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

Chaotic search method using the Lin-Kernighan algorithm for traveling salesman problems

Shun Motohashi†‡, Takafumi Matsuura† and Tohru Ikeguchi†

†Graduate School of Science and Engineering, Saitama University
255 Shimo-Ohkubo, Saitama, 338–8570 Japan
‡Email: motohashi@nls.ics.saitama-u.ac.jp

Abstract—The traveling salesman problem (TSP) belongs to a class of $\mathcal{NP}$-hard. Thus, it is required to develop effective algorithms for finding near optimal solutions or approximate solutions in a reasonable time frame. We have already proposed several methods which use chaotic dynamics to drive a local search method, such as the 2-opt algorithm or the adaptive k-opt algorithm. In these methods, chaotic dynamics efficiently controls to avoid local minima. In this paper, extending this idea, we propose a new method for solving the TSP. In the proposed method, one of the most powerful local search methods, the Lin-Kernighan algorithm, is driven by chaotic dynamics. As a result, the proposed method obtains better solutions than the conventional chaotic search methods.

1. Introduction

In our world, we often face to various situations in which reduction of operating costs is asked, for example, scheduling, routing problem, circuit designing, computer wiring and so on. These examples are formulated as the traveling salesman problem(TSP) which is one of the famous combinatorial optimization problems.

The TSP is descried 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, this problem becomes a symmetric TSP, otherwise an asymmetric TSP. In this paper, we deal with the symmetric TSP.

The TSP generally belongs to a class of $\mathcal{NP}$-hard. Thus, it is believed to be almost impossible to obtain an optimal solution in a reasonable time frame. Therefore, it needs to develop effective approximate algorithms for finding near optimal solutions or approximate solutions. As approximate algorithms, many local search methods, for example, the 2-opt algorithm, the 3-opt algorithm and the Lin-Kernighan algorithm[1], have already been proposed. However, these algorithms rarely find optimal solutions, because the local search methods usually get stuck at local minima. To escape from the local minima, several strategies have already been proposed, for example, a tabu search method[2], a simulated annealing[3], a genetic algorithm[4] and so on.

In recent years, we have already proposed several effective algorithms for solving TSP[6, 7, 8]. In these methods, chaotic dynamics is used to escape from the local minima. To realize a chaotic search method, a chaotic neuron model is used. The chaotic neuron model is proposed by Aihara et al[5], which can reproduce an important property which real nerve cells have: refractoriness. The refractoriness of the chaotic neuron works to escape from the local minima and to lead better solutions. Using this property, the chaotic search method in which the most simple local search method, the 2-opt algorithm (Fig. 1), is driven by chaotic dynamics, has already been proposed[6, 7], and exhibits better performance than the tabu search method which has almost the same tactics of searching solutions as the chaotic search method in a state space. Next, to improve the performance of the chaotic search method, an adaptive k-opt algorithm driven by chaotic dynamics has been proposed[8]. The adaptive k-opt algorithm is a powerful local search method which adaptively changes the number of exchanged links k. As a result, this method shows better performance than the chaotic search method using the 2-opt algorithm.

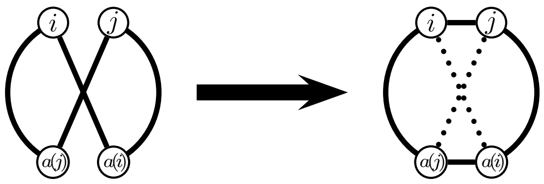

Figure 1: An example of the 2-opt algorithm. In this figure, $a(i)$ is the next city of the city i in the current tour. Two links i-$a(i)$ and j-$a(j)$ are deleted from the current tour, then two links i-j and $a(i)$-$a(j)$ are added to obtain a shorter tour.

The Lin-Kernighan algorithm is one of the most effective local search methods for the symmetric TSP. The Lin-Kernighan algorithm improves a tour by exchanging k links in the current tour for other k links. Then, the value of

k is dynamically changed. In the Lin-Kernighan algorithm, as the value of k increases, better improvements are searched. Although the Lin-Kernighan algorithm shows the most highest performance for the TSP, this method also gets trapped into the local minima, because this method is a local search method. Then, if we introduce chaotic dynamics to avoid the local minima for the Lin-Kernighan algorithm, we realize a more effective chaotic search method. As a result, the proposed method shows higher performance than the conventional chaotic search methods[7, 8].

2. The Lin-Kernighan algorithm

The k-opt algorithm generally gets better solutions when the value of k increases. However, for an n-city TSP, the k-opt algorithm has a time complexity of $O(n^k)$ to search a new tour. Thus, if the value of k increases to obtain the better solutions, the computational costs increase exponentially. To avoid such an issue, Lin and Kernighan proposed a powerful variable k-opt algorithm[1]. This algorithm is called the Lin-Kernighan algorithm, and considered to be one of the most effective local search methods in the field of operations research.

In the Lin-Keringhan algorithm, k links in the current tour are exchanged for other k links to improve a solution. The Lin-Kernighan algorithm searches better improvements as the value of k increases. Through computational experiments, it has been indicated that the Lin-Kernighan algorithm has a time complexity of $O(n^{2.2})$ for n-city TSPs[1].

In the Lin-Kernighan algorithm, a possible improvement is searched by repeating a choice of a deleted link and an added link. The procedure of the Lin-Kernighan algorithm is shown below.

1. Let $G^* = 0$ and $m = 1$. Here, G^* is a value of the best improvement in the previous search, and m is the number of sets of deletion and addition of links.

2. Choose any city t_1 from the current tour T.

3. Delete a link $x_1 = (t_1, t_2)$ from T. Here, x_m is a link between the cities t_{2m-1} and t_{2m}.

4. Add a link $y_1 = (t_2, t_3)$ with the condition that $g_1 > 0$, where $g_m = |x_m| - |y_m|$ (Fig. 2(b)). Here, y_m is a link between the cities t_{2m} and t_{2m+1}, and $|x_m|$ is a length of the link x_m. If no such y_1 exists, go to Step 10.

5. Let m increase by one. Delete x_m and add y_m by the following steps (a)–(d). If such x_m and y_m do not exist, go to Step 6.

   (a) Delete x_m to satisfy the following conditions (Fig. 2(a)):

      i. x_m is not previously added.

      ii. If t_{2m} is connected to t_1, the resulting configuration is a feasible tour.

   (b) Let T' be a tour constructed by connecting t_{2m} to t_1. If $f(T) - f(T') > G^*$, set $G^* = f(T) - f(T')$ and $k = m$, where $f(T)$ is a length of T and k is the number of exchanged links to achieve G^*.

   (c) Add y_m to satisfy the following conditions (Fig. 2(b)):

      i. y_m is not previously deleted.

      ii. $G_m > 0$, where $G_m = \sum_{j=1}^{m}(|x_j| - |y_j|)$.

      iii. If y_m is added, a next link x_{m+1} exists.

      iv. $|x_{m+1}| - |y_m|$ is maximum for all candidates of y_m.

   (d) If $G_m > G^*$, go to Step 5.

6. If $G^* > 0$, construct a new tour by executing the k-opt exchange for T, and go to Step 1.

7. If another link can be selected as y_2, go to Step 5(c).

8. If another link can be selected as x_2, go to Step 5(a).

9. If another link can be selected as y_1, go to Step 4.

10. If another link can be selected as x_1, go to Step 3.

11. If another city can be selected as t_1, go to Step 2.

12. The procedure terminates.

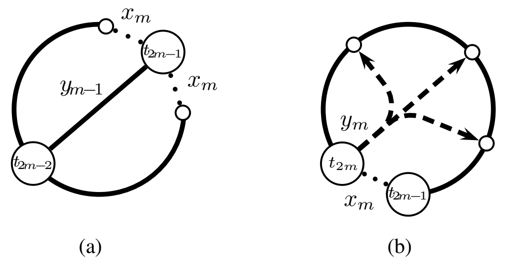

Figure 2: Examples of how to choose added and deleted links in the Lin-Kernighan algorithm. (a) Choose a deleted link x_m. (b) Choose a added link y_m

3. The proposed method

In the proposed method, the Lin-Kernighan algorithm is driven by chaotic dynamics. To realize a method based on chaotic dynamics, we use a chaotic neural network (CNN) constructed by chaotic neurons[5]. In the proposed method, each neuron is assigned to each city, and an execution of the Lin-Kernighan algorithm is controlled by an output of the assigned chaotic neuron. If a neuron fires, the Lin-Kernighan algorithm is executed on the corresponding city.

The chaotic neuron has a gain effect and a refractory effect. These effects are defined by the following equations:

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

$$\Delta_{ij}(t) = D_0(t) - D_{ij}(t), \quad (3)$$

$$\zeta_i(t+1) = -\alpha \sum_{d=0}^{s-1} k_r^d x_i(t-d) + \theta, \quad (4)$$

where $\beta(t)$ is a scaling parameter of the gain effect at time t ($\beta(t) > 0$). This parameter increases with time t: $\beta(t+1) = \beta(t) + \gamma$ ($\gamma > 0$). If the value of $\beta(t)$ gradually increases, a searching space is increasingly limited as the simulated annealing[3]. In Eq.(2), $\Delta_{ij}(t)$ is defined as a difference between a length of a current tour and a new tour. $D_0(t)$ is a length of the current tour at time t, and $D_{ij}(t)$ is a length of the new tour offered by the Lin-Kernighan algorithm which links cities i and j. In the original Lin-Kernighan algorithm, if an improvement is not found, an exchange is not executed. In such a case, in the proposed method, $\Delta_{ij}(t)$ is set to a value of the improvement by the 2-opt algorithm which links the cities i and j. $\zeta_j(t+1)$ is a refractory effect of the city j at time $t+1$. In Eq.(2), the city j is chosen so as to maximize the value of the gain effect. If the length of the new tour is shorter than the current tour, the value of the gain effect becomes positive ($\Delta_{ij}(t) > 0$). Then, the gain effect encourages to fire the chaotic neuron.

In Eq.(4), α is a scaling parameter of the refractory effect after a neuron firing ($\alpha > 0$), 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, and θ is a threshold value. If a neuron has fired for the past s steps, the right hand side of Eq.(4) becomes negative. Namely, the refractory effect inhibits to fire the neuron for a while.

In Eq.(4), if $s - 1 = t$, it means that the neuron memorizes its all history from $t = 0$. If we use Eq.(4) directly, it needs much amount of memory to memorize its all history. However, Eq.(4) can be reduced to the following simple one-dimensional difference equation:

$$\zeta_i(t+1) = k_r \zeta_i(t) - \alpha x_i(t) + (1 - k_r)\theta. \quad (5)$$

Then, the output of the ith neuron is defined by the following equation:

$$x_i(t+1) = f(\xi_i(t+1) + \zeta_i(t+1)), \quad (6)$$

where $f(y) = 1/(1 + e^{-y/\epsilon})$. If $x_i(t+1) > 1/2$, the ith neuron fires at time $t+1$ and the Lin-Kernighan algorithm which links the cities i and j is executed. Each neuron is updated asynchronously.

In the proposed method, $\beta(t+1)\Delta_{ij}(t) + \zeta_j(t+1)$ is calculated for all city j, then the maximum value is used as the gain effect. If the number of cities increases, the computational costs of the gain effect increase. In general, if the city i and the city j are located far away, the city j is not chosen, because the value of $\Delta_{ij}(t)$ is smaller than other cities. Even if the far city j is chosen and the ith neuron fires, an obtained new tour might be different from an optimal tour. Then, in Eq.(2), the cities j are not necessary to be used for execution of the algorithm. Thus, in the proposed method, a neighborhood list is used to reduce computational costs. If the neighborhood list size is r, only r nearest neighbor cities of the city i are used for Eq.(2). For the same reason, this neighborhood list is used in the Lin-Kernighan algorithm. If y_m which links the cities t_{2m} and t_{2m+1} adds, the city t_{2m+1} is chosen among the neighborhood list of the city t_{2m}. In this paper, the neighborhood list size is fixed to 10.

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

1. Let $i = 1$.
2. Choose the city j which maximizes the value of the gain effect of the ith neuron from the neighborhood list of the city i.
3. The output of the ith neuron $x_i(t+1)$ is calculated.
4. If $x_i(t+1) > 1/2$, the ith neuron fires, and the Lin-Kernighan algorithm which links cities i and j is executed.
5. If $i < n$, let $i = i + 1$ and go to Step 2. Otherwise finish this iteration.

4. Results

To evaluate the performance of the proposed method, we used TSPLIB instances[9]. Then, we compared the proposed method with the conventional method: the adaptive k-opt algorithm driven by chaotic dynamics[8]. The adaptive k-opt algorithm is one of the local search methods which changes the number of exchanged links k similarly to the Lin-Kernighan algorithm. However, a criterion for how deep the two algorithm are executed is different. This conventional method includes a tuning and an annealing of parameters. Moreover, to diversify solutions, when better solutions could not be found for more than 100 iterations, the double bridge algorithm is applied. The double bridge algorithm is a special case of the 4-opt algorithm (Fig. 4).

In the proposed method, the parameters α, k_r, θ and ϵ are fixed for all instances: $\alpha = 0.95, k_r = 0.3, \theta = 1.0, \epsilon = 0.002$. However, $\beta(0)$ and γ are changed for each instance, because a size of $\Delta_{ij}(t)$ in Eq.(2) depends on the instance size. The value of $\Delta_{ij}(t)$ represents a difference between the length of the current tour and that of the new tour. Then, if the length of the exchanged links varies greatly, the scaling parameters must be tuned to small values, because the value of $\Delta_{ij}(t)$ becomes larger. Namely, the value of $\Delta_{ij}(t)$ depends on the lengths of the links. Therefore, we use a standard deviation(SD) of the lengths of the links to decide the values of the scaling parameters. If the value of the SD becomes larger, the value of $\Delta_{ij}(t)$ becomes larger, because the lengths of the links have wide fluctuations. Thus, the scaling parameters should be tuned to smaller values. To decide the values of the scaling parameters, first, we investigated good values of $\beta_c(0)$ and γ_c for an instance manually. In this paper, we used pcb1173. Next, we calculated the SD of the links for pcb1173. Then, we decided the values of $\beta_I(0)$ and γ_I for an instance I by the following equations:

$$\beta_I(0) = \beta_c(0) \times \frac{35.194410}{(\text{the SD of an instance } I)}, \quad (7)$$

$$\gamma_I = \gamma_c \times \frac{35.194410}{(\text{the SD of an instance } I)}, \quad (8)$$

where 35.194410 is the SD of pcb1173. For example, for pcb442, $\beta_{pcb442}(0) = 0.008 \times \frac{35.194410}{80.716443} = 0.0034882$, $\gamma_{pcb442} = 0.0015 \times \frac{35.194410}{80.716443} = 0.00065404$. Table 1 shows the scaling parameters and the SDs for instances in this paper.

Table 2 shows results of the adaptive k-opt algorithm driven by chaotic dynamics with the double bridge algorithm[8](k-opt+DB), the proposed method and the Lin-Kernighan algorithm(LK). The Lin-Kernighan algorithm is obtained by setting $\alpha = k_r = \theta = 0$ in the proposed method. Then it works same as the original Lin-Kernighan algorithm. The conventional chaotic method is applied for 5,000 iteration, and the proposed method is applied for 200 iteration. The results of Table 2 are expressed by percentages of average gaps between obtained solutions and the optimal solutions. From Table 2, the proposed method obtains better solutions than the conventional chaotic search method by shorter iterations for all problems.

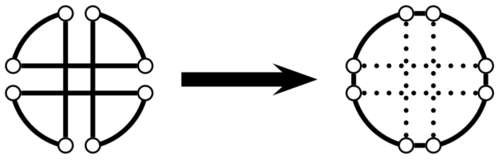

Figure 3: An example of the double bridge algorithm.

Table 1: The values of the scaling parameters and the SD.

problem	β	γ	SD
pcb442	0.0034882	0.00065404	80.716443
pcb1173	0.0080000	0.00150000	35.194410
pr2392	0.0021511	0.00040334	130.886889
rl5915	0.0028917	0.00054220	97.365513
rl11849	0.0042271	0.00079259	66.606484

5. Conclusion

In this paper, we proposed a new method which drives the Lin-Kernighan algorithm by chaotic dynamics. From the computational experiments, the proposed method shows better performance than the conventional chaotic search method. In the future work, we apply the proposed method for larger-size TSPs and compare its performance with other heuristics.

Table 2: Results of the adaptive k-opt algorithm driven by chaotic dynamics with the double bridge algorithm[8](k-opt+DB), the proposed method and the Lin-Keringhan algorithm(LK). For the conventional method, 5,000 iterations are applied, and for the proposed method, 200 iterations are applied. The results are expressed by the percentages of average gaps between the obtained solutions and the optimal solutions. The best values are described in bold faces.

problem	k-opt+DB[8]	LK	the proposed method
pcb442	0.8246	1.2633	**0.5109**
pcb1173	1.5692	2.3290	**1.1035**
pr2392	1.8390	2.6897	**0.8080**
rl5915	1.7418	3.5707	**1.0813**
rl11849	1.1855	3.0118	**1.0923**

Acknowledgments

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

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