j(t)\} & \text{(Or-opt)} \end{cases}, \tag{2}$$

where β_2 and β_{Or} are scaling parameters; $\Delta_{ij}(t)$ is a difference between the length of a current tour and that of a new tour when city i and city j are connected after applying the 2-opt algorithm to city i (Fig.1); $\Delta_{ijk}(t)$ is a difference of the tour length in case of the Or-opt algorithm (Fig.2). $\zeta_j(t)$ is a refractory effect of the neuron j at time t which is defined by Eq. (3).

The second factor is a refractory effect which works to avoid the local minima. The refractory effect has a similar effect as a memory effect in the tabu search [3, 4]. In the tabu search, previous states are memorized in a tabu list. Then, to avoid a local minimum, the solutions in the tabu list are not allowed for a certain temporal duration called a tabu tenure. In case of chaotic search, past firings are memorized as previous states to decide strength of the refractory effect. The strength of the refractory effect increases just after firing and recovers exponentially with time. Thus, although the tabu search perfectly inhibits to go back to the same solutions for the certain temporal, the chaotic search might permit to select the same solutions if a corresponding neuron fires due to a larger gain than the refractory effect or an exponential decay of the refractory effect. The refractory effect is expressed as:

$$\begin{aligned} \zeta_i(t+1) &= -\alpha \sum_{d=0}^{t} k_r^d x_i(t-d) + \theta & (3) \\ &= k_r \zeta_i(t) - \alpha x_i(t) + \theta(1 - k_r), & (4) \end{aligned}$$

where α controls a strength of the refractory effect after the firing ($\alpha > 0$); the parameter k_r is a decay parameter of the refractory effect ($0 < k_r < 1$); θ is a threshold value. Then, $\zeta_i(t+1)$ expresses a refractory effect with a factor k_r, because the first term of the right hand side of Eq. (3) becomes negative, if the neuron frequently fires in its past history, which then depresses the value of $\zeta_i(t+1)$, and relatively leads the neuron state to a resting state.

The procedure of the proposed method is described as follows:

1. Given an n-city TSP, and make an initial solution by the nearest neighbor method.
2. Improve a current tour by the 2-opt algorithm driven by a chaotic dynamics.
   (a) A city i is selected from the neurons whose internal state has not been updated yet.
   (b) A city j is selected in such a way $\xi_i(t+1)$ is maximum.
   (c) If the ith neuron fires ($x_i(t+1) > \frac{1}{2}$), city i and city j are connected by the 2-opt algorithm. Then, if the best solution is obtained, 2-opt algorithm is applied for the tour until no further improvement can be obtained.
   (d) Repeated the steps (a)-(c) until there exist neurons whose internal state has not been updated yet.
3. Improve a current tour by the Or-opt algorithm driven by a chaotic dynamics.
   (a) A city i is selected from the neurons that have not updated yet.
   (b) Cities j and k are selected in such a way $\xi_i(t+1)$ is maximum.
   (c) If the ith neuron fires ($x_i(t+1) > \frac{1}{2}$), the Or-opt algorithm is executed for the city i. Then, if the best solution is obtained, the 2-opt algorithm is applied to the tour until no further improvement can be obtained.
   (d) Repeated the steps (a)–(c) until there exist neurons that have not been updated yet.
4. Finish one iteration, and repeated the steps 2 and 3 for sufficiently many times.
5. The Or-opt and the 2-opt algorithms are applied to the best solution until no further improvement can be obtained.

3. Results

To evaluate the performance of the proposed method, we solved benchmark problems in TSPLIB [16]. First, we confirmed the performance of the Or-opt algorithm.

Table 1 shows the results of the 2-opt algorithm, and the 2-opt and Or-opt algorithm for several types of the problems. From Table 1, the method introducing the Or-opt algorithm is obtained better solutions than the 2-opt algorithm for all problems. Table 2 shows the results of the conventional chaotic search methods: the 2-opt algorithm driven by chaotic dynamics which includes control and annealing of parameters [8]; an adaptive k-opt algorithm driven by chaotic dynamics which includes control and annealing of parameters [8, 7]. The adaptive k-opt algorithm is almost the same as the Lin-Kernighan algorithm which is considered to be the best local search for the TSP. Moreover, in these methods, a double bridge (DB) algorithm is applied to change the solution space, when a better solution could not be obtained within 100 iteration. The double bridge algorithm is a special case of the 4-opt algorithm. From Table 2, the chaotic dynamics improves the performance of the 2-opt algorithm. These results indicate that if we introduce the chaotic dynamics to avoid local minima for the 2-opt and Or-opt algorithm, it is possible to realize an effective chaotic search method.

In the proposed chaotic method, although many parameters exist, we set the same values of parameters in Eq.(3) for all instances: $\alpha = 0.95$, $k_r = 0.30$, $\theta = 1.0$ and $\epsilon = 0.002$. Then, scaling parameters in Eq(2) were changed depending on the problem size. The scaling parameters, β_2 and β_{Or}, are adjusted by the following equations:

$$\beta_2 = 0.0625 \times \sqrt{\frac{(\text{the density for an instance})}{0.00021418}}, \quad (5)$$

$$\beta_{\mathrm{Or}} = 0.0950 \times \sqrt{\frac{(\text{the density for an instance})}{0.00021418}}, \quad (6)$$

where 0.0625 and 0.0950 are the good parameters for pcb1173; 0.00021418 is the density of pcb1173 (Table 3). The key idea of Eqs.(5) and (6) is based on spatial ranges and spatial densities of city-distributions of a TSP instance. If the spatial range becomes larger, the value of $\Delta_{ij}(t)$ (or $\Delta_{ijk}(t)$) becomes larger. Then, the scaling parameters must be tuned to smaller values. In addition to the spatial range, the spatial density of the cities also affects the scaling of $\Delta_{ij}(t)$ (or $\Delta_{ijk}(t)$), because the value of $\Delta_{ij}(t)$ with lower densities is larger than that with higher density. In this paper, at first, we searched good parameters set for pcb1173. Then, we adjusted the scaling parameters for the other instances by Eqs.(5) and (6). Table 3 shows the scaling parameter for each instance. Table 4 shows the results of the proposed method. From Tables 2 and 4, the proposed method shows better solutions than the chaotic search methods based on the 2-opt algorithm for pcb442.

However, we cannot improve the performance of the conventional chaotic search method for the other instances.

Table 1: The results of the 2-opt algorithm, and the 2-opt and Or-opt algorithm. Each method is carried out until no further improvement can be obtained. Results are expressed by percentages of gaps between obtained solutions and the optimal solutions (%).

	2-opt			2-opt + Or-opt		
Problem	Ave.	Best	Worst	Ave.	Best	Worst
pcb442	7.473	4.858	10.258	3.970	2.229	6.588
pcb1173	9.885	8.541	11.167	6.238	4.415	8.079
pr2392	9.563	8.206	11.232	5.294	3.750	6.366
rl5915	9.395	8.206	11.416	6.244	5.345	7.436
rl11849	8.752	8.016	9.367	5.567	4.609	6.396

Table 2: The results of the conventional chaotic search: 2-opt based chaotic search [8]; 2-opt based chaotic search with double bridge [7], and the adaptive k-opt based chaotic search with double bridge [7] for $5,000$ iterations. Results are expressed by percentages of gaps between obtained solutions and the optimal solutions (%).

Problem	2-opt [8] Ave.	2-opt+DB [7] Ave.	adaptive k-opt+DB [7] Ave.
pcb442	1.034	0.982	0.825
pcb1173	1.692	1.748	1.569
pr2392	1.952	2.000	1.839
rl5915	2.395	2.273	1.742
rl11849	2.223	1.730	1.186

One of the possible reasons for the lower performance is that the proposed method does not implement an annealing effect. If we implement the annealing effect to the proposed method, the searching space is gradually limited as the simulated annealing [17], and the proposed method searches the solution space more deeply. Then, we replaced the scaling parameters in Eq.(2) by the following equations:

$$\beta_2(t+1) = \beta_2(t) + \lambda \quad \text{(2-opt)}, \quad (7)$$

$$\beta_{\mathrm{Or}}(t+1) = \beta_{\mathrm{Or}}(t) + \gamma \quad \text{(Or-opt)}. \quad (8)$$

$\beta_2(t)$ and $\beta_{\mathrm{Or}}(t)$ are new scaling parameters. Both of them increase in proportion to time t. Table 5 shows the scaling parameters of each instance. These parameters are adjusted by the density of each instance. The other parameters are set to the same values for all instances: $\alpha = 0.95$, $k_r = 0.30$, $\theta = 1.0$ and $\epsilon = 0.002$. Table 6 shows the results of the proposed method with the annealing effect. From Tables 2 and Table 6, proposed method obtains better solutions than the conventional chaotic search methods.

4. Conclusions

In this paper, we proposed a new method for solving the TSP using two local searches, the 2-opt algorithm and the

Table 3: The value of the scaling parameters in Eq.(2).

Problem	β_2	β_{Or}	Density
pcb442	0.026553	0.040360	0.00003877
pcb1173	0.062500	0.095000	0.00021481
pr2392	0.017839	0.027116	0.00001750
rl5915	0.022336	0.033995	0.00002744
rl11849	0.031916	0.048512	0.00005601

Table 4: The results of the proposed method for $5,000$ iterations. Results are expressed by percentages of gaps between obtained solutions and the optimal solutions (%). The middle column of this table shows the results without step 5 in the proposed method.

	without greedy			with greedy		
Problem	Ave.	Best	Worst	Ave.	Best	Worst
pcb442	0.912	0.323	1.721	0.854	0.313	1.660
pcb1173	2.039	1.454	2.821	1.838	1.134	2.821
pr2392	2.606	2.030	3.309	2.487	1.874	3.308
rl5915	2.895	1.906	3.927	2.775	1.906	3.927
rl11849	2.782	2.326	3.274	2.470	1.982	3.027

Table 5: The values of the scaling parameters for each instance.

Problem	$\beta_2(0)$	λ	$\beta_{\mathrm{Or}}(0)$	γ
pcb442	0.00339	0.0000084	0.00339	0.0000127
pcb1173	0.00800	0.0000200	0.00800	0.0000300
pr2392	0.00228	0.0000057	0.00228	0.0000085
rl5915	0.00285	0.0000071	0.00285	0.0000107
rl11849	0.00408	0.0000102	0.00408	0.0000153

Table 6: The results of the the proposed method with the annealing effect for $5,000$ iterations. Results are expressed by percentages of gaps between obtained solutions and the optimal solutions (%). The middle column of this table shows the results without step 5 in the proposed method.

	without greedy			greedy		
Problem	Ave.	Best	Worst	Ave.	Best	Worst
pcb442	0.469	0.021	1.085	**0.451**	0.021	0.906
pcb1173	0.898	0.520	1.489	**0.840**	0.436	1.366
pr2392	1.339	0.822	1.834	**1.153**	0.716	1.614
rl5915	1.430	0.912	1.962	**1.291**	0.824	1.825
rl11849	1.276	0.815	1.647	**1.160**	0.858	1.496

Or-opt algorithm, driven by chaotic neurodynamics. From the computational results, we confirmed that it obtains better solutions than the previous chaotic search methods. In the future work, it is important to clarify why the proposed method can improve performance with statistical methods [13]. To improve the proposed method, we also consider how to control different types of two local search algorithms.

Acknowledgments

The authors thank K. Aihara, Y. Horio, M. Adachi and M. Hasegawa for their valuable comments and discussions. The research of T.M. is partially supported by a Grant-in-Aid from the JSPS Fellows (No.20·6863). The research of T.I. is partially supported by a Grant-in-Aid for Scientific Research (B) (No.20300085) from the the JSPS.

References

[1] Or, I. (1967), Ph.D. thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, Illinois.

[2] Lin, S. and Kernighan, B. (1973): *Operations Research*, **21**, 498–516.

[3] Glover, F. (1989): *ORSA Jour. on Computing*, **1** (3), 190–206.

[4] Glover, F. (1990): *ORSA Jour. on Computing*, **2** (1), 4–32.

[5] Aihara, K. et al. (1990): *Phys. Lett. A*, **144**, 333–340.

[6] Hasegawa, M. et al. (1997): *PRL*, **79** (12), 2344–2347.

[7] Hasegawa, M. et al. (2001): *Tech. Rept. of IEICE*, **101** (229), 25–32.

[8] Hasegawa, M. et al. (2002): *Neural Networks*, **15** (2), 271–283.

[9] Hasegawa, M. et al. (2000): *Control and Cybernetics*, **29** (3), 773–788.

[10] Hasegawa, M. et al. (2002): *EJOR*, **39** (3), 543–556.

[11] Matsuura, T. et al. (2004): *Proc. of NCSP2004*, 347–350.

[12] Matsuura, T. et al. (2005): *Proc. of MIC2005*, 677–682.

[13] Matsuura, T. and Ikeguchi, T. (2006): *Lecture Notes in Computer Science*, **4224**, 1103–1110.

[14] Matsuura, T. et al. (2007): *Proc. of MIC2007*.

[15] Matsuura, T. et al. (2007): to appear in *Proc. of the 14th ICONIP*.

[16] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

[17] Kirkpatrick, S. et al. (1983): *Science*, **220**, 671–680.