2009 International Symposium on Nonlinear Theory and its Applications
NOLTA'09, Sapporo, Japan, October 18-21, 2009

Chaotic Search based on the Ejection Chain Method for Traveling Salesman Problems

Shun Motohashi†, Takafumi Matsuura† and Tohru Ikeguchi†

†Graduate School of Science and Engineering, Saitama University
255 Shimo-ohkubo, Sakura-ku, Saitama, 338-8570 Japan
Email: motohashi@nls.ics.saitama-u.ac.jp, takafumi@nls.ics.saitama-u.ac.jp, tohru@ics.saitama-u.ac.jp

Abstract—In the real world, we are often asked to solve difficult problems of combinatorial optimization: for example, vehicle routing, computer wiring, circuit drilling, scheduling, and so on. To solve these problems, the traveling salesman problem (TSP) has been studied widely. Because the TSP belongs to a class of NP-hard, it is required to develop an effective approximate algorithm for obtaining near optimal solutions in a reasonable time frame. As an approximate algorithm, a method using chaotic dynamics shows good performance. In this method, chaotic dynamics controls local search methods to avoid local minima. In this paper, we propose a new chaotic search method using the stem-and-cycle structure for solving TSPs. Evaluating performance, we show that the new chaotic search method is very effective for solving TSPs.

1. Introduction

The traveling salesman problem (TSP) is one of the most typical combinatorial optimization problems. The TSP is described as follows: given a set of n cities and distances d_{ij} between cities i and j, find an optimal solution, or a shortest-length tour. Thus, the goal of the TSP is to find a permutation σ of the cities that minimizes the following quantity:

$$\sum_{k=1}^{n-1} d_{\sigma(k)\sigma(k+1)} + d_{\sigma(n)\sigma(1)}, \tag{1}$$

where $\sigma(k)$ is the kth city in a tour. If $d_{ij} = d_{ji}$ for all i and j, the TSP is symmetric, otherwise asymmetric. In this paper, we deal with the symmetric TSP.

Because the TSP belongs to a class of $\mathcal{NP}$-hard, it is almost impossible to find optimal solutions in a reasonable time frame. Thus, it is required an effective approximate algorithm for finding near optimal solutions in a reasonable time frame. As such approximate algorithms, several heuristic algorithms have already been proposed: for example, the 2-opt algorithm, the 3-opt algorithm, the Lin-Kernighan algorithm[1], and so on. These algorithms are called the local search method whose basic mechanism is to search better solutions from near neighbors of a current solution. For example, the 2-opt algorithm, which is one of the simplest local search methods, is described as follows: two links are deleted from a current tour, and other two links are added in such a way that a length of a new tour is shorter than that of the current tour (Fig. 1). Such exchanges of links continue until no further shorter tours can be found.

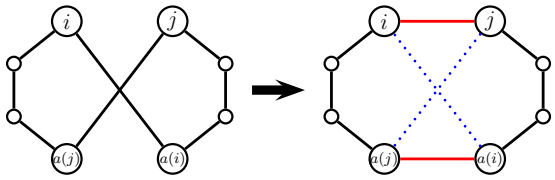


Figure 1: An example of the 2-opt algorithm. In this figure, $a(i)$ is the next city of the city i in a current tour. Two links $(i, a(i))$ and $(j, a(j))$ are deleted from the current tour, then two links (i, j) and $(a(i), a(j))$ are added to obtain a shorter tour.

However, it is almost impossible to obtain the optimal solution with these local search methods, because the local search methods get trapped into local minima. Thus, to avoid such local minima, various meta-heuristic strategies have been proposed. For example, tabu search[2], simulated annealing[3], genetic algorithm[4], chaotic search [6–10], and so on.

Among them, the chaotic search is one of effective and practical strategies to avoid local minima. We have already proposed effective algorithms for solving TSP using the chaotic search[6–10]. In the chaotic search method, to avoid local minima, local search methods are driven by chaotic neurodynamics. To realize the chaotic search method, a chaotic neuron model proposed by Aihara, Takabe and Toyoda[5] is introduced. This model can reproduce refractoriness, one of important properties which real nerve cells have. In the proposed methodologies, the refractoriness plays a central role of avoiding local minima.

First, the chaotic search method using the 2-opt algorithm was proposed[6, 7]. Although the 2-opt algorithm is the simplest local search method, this chaotic method[6, 7] shows good performance. It is also shown that this chaotic search method has higher performance than the tabu search method[2].

Next, a chaotic search method was proposed using a lo-

cal search method with higher performance than the 2-opt algorithm, the adaptive k-opt algorithm[8]. The adaptive k-opt algorithm changes the number of exchanged links adaptively. This method[8] shows higher performance than the chaotic search method using the 2-opt algorithm[6, 7].

Then, to extend this idea, we have proposed a chaotic search method[9] based on the Lin-Kernighan algorithm[1]. The Lin-Kernighan algorithm is one of the most effective local search methods, because it changes the number of exchanged links dynamically depending on current states. The Lin-Kernighan algorithm has higher performance than the adaptive k-opt algorithm because the Lin-Kernighan algorithm explores a searching space widely. As a result, this chaotic method[9] showed the highest performance in all the chaotic search methods[6–9].

The stem-and-cycle(S&C) ejection chain method[11, 12] is also a variable depth search method which changes the number of exchanged links adaptively. It is reported that the S&C ejection chain method can explore a searching space widely than the Lin-Kernighan algorithm. Then, in this paper, we propose a new chaotic search method based on the S&C ejection chain method. As a result, the proposed method shows higher performance than the conventional chaotic search method based on the Lin-Kernighan algorithm[9].

2. Proposed method

2.1. Stem-and-cycle ejection chain method

The stem-and-cycle(S&C) structure[11, 12] is constructed from a path $(v_t, \ldots, v_r)$ called *stem* and a cycle $(v_r, v_{s_1}, \ldots, v_{s_2}, v_r)$ called *cycle* (Fig. 2). The vertex v_r which belongs to both stem and cycle is called *root*. The two adjacent vertices of the root on the cycle v_{s_1} and v_{s_2} are called *subroots*. The vertex v_t is called *tip* of the stem.

The S&C ejection chain method has two types of improvements: a stem ejection and a cycle ejection. The first improvement is the stem ejection described as follows (Fig. 3 (a)): add a link (v_t, v_p), where v_p belongs to the stem. Next, delete a link (v_p, v_q), where v_q is the adjacent vertex of v_p and on the subpath $(v_t, \ldots, v_p)$. Then, the vertex v_q is a new tip. The second improvement is the cycle ejection described as follows (Fig. 3 (b)): add a link (v_t, v_p), where v_p belongs to the cycle. Next, delete a link (v_p, v_q), where v_q is one of the adjacent vertices of v_p. Then, the vertex v_q is a new tip.

The structure obtained by the ejection chain improvement does not offer a feasible solution, or a tour. Thus, it is required to generate a feasible solution from the S&C structure. Such trial solutions are generated as follows (Fig. 4): add a link (v_t, v_s), where v_s is the subroots. Next, delete a link (v_r, v_s).

In the proposed method, the implementation of the S&C ejection chain method is the same as the Lin-Kernighan algorithm. Namely, the proposed method searches better solutions by repeating the ejection improvements. If a current solution is worse than the best solution in the previous searches, the search is stopped. Thus, the proposed method is difference from the original algorithm in Ref. [12] because the depth of the search is decided depending on solution states. The procedure of the S&C ejection chain method of the proposed method is described below.

1. Let $G^* = 0$ and $m = 1$. Here, G^* is a value of the best improvement in the previous searches, and m is the number of depth of this search.

2. To construct a S&C structure, choose an initial tip vertex v_t from an initial tour T.

3. Choose a root vertex v_r from neighbors of $p(v_t)$ so that an improvement value $d(v_t, p(v_t)) - d(p(v_t), v_r)$ is maximum. Here, $p(v)$ is the previous vertex of v, and $d(v_1, v_2)$ is a distance between vertices v_1 and v_2. Next, delete a link $(p(v_t), v_t)$, and add a link $(p(v_t), v_r)$ to generate the S&C structure from a tour. Then, let $e_0 = d(v_t, p(v_t)) - d(p(v_t), v_r)$.

4. Repeat the ejection improvements by the following steps (a)-(e). If such an improvement does not exist, go to Step 5.

   (a) Choose v_p and v_q to satisfy the following conditions:

      i. v_p is a neighbor vertex of v_t.
      ii. A link (v_t, v_p) is not previously deleted.
      iii. A link (v_p, v_q) is not previously added.
      iv. An ejection value e_m at the depth m is maximum, where e_m is calculated by $e_m = d(v_p, v_q) - d(v_t, v_p)$.

   (b) Add a link (v_t, v_p), and delete a link (v_p, v_q).

   (c) Let T' be a tour constructed by trial exchanges. If $f(T) - f(T') > G^*$, set $G^* = f(T) - f(T')$ and let $T^* = T'$, where $f(T)$ is a length of the tour T and T^* is the tour to achieve G^*.

   (d) If $G < G^*$, go to Step 5, and $G = \sum_{i=0}^{m} e_i$.

   (e) Let m increase by one, and set v_q to a new tip v_t.

5. If $G^* > 0$, construct a new tour T^*.

2.2. Chaotic search method

In the proposed method, we drive the stem-and-cycle (S&C) ejection chain method by chaotic neurodynamics. To generate chaotic neurodynamics, we use a chaotic neural network constructed by chaotic neurons[5]. The number of chaotic neurons is the same as the number of cities, and each neuron is assigned to each city. If the chaotic neuron fires, the S&C ejection chain method for the corresponding city is executed.

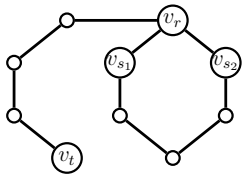
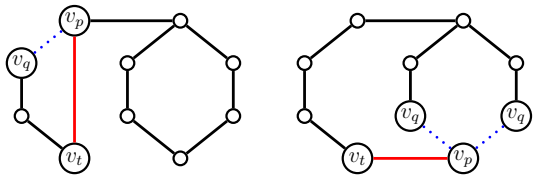
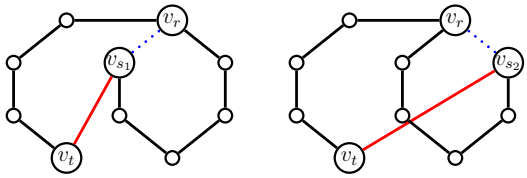


Figure 2: An example of S&C structure. A path $(v_t, \ldots, v_r)$ is the *stem* and a cycle $(v_r, v_{s_1}, \ldots, v_{s_2}, v_r)$ is the *cycle*. v_r is the *root*, v_{s_1}, v_{s_2} are the *subroots* and v_t is the *tip*.



(a) Stem ejection (b) Cycle ejection

Figure 3: Examples of S&C ejection improvements. Red lines represent added links, and blue dotted lines represent deleted links.



(a) (b)

Figure 4: Examples of trial solutions. Red lines represent added links, and blue dotted lines represent deleted links.

An internal state of the chaotic neuron is constructed by a gain effect and a refractory effect. First, the gain effect is described as follows:

$$\xi_i(t+1) = \max_j\{\beta(t)\Delta_{ij}(t) + \zeta_j(t)\}, \quad (2)$$

$$\beta(t+1) = \beta(t) + \frac{q}{\overline{\Delta}(t)}, \quad (3)$$

$$\overline{\Delta}(t) = \frac{1}{n}\sum_{i=1}^{n}|\Delta_{ij}(t)|, \quad (4)$$

where $\beta(t)$ is a scaling parameter of the gain effect at time t ($\beta(t) > 0$). This parameter increases with time t. If the value of $\beta(t)$ gradually increases, a searching space is increasingly limited as the simulated annealing[3]. $\Delta_{ij}(t)$ is defined as a difference of a length between a current tour and a new tour, or $\Delta_{ij}(t) = D_0(t) - D_{ij}(t)$, where $D_0(t)$ is a length of the current tour at time t, and $D_{ij}(t)$ is that of the new tour obtained by the S&C ejection chain method which links cities i and j, where i and j are v_t and v_p at $m = 1$ of the S&C procedure, respectively. $\zeta_j(t)$ is a refractory effect of the city j at time t.

To obtain the same range of $\xi_i(t)$ for all instances, the scaling parameter $\beta(t)$ is adjusted by $\Delta_{ij}(t)$ (Eq. (3)), because the range of $\Delta_{ij}(t)$ depends on each instance. In Eq. (3), q is a scaling parameter of the annealing effect. In Eq. (4), n is the number of neurons. If the length of the new tour is shorter than the current tour ($\Delta_{ij}(t) > 0$), the value of the gain effect becomes positive. Then, the gain effect encourages the chaotic neuron to fire.

Next, the refractory effect is described as follows:

$$\zeta_i(t+1) = -\alpha\sum_{d=0}^{s-1}k_r^d x_i(t-d) + \theta, \quad (5)$$

where α is a scaling parameter of the refractory effect ($\alpha > 0$); k_r is a decay parameter of the refractory effect ($0 < k_r < 1$); s is a temporal period for memorizing past outputs; $x_i(t)$ is an output of the ith neuron at time t; θ is a threshold value. If a neuron has fired for the past s steps, Eq. (5) tends to be negative. Namely, the refractory effect inhibits the neuron from firing for a while. In Eq. (5), if $s - 1 = t$, it means that the neuron memorizes its all history from $t = 0$. If we use Eq. (5) directly, it needs much amount of memory to memorize its all history. However, Eq. (5) can be transformed into the following simple one-dimensional difference equation:

$$\zeta_i(t+1) = k_r\zeta_i(t) - \alpha x_i(t) + (1-k_r)\theta. \quad (6)$$

Then, the output of the ith neuron is defined as follows:

$$x_i(t+1) = f\left(\xi_i(t+1) + \zeta_i(t+1)\right), \quad (7)$$

where $f(y) = 1/(1+e^{-y/\epsilon})$. If $x_i(t+1) \geq 1/2$, the ith neuron fires at time $t+1$ and the S&C ejection chain method which links the cities i and j is executed. Each neuron is updated asynchronously.

To solve an n-city TSP, the procedure of a single iteration in the proposed method is shown below.

1. Let $i = 1$.
2. To calculate the gain effect of the ith neuron (Eq. (2)), select a city j which maximizes the gain effect from the neighbor cities of i.
3. Calculate the output of the ith neuron (Eq. (7)).
4. If $x_i(t+1) \geq 1/2$, the ith neuron fires, then the S&C ejection chain method which links cities i and j is executed.
5. If $i < n$, let i increase by one and go to Step 2. Otherwise finish this iteration.

3. Results

To evaluate the performance of the proposed method, we use the benchmark problems of TSPLIB[13]. We compare the proposed method with the conventional chaotic search method based on the Lin-Kernighan algorithm[9]. In these methods, initial solutions are constructed by the nearest neighbor method. Parameters of the proposed method $\beta(0)$, α, k_r, θ, q and ϵ are set to 0, 1.0, 0.5, 1.0, 0.060 and 0.002, respectively. These methods are applied for 200 iterations. To reduce computational costs, we use two candidate lists: 10 nearest neighbors (10NN) and 8 quadrant neighbors (8QN). In the 10NN, the 10 nearest cities for each city are added in the candidate list. In the 8QN, the 2 nearest cities are added from each of the four quadrants for each city in the candidate list. These candidate lists are used when we decide which link is added: for example, Step 4(a) of the stem-and-cycle ejection chain method of Sec. 2.1 and Step 2 of the chaotic search method of Sec. 2.2.

Table 1 summarizes results of the proposed method (CS-S&C) and the conventional chaotic search method based on the Lin-Kernighan algorithm (CS-LK). In these methods, the obtained best solutions are further improved by its local search method (LS) until no further improvements are found. In other words, in the chaotic search method based on the Lin-Kernighan algorithm, the Lin-Kernighan algorithm is applied as the local search method. In Table 1, the results are expressed by percentages of average gaps between obtained solutions and the optimal solutions.

From Table 1, using 8QN, the proposed method shows higher performance than the conventional chaotic search method with the local search method for all instances. However, using 10NN, the proposed method shows worth performance than the conventional chaotic search method for rl5915 and rl11849.

Table 1: The results of the conventional chaotic search method based on the Lin-Kernighan algorithm without the local search method (CS-LK), that with the local search method (LS), the proposed method without the local search method (CS-S&C) and that with the local search method (LS). CL represents the candidate lists (10NN and 8QN).

		CS-LK[9]		CS-S&C	
Instance	CL	w/o LS	w/ LS	w/o LS	w/ LS
pcb1173	10NN	0.759	0.689	0.497	**0.452**
	8QN	0.855	0.785	0.529	0.487
pr2392	10NN	0.807	0.763	0.676	**0.647**
	8QN	0.882	0.832	0.795	0.756
rl5915	10NN	1.011	0.968	1.354	1.334
	8QN	0.859	0.780	0.673	**0.651**
rl11849	10NN	0.890	0.841	0.995	0.965
	8QN	0.784	0.707	0.678	**0.646**

4. Conclusion

In this paper, we propose a new chaotic search method using the stem-and-cycle structure. As a result, the proposed method shows higher performance than the conventional chaotic search method based on the Lin-Kernighan algorithm[9]. In the future works, we will improve the performance of the proposed method. Moreover, we will develop a new parameter tuning method and a different search method.

Acknowledgement

The research of T.I. is partially supported by Grant-in-Aid for Scientific Research (B) (No. 20300085) from the JSPS.

References

[1] S. Lin and B. Kernighan, *Operations Research*, **21**, pp.498–516, 1973.

[2] F. Glover, *ORSA J. Computing*, **1**, pp.190–206, 1989.

[3] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, *Science*, **220**, pp.671–680, 1983.

[4] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, 1975, and MIT Press, 1992.

[5] K. Aihara, T. Takabe and M. Toyoda, *Physics Letters A*, **144**, pp.333–340, 1990.

[6] M.Hasegawa, T. Ikeguchi and K. Aihara, *Physical Review Letters*, **79**, pp.2344–2347, 1997.

[7] M. Hasegawa, T. Ikeguchi and K. Aihara, *Neural Networks*, **15**, pp.271–283, 2002.

[8] M. Hasegawa, T. Ikeguchi and K. Aihara, *Technical Report of IEICE*, **101**, pp.25–32, 2001.

[9] S. Motohashi, T. Matsuura and T. Ikeguchi, *Proceedings of International Symposium on Nonlinear Theory and its Applications*, **21**, pp.144–147, 2008.

[10] S. Motohashi, T. Matsuura and T. Ikeguchi, to appear in *Proceedings of International Conference on Artificial Neural Networks (LNCS)*, 2009.

[11] F. Glover, *Computer Science and Operations Research*, pp. 449–509, 1992.

[12] C. Rego, *European Journal of Operational Research*, **106**, pp. 522–538, 1998.

[13] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`