2005 International Symposium on Nonlinear
Theory and its Applications (NOLTA2005)
Bruges, Belgium, October 18-21, 2005

Parallel Algorithms for Chaotic Exponential Tabu Search Hardware for Quadratic Assignment Problems

Naoki Ogawa†, *Yoshihiko Horio*† and *Kazuyuki Aihara*‡*

†Graduate School of Engineering, Tokyo Denki University, 2–2 Kanda–Nishiki–cho, Chiyoda–ku,
Tokyo, 101–8457, Japan. Email: 05gmd06@ed.cck.dendai.ac.jp Tel: +81–3–5280–3362
‡Aihara Complexity Modeling Project, ERATO, JST, 3–23–5 Uehara, Shibuya–ku, Tokyo, 151–0064, Japan
∗ Institute of Industrial Science, University of Tokyo, 4–6–1 Komaba, Meguro–ku, Tokyo, 153–8505, Japan

Abstract—The quadratic assignment problem (QAP) is one of the nondeterministic polynomial (*NP*)-hard combinatorial optimization problems. One of the heuristic algorithms for the QAP is tabu-search. Exponential tabu-search, which is one of improved versions of the tabu search, was proposed using a neural network, and it was further extended to chaotic tabu search with a chaotic neural network. Chaotic dynamics shows efficient solution search ability, and at the same time, it avoids the local minima problem. We have proposed and built a chaotic exponential tabu search hardware system with switched-current chaotic neuron ICs for size-10 QAPs. The measurement results have confirmed the validity and good performance of the system. In this paper, we propose two parallel processing algorithms for the chaotic exponential tabu search hardware to accelerate the processing speed of the system, in order to solve large-scale QAPs.

1. Introduction

The quadratic assignment problem (QAP) is one of the *NP*-hard combinational optimization problems [1]. Obtaining the optimal solution of the QAP requires impractical computational time. Currently, the optimum solution of the QAP with size more than 36 is unknown [1]. Therefore, heuristic methods are very important to find good near optimum solutions in reasonable time.

Tabu search with 2-opt algorithm is one of such heuristic methods [2], which escapes from undesirable local minima by using a tabu list. Once a certain 2-opt exchange is stored in the tabu list, the same exchange is forbidden for x iterations, where x is a tabu list size, and becomes available after x iterations.

Hasegawa et al. proposed an implementation of the tabu search on a neural network [3]–[5]. Furthermore, they extended the ordinary tabu to an exponential tabu by using exponential decay of the refractoriness of a neuron model. Moreover, they introduced chaotic search into the exponential tabu search by replacing static neurons with chaotic neurons [3]–[5]. As a result, they confirmed superior performance of the chaotic exponential tabu search in solving the QAP through computer simulations.

We proposed an efficient mixed analog/digital circuit hardware for the chaotic exponential tabu search algorithm [6], [7]. The hardware system uses switched-current chaotic neuron integrated circuits for a rapid implementation of physical chaotic dynamics [8], [9]. However, the hardware system uses a sequential update of a neuronal states, so that $n \times n$ updates (n is the size of the QAP) are required for one iteration of the algorithm. Therefore, it would take a large amount of time for a large-size QAP.

In this paper, we propose two parallel processing algorithms suitable for the hardware implementation of the chaotic exponential tabu search. We confirm by simulations that the speed of the solution search is greatly improved with the proposed methods.

2. Quadratic Assignment Problem

The QAP of size n consists of n locations and n units. An $n \times n$ "distance" matrix denotes mutual distances among the locations. Moreover, an $n \times n$ "flow" matrix expresses mutual relations among the units. The QAP is defined such that we should find an assignment of the units to the locations that minimizes the cost function $F^{\boldsymbol{p}}$ given by

$$F^{\boldsymbol{p}} = \sum_{g=1}^{n} \sum_{h=1}^{n} a_{gh} b_{p(g)p(h)}, \tag{1}$$

where $\boldsymbol{p}$ is a permutation of n elements given by eq. (2), which expresses one of the feasible solutions, a_{gh} is the distance between the locations g and h, $b_{p(g)p(h)}$ is the flow between the units $p(g)$ and $p(h)$, and $p(g)$ represents the gth element of the permutation $\boldsymbol{p}$.

$$\boldsymbol{p} : (p(1),\ p(2), \cdots,\ p(g), \cdots,\ p(h), \cdots,\ p(n)). \tag{2}$$

If the current permutation $\boldsymbol{p}$ gives the minimum of $F^{\boldsymbol{p}}$, $\boldsymbol{p}$ is the optimal solution.

3. Chaotic Exponential Tabu Search

Assuming the size of the problems is n, the neural network composed of $n \times n$ chaotic neurons as shown in Fig. 1(a) is prepared. In [3]–[5], the state of each neuron is updated one by one from the $(1, 1)$th neuron in the network to the (n, n)th neuron. We denote this update sequence as

one "iteration." If the (i, j)th neuron in the network fires on the course of updating, the element i of the permutation $\boldsymbol{p}$ is assigned to the index j as shown in Fig. 1(b). At the same time, the element $p(j)$ is assigned to the index $q(i)$. These exchanges are referred to "(i, j)" and "$(p(j), q(i))$" assignments, respectively.

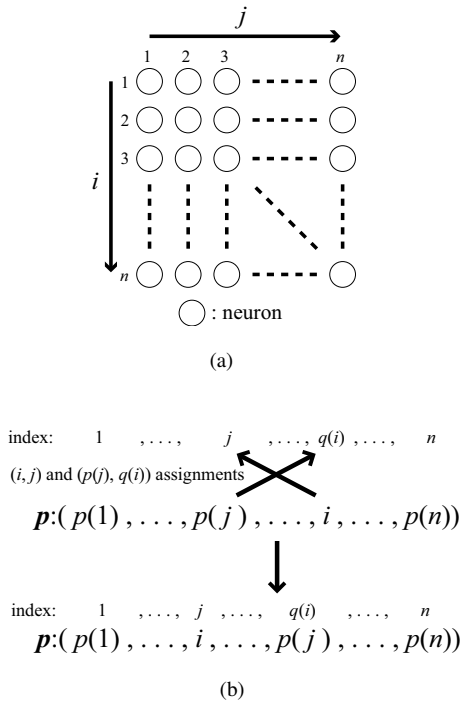

Figure 1: (a) The neural network for the QAP of size n, and (b) the permutation $\boldsymbol{p}$ and the (i, j) and $(p(j), q(i))$-assignments.

The chaotic dynamics of the (i, j)th neuron in the network used in our hardware system is defined as follows [10]:

$$\xi_{ij}(t+1) = \beta\{F_1^{\boldsymbol{p}}(t) - F_{ij}^{\boldsymbol{p}}(t)\}, \quad (3)$$

$$\eta_{ij}(t+1) = -W\sum_{k=1}^{n}\sum_{l=1}^{n} x_{kl}(t) + k_f\eta_{ij}(t) - \alpha x_{p(j)q(i)}(t) + R, \quad (4)$$

$$\zeta_{ij}(t+1) = k_r\zeta_{ij}(t) - \alpha x_{ij}(t) + R, \quad (5)$$

$$x_{ij}(t+1) = f\{\xi_{ij}(t+1) + \eta_{ij}(t+1) + \zeta_{ij}(t+1)\}, \quad (6)$$

where $F_1^{\boldsymbol{p}}(t)$ is the current cost function, $F_{ij}^{\boldsymbol{p}}(t)$ is the cost after the (i, j)-assignment. Therefore, the internal state $\xi_{ij}(t+1)$ gives the gain of the cost function resulting from the (i, j)-assignment, where β is a scaling parameter of the gain. Moreover the internal state $\eta_{ij}(t)$ is a sum of feedback from other neurons and tabu effect for the $(p(j), q(i))$-assignment, the internal state $\zeta_{ij}(t)$ is the tabu effect for the (i, j)-assignment, α is a scaling parameter for the tabu effect, k_f and k_r are decay parameters for the tabu effect, R is an external bias, W is a coupling coefficient between neurons, and $f(\cdot)$ is a monotonically increasing nonlinear output function of the neuron. During the sequential updating process, the (i, j)th neuron fires if $x_{ij}(t+1) \geq 0.5$.

4. Simplified Gain Calculation

Eq. (3) gives a gain resulting from the (i, j)-assignment. By substituting eq. (1) to eq. (3), the gain can be simplified as shown by the following equation.

$$\begin{aligned} F_1^{\boldsymbol{p}}(t) - F_{ij}^{\boldsymbol{p}}(t) = {} & a_{jj}(b_{p(j)p(j)} - b_{ii}) \\ & + a_{jq(i)}(b_{p(j)i} - b_{ip(j)}) \\ & + a_{q(i)j}(b_{ip(j)} - b_{p(j)i}) \\ & + a_{q(i)q(i)}(b_{ii} - b_{p(j)p(j)}) \\ & + \sum_{k=1,k\neq i,j}^{n}\{a_{kj}(b_{p(k)p(j)} - b_{p(k)i}) \\ & + a_{kq(i)}(b_{p(k)i} - b_{p(k)p(j)}) \\ & + a_{jk}(b_{p(j)p(k)} - b_{ip(k)}) \\ & + a_{q(i)k}(b_{ip(k)} - b_{p(j)k})\}, \end{aligned} \quad (7)$$

where a_{ij} and b_{ij} are the (i, j)th elements of the distance matrix and the flow matrix, respectively. We use eq. (7) in the hardware implementation of the system.

5. Parallel Algorithms

We constructed the hardware system for size-10 QAPs, and confirmed good performance of the system [6], [7]. However, the current system updates one neuron at a time. Therefore, if a large number of neurons are used for a large-scale QAP, it would take long time to complete one iteration, that is, to update all the neurons once.

When one of the neurons is updated, all of the remaining neurons are idle and wasting time. Therefore, we can exploit this idling time to realize parallel updatings of some neuronal states. The parallel calculation of the gains given by eq. (7) for some neurons, of which internal states are updated simultaneously, realizes a parallel processing scheme for the exponential chaotic tabu search hardware system. We propose herein two such parallel algorithms.

5.1. m-Neuron-Parallel Algorithm

The first algorithm is the m-neuron-parallel algorithm. In the algorithm, we first pick m consecutive neurons in the network starting from the first neuron of the network, i. e. the $(1, 1)$th neuron. Then, we update all of these neurons, and calculate the gains of the cost function for these neurons simultaneously. Finally, we chose the neuron that has the largest output value. If the output of this neuron exceeds the firing threshold, i. e. 0.5, this neuron fires, and the elements of the permutation $\boldsymbol{p}$ corresponding to this neuron

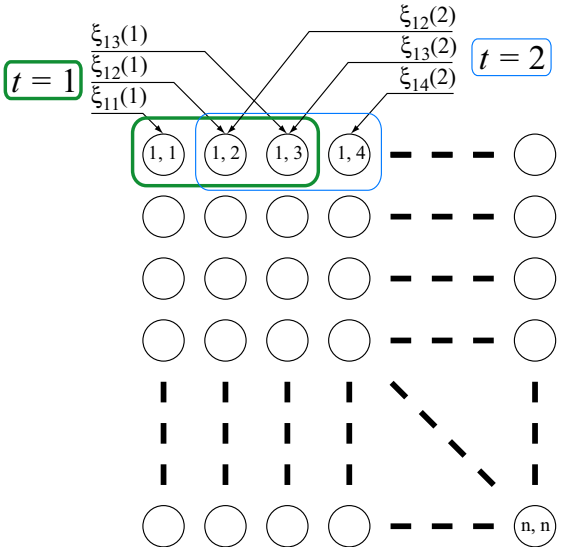

Figure 2: The m-neuron-parallel algorithm (m=3).

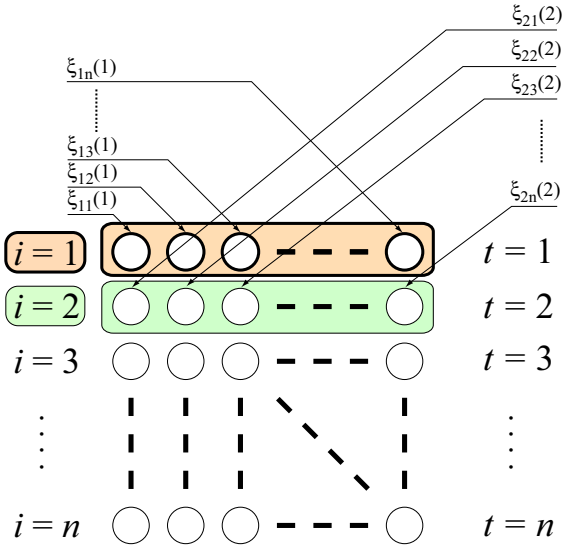

Figure 3: The row-parallel algorithm.

are exchanged. After this process, we chose next m neurons from the network, and continue the process. For example, as shown in Fig. 2, when $m = 3$, the first neuron group consists of the $(1, 1)$th, $(1, 2)$th and $(1, 3)$th neurons, the second group is composed of the $(1, 2)$th, $(1, 3)$th and $(1, 4)$th neurons, and so on. Thus, the final neuron group consists of the (n, n)th, $(1, 1)$th and $(1, 2)$th neurons. One iteration is completed when the group of m neurons returns to the first group. Therefore, a total of m updates are executed for each neuron in one iteration. In contrast, the original algorithm updates each neuron only once in one iteration.

5.2. Row-Parallel Algorithm

The second algorithm is the row-parallel algorithm. In this algorithm, neurons in the same row are independently updated simultaneously, according to the original algorithm. As shown in Fig. 3, the first neuron group consists of $(1, 1)$th to $(1, n)$th neurons, the second neuron group is composed of the $(2, 1)$th to $(2, n)$th neurons and so on. Like the m-neuron-parallel algorithm, the neuron that has the largest output in each group fires if the output exceeds the firing threshold, and the corresponding exchanges are executed for the permutation $\boldsymbol{p}$.

In the m-neuron-parallel algorithm, each neuron is included in the updating neuron group m times in one iteration. In contrast, in the row-parallel algorithm, each neuron is updated only once in one iteration, because the index of the head of the updating neuron group is incremented by n, which is the size of the problem, so that each neuron is included in the group only once throughout one iteration.

6. Simulation Results

We solved the benchmark problems, Tai10a, Tai12a, and Tai15a from the QAP library [1] using the proposed parallel processing algorithms. Here, we define the normalized average iteration number (NAIN) for obtaining the optimal solution as

$$\mathrm{NAIN} = \frac{\mathrm{AIN}}{\mathrm{AINWOPP}}, \tag{8}$$

where AIN is the average iteration number to obtain the optimal solution using the proposed parallel processing algorithm, and AINWOPP is that without any parallel scheme, that is, using the original sequential update. These iteration numbers are averaged over 100 trials with different initial conditions.

Fig. 4 shows NAIN with the m-neuron-parallel algorithm when the number of parallel updated neurons m is changed. As shown in the figure, the processing speed is improved as m is increased. Furthermore, a proper number of the parallel neurons m would be equal or around the size of the problem n, i. e. $m \approx n$, because the improvement seems to saturate even if we further increase m.

Results for the m-neuron-parallel algorithm with $m = 20$ and the row-parallel algorithm are summarized in Table. 1. The NAINs of row-parallel algorithm are calculated such that we divide the AINs by 20 so that each neuron is updated with the same number of counts as in the m-neuron-parallel algorithm with $m = 20$. As shown in the table, the speed of the m-neuron-parallel algorithm is about ten times faster than the original non-parallel algorithm. Moreover, the m-neuron-parallel algorithm is superior to the row-parallel algorithm in terms of the efficiency of the chaotic search. However, the row-parallel algorithm would have an advantage for the hardware implementation, be-

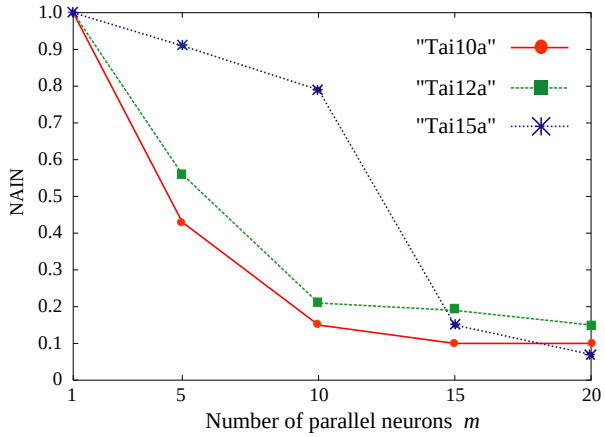

Figure 4: Simulation results for the m-neuron-parallel algorithm. The normalized average iteration numbers defined by eq. (8) are plotted with different m for three QAP instances.

Table 1: Summary of the simulation results. NAINs given in eq. (8) are shown.

Instances	Tai10a	Tai12a	Tai15a
m-neuron-parallel (m=20)	0.10	0.15	0.07
row-parallel	0.72	0.69	0.79

cause the modularity of the row-parallel algorithm is suitable for a modular structure of the hardware system. The overall assessment of the total performance of these parallel algorithms considering the actual hardware restrictions is an important future problem.

7. Conclusions

We have proposed two parallel processing algorithms, that is, m-neuron-parallel and row-parallel algorithms, for the chaotic exponential tabu search hardware. We have shown that the m-neuron-parallel algorithm accelerates the processing speed about ten times as fast as the original sequential algorithm.

In a future investigation, we intend to use different sizes and instances of QAPs for further analyses of the parallel algorithms. Moreover, we will improve the proposed algorithms. In addition we will develop a parallel processing hardware system based on the proposed algorithms.

Acknowledgments

The authors would like to thank T. Ikeguchi of Saitama University and M. Hasegawa of Communications Research Laboratory for their valuable discussions. This work was supported in part by Grant-in-Aid no. 16300072 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] R. E. Burkard, S. E. Karisch and F. Rendl, "QAPLIB - A quadratic assignment problem library," available via world wide web to http://www.opt.math.tu-graz.ac.at/qaplib/

[2] E. Taillard, "Robust tabu search for the quadratic assignment problems," *Parallel computing,* vol. 17, pp. 443–455, 1991.

[3] M. Hasegawa, T. Ikeguchi and K. Aihara, "A novel chaotic search for quadratic assignment problems," *European Journal of Operational Research,* vol. 139, pp. 543–556, 2002.

[4] M. Hasegawa, T. Ikeguchi and K. Aihara, "A novel chaotic search for combinatorial optimization," in *Proc. Int. symp. Nonlinear Theory and Its Applications,* pp. 613–616, 1997.

[5] M. Hasegawa, T. Ikeguchi and K. Aihara, "Exponential and chaotic neurodynaminal tabu searches for quadratic assignment problems," *Control and Cybernetics,* vol. 29, no. 3, pp. 773–788, 2000.

[6] S. Matsui, Y. Kobayashi, K. Watanabe, and Y. Horio, "Exponential chaotic search hardware for quadratic assignment problems using switched-current chaotic neuron IC," in *Proc. IJCNN,* vol. 3, pp. 2221–2226, 2004.

[7] S. Matsui, Y. Horio, and K. Aihara, "Parameter settling for chaos-driven tabu search hardware system," in *Proc. RISP Int. Workshop on Nonlinear Circuit and Signal Processing,* pp. 151–154, 2005.

[8] R. Herrera, Y. Horio and K. Suyama, "IC implementation of a current-mode chaotic neuron," in *Proc. IEEE ISCAS,* vol. 3, pp. 546–549, 1998.

[9] R. Herrera, Y. Horio and K. Suyama, "Realizing the chaotic neuron model: IC solution," in *Proc. Int. symp. Nonlinear Theory and Its Applications,* vol. 1, pp. 625–628, 1997.

[10] K. Tanaka, Y. Horio and K. Aihara, "A modified algorithm for the quadratic assignment problem using chaotic-neuro-dynamics for VLSI implementation," *in Proc. IJCNN,* pp. 240–245, 2001.