Chaotic Neurodynamical search with small number of neurons for solving QAP

Hikaru Ohnishi¹a), Yutaka Shimada¹,², Kantaro Fujiwara¹,², and Tohru Ikeguchi¹,²

¹ Department of Management Science, Graduate School of Engineering, Tokyo University of Science, 1-3 Kagurazaka, Tokyo 162-0825, Japan
² Department of Information and Computer Technology, Faculty of Engineering, Tokyo University of Science, 6-3-1 Nijuku, Katsushika-ku, Tokyo 125-8585, Japan

a) hohnishi@hisenkei.net

Received January 11, 2017; Revised April 10, 2017; Published July 1, 2017

Abstract: Quadratic Assignment Problem (QAP) is one of the combinatorial optimization problems which are classified into Nondeterministic Polynomial time solvable (NP)-hard problems. Therefore, it is important to develop algorithms for finding good approximate solutions in short time. In this paper, we proposed an algorithm for approximately solving QAP by using chaotic neurodynamics. The proposed algorithm has three characteristics. First, compared with the conventional method, the number of neurons was substantially reduced. Second, the effect of external inputs to neurons was changed. Third, a new parameter tuning method was introduced. As a result, our algorithm can find good solutions compared with the conventional method using chaotic neurodynamics.

Key Words: QAP, optimization, chaotic neural network

1. Introduction

There are many combinatorial optimization problems, for example, scheduling, vehicle routing, facility location problem, and so on. It is important to solve these problems, because the cost can be reduced. These problems can be modeled by Quadratic Assignment Problems (QAP) [1, 2], and QAP is classified into Nondeterministic Polynomial time solvable (NP)-hard problems. The QAP is formulated as follows: when two \( n \times n \) matrices, a distance matrix \( D \) and a flow matrix \( R \), are given, we are asked to find an assignment \( p = \{p(1), p(2), \ldots, p(n)\} \) that minimizes the objective function of QAP defined by Eq. (1):

\[
F(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} R_{p(i)p(j)},
\]

where \( p(i) \) is the element assigned to location \( i \). Namely, if \( p(i) = j \), the element \( j \) is assigned to the location \( i \). In addition, QAP includes many other combinatorial optimization problems as a special case, for example, the Traveling Salesman Problem (TSP). There are several algorithms
for solving QAP. Algorithms for solving combinatorial optimization problems such as QAP can be classified into exact algorithms and approximate algorithms. The exact algorithms can find an exact solution. However, it takes a huge amount of computational time to obtain an exact solution. On the other hand, the approximate algorithms such as heuristic methods do not guarantee to find an exact solution. However, it takes less time to find approximate solutions. Namely, it is one of the important issues to develop effective approximate algorithms for obtaining approximate solutions in short time.

In this paper, we used a 2-exchange method which is one of well-known heuristic methods for solving QAP. The 2-exchange method is simple but effective to find a reasonable solution. However, in general, heuristic methods such as the 2-exchange method are trapped into undesirable local minimas, which makes it difficult to find a global minimum. Therefore, many methods for escaping from the local minimas have also been proposed: for example, a tabu search [3, 4], a genetic algorithm [5], chaotic dynamics [6–8], and so on.

In this paper, we focus on the methods proposed by Hasegawa et al. [6–8] that use chaotic dynamics of neurons. These methods control a heuristic method by chaotic neurodynamics. In [6], Hasegawa et al. proposed a method for controlling execution of the 2-opt method which is one of the heuristic methods for solving TSP, by chaotic neurodynamics. Although this method performed well in problems whose sizes are about 300 cities at most, it was suffering from the limitation of computational resources, because $n^2$ neurons are required when the problem size is $n$. Then, Hasegawa et al. applied this method to solve QAP [7]. The method applied for QAP can also get good performance. The method for solving QAP by chaotic neurodynamics still requires $n^2$ neurons when the size of QAP is $n$. After that, a method for controlling execution of the 2-opt method for solving TSP by chaotic neurodynamics only with $n$ neurons has been proposed [8].

In this paper, we propose a new method for solving QAP using chaotic neurodynamics. The proposed method has the same strategy as [6–8]. Namely, the method controls the execution of the 2-exchange method by using the chaotic dynamics. However, the number of neurons in the proposed method is substantially reduced compared with the method in [7]. In addition, to improve the performance of the method, we incorporated a new strategy into this proposed method. We changed the effect of an external input to neurons that effectively encourages the 2-exchange method to escape from undesirable local minimas. Although this strategy works well, its performance largely depends on the parameter. Thus, we further introduced a parameter tuning method. As a result, we succeeded to improve the performance without selecting suitable parameter values.

2. Heuristic method for QAP

There are many heuristic methods for solving QAP. In the following, we introduced a 2-exchange method which is a basic algorithm for solving QAP and our proposed method is based on this method. The algorithm of the 2-exchange method is described as follows:

**Step1:** Generating a random solution (assignment) $p_r = \{p_r(1), p_r(2), \cdots, p_r(n)\}$.

**Step2:** Calculating the objective function $F(p_r)$.

**Step3:** Selecting two elements $p_r(i)$ and $p_r(j)$ from all the elements of $p_r$ at random, and then $p_r(i)$ and $p_r(j)$ are exchanged. Let $p'_r$ be a solution after the exchange of $p_r(i)$ and $p_r(j)$.

**Step4:** Calculating the objective function $F(p'_r)$.

**Step5:** If a solution is improved, that is, $F(p_r) > F(p'_r)$, then $p_r \leftarrow p'_r$ and return to **Step3**. If there is no pair of $p_r(i)$ and $p_r(j)$ that makes the objective function decrease, this algorithm is terminated.

Generally, heuristic methods such as the 2-exchange method are carried out only when the objective function decreases. In this sense, many heuristic methods can basically make a present solution better step-by-step. However, the problem is that these heuristic methods are generally trapped into undesirable local minimas. Then, to escape from the undesirable local minimas, for example, tabu
search [3, 4] is introduced as follows: if there is no pair of elements which makes the objective function decrease, the pair of elements that least degrades the objective function is chosen. In the tabu search, the most recent exchanges are stored in a data structure called a tabu list, and the pairs included in the tabu list are not exchanged. Once the pair is stored in the tabu list, it becomes available $s$ iterations later, where $s$ is the size of the tabu list. This tabu effect prevents the 2-exchange method from returning to recently visited local minima, and thereby the 2-exchange method can escape from local minima. In the next section, we introduce a method based on neural networks [6] and chaotic search method [9] that includes both tabu effect and the chaotic dynamics.

3. Conventional methods

3.1 Realizing tabu search by neural networks

First, we introduce a method realizing the tabu search by the neural network. When an element is assigned to a location, there are $n \times n$ ways for selecting a pair of the element $i$ and the location $j$, where $n$ is the size of the problem. In [7], $n \times n$ neurons are prepared for representing each assignment $(i,j)$. Let $q(i)$ be a label of a location occupied by the element $i$. If the neuron corresponding to the assignment $(i,j)$ fires, the element $i$ is assigned to the location $j$, and simultaneously, to make a feasible solution, the element $p(j)$ is assigned to the location $q(i)$. In this case, in order to realize the tabu search, an assignment of $i$ to $j$ and $p(j)$ to $q(i)$ is forbidden. Then, the neurons $(i,j)$ and $(p(j),q(i))$ should not fire during some iterations after the firing of the neuron $(i,j)$. This tabu effect is realized by introducing a refractory effect of neurons, that is, neurons do not fire for a while just after the neurons fire. Then, the tabu search by neural networks can be realized by the following Eqs. (2)–(4).

\[
\begin{align*}
\xi_{ij}(t+1) &= \beta \Delta_{ij}(t), \\
\gamma_{ij}(t+1) &= k \zeta_{q(i)p(j)}(t) - \alpha x_{p(j)q(i)}(t), \\
\zeta_{ij}(t+1) &= k \zeta_{ij}(t) - \alpha x_{ij}(t),
\end{align*}
\]

where $x_{ij}(t)$ is the output of the neuron $(i,j)$, and $\xi_{ij}(t)$, $\gamma_{ij}(t)$, and $\zeta_{ij}(t)$ are internal states of the neuron $(i,j)$ at time $t$: $\xi_{ij}(t)$ is the gain effect, $\gamma_{ij}(t)$ is the tabu effect of the assignment of $p(j)$ to $q(i)$, and $\zeta_{ij}(t)$ is that of $i$ to $j$. The parameter $\beta$ is the scaling parameter for the gain effect, $\alpha$ is the decay parameter of the tabu effect, $\xi$ is the scaling parameter of the tabu effect, and $\Delta_{ij}(t)$ is the gain of the value of the objective function defined by $\Delta_{ij}(t) = G_0(t) - G_{p(j)q(i)}(t)$, where $G_0(t)$ is the present value of the objective function $F(p)$ at time $t$ and $G_{p(j)q(i)}(t)$ is the value of $F(p')$ where $p'$ is generated by exchanging the elements $i$ and $p(j)$ of $p$. The output of the neuron $(i,j)$, $x_{ij}(t+1)$, is calculated by

\[
x_{ij}(t+1) = \begin{cases} 
1 & \text{if a pair } (i,j) \text{ maximizes } \{ \xi_{ij}(t+1) + \gamma_{ij}(t+1) + \zeta_{ij}(t+1) \}, \\
0 & \text{(otherwise)}. 
\end{cases}
\]

3.2 Chaotic search including tabu search

The tabu search neural network model shown in Eqs. (2)–(4) has a similar form to the chaotic neural network model in [9]. However, because the output of the neuron $x_{ij}(t)$ takes only 0 or 1, the chaotic behavior is not observed in the tabu search neural network. Therefore, to produce chaotic behavior, Hasegawa et al. introduced a sigmoidal functional form into the output of neurons. They showed that the algorithm with chaotic behavior achieves better performance than the normal tabu search and the tabu search neural network [7]. Their method includes both the tabu effect and the chaotic dynamics, which is realized by the following Eqs. (6)–(10).

\[
\begin{align*}
\xi_{ij}(t+1) &= \beta \Delta_{ij}(t), \\
\eta_{ij}(t+1) &= -W \sum_{l=1}^{n} \sum_{m=1}^{n} x_{lm}(t) + W, \\
\gamma_{ij}(t+1) &= k \zeta_{q(i)p(j)}(t) - \alpha \{ x_{p(j)q(i)}(t) + z_{p(j)q(i)}(t) \} + \theta(1-k), \\
\zeta_{ij}(t+1) &= k \zeta_{ij}(t) - \alpha \{ x_{ij}(t) + z_{ij}(t) \} + \theta(1-k),
\end{align*}
\]

257
where $\eta_{ij}(t+1)$ is the feedback input to the chaotic neuron $i$ from other neurons in the network, $W$ is the connection weights, $\theta$ is a positive bias, $z_{ij}(t)$ is the accumulated output value corresponding to assignments of $i$ to $j$, and $f$ is a sigmoidal function defined by $f(y) = 1/(1 + e^{-y/\epsilon})$. If $x_{ij}(t + 1) > \frac{1}{2}$, the neuron $(i, j)$ fires and the element $i$ is assigned to the location $j$, and $p(j)$ to $q(i)$. We refer to this chaotic search (CS) method as a conventional method throughout this paper.

This neural network should be asynchronously updated. This is because in an asynchronous update, every element can be exchanged even if more than two neurons fire in a single iteration. In this method, simple cyclic updating method is adopted with a fixed order, namely, $(1, 1) \rightarrow (1, 2) \rightarrow \cdots \rightarrow (1, n) \rightarrow (2, 1) \rightarrow (2, 2) \rightarrow \cdots \rightarrow (2, n) \rightarrow \cdots \rightarrow (n, 1) \rightarrow (n, 2) \rightarrow \cdots \rightarrow (n, n)$.

The accumulated output $z_{p(j)q(i)}(t)$ is one of the refractory effects which inhibit the neuron $(p(j), q(i))$ from firing when the neuron $(i, j)$ fires, because the assignment of $i$ to $j$ is equal to $p(j)$ to $q(i)$. Therefore, if the neuron $(i, j)$ fires, the neuron $(p(j), q(i))$ cannot fire by introducing $z_{ij}(t)$. The accumulated output $z_{ij}(t)$ is calculated as follows: when the state of the neuron $(i, j)$ is updated, $z_{ij}(t + 1)$ is reset to 0. On the other hand, after the neuron $(i, j)$ is updated, if the neuron $(p(j), q(i))$ is already updated in this iteration $t$, $x_{ij}(t + 1)$ is added to $z_{p(j)q(i)}(t + 1)$, otherwise $x_{ij}(t + 1)$ is added to $z_{p(j)q(i)}(t)$, where $z_{ij}(0) = 0$.

4. Proposed method

4.1 Algorithms of the proposed method

In the conventional method [7], when the problem size is $n$, $n^2$ chaotic neurons are assigned to any pairs of elements and locations. Although this conventional method [7] shows good performance, it uses a large number of neurons. In our method, we assign chaotic neurons to each location and thereby reduce the number of neurons from $n^2$ to $n$.

We use a mutually coupled neural network of the $n$ units, where each neuron mutually connects to other neurons. The dynamics of chaotic neurons in our method obeys the following equations of Eqs. (11)–(14).

\[
\begin{align*}
\xi_j(t+1) &= \beta \Delta_{ij}(t), \\
\eta_j(t+1) &= W \sum_{l=1, l \neq j}^{n} x_l(t) + W, \\
\zeta_j(t+1) &= k\zeta_j(t) - \alpha x_j(t) + (1-k)\theta,
\end{align*}
\]

\[x_j(t+1) = f(\xi_j(t+1) + \eta_j(t+1) + \zeta_j(t+1)), \quad (14)\]

where $\xi_j(t)$ is an external input to the chaotic neuron $j$, $\eta_j(t)$ is a feedback input to the chaotic neuron $j$ from other neurons in the network, $W$ is the connection weight, $\zeta_j(t)$ is a refractoriness term of the chaotic neuron $j$, and $x_j(t)$ is the output of the chaotic neuron $j$. Note that the sign of $W$ in the first term of the right-hand side of Eq. (7) is negative, but positive in Eq. (12). In our method, we first select a reference neuron $i$ and update states of all neurons except for the reference neuron $i$. Then, if the neuron $j$ fires, we perform the 2-exchange method for the elements $p(i)$ and $p(j)$. The reference neuron $i$ is selected in the order of the indexes of elements in this paper.

Besides reducing the number of neurons, we changed the calculation method of $\Delta_{ij}(t)$. If we have a bad exchange, we applied a strong input to chaotic neurons. As mentioned in the Sec. 3.1, in [7], $\Delta_{ij}(t)$ is defined as $\Delta_{ij}(t) = G_0(t) - G_{ip(j)}(t)$. On the other hand, in the proposed method, we define $\Delta_{ij}(t)$ as $\Delta_{ij}(t) = G_{p(i)p(j)}(t) - G_0(t)$, and $\beta > 0$. Due to this effect, bad exchanges are frequently executed, which means that our method can widely search the solution space. The algorithm of our proposed method is explained as follows.

**Step 1:** Let $i = 1$. 

258
Step2: Internal states of all chaotic neurons except for the chaotic neuron $i$ are updated asynchronously by Eqs. (11)–(13).

Step3: The outputs of all the chaotic neurons except for the chaotic neuron $i$ are calculated by Eq. (14).

Step4: If $\min_j (\Delta_{ij}(t)) < 0$, the elements $p(i)$ and $p(j)$ are exchanged by the 2-exchange method and go to Step6. Otherwise go to Step5.

Step5: If $x_j^* (t+1) > \frac{1}{2}$ where $j^* = \arg \max_j \{ \xi_j(t+1) + \eta_j(t+1) + \zeta_j(t+1) \}$, the chaotic neuron $j^*$ fires, and the elements $p(i)$ and $p(j^*)$ are exchanged by the 2-exchange method.

Step6: If $i = n$, this algorithm is terminated, otherwise $i \leftarrow i + 1$ and return to Step2.

We defined this cycle as one iteration and continued this cycle until $100 \times n$ exchanges are carried out as is the same with [7]. An important different point of the proposed method from the conventional method [7] is that we added Step4 to introduce the steepest descent dynamics. In [7], the steepest decent dynamics is expressed by the gain effect. However, we changed the effect of gain, and the steepest decent dynamics disappears. Therefore, we added Step4 in our proposed method.

4.2 Evaluating performance
We evaluated the performance of the proposed method, using benchmark problems from QAPLIB [2]. To evaluate the performance, we used the gap from optimal solution, which is defined by the following equation.

$$\text{gap}[%] = \frac{\text{obtained solution} - \text{optimal solution}}{\text{optimal solution}} \times 100.$$  \hspace{1cm} (15)

In this paper, we used the following parameter values: $w = 0.0005$, $k = 0.5^{\frac{1}{n}}$, $\alpha = 1$, $\theta = 0.05$, and $\epsilon = 0.002$. We changed the value of $\beta$ and calculated the average gaps of 100 trials for each value of $\beta$. In the numerical experiments, the elements $(i, j)$ of the distance matrix $D = (d_{ij})$ and the flow matrix $R = (r_{ij})$ are normalized by $\max_{ij} \{ d_{ij} \}$ and $\max_{ij} \{ r_{ij} \}$, respectively.

Table I shows the obtained best average gaps for various parameter values of $\beta$. Numerals with bold faced types indicate the best gap. The best parameter $\beta$ for each problem is shown in parentheses. From Table I, if we compared performance of our method with the stochastic simulated annealing and the robust tabu search, our proposed method can get better performance. In addition, even though the proposed method controls the 2-exchange method with a small number of neurons, we can get better performance compared to the conventional chaotic search (CS) [7]. We emphasize that in [7], the parameter tuning method has also been proposed, and thereby the parameters of CS [7] shown in Table I and Fig. 1 are tuned enough.

To investigate how the performance of the proposed method depends on $\beta$, we changed the value of $\beta$ and evaluated the performance. Figure 1 shows the results of the average gaps for each value of $\beta$. In Fig. 1, the ordinate axes show the gap and the abscissas show the values of the parameter $\beta$. The results of the proposed method are shown by the red lines and those of the method in [7] are shown by the blue lines. These figures show that our proposed method can get better performance than that of the methods in [7] except for Bur26b.

Although the performance of the proposed method is better, the performance largely depends on the parameter $\beta$. In order to clarify the relation between the performance and the parameter $\beta$, we investigated the firing rates of neurons in each iteration, because the firing rate of neurons affects the search space of the proposed method: when neurons frequently fire, the proposed method widely search the solution space. Figures 2 and 3 show the gaps and the firing rates of each iteration. In Figs. 2 and 3, the ordinate axes show the gaps and firing rates, and the abscissas show the iteration. In Figs. 2(a) and 3(a), the firing rates converge to around 0 when the parameter $\beta$ is small. Therefore, it is difficult to escape from the local minima, because the neuron hardly fires. On the other hand,
Table I. Results of the average gaps [%] for the stochastic simulated annealing (SSA), the robust tabu search (Ro-TS), the chaotic search (CS) and the proposed method. The best parameter values of $\beta$ for each problem are shown in parenthesis. Numerals with bold faced types indicate the best gap, and italic faced types indicate the second best gap.

| Problem | SSA [7] | Ro-TS [7] | CS [7] | Proposed method |
|---------|---------|-----------|--------|-----------------|
| Bur26a  | 0.162   | 0.147     | 0.159  | **0.128 (0.11)**|
| Bur26b  | 0.246   | 0.102     | **0.0814** | 0.165 (0.32)    |
| Bur26c  | 0.0922  | 0.0362    | 0.0496 | **0.0161 (0.12)**|
| Bur26d  | 0.106   | 0.0271    | 0.0234 | **0.00858 (0.12)**|
| Ste36a  | 5.68    | **3.85**  | 5.65   | 4.07 (0.05)     |
| Ste36b  | 7.72    | 7.42      | 12.7   | **5.10 (0.08)** |
| Ste36c  | 3.99    | 3.09      | 4.40   | **2.71 (0.06)** |
| Tai20b  | 8.78    | **0.626** | 1.80   | 0.989 (0.03)    |
| Tai30b  | 8.48    | 2.28      | 2.33   | **1.12 (0.03)** |
| Tai40b  | 7.26    | 2.88      | 3.70   | **1.84 (0.02)** |
| Tai50b  | 4.16    | 1.85      | 2.21   | **1.15 (0.02)** |
| Tai60b  | 3.67    | 3.13      | 2.52   | **1.16 (0.02)** |
| Tai80b  | 3.81    | 2.64      | 2.88   | **1.71 (0.02)** |

Fig. 1. Results of the average gaps for each $\beta$. In Figs. 2(c) and 3(c), the firing rates are very high when the parameter $\beta$ is large. In this case, because the firing rate is too high, the neurons fire and searches another area frequently, and the proposed method cannot reach local minima. Therefore, the balance of searching range and depth is important.

5. Parameter tuning method

5.1 Strategy to select the value of $\beta$

From Sec. 4.2, the performance of the proposed method largely depends on the scaling parameter $\beta$ which controls the firing of neurons. Based on the results in Sec. 4.2, we introduced the following tuning method in which the value of $\beta$ is controlled at each iteration.

Let the value of $\beta$ at $t$ be $\beta(t)$ and its initial value be $\beta(0) = 0.01$. Our tuning method consists of three rules which is described in Table II:

(Rule 1) If there is no pair which makes the objective function decrease and all neurons do not fire at $t$, $\beta(t)$ increases by $\beta(t + 1) \leftarrow \beta(t) + \beta(t)/n$.

(Rule 2) If there is no pair which makes the objective function decrease but at least one neuron fires at $t$, $\beta$ is reset by $\beta(t + 1) \leftarrow \beta(0)$.

(Rule 3) However, if there are one or more pairs which make the objective function decrease by exchanging them, $\beta$ does not change, namely $\beta(t + 1) \leftarrow \beta(t)$.

By Rule 3, our method can perform the 2-exchange method until the method reaches a local

![Graphs showing average gaps for different problems](image-url)
Fig. 2. Results of the gaps and firing rates of the problem Ste36b.

(a) $\beta = 0.01$, best gap = 19.5

(b) $\beta = 0.08$, best gap = 5.68

(c) $\beta = 1.00$, best gap = 18.9

Fig. 3. Results of the gaps and firing rates of the problem Tai50b.

(a) $\beta = 0.01$, best gap = 2.54

(b) $\beta = 0.02$, best gap = 0.91

(c) $\beta = 1.00$, best gap = 2.83
Table II. How the value of $\beta$ is decided in the proposed parameter tuning method.

| Rule | $\min_{ij} (\Delta_{ij})$ | Update rule of $\beta$ |
|------|--------------------------|------------------------|
| Rule 1 | $\geq 0$ | $\beta(t+1) \leftarrow \beta(t) + \beta(t)/n$ |
| Rule 2 | $\geq 0$ | $\beta(t+1) \leftarrow \beta(0)$ |
| Rule 3 | $< 0$ | $\beta(t+1) \leftarrow \beta(t)$ |

Table III. Results of the average gaps[\%] for (i) the chaotic search(CS) in [7], (ii) the proposed method without the parameter tuning method and (iii) the proposed method with the parameter tuning method. In the method (ii), the best parameter $\beta$ for each problem is shown in parenthesis. Numerals with bold faced types indicate the best gap, and italic faced types indicate the second best gap.

| Problem | (i) CS [7] | (ii) Proposed method without the parameter tuning | (iii) Proposed method with the parameter tuning |
|---------|-------------|-----------------------------------------------|-----------------------------------------------|
| Bur26a  | 0.159       | 0.128 (0.11)                                 | 0.0968                                        |
| Bur26b  | **0.0814**  | 0.165 (0.32)                                 | 0.131                                         |
| Bur26c  | 0.0496      | **0.0161 (0.12)**                           | 0.0521                                        |
| Bur26d  | 0.0234      | **0.00858 (0.11)**                          | 0.0327                                        |
| Ste36a  | 5.65        | 4.07 (0.05)                                  | 3.01                                          |
| Ste36b  | 12.7        | 5.10 (0.08)                                  | 3.01                                          |
| Ste36c  | 4.40        | 2.71 (0.06)                                  | 1.74                                          |
| Tai20b  | 1.80        | **0.989 (0.03)**                             | 0.948                                         |
| Tai30b  | 2.33        | **1.12 (0.03)**                              | 0.799                                         |
| Tai40b  | 3.70        | **1.83 (0.02)**                              | 1.76                                          |
| Tai50b  | 2.21        | **1.15 (0.02)**                              | 1.24                                          |
| Tai60b  | 2.52        | **1.16 (0.02)**                              | 1.13                                          |
| Tai80b  | 2.88        | **1.71 (0.02)**                              | 1.60                                          |

minimum in the present searching area. After reaching the local minimum, Rule 1 is applied to the parameter $\beta(t)$. This Rule 1 makes neurons easier to fire and thereby encourages our method to perform bad exchanges. By Rules 1 and 3, searching areas are effectively switched after our method reaches one of good local minima. However, if we use only Rules 1 and 3, the value of $\beta(t)$ monotonically increases without limitation. We thus introduced Rule 2.

In our method, $\beta(t+1)$ is recalculated between Step5 and Step6 in the algorithm described in Sec. 4. Note that the value of $\beta(t)$ does not converge, but fluctuate through the searching process.

5.2 Results

Table III shows the obtained best average gaps. Numerals with bold faced types indicate the best gap, and italic faced types indicate the second best gap. The third column of Table III shows average gaps of 100 trials for the best $\beta$ of the proposed method without the parameter tuning method. The fourth column of Table III shows average gaps of 100 trials of the proposed method including the parameter tuning method. The initial values of $\beta$ are 0.01 for all problems.

From Table III, even though the initial value of $\beta$ is the same for all problems, we can get better performance compared to the method in [7]. Next, we investigated how the parameter tuning method affects the firing rate.

In Sec. 4.2, we mentioned that the balance of searching range and depth is important. Our tuning method is aimed at keeping the balance as follows: when $\beta$ is small, the firings of neurons are inhibited and the normal 2-exchange method mainly operates repeatedly. After several iterations, the method is trapped into a local minimum. In this case, by our tuning method, the value of $\beta$ is gradually increased. At the same time, neurons are stimulated by gradually increasing external inputs because
Fig. 4. Gaps and firing rates for each iteration in Bur26b. The small figures included in (a) and (b) are enlargements of each figure.

(a) $\beta = 0.32$, best gap = 0.190

(b) $\beta$ under control by the parameter tuning method, best gap = 0.182

Fig. 5. Gaps and firing rates for each iteration in Ste36b. The small figures included in (a) and (b) are enlargements of each figure.

(a) $\beta = 0.08$, best gap = 5.68

(b) $\beta$ under control by the parameter tuning method, best gap = 1.85

of the increase of the value of $\beta$. After that, the neurons start firing and the exchanges that make the present solution worse are carried out. After several iterations, due to the refractory effect of neurons, the firings of neurons are inhibited and the normal 2-exchange method operates again. In other words, our tuning method can create a positive feedback loop: finding a good local minimum and then go to other searching space by the neurodynamics. In Figs. 4–6, the ordinates show the gaps and the firing rates, and the abscissas show the iteration. Figures 4(a), 5(a), and 6(a) show the results for the case that $\beta$ is constant. On the other hand, Figs. 4(b), 5(b), and 6(b) show the results for the case that $\beta$ is controlled by the parameter tuning method. In Figs. 4–6, the small figures included
in (a) and (b) are enlargements of each figure. Using the parameter tuning method, the firing rate is decreased and the proposed method can reach one of good local minima before switching searching space. Then, $\beta$ is gradually increased. This makes neurons fire easily after reaching a local minimum.
and escape from the local minima. Figure 7 shows the transition of the value of \( \beta \). From Fig. 7, the value of \( \beta \) increases and decreases repeatedly by our tuning method. As compared with the gaps, when the values of \( \beta \) increase, the gaps decrease, that is, our tuning method works well so that our neurodynamics-based method can achieve a deep and wide search.

6. Conclusion

In this paper, we proposed a new algorithm for solving Quadratic Assignment Problem (QAP) by chaotic neurodynamics. In our method, we reduced the number of neurons and changed the effect of external input: the worse the exchange is, the stronger the input is applied. This modification enables the proposed method to widely search the solution space. By comparing with the performance of the method of chaotic search with \( n^2 \) neurons in [7], our method can get better performance. However, the performance of the proposed method largely depends on the value of the parameter \( \beta \). Therefore, we introduced a parameter tuning method. As a result, we can get better performance compared to the method in [7] without manual parameter tuning of \( \beta \).

As we have already mentioned that neurons in our method are likely to fire when the exchange is bad, which is different from the previous methods [6–8]. In the present work, we only proposed a parameter tuning method in Sec. 5, it is one of the important future works to improve the tuning method and develop other effective parameter tuning methods.

Acknowledgments

The authors would like to thank Profs. Y. Horio and M. Hasegawa for their valuable comments and discussion. The authors would like to thank Mr. S. Ishii and AGS Corporation for their kind encouragements on this research. The research is supported by JSPS KAKENHI (Nos.26880020, 15K12137, 16K16138, 24650116, 15KT0112 and 17K00348).

References

[1] E.L. Lawler, “The quadratic assignment problem,” Management science, vol. 9, pp. 586–599, 1963.
[2] “QAPLIB - A Quadratic Assignment Problem Library,” http://www.opt.math.tu-graz.ac.at/qaplib/
[3] F. Glover, “Tabu search-part I,” ORSA Journal on Computing, vol. 1, pp. 190–206, 1989.
[4] F. Glover, “Tabu search-part II,” ORSA Journal on Computing, vol. 2, pp. 4–32, 1990.
[5] D.M. Tate and A.E. Smith, “A genetic approach to the quadratic assignment problem,” Computers & Operations Research, vol. 22, pp. 73–83, 1995.
[6] M. Hasegawa, T. Ikeguchi, and K. Aihara, “Combination of chaotic neurodynamics with the 2-opt algorithm to solve traveling salesman problems,” Physical Review Letters, vol. 79, pp. 2344–2347, 1997.
[7] M. Hasegawa, T. Ikeguchi, K. Aihara, and K. Itoh, “A novel chaotic search for quadratic assignment problems,” European Journal of Operational Research, vol. 139, pp. 543–556, 2002.
[8] M. Hasegawa, T. Ikeguchi, and K. Aihara, “Solving large scale traveling salesman problems by chaotic neurodynamics,” Neural Networks, vol. 15, no. 2, pp. 271–283, 2002.
[9] K. Aihara, T. Takabe, and M. Toyoda, “Chaotic neural networks,” Physics Letters A, vol. 144, pp. 333–340, 1990.
[10] H. Ohnishi, Y. Shimada, K. Fujiwara, and T. Ikeguchi, “A chaotic search with small memory consumption for solving QAP,” Proceedings of International Symposium on Nonlinear Theory and its Applications (NOLTA2015), pp. 610–613, 2015.