Neurocomputing 4 (1992) 167–179
Elsevier

A parallel algorithm for solving Unfriendly Beehive Problems

J.D. Rofkar and Y. Takefuji

Department of Electrical Engineering, Case Western Reserve University, Cleveland, OH 44106, USA

Received 25 July 1991
Revised 20 September 1991

Abstract

Rofkar, J.D. and Y. Takefuji, A parallel algorithm for solving Unfriendly Beehive Problems, Neurocomputing 4 (1992) 167–179.

This paper presents a general purpose algorithm for solving the Unfriendly Beehive Game. The proposed algorithm will utilize an artificial neural network to solve the problem. The neural network will use a simple partial differential (motion) equation expressed in terms of its natural constraints. Each constraint within the equation represents a simple connection to an artificial neuron. Each connection strength (or synaptic strength) is weighted by multiplicative constants and summed together. The result is an input that is adjusted in the direction that decreases the error or conflict. Using this information, the system derives a simple binary state output in an attempt to solve the puzzle. Given an overall time slot in which to resolve all conflicts, the system iteratively strives to arrive at the state of the global minimum. The proposed algorithm will be able to solve a variety of real-world problems, including: facility layouts for maximizing productivity and safety, classroom assignment for minimizing pupil conflict, crop and plant placement for maximizing yields, and chemical placement within shipping boxes to reduce the possibility of chemical interactions.

Keywords. Parallel algorithm; Unfriendly Beehive Problem; combinatorial optimization.

1. Introduction

The problem behind the Unfriendly Beehive Game (a.k.a. the 1 to 19 Game) is to place the integers 1 through 19 into a 19-element hexagonal matrix (see *Fig. 1*) so that the difference between any element and a neighboring element is four (4) or greater. The actual story behind the problem goes: There are workers in a beehive colony who are strictly ranked from 1 to 19. Because they are so competitive, no worker wants to be near another who is close to his/her in rank. Therefore, arrange them in the cells so that no two adjacent workers are closer than some pre-defined rank apart (in this case 4). The oldest known publication of this problem appeared in the British magazine *Games and Puzzles* in 1971, but the best that could be achieved at that time was a rank difference of four. Jellis [1] and Pritchard [2] clearly defined the rank difference (rd) as four. However, Rubin [3] was able to show that this problem could be solved with a rank difference of 5 (rd = 5), instead of 4. Although solutions to the Unfriendly Beehive Problem do exist [1–3], there have been no proposed algorithms for solving it.

Correspondence to: Y. Takefuji, Department of Electrical Engineering, Case Western Reserve University, Cleveland, OH 44106, USA.

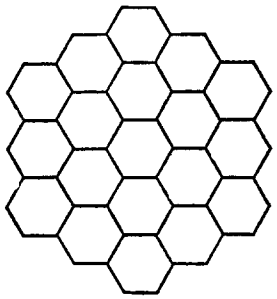


Fig. 1. Can you place the numbers 1 to 19 in the honeycomb's nineteen cells in such a way that there is a difference of at least 4 between anyone cell and its neighboring cells?

2. Approach

This paper will present a parallel algorithm based on a two-dimensional artificial neural network for finding a conflict-free solution to the Unfriendly Beehive Problem. The artificial neural network model uses a large number of massively interconnected simple processing elements. The processing elements in the neural network are called neurons because they perform the same function as simplified biological neurons. The interconnections between processing elements are determined by the motion equation:

$$\frac{dU_{ij}}{dt} = - \frac{\partial E(V_{11}, V_{12}, ..., V_{nn})}{\partial V_{ij}} ,$$

where n is the number of nodes in the hexagonal matrix, U_{ij} is the input and V_{ij} is the output of the ijth neuron. Note, E is called the computational energy function given by the necessary and sufficient constraints of the Unfriendly Beehive Problem.

The goal of using the artificial neural network is to minimize the energy function E (see *Fig. 2* for a representation of a typical energy landscape for an optimization problem). Theorem 1 in the Appendix shows that the motion equation performs a gradient descent method to minimize the energy function E – thus, converging at global minimum and, consequently, a solution to the Unfriendly Beehive Problem.

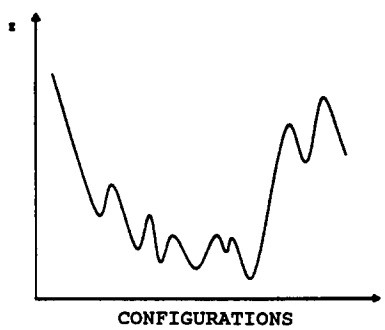


Fig. 2. Representation of a typical energy landscape for an optimization problem.

The first mathematically based neuron model was proposed by McCulloch and Pitts in 1953 [4]. Their input/output function of their ijth neuron was given by:

$$V_{ij} = \begin{cases} 1\,, & \text{if } U_{ij} > 0 \\ 0\,, & \text{otherwise} \end{cases} .$$

The sigmoid neural network model for solving combinatorial optimization problems was first introduced by Hopfield and Tank [5]. The input/output function of the ijth processing element in their neural model is given by:

$$V_{ij} = \tfrac{1}{2}\,[1 + \tan h(\lambda U_{ij}]\,,$$

where λ is a gain parameter. Takefuji proved that the decay term in the Hopfield neural network is harmful for the convergence of the system [6]. In other words, the decay term disturbs the system convergence under certain conditions. Therefore, the decay term is not used in our algorithm. Our method, however, is based on the Hopfield neural network model and the McCulloch-Pitts binary neuron since both have been used for finding near-optimum solutions of several NP-complete and/or optimization problems [6–14, 20–22].

However, the McCulloch-Pitts model sometimes produces undesirable oscillatory behavior. To suppress the oscillatory behavior of the neural network, we have introduced the property of hysteresis – which consequently shortens convergence times! Theorem 2 in the Appendix shows that the discrete motion equation based on the first-order Euler method forces the state of the system composed of the hysteresis McCulloch-Pitts neurons to converge to the local minimum.

3. System representation

The proposed algorithm utilizes a hill-climbing term (h). The hill-climbing term forces the state of the system out of a local minimum to converge into the state of the global minimum. The main difference of the proposed hill-climbing and the hill-climbing in [9] is its unique ability to both shrink and grow. By giving this property to the hill-climbing term, it increases the system's chances of 'climbing' out of a local minimum successfully – at the right time. This property will become more evident as we discuss the motion equation below. The hill-climbing's influence (C) is controlled by the number of program iterations (t).

The algorithm uses two $N \times N$ square matrices for both input (U_{ij}) and output (V_{ij}) composed of N^2 simple processing elements each and extra N elements are used to store current values and to help determine the output state. N is the number of nodes in a perfect hexagonal matrix ($N = 19$ for the original Unfriendly Beehive Problem). A perfect hexagonal matrix can be defined in terms of the number of layers (L) surrounding the middle hexagon, where each layer is composed of the minimum number of hexagons required to surround the previous layer. Thus, $N = 3L(L + 1) + 1$. Output (V) of the ijth neuron follows:

$$V_{ij}(t) = \begin{cases} 1\,, & \text{if } U_{ij}(t) > \text{UTP (Upper Trip Point)} \\ 0\,, & \text{if } U_{ij}(t) < \text{LTP (Lower Trip Point)} \\ \text{Unchanged}\,, & \text{otherwise} \end{cases} .$$

Note that $U_{ij}(t)$ is the input of the ijth neuron at time (t).

The motion equation for the ijth neuron is as follows:

$$\frac{dU_{ij}}{dt} = -A\left(\sum_{k=1}^{N} V_{ik} - 1\right) - A\left(\sum_{k=1}^{N} V_{kj} - 1\right) \times$$

$$- B \left(6 - \sum_{k=1}^{6} g(|N_j - N_{j_k}|) \right) +$$

$$+ C \left[h \left(\sum_{k=1}^{N} V_{ik} \right) \right] + C \left[h \left(\sum_{k=1}^{N} V_{kj} \right) \right] +$$

$$+ C \left[\sum_{k=1}^{6} h\Big(g(|N_j - N_{j_k}|) \Big) \right] ,$$

where $g(x)$ is a boolean function:

$$g(x) = \begin{cases} 1, & \text{if } x \geq \text{rd} \\ 0, & \text{otherwise} \end{cases} .$$

N_j is the value of node j, and N_{jk} is the value of node j's k^{th} neighbor (for $k = 1...6$). $h(x)$ is the hill-climbing funtion:

$$h(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise} \end{cases} .$$

The first and second terms force the system to fire one and only one neuron per row and comumn, respectively. The third term forces the system to fire on values that make the difference between any two neighboring elements (nodes) greater than or equal to the rank difference (rd). It achieves this by finding the difference between the current node values (N_j) and all the k^{th} edge values for $k = 1...6$. The last three terms are the hill-climbing terms which help the state of the system to escape from local minima by increasing the frequency to converge to the state of the global minimum. A, B, C, MIN, and MAX are empirically derived constant coefficients which act as the weights between processing elements.

The following procedure outlines the proposed parallel algorithm for the *Unfriendly Beehive Problem*:

1. Set $t = 0$, $A = 1$, $B = 2$, $C = 1$, MIN $= -13$, MAX $= 13$.
2. Randomize the input matrix (U_{ij}) for $i = 1...N$ and for $j = 1...N$ with values between MIN and MAX.
3. Initialize the output matrix (V_{ij}) along with the additional N_j nodes for $i = 1...N$ and for $j = 1...N$ to the value 0.
4. Compute the output $(V_{ij}(t))$ based on this initialization for $i = 1...N$ and for $j = 1...N$ for time t.
5. Compute the input matrix $(U_{ij}(t+1))$ for $i = 1...N$ and for $j = 1...N$ for time $t+1$ based on the first order Euler method.
6. Confine the input matrix (U_{ij}) to MAX and MIN levels for $i = 1...N$ and for $j = 1...N$.
7. If all conflicts are resolved then END, else increment time by 1 ($t = t + 1$) and GOTO step 4.

The following pseudo-code algorithm outlines the basis for simulating the synchronous UFBP parallel algorithm on a sequential machine:

```
Program Parallel_Simulator_on_a_Sequential_Machine
...
{INITIALIZATION}
for i := 1 to N do begin
    for j := 1 to N do begin
        {Initialize U_ij, V_ij, and additional N_j nodes}
    end;
end;
...
{MAIN}
while (conflicts remain) do begin
    ...
    {LOOP 1: Update all input values}
    for i := 1 to N do begin
        for j := 1 to N do begin
            U_ij := U_ij + (delta)U_ij
        end;
    end;
    ...
    {LOOP 2: Update all output values}
    for i := 1 to N do begin
        for j := 1 to N do begin
            if (U_ij > UTP) then V_ij := 1
            else if (U_ij < LTP) then V_ij := 0;
        end;
    end;
    ...
end;
```

It is quite simple to simulate a synchronous parallel system on a sequential machine. In the first loop, all input values U_{ij} are sequentially updated while all output values V_{ij} are fixed. In the second loop, all output values V_{ij} are sequentially updated while all input values U_{ij} are fixed. It is equivalent to simultaneously updating the values of all inputs and outputs. Currently, an asynchronous parallel system simulator is under investigation on a sequential machine.

4. Testing and validation

We verified our algorithm by solving 1000 cases for each rank difference (4 and 5) where the size of the hexagonal matrix remained constant ($N = 19$). In each case, the input matrix (U_{ij}) was randomly initialized with values between MIN and MAX, and the output matrix (V_{ij}) along with the additional N_j nodes were initialized to 0 for $i = 1...N$ and for $j = 1...N$. Results of these tests can be seen in *Figs. 3* and *4* for rank differences of 4 and 5, respectively.

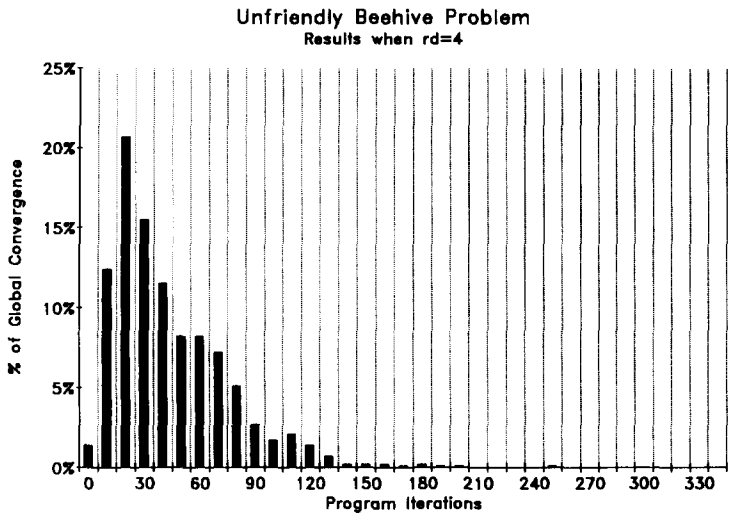


Fig. 3. Relationship between program iterations and frequency that the program will converge to global minimum. Less than 100 iterations accounted for over 90% of all solutions. RD = 4, $N = 19$.

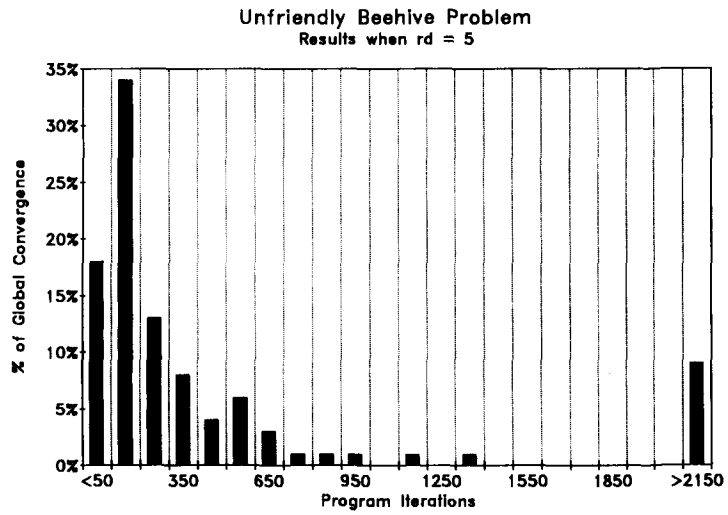


Fig. 4. Relationship between program iterations and frequency that the program will converge to global minimum. RD = 5, $N = 19$.

Accompanying these results are possible solutions for rd = 4 and rd = 5, where $N = 19$ (*Figs. 5* and *6*) as well as *Fig. 7* which shows snapshots of two sample runs for rd = 4. Once our algorithm was validated, we increased the number of layers (L). We present a few solutions for $N > 19$ (*Figs. 8–14*). It must be noted that the rank difference can also increase for $N > 19$. Because of this fact, we have included *Table 1* which lists the maximum rank difference for $19 \leq N \leq 91$ for which our algorithm was able to produce solutions.

Table 1. Empirical results for maximum rank difference per layer.

Layers	Number of nodes	Rank difference
2	19	5
3	37	8
4	61	13
5	91	20

Table 2. Summary of simulation results.

N	RD	Average iteration steps to convergence	convergence frequency to solutions
19	4	57.7	99.6%
19	5	105.8	66.2%
37	7	89.2	99.5%
37	8	149.0	91.5%
61	13	189.2	54.0%
91	20	238.8	18.0%

Empirical results indicate that the rank difference increases by a factor of the next odd number for each additional layer. Thus, it appears that the maximum rank difference can be calculated as:

$$\mathrm{rd} = 5 + \sum_{k=1}^{L-2} (2k + 1),$$

where L equals the number of layers, and $L > 2$. *Table 2* shows some simulation results including average iteration steps and convergence frequency.

The Unfriendly Beehive Game computer implementation was developed on an IBM PC/AT compatible 386SX-20 using Borland's Turbo C++.

Appendix

Theorem 1. *The system always satisfies* $\frac{dE}{dt} \leq 0$ *under two conditions:*

(i) $\frac{dU_{ij}}{dt} = -\frac{\partial E}{\partial V_{ij}}$, *and*

(ii) $V_{ij} = f(U_{ij})$,

where E is the computational Lyapunov energy function and $f(U_{ij})$ is a nondecreasing function.

Proof. Consider the derivatives of the computational energy E with respect to time t.

$$\frac{dE}{dt} = \sum_{i}^{N} \sum_{j}^{N} \frac{dV_{ij}}{dt} \frac{\partial E}{\partial V_{ij}} =$$

$$= \sum_{i}^{N} \sum_{j}^{N} \frac{dU_{ij}}{dt} \frac{dV_{ij}}{dU_{ij}} \frac{\partial E}{\partial V_{ij}} =$$

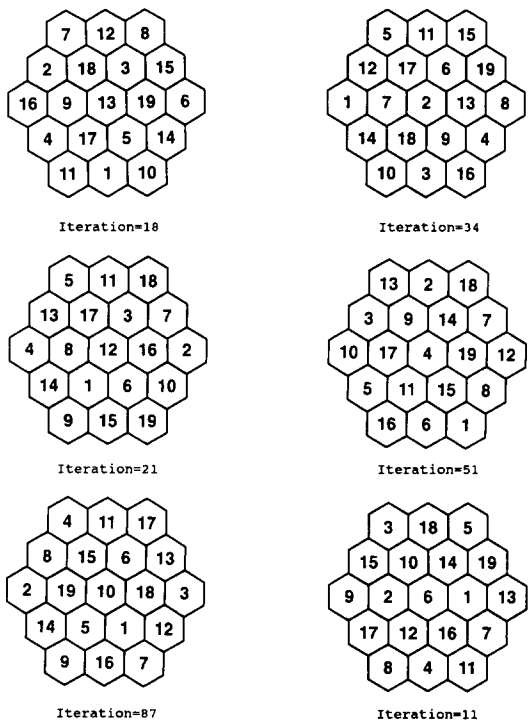


Fig. 5. Sample solutions for rank difference = 4.

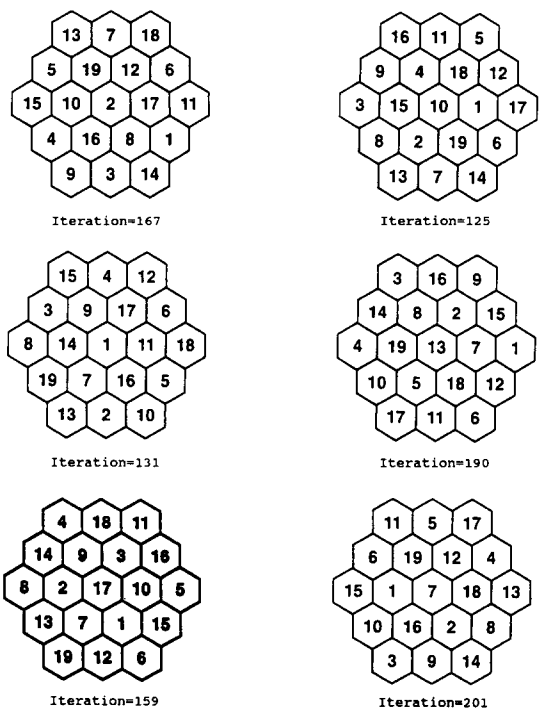


Fig. 6. Sample solutions for rank difference = 5.

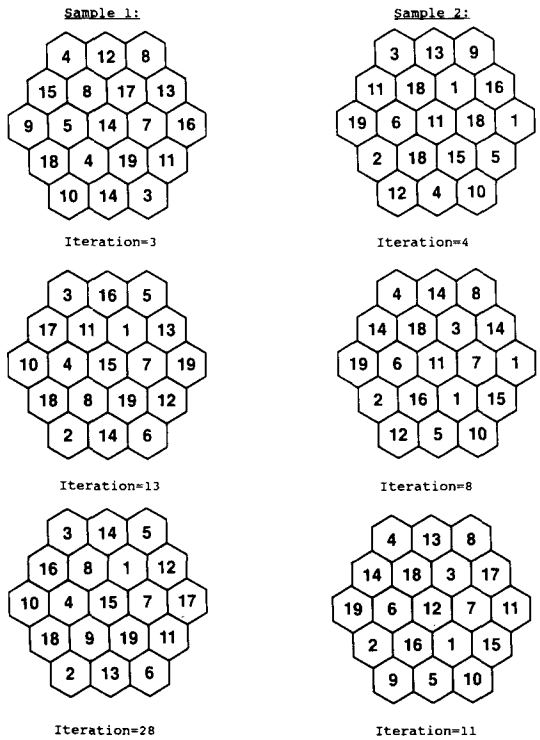


Fig. 7. Three snapshots and solutions from two sample runs. rd = 4.

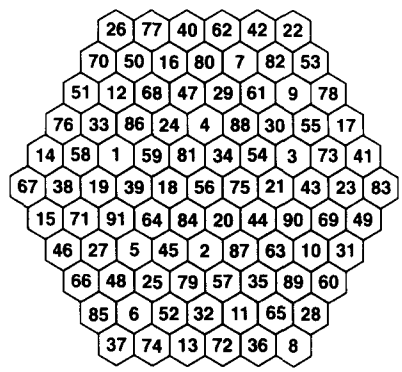


Fig. 8. Solution for $N = 91$, $L = 5$, RD = 18. Iteration = 67.

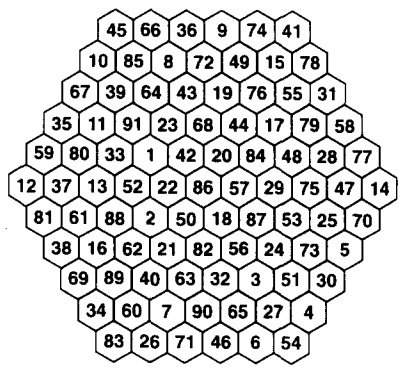


Fig. 9. Solution for $N = 91$, $L = 5$, RD = 19. Iteration = 64.

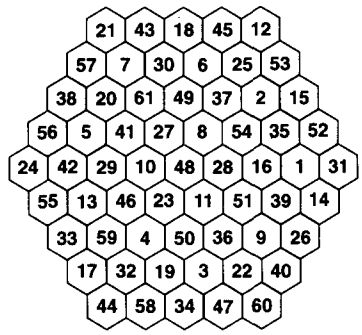


Fig. 10. Solution for $N = 61$, $L = 4$, RD $= 12$. Iteration $= 88$.

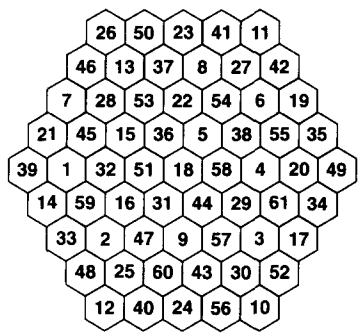


Fig. 11. Solution for $N = 61$, $L = 4$, RD $= 13$. Iteration $= 78$.

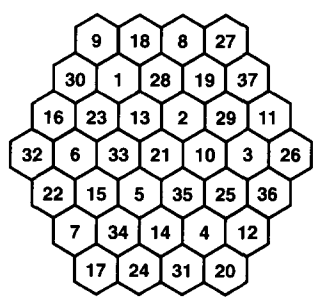


Fig. 12. Solution for $N = 37$, $L = 3$, RD $= 7$. Iteration $= 142$.

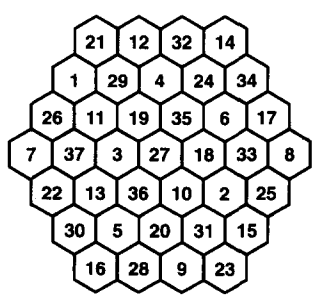


Fig. 13. Solution for $N = 37$, $L = 3$, RD $= 8$. Iteration $= 79$.

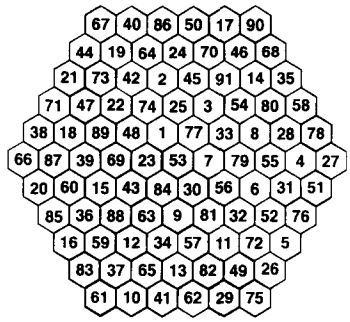


Fig. 14. Solution for $N = 91$, $L = 5$, RD $= 20$. Iteration $= 208$.

$$= -\sum_i^N \sum_j^N \left(\frac{dU_{ij}}{dt}\right)^2 \frac{dV_{ij}}{dU_{ij}} \quad \text{where } \frac{\partial E}{\partial V_{ij}} \text{ is replaced by } -\frac{dU_{ij}}{dt}$$

$$\leq 0 \qquad \text{where } \frac{dV_{ij}}{dU_{ij}} \geq 0 \,. \qquad \square$$

Theorem 2. *The system always satisfies* $\dfrac{\Delta E}{\Delta t} \leq 0$ *under two conditions:*

i) $\dfrac{\Delta U_{ij}}{\Delta t} = -\dfrac{\Delta E}{\Delta V_{ij}}$, *and*

ii) $V_{ij} = f(U_{ij})$,

where E is the computational Lyapunov energy function and $f(U_{ij})$ is a binary hysteresis McCulloch-Pitts function:

$$f(U_{ij}) = \begin{cases} 1\,, & \text{if } U_{ij} > \text{UTP} \\ 0\,, & \text{if } U_{ij} < \text{LTP} \\ \textit{unchanged}\,, & \textit{otherwise} \end{cases} .$$

Proof. Consider the derivatives of the computational energy E with respect to time t.

$$\frac{\Delta E}{\Delta t} = \sum_i^N \sum_j^N \frac{\Delta V_{ij}}{\Delta t} \frac{\Delta E}{\Delta V_{ij}} =$$

$$= \sum_i^N \sum_j^N \frac{\Delta V_{ij}}{\Delta t}\left(-\frac{\Delta U_{ij}}{\Delta t}\right) \qquad \text{where } \frac{\Delta E}{\Delta V_{ij}} \text{ is replaced by } -\frac{\Delta U_{ij}}{\Delta t}$$

$$= -\sum_i^N \sum_j^N \left(\frac{\Delta U_{ij}}{\Delta t}\frac{\Delta V_{ij}}{\Delta U_{ij}}\right)\left(\frac{\Delta U_{ij}}{\Delta t}\right) =$$

$$= -\sum_i^N \sum_j^N \left(\frac{\Delta V_{ij}}{\Delta U_{ij}}\right)\left(\frac{\Delta U_{ij}}{\Delta t}\right)^2 .$$

Let $\frac{\Delta U_{ij}}{\Delta t}$ be $\frac{U_{ij}(t+\Delta t)-U_{ij}(t)}{\Delta t}$. Let $\frac{\Delta V_{ij}}{\Delta U_{ij}}$ be $\frac{V_{ij}(t+\Delta t)-V_{ij}(t)}{U_{ij}(t+\Delta t)-U_{ij}(t)}$. It is necessary and sufficient to consider the following four regions:

Region (1). $U_{ij}(t) >$ UTP and $V_{ij}(t) = 1$
Region (2). LTP $\leq U_{ij}(t) \leq$ UTP and $V_{ij}(t) = 1$
Region (3). LTP $\leq U_{ij}(t) \leq$ UTP and $V_{ij}(t) = 0$
Region (4). $U_{ij}(t) <$ LTP and $V_{ij}(t) = 0$.

In region (1), we must consider the four possible cases for $U_{ij}(t+\Delta t)$:

Case (1). $U_{ij}(t+\Delta t) > U_{ij}(t)$
Case (2). LTP $\leq U_{ij}(t+\Delta t) < U_{ij}(t)$
Case (3). $U_{ij}(t+\Delta t) <$ LTP $< U_{ij}(t)$
Case (4). $U_{ij}(t+\Delta t) = U_{ij}(t)$.

In cases (1) and (2), $V_{ij}(t+\Delta t) = V_{ij}(t) = 1 \Rightarrow \frac{\Delta V_{ij}}{\Delta U_{ij}} = 0$. Therefore, $\frac{\Delta E}{\Delta t} = 0$.

In case (4), $\frac{\Delta U_{ij}}{\Delta t} = 0 \Rightarrow \frac{\Delta E}{\Delta t} = 0$.

In case (3), $V_{ij}(t+\Delta t) = 0 \Rightarrow \frac{\Delta V_{ij}}{\Delta U_{ij}} = \frac{0-1}{\text{negative number}} > 0$ and $\frac{\Delta U_{ij}}{\Delta t} < 0$. Therefore, $\frac{\Delta E}{\Delta t} < 0$.

It is concluded that $\frac{\Delta E}{\Delta t} \leq 0$ is always satisfied in region (1). Similarly, in regions (2), (3), and (4), $\frac{\Delta E}{\Delta t} \leq 0$ is always satisfied. □

References

[1] G.P. Jellis, ed., The Unfriendly Beehive Problem, *Games and Puzzles J.* (1971).

[2] D. Pritchard and D. Francis, 1 to 19 – Puzzle #75, *Puzzles for Geniuses: II* (1984) 75.

[3] F. Rubin, The Unfriendely Beehive Problem, J. Recreational Mathemat. 13 (3) (1982) 219.

[4] W.S. McCulloch and W. Pitts, A logical calculus of the ideas imminent in nervous activity, *Bull. Mathemat. Biophys.* 5 (1943) 115–133.

[5] J.J. Hopfield and D.W. Tank, Neural computation of decisions in optimization problems, *Biol. Cybernet.* 52 (1985) 141–152.

[6] Y. Takefuji and K.C. Lee, Artificial neural networks for four-coloring map problems and K-colorability problems, *IEEE Trans. Circuits Syst.* 38 (3) (Mar. 1991) 326–333.

[7] Y.-P.S. Foo, Y. Takefuji and H. Szu, Binary neurons with analog communication links for solving large-scale optimization problems, *Proc. Internat. Neural Network Soc. Meeting* (Sept. 1988).

[8] Y. Takefuji and K.C. Lee, A near-optimum parallel planarization algorithm, *Science* (245) (Sept. 1989) 1221–1223.

[9] Y. Takefuji and K.C. Lee, A parallel algorithm for tiling problems, *IEEE Trans. Neural Networks* 1 (1) (Mar. 1990) 143–145.

[10] Y. Takefuji, C.W. Lin and K.C. Lee, A parallel algorithm for estimating the secondary structure in Ribonucleic Acids, *Biol. Cybernet.* 63 (5) (1990) 337–340.

[11] Y. Takefuji, L.L. Chen, K.C. Lee and J. Huffman, Parallel algorithms for finding a near-maximum independent set of a circle graph, *IEEE Trans. Neural Networks* 1 (3) (Sep. 1990) 263–267.

[12] Y. Takefuji and K.C. Lee, An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviors of neural dynamics, *Biol. Cybernet.* 64 (5) (1991).

[13] Y. Takefuji and K.C. Lee, A super parallel sorting algorithm based on neural networks, *IEEE Trans. Circuit Syst.* 37 (11) (Nov. 1990) 1425–1429.

[14] N. Funabiki and Y. Takefuji, A parallel algorithm for allocation of spare cells on memory chips, *IEEE Trans. Reliability* 40 (3) (Aug. 1991) 338–346.

[15] R.P. Lipmann, An introduction to computing with neural nets, *IEEE ASSP Mag.* (Apr. 1987) 4–22.

[16] D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vols. I & II* (MIT, Cambridge, MA, 1986).

[17] J.J. Hopfield and D.W. Tank, Computing with neural circuits: A model, *Science* 233 (Aug. 1986) 625–633.

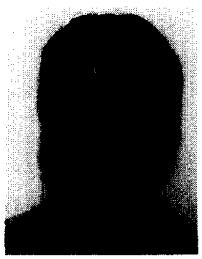
[18] J.J. Hopfield and D.W. Tank, 'Neural' computation of decisions in optimization problems, *Biol. Cybernet.* 52 (1985) 141–152.

[19] C. Peterson and J.R. Anderson, Neural networks and NP-complete optimization problems; A performance study on the graph bisection problem, submitted to *Complex Syst.*

[20] Y. Takefuji, K.C. Lee and Y.B. Cho, Comments on $O(n^2)$ algorithms for graph planarizations, *IEEE Trans. Comput. Aided Design* 10 (12) (Dec. 1991) 1582–1583.

[21] K.C. Lee, N. Funabiki and Y. Takefuji, A parallel improvement algorithm for the bipartite subgraph problem, *IEEE Trans. Neural Networks* 3 (1) (Jan. 1992) 139–145.

[22] N. Funabiki and Y. Takefuji, A parallel algorithm for solving the 'Hip' games, *Neurocomput.* 3 (2) (1991) 97–106.



Mr. Rofkar is currently an undergraduate student of computer science at Case Western Reserve University working toward his Master's degree in computer science. He is also working for State Chemical Mfg., Co., Cleveland, OH (USA) as a systems operator; and also as a consultant in the areas of computer-aided design and graphical user interfaces as they pertain to the field of commercial art. His research interests include resource allocation problems, Ramsey theory, the visual aspects of artificial reality, and puzzles. He is a prospective member of the IEEE Computer Society, and ACM.

Yoshiyasu Takefuji is a faculty member of Electrical Engineering at Case Western Reserve University. Before joining Case in 1988, he taught at the University of South Florida and the University of South Carolina. He received his BS (1978), MS (1980), and Ph.D. (1983) from Electrical Engineering from Keio University (Japan). His research interests focus on neural network parallel computing for solving real-world problems. He is interested in VLSI applications and silicon architecture. He received the National Science Foundation/Research Initiation Award in 1989 and is an NSF advisory panelist. A member of the IEEE Computer Society, ACM, International Neural Network Society, and American Association for the Advancement of Science, he received the Information Processing Society of Japan's best paper award in 1980. He coauthored two books, *Digital Circuits* (Ohm-Sha Publishers) in 1984 and *Neural Network Computing* (Baifukan Publishers) in 1991. He was an Editor of the *Journal of Neural Network Computing* and is an associate editor of the *IEEE Transactions on Neural Networks and Neurocomputing*, and a guest editor of the *Journal of Analog Integrated Circuits and Signal Processing* on VLSI neural network issue. He has published more than 80 papers and has authored the book *Neural Network Parallel Computing* from Kluwer Publishers (Jan. 1992).