NEURAL NETWORK MODELLING OF OPTIMAL ROBOT MOVEMENT USING BRANCH AND BOUND TREE

S. Khanmohammadi

Department of Electrical Engineering
Tabriz University
Tabriz, Iran

Abstract In this paper a discrete competitive neural network is introduced to calculate the optimal robot arm movements for processing a considered commitment of tasks, using the branch and bound methodology. A special method based on the branch and bound methodoloy, modified with a travelling path for adapting in the neural network, is introduced. The main neural network of the system consists of different subnets, each of which is designed for a special propose. The common neuron for competitive layers and the state presentation neurons for different layers are also presented and used in the design of neural network architecture. Sigma PI neurons are used to increase the calculation's performance. A case study with different commitments of tasks is simulated and the results are compared with Hopfield and Tank net from different view points.

Key Words Branch and Bound, Sequence Path, Compatative NNT, Cyclic Competition, Sigma PI Neurons

چکیده در این مقاله برای محاسبه حرکت بهینه بازوی روبوت جهت انجام عملیات بر روی ترکیبهای مورد نظر هر قطعه کار، یک شبکه عصبی گسسته رقابتی معرفی می شود. روش خاصی از محاسبات شاخه و حد با معرفی درخت همراه با پیاده کردن این روش بر روی شبکه های عصبی ارائه می گردد. شبکه عصبی اصلی شامل زیرشبکه های مختلف است که هر کدام برای منظور خاصی طراحی شده اند. سلول مشترک برای لایه های رقابتی و نیز سلولهای نمایش حالت برای لایه های مختلف معرفی شده و در طراحی سیستم بکار می روند. برای بالا بردن کارایی سیستم از سلولهای سیگما پی Sigma Pi استفاده می شود. در یک بررسی نوعی از ترکیبهای مختلف قطعه کارها مدلسازی شده و نتایج حاصله از دیدگاههای مختلف با شبکه هوپفیلد و تانک مقایسه می شود.

INTRODUCTION

Artificial neural networks take their names from the networks of nerve cells in the human brain and have structures similar to biological systems [1]. Recently the neural networks are finding applications in various fields including adaptive pattern recognition, adaptive signal processing, adaptive dynamic modelling, adaptive control, expert systems, and process control [2]. One of the other specific applications of neural networks is the robot arm movement. Igor Aleksander and Helen Morton have suggested an architecture of three major parts for the system of robot arm movement [3] as follows:

a sensory neural level,
a cognitive neural level, and
a conventional computing level.

In this paper the computational feature of the neural network is exploited as the computing level to calculate the optimal robot arm movement for processing different commitments of tasks. This problem is an NP complete problem settled in Travelling Salesman Problem (TSP) category.

Hopfield and Tank have used the neural

network to solve the TSP, considering the energy equation and using the heuristic methodology to decrease the energy of the system at each step [4]. Wilson and Pawley have reported that the performance of Hopfield network is poor and it gets poorer as the problem gets larger. It does not solve larger problems reliably, however in such a case the network often comes up with a solution that contains segments that are locally optimal [5].

This paper presents a neural network to calculate the optimal sequencing of robot arm movements using the branch and bound methodology. The robot processes different commitments of ***m*** tasks that are chosen from a set of ***n*** types ($m \subseteq n$). The simulation results are compared with those of Hopfield and Tank net from different view points. The calculation time through the present network is less than that of the Hopfield and Tank method and it preserves its relative speed even for $m > 10$. Beyond this value, however, both methods become slow.

STATE OF THE SYSTEM

A commitment of ***m*** tasks from ***n*** types is presented to an intelligent robot for processing at a particular time. For convenience, the tasks are of different types without loosing the generality. The objective is the calculation of the optimal sequence of robot arm movements to minimize the total operation time. Clearly this is an NP complete TSP where the robot arm movements are considered as the travelling salesman and the tasks are playing the role of different cities in TSP.

The objective function to be minimized is:

$$T = \sum_{k=1}^{m} p_k + \sum_{k=0}^{m-1} d_{k(k+1)} + d_{mo} \qquad (1)$$

where T is the total operation time of robot on ***m*** tasks. $p_k = p_i$, if task i is considered in the k^{th} position of the sequence and $d_{k(k+1)} = d_{ij}$ if the tasks i and j are considered in the k^{th} and $(k+1)^{th}$ positions of the sequence respectively. p_i is the operation time on task i and d_{ij} is found from the following formula:

$$d_{ij} = max\ \{r_{ij}\ ,\ v_{ij}\} \qquad (2)$$

where r_{ij} is the preparation time of robot for operation on taks j after task i, and v_{ij} is the movement time of the robot arm from the position of task i to the position of taks j.

It must be noted that during the movement of the robot arm, all, or some parts of the preparation proceduce to the new task takes place. It depends on which one (movement or preparation) takes more time. In the last formula, d_{oj} is the movement (or preparation) time from the start position of the robot arm to the position of task j (if it is considered in the first position of the sequence). In the same way the d_{jo} is the movement time of the robot arm from the position of the last task to the start position. Therefore, for three tasks and for the sequence 4, 2, 5; the objective function will be written as:

$$T = p_4 + p_2 + p_5 + d_{04} + d_{42} + d_{25} + d_{50} \qquad (3)$$

Noting that the operation times pis are fixed, and they don't affect the optimal conditions, the objective function may be reduced to:

$$T = \sum_{k=0}^{m-1} d_{k(k+1)} + d_{mo} \qquad (4)$$

In the literature of optimization, the minimization objective function is called the cost function. In this paper the optimal movement duration of the robot arm is called the cost function to respect this routine.

BRANCH AND BOUND TREE AND SEQUENCING PATH

In this section, a branch and bound tree and the adjacent travelling path to find different possible sequences or subsequences of robot arm movements are introduced. Figure 1 shows the branch and bound tree and its adjacent travelling path for sequencing of 4 movements. This tree consists of a root node and the sequencing nodes divided at different levels, where level i denotes the i^{th} position in the sequence. Therefore, the node 4 in the 2nd branch from the top of the 3rd level in Figure 1 means that the movement of the arm for task 4 is considered in the 3rd position of the sequence in that branch. The travelling path, shown with the directed line, starts from the root node and returns back to it. Each node becomes active (considered in the sequence) passing from over it and is inactivated (removed from the sequence) passing from its underside. When the path is travelled completely, all of $m!$ permutations for m movements are obtained [6].

In the branch and bound methodology, after activation of each node the cost function (the movement duration in the case of this paper) is calculated for the subsequences generated by the active nodes. If the calculated (current) cost exceeds the last one obtained, the node is bypassed and inactivated (removed from the sequence). In the branch and bound tree of Figure 1, when the continued line is travelled, the sequence 1, 2, 4 is obtained. If the cost function of this subsequence exceeds the last one obtained and saved as an optimal (best) cost, node 4 will be bypassed. Continuing this procedure, the subsequences (1, 2), (1), (1, 3), (1, 3, 2), ... will be obtained respectively.

Using a set of m layers of m binary cells, the whole branch and bound tree can be represented by the layered $m \times m$ cells. Each layer represents the corresponding level in the branch and bound tree. Forward and backward movements through the layers will be the projections of the travelling in the left to right and right to left directions of the branch and bound tree respectively. Activation of the i^{th} cell in the j^{th} layer corresponds to the activation of node i in the j^{th} level of branch and bound tree. This mapping is used in the design of the neural network for branch and bound calculations. To put the concept in an appropriate perspective, first a general foundation of neural network along with some special proposed neurons and activation rules, is in the next section.

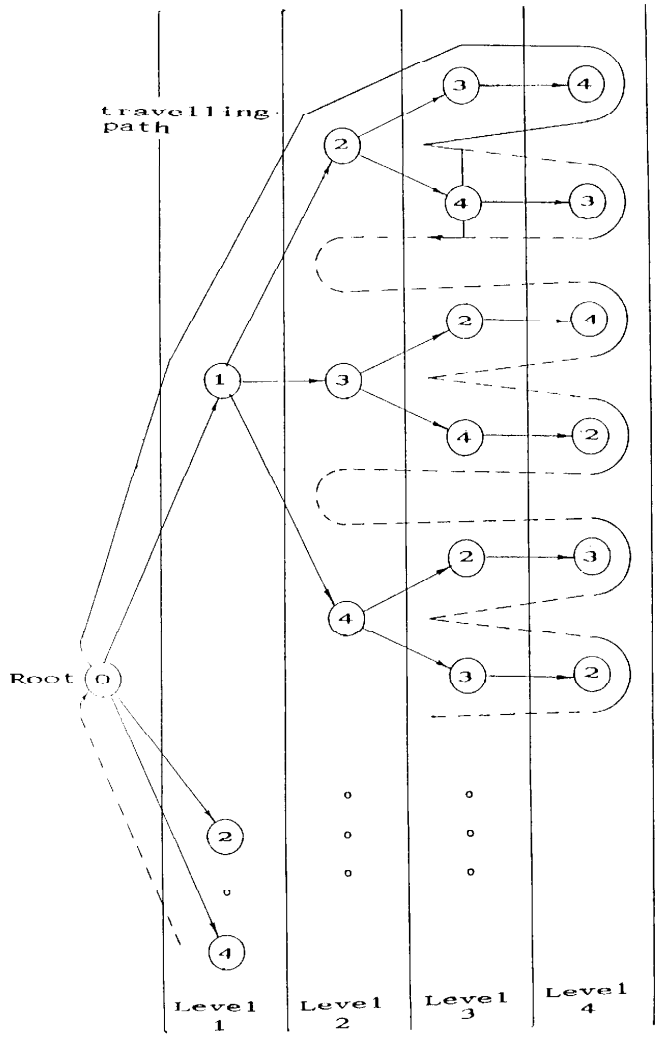

Figure 1. Branch and bound tree for 4 movements and its adjacent travelling path to find different sequences. Level i represents the i^{th} position in the sequence. The nodes of the last level are leaves. Reaching a leave means that all the movements are taken into account.

NEURAL NETWORK FOUNDATIONS

Processing elements (PEs), also referred to as nodes, short term memories (STM), neurons, populations, or threshold logic units, which perform simple computations, are the basic components of artificial neural networks [7]. Figure 2 displays the anatomy of a generic processing unit, which will be called simply "a unit" in this paper, and form an input vector $A = (a_1, ..., a_i, ..., a_n)$, where a_i is the activity level (output) of the unit i. Associated with each connected pair of units is a fixed or adjustable value called a weight (also referred to as a long term memory). The collection of weights that abuts the unit j forms a vector $W_j = (w_{1j}, ..., w_{ij}, ..., w_{nj})$, where the weight w_{ij} represents the connection strength from the unit a_i to the unit b_j. An additional parameter θ_j modulated by the weight w_{tj}, which is commonly taken to be unity, is considered for some units as internal threshold. The weights W_j, their associated unit values A, and the possible extra parameter θ_j, are used to compute the output value b_j. This computation is typically taking the dot product of A and W_j, subtracting the threshold to find the net value absorbed by the unit and passing it through an activation (threshold) function $f(\,)$. Mathematically these procedures are defined as

$$net_j = \Sigma a_i w_{ij} - w_{tj}\theta \qquad (5)$$

or more precisely,

$$net_j = \Sigma a_i w_{ij} - \theta \qquad (6\text{-a})$$

and

$$b_j = f(net_j) \qquad (7)$$

where the net_j is the net value absorbed by the unit j. The threshold values will be considered as θ instead of $w_{tj}\theta$ from now. The negative threshold value is commonly called the bias value and in this case the net_j will be found from the following equation

$$net_j = \Sigma a_i w_{ij} + \theta \qquad (6\text{-b})$$

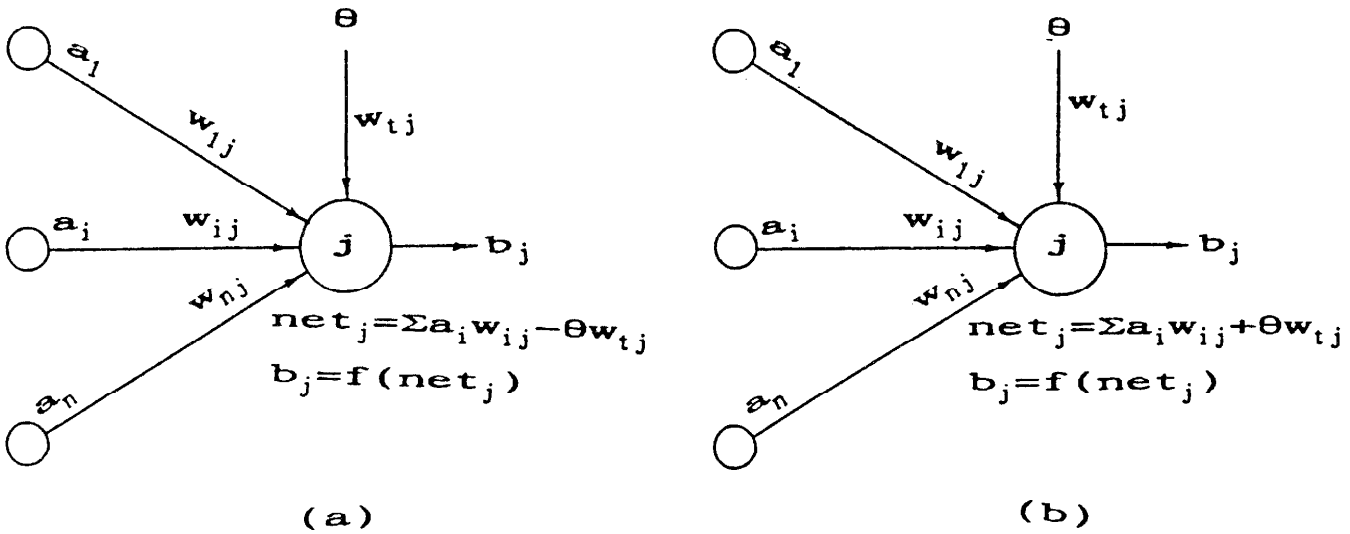

Figure 2. Anatomy of the generic unit j. Input units from the vector $A = (a_1, ..., a_n)$. The threshold is θ_j. The connection strengths are represented by the weight values w_{ij} and the threshold is being assigned to the weight w_{ij}. **(a)** θ_j is considered as threshold value. **(b)** θ_j is considered as bias value.

Sigma PI Units

In the anatomy of Figure 2, a simple additive unit is assumed where the net value absorbed by the unit is given by (6). This is certainly the most common form used in most of the models. Sometimes however, multiplicative connections are used where the output values of two (or more) units are multiplied before entering into the sum. Such a multiplicative connection allows one unit to gate another. Thus, if one unit of the multiplicative pair is zero, the other unit of the gate can have no effect. The Sigma Pi (SP) units are suggested by Crick and Asanuma for this type of connections [8]. The net value of a SP unit is

$$net_j = \Sigma w_{ij} \Pi a_{i1} a_{i2} \ldots a_{ik} \qquad (8)$$

where a_{i1}, a_{i2},..., a_{ik} are the outputs of k conjunct units connected to the i^{th} gate of the SP unit. If the SP unit contains only one gate, the net value will be simply the weighted product of the outputs of the conjunct units

$$net_j = w_{ij} \Pi a_{i1} a_{i2} \ldots a_{ik} \qquad (9)$$

and it will be called the Pi unit. The anatomy of SP and Pi units are shown in Figure 3.

Activation Rules

Activation functions, also referred to as threshold functions, map the net values of units to their output values. Four common activation functions are the linear, ramp, step and sigmoid (S shaped) functions [7].

The linear function has the equation

$$f(x) = \alpha x \qquad (10)$$

where α is a real-valued constant that regulates the magnification of the unit's activity x.

When the linear function is bounded to the range $[-\gamma,+\gamma]$, (10) becomes the nonlinear ramp function described by

$$f(x) = \begin{cases} +\gamma & \text{if } x > \gamma \\ x & \text{if } x < \gamma \\ -\gamma & \text{if } x < -\gamma \end{cases} \qquad (11)$$

where $\gamma(-\gamma)$ is the maximum (minimum) output value of the unit, which is commonly referred to as the saturation level.

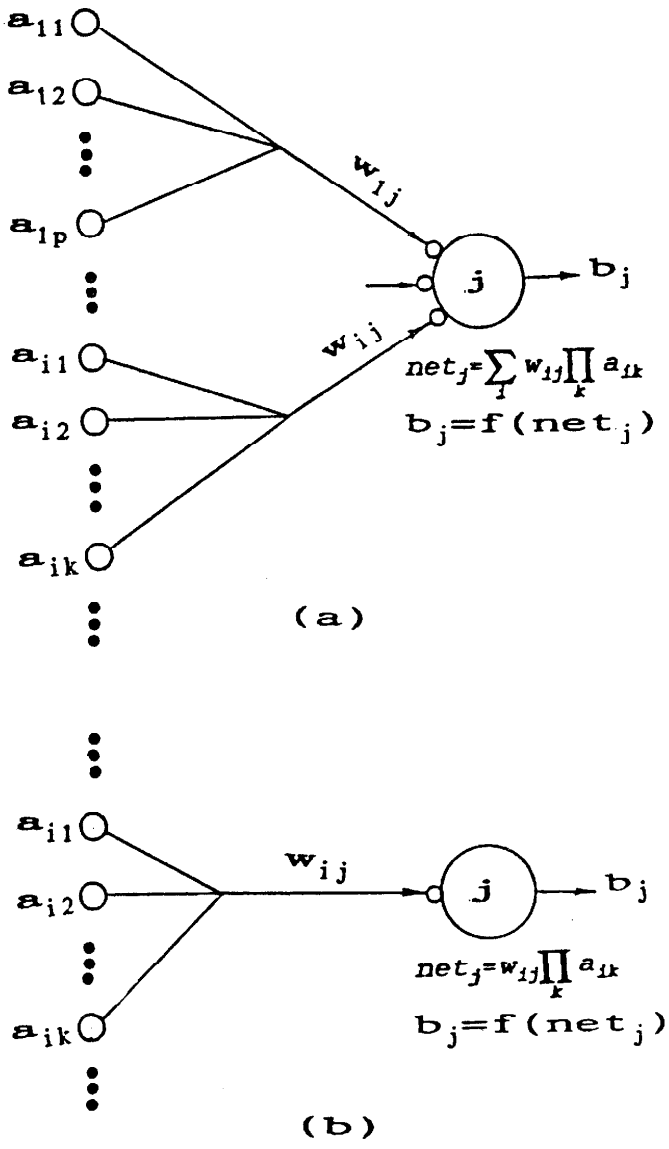

Figure 3. The Anatomy of Sigma Pi and Pi units. (**a**) Sigma Pi unit, (**b**) Pi unit

If the activation function responds only to the sign of the input, emitting $+\gamma$ if the x is positive and $-\delta$ if it is not, then the activation function is called a step function, where γ and δ are positive scalars. The step activation function is mathematically characterized by

$$f(x)=\begin{cases}+\gamma & \text{if } x>0\\ -\delta & \text{otherwise}\end{cases} \qquad (12)$$

Equation 12 is often binary, emitting 1 if $x > 0$, and 0 if otherwise.

The sigmoid function is a bounded monotonic, non-decreasing function that provides a graded, nonlinear response. A common sigmoid function is the logistic funciton

$$f(x) = (1+e^{-x})^{-1} \qquad (13)$$

which has its saturation levels at 0 and 1. Two other sigmoid functions are the hyperbolic tangent

$$f(x) = \tanh(x) \qquad (14)$$

with the saturation levels at -1 and 1; and the augmented ratio of squares

$$f(x)=\begin{cases}x^2/(1+x^2) & \text{if } x>0\\ 0 & \text{otherwise}\end{cases} \qquad (15)$$

In this paper the linear and the binary step functions are used as activation functions.

Competition Rules

A competitive set is a set of units which act through competition and inhibition in opposition to one another. The activation rule tor the unit j of a competitive set is the following modification of the activation rule using the binary step activation function:

$$b_j=\begin{cases}1 & \text{if } net_j>0 \text{ and } net_j = max\{net_i\}\\ 0 & \text{otherwise}\end{cases} \qquad (16)$$

This equation denotes that unit j of the competitive set will be active (it wins the competition) if its net value is positive and has a miximum value among all other units of the set. Commonly each unit has an inhibitive connection (normally with negative unity weight) to all other units and an exciting recurrent connection (normally with unity weight) as considered in this paper and shown in Figure 4. Under the same conditions (equal net values) the topper unit, with the least index, is considered as the winner.

Cyclic Competition Rule

This is another version of competition procedure used by Rumelhart and Zipser [9], and Grosberg [10] for pattern classification, manipulated for branch and bound calculations and other applications.

According to this rule, if the winner becomes inactive instantaneously (shocked) by an external unit, the state of the set will change to the competition state, Figure 4(a), and the competition will take place between the units with positive net values excluding the last winner. By this way after imposing a sequence of shocks to the net, the winner units of Figure 4 will be i, k, and n, Figure 4(b), respectively. Regarding that units 1 and n are neighbors, if the net value of the first unit was 2 ($net_1 = 2$), it would be the next winner after imposing a new shock.

NEURAL NETWORK ARCHITECTURE AND DYNAMICS OF BRANCH AND BOUND CALCULATIONS

The neural network of branch and bound sequencing is a discrete network consisting of a hierarchical collection of four main subnets called: Sequencing, layer recognition, cost calculation, and processing control subnets. It also contains two special proposed units named common unit and optimal cost memory. Figure 5 shows the schematic representation of this architecture. The doubled lines show the communication links between different subnets.

As it is shown in Figure 5, the input layer (I) and layer recognition subnet (R) have single array representations while the sequencing subnet (S) is presented in a grid form as a set of different layers for convenience. The common unit (P) is also included in this subnet. The cost calculation subnet

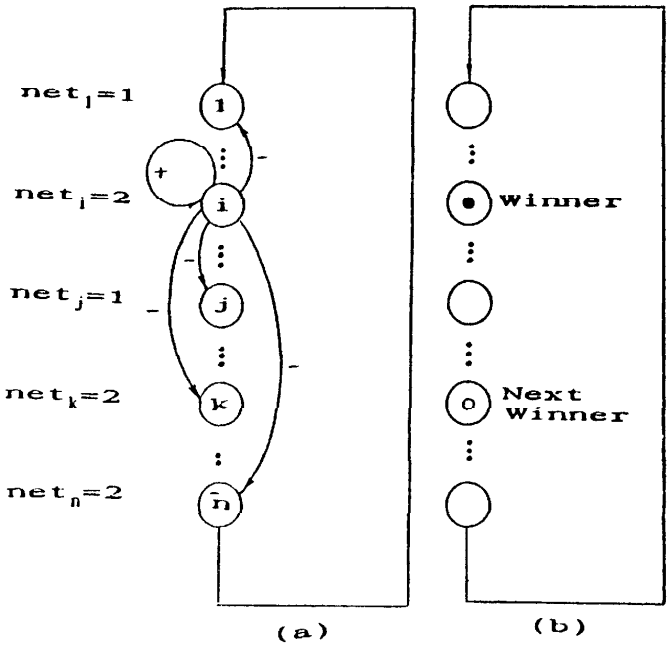

Figure 4. A set of competitive units with corresponding net values. Each unit has a negative unity inhibitive connection to other units and a unity exciting recurrent connection. Units 1 and *n* are neighbors. **(a)** competition takes place between the units with positive net values. **(b)** unit *i* wins and becomes active (its output value is set to 1). The cycle line shows the direction of cyclic competition, which implies that after the unit *i*, units *k* and *n* will be the winners.

consists of a grid of data table units (T), and a single unit called current cost unit (C). The process control subnet (PCS) includes the optimal cost memory (B), counter (U), comparison (M), backward (F) and the end of optimization procedure (E) units. In this scheme the common and optimal cost units are specially proposed units.

The generalized dynamics of the system may be described as follows: a commitment of ***m*** tasks is presented to the system by the input pattern. At each iteration, a new sequence (subsequence) of robot arm movements is found by activation of appropriate unit in each layer of the sequencing grid. If the cost of sequence (subsequence) is less than the last cost saved as the optimal (best) one, and the number of layers containing a winner is less than ***m*** (a leaf of branch and bound tree is not reached), the forward process will continue. Otherwise, the backward process is performed.

In the following subsections, the dynamics of different subnets, the activation rules of various units, their interconnections in the mentioned subnets and their usage are presented in more details, where the following symbols are used to clarify the descriptions.

I_i unit *i* of input layer

i_i output of I_i

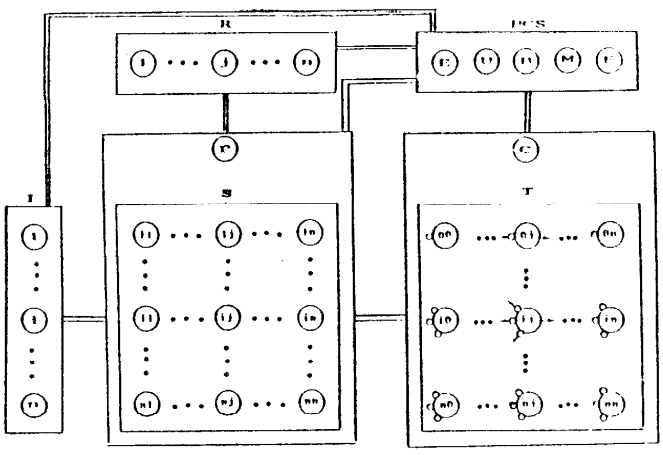

Figure 5. The schematic representation of the branch and bound sequencing neural system

R_i	unit *i* of layer recognition subnet
r_i	output of R_i
S_{ij}	unit in the i^{th} row and j^{th} column of sequencing grid
s_{ij}	output of S_{ij}
T_{ij}	unit in i^{th} row and j^{th} column of data table grid
t_{ij}	output of T_{ij}
P	common (public!) unit
p	output of P
C	current cost unit
c	output of C
B	optimal (best) cost's memory.
b	output of B
F	backward unit
f	output of F
U	counter unit
u	output of U
M	comparison unit
m	output of M
E	end of optimization (movement signal) unit
e	output of E

The activation functions of all the units, except the current cost and optimal cost units, which follow the linear activation functions, are binary step functions.

Appropriate numbers of indexes are used for weights (one for single arrays and two for grids).

Characteristic letters of different subnets are used in parentheses as subscripts of weights to clarify the origins and targets of the connection. For example, $w_{(s)\,ij\,,\,(R)\,i}$ means the weight of the connection link from the unit S_{ij} of the sequencing grid to the unit R_i of the layer recognition subnet, $w_{(T)\,ij\,,\,(c)}$ represents the weight of the unit T_{ij} of the data table grid to the current cost unit and so on.

Finally, when a set of weights are considered to follow a special constraint, such as (18), they are assumed to have the same values.

Input Layer

The units of this layer are connected to all appropriate units in the sequencing grid such that each I_i is connected to the appropriate units S_{ij}. Figure 6 shows these connections.

Sequencing Grid

All the units of this grid have a threshold value of θ and are classified into different layers. Considering the following connections,

$$i_i w_{(r)i,(s)ij} = \theta, \quad j \neq 1 \tag{17}$$

$$i_i w_{(r)i,(s)i1} > \theta, \quad j = 1 \tag{18}$$

according to the cyclic competition rule, after representing a new commitment to the input layer, the competition takes place only between the units corresponding to the input pattern in the first layer (with positive net values) and one of them becomes active (fires) as the winner.

In this grid, each unit of layer is fully connected to all the units of the next layer, and has inhibitive

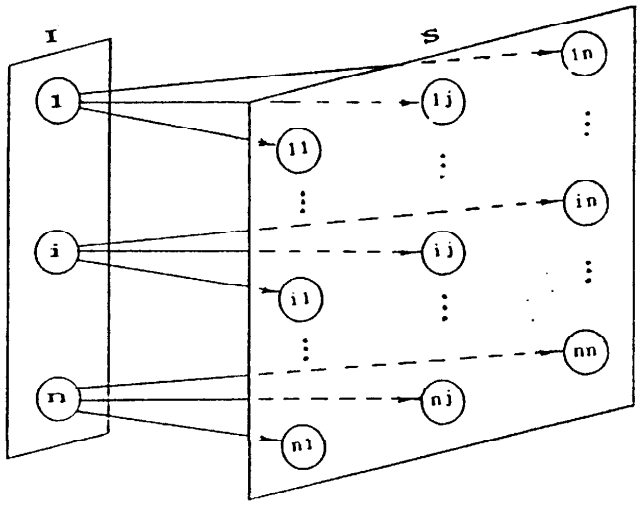

Figure 6. Connections between the units of input layer and the sequencing grid. Each unit I_i is connected to the appropriate units S_{ij}.

connection to all the units of the same row and the same layer and a recurrent exciting connection. Figure 7 shows the general architecture of this grid.

By choosing appropriate weights satisfying the following constraint

$$s_{ij}w_{(s)ij,(s)hk} + i_n w_{(r)h,(s)hk} > \theta, \qquad (19)$$

for $i = 1, \ldots, n\text{-}1$, $j = 2, \ldots, n$,
$h = 1, \ldots, n$, $h \neq i$, $k = j + 1$

after activation of the winner unit of each layer, the competition takes place only between the units of the next layer corresponding to the input pattern (commitment). Figure 8 represents an illustrative excample of the active units for the sequence 3, 8, 1, 5, 4 in a commitment of m=5 tasks.

The layer Recognition Subnet

This subnet is a set of n units where its j^{th} unit corresponds to the j^{th} layer of the sequencing grid with exciting and inhibitive connections as shown in Figure 9;

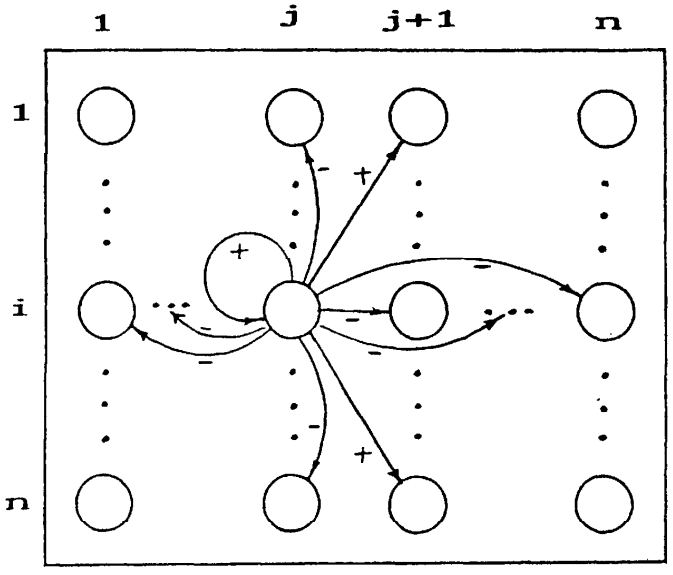

Figure 7. The units of the sequencing grid are classified into different layers. Each unit S_{ij} has a threshold value θ. It is fully connected to all the units of the next layer, has an inhibitive connection to all the units in the same row and the same layer and an exciting recurrent connection.

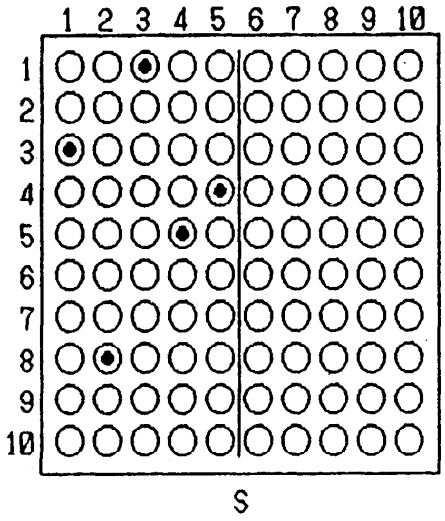

Figure 8. Active units marked by tokens in circles, represent the sequence 3, 8, 1, 5, 4 of a commitment of 5 tasks.

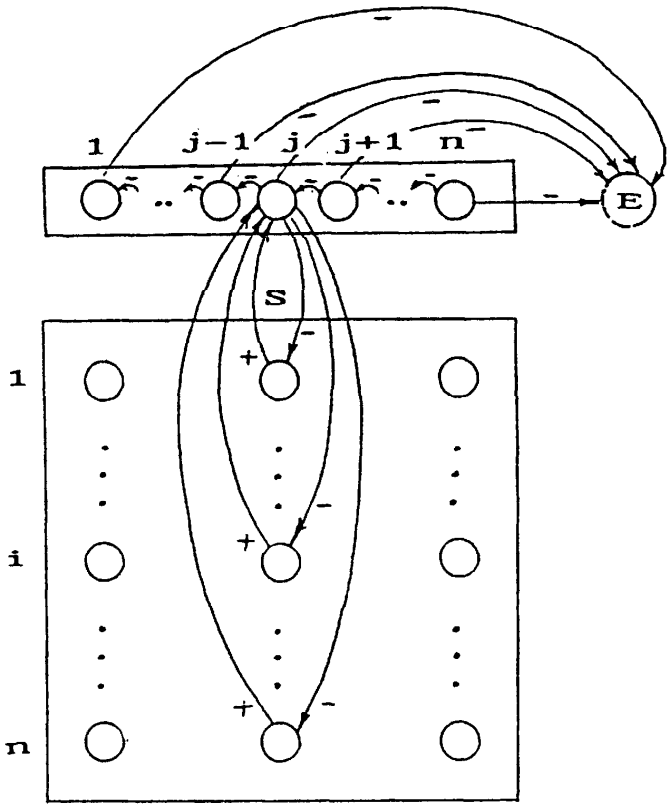

Figure 9. Schematic representation of the layer recognition subnet and its connections. Each unit j of this subnet receives and exciting connection from all the units of the j^{th} layer in the sequencing grid and has an inhibitive connection to the previous unit of the subnet and to all units of the j^{th} layer in the sequencing grid.

where

$$w_{(R)j,(R)i} = w_{(S)hi,(R)i} \qquad (20)$$

for $i, h = 1, \ldots, n, \quad j = i + 1$

According to these connections, when one of the units of a layer (say layer j) in sequencing grid becomes active (wins), the corresponding unit of the layer recognition subnet becomes active as well. It inactivates the preceding unit of the subnet while decreasing the net value of the units of the j^{th} layer. This procedure will be useful in the backward procedure. The schematic representation of this subnet is shown in Figure 9.

Common Unit

This unit is common between all the layers of sequencing grid. It participates in the competition with the units of each layer, which is in the competition state (its units are competing). This unit has a bias value of $\delta > 0$ such that

$$s_{ij} w_{(s)ij,(s)hk} + i_h w_{(r)h,(s)hk} - \theta > \delta, \qquad (21)$$

for $i, j, h = 1, \ldots, n, \quad h \neq i, \quad k = j + 1$

This condition forces the common unit to be the last one that wins in the competition of a layer. After inactivation of this unit, the next cycle of competition will start from the top of the layer in the next shock imposed to its units. Considering that the j^{th} layer was in the competition state, activation of this unit means that it was not possible to consider a new task (arm movement) in the j^{th} position of the sequence (there was not a real winner). This unit has a bilateral connection to the backward unit with the following weights

$$w_{(P),(F)} = \text{a great value} \qquad (22)$$

$$w_{(F),(P)} < -\delta \qquad (23)$$

According to (22) and (23) the common unit activates only the backward unit and becomes inactive instantly by the activation of the backward unit, which again inactivates the backward unit in turn. This mutual activation and inactivation between the two units causes the generation of a backward impluse (shock) by the backward unit. When the common unit becomes the winner of the first layer of the sequencing grid, the optimization procedure terminates.

Cost Calculation Subnet

This subnet consists of an (n+1) × (n+1) grid of the data table units and the current cost unit as shown in Figure 10.

All the units of the data table grid are Pi type with the following connections to the current cost unit

$$w_{(T)ij,(c)} = d_{ij} \qquad (24)$$

for $i, j = 0, \ldots, n$

consequently,

$$net_{(c)} = \Sigma t_{ij} \, w_{(T)ij,(c)} \qquad (25)$$

or considering (24) and binary step activation rule for T_{ijs},

$$net_{(c)} = \Sigma d_{ij} \qquad (26)$$

where the summation is done for the active units of the data table grid. On the other hand, all of the i^{th} and j^{th} units of two adjacent layers of the

sequencing grid, say q^{th} and $(q+1)^{th}$ layers respectively, are connected to one of the gates of the unit T_{ij} in this grid. Each unit j of the first column in the sequencing grid is connected to the single gate of the unit T_{0j} in this grid. Each unit R_i of the layer recognition unit is also connected to one of the gates of T_{i0} in conjunction with one of the S_{ijs}. In this way, considering as an example the active units of the sequencing grid in Figure 8, the active units of the data table grid will be $t_{03} = t_{38} = t_{81} = t_{15} = t_{54} = t_{40} = 1$, and $net_c = d_{03} + d_{38} + d_{81} + d_{15} + d_{54} + d_{40}$ as shown in Figure 11. The current cost unit uses the linear rule for activation, that is

$$c = net_{(c)} \tag{27}$$

where $net_{(c)}$ and c represent the net and output values of the current cost unit, respectively.

Process Control Subnet

The purpose of this subnet is to control forward and backward processing of the system and save the updated optimal cost. This subnets includes the optimal cost memory, counter, comparison, backward and end of optimization (movement signal) units, as have been mentioned previously. The connections of these units are shown in Figure 12 and will be discussed below.

Counter unit counts the number of tasks in the

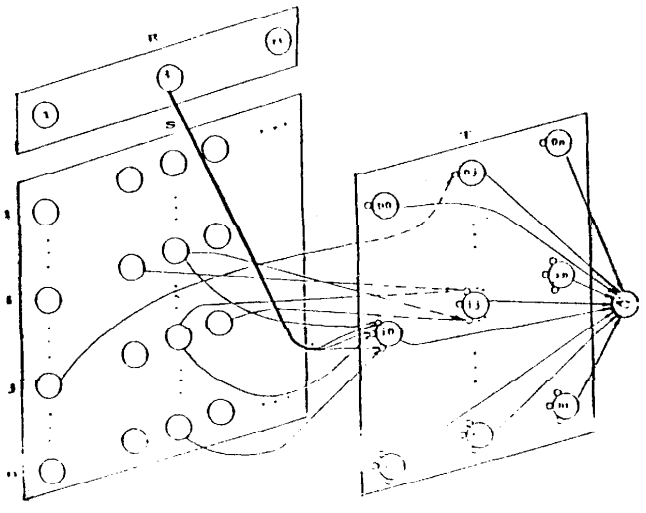

Figure 10. Schematic representation of the cost calculations subnet containing the data table grid and current cost unit. The units of data table grid are Sigma Pi type with conjunct inputs to different gates. The units i and j of two successive layers are connected to different gates of T_{ij}. S_{ij} units are connected to T_{ojs} while R_j and S_{ij} units are connected to the different gates of T_{io}.

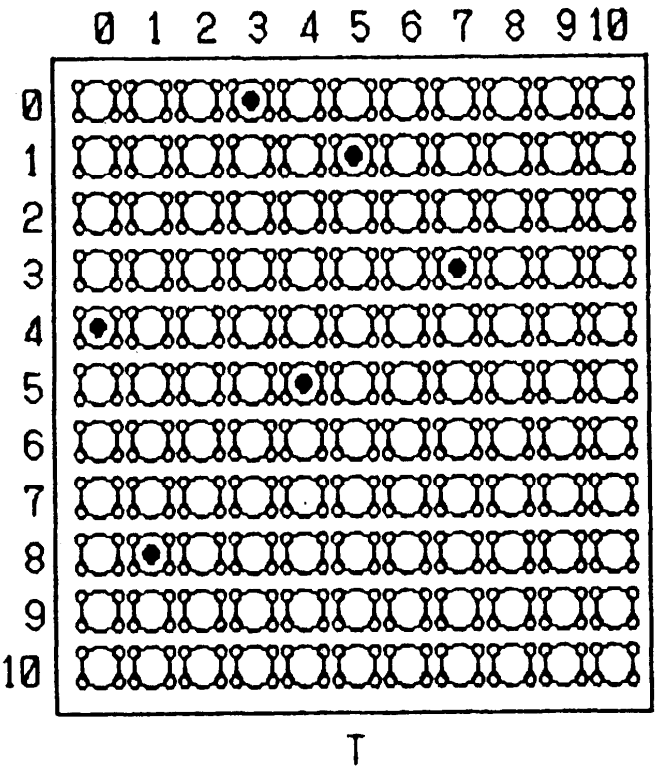

Figure 11. Active units of data table grid (marker by tokens) corresponding to the active units of sequencing grid in Figure 8.

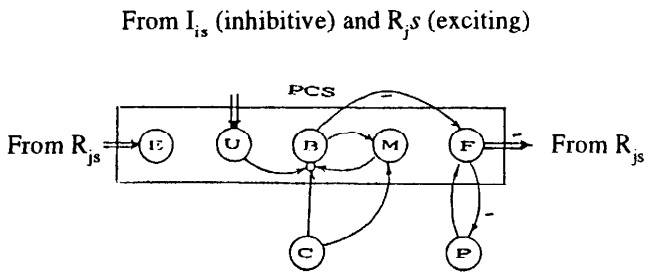

Figure 12. Process control subnet consisting of the optimal cost memory (B), counter (U), comparison (M), backward (F) and end of optimization (E) units.

commitment and limits the sequencing grid processing to the first m layers for m tasks. It has a threshold value of c (a very small positive value) and becomes active if all the tasks are considered in the sequence. That is, if the commitment consists of m tasks, then $net_{(u)}$ becomes positive if each of the first m layers has an active unit (winner). Activation of this unit causes updating of the optimal cost (if it is modified) and performing the backward processing.

Comparison unit compares the current cost and last optimal (best) cost, saved in the optimal cost unit. It becomes active if the current cost is less than the saved cost.

Optimal cost memory is a collection of cells to save the optimal cost. Takeda and Goodman [11], proposed several schemes for developing any real number on to the unit state space and Takashi Nakatsuji and Terutoshi Kaku [12], have introduced an adaptation of the group-and-weight scheme for data handling. This paper does not deal with the methodology of handling the neural nets as a memory place. The interested readers are referred to the mentioned references. Hear it is considered that the last cost value is saved in an optimal cost unit as its net value and can be retrieved from its output; That is:

$$b = net_{(B)} \tag{28}$$

where $net_{(B)}$ and b are the net and output values of the optimal cost unit, respectively. The $net_{(B)}$ is calculated following the PI rule that is:

$$net_{(B)} = w_{(C.U.M),(B)} \cdot c.\ u.\ m \tag{29}$$

where the $w_{(C,\ U,\ M)\ ,\ (B)}$ denotes the weight of the conjunct units C, U and M entering the single gate of the optimal cost memory. Considering the unity weight for this connection (29) may be written more precisely as:

$$net_{(B)} = c.\ u.\ m \tag{30}$$

Equation 30 denotes that if counter and comparison units are active (all the tasks are taken into account and the current cost is less than the saved on) then the current cost is assigned to the net_b. The net_b remains unchanged if the right hand side of (30) is zero.

The backward unit follows the binary step activation function with the bias

$$\theta_f = \varepsilon \tag{31}$$

where θ_f is the bias of the backward unit and ε is a very small positive value. The net value of this unit is computed from the following equation

$$net_{(F)} = pw_{(P),\ (F)} + cw_{(B)\ ,\ (F)} - bw_{(B)\ ,\ (F)} + \theta_f \tag{32}$$

which means that this unit becomes active if the common unit is active or current cost is greater than or equal to the optimal cost (saved in the optimal cost memory). Evidently, when all the tasks are considered in the sequence, the optimal cost is updated as mentioned earlier, and the backward unit becomes active immediately.

This unit has inhibitive connections to all the units of the sequencing grid such that

$$i_i\ w_{(r)\ i,\ (g)\ hk} + s_{ij}\ w_{(s)\ ij,\ (s)\ hk} + f w_{(F)\ ,\ (s)\ hk} > 0 \tag{33}$$

and

$$i_i w_{(r)\ i,\ (s)\ hk} + s_{ij}\ w_{(s)\ ij,\ (s)hk} + f w_{(F)\ .\ (s)\ hk} + I_k\ w_{(R)k,(s)hk} < 0 \tag{34}$$

Considering (33) and (34), activation of this unit causes the net values of the last layer in the

competition state, which is recognized by the layer recognition subnet, to become negative, leading to inactivation of the winner. The remaining net values remain positive and their winners remain active. If the last winner is the common unit, according to its mutual activation and inactivation with the backward unit (as mentioned in subsection 4) the backward unit becomes inactive after producing a backward shock. If the last winner is a S_{ij} unit, its inactivation causes the inactivation of the corresponding T_{ij} and consequently subtracting d_{ij} from the current cost. This procedure causes the net_f to become negative and the backward unit to be inactivated. Therefore in any way the backward unit generates a backward shock to the active layer of the sequencing grid (the last layer, which was in the competition state) to continue the cyclic competition.

Inactivation of the last winner in a layer j, causes the layer recognition unit of the preceding layer to become active instantly, and reasoning so until the activation of a new winner in layer j. If the last winner was the common unit, the cyclic competition will continue in the preceding layer. The end of optimization unit has a bias value of ε, that is, when all of its input values becom zero, then $net_{(E)} = \varepsilon > 0$. All the R_{js} have an inhibitive connection to the end of optimization unit; therefore, if at least one of the R_{js} is active (system is in the optimization state), this unit remains inactive and it becomes active if all the R_{js} become inactive (end of the optimization procedure).

SIMULATION RESULTS

The designed system is simulated for different commitments of tasks. The sets of ten types of tasks are considered as the main collections, out of which five different commitments of 4, 5, 9 and 10 tasks are simulated in turn. The results are compared with those of Hopfield and Tank method. Figure 13 shows the results.

All the values are normalized and in each box the maximum value is set to unity. The dark bars represent the values of the branch and bound method where the clear bars depict the Hopfield and Tank method. The bars of (a) boxes compare the total number of iterations to find all the possible permutations considering that in the Hopfield method $\boldsymbol{m}!$ random sequences are generated for the commitment of $\boldsymbol{m}$ tasks. The (b) boxes compare the iteration number in which the optimal cost (usually the best cost in the case of Hopfield and Tank method) is obtained. The (c) boxes show the optimal cost obtained using each of the methods. The (d) boxes show the cost value in the Hopfield and Tank method when (that is the same iteration) the optimal cost is obtained by the branch and bound net.

The critical condition for the branch and bound net arises when the optimal sequencing is the descending order obtained in the last branch (ex. sequence 4, 3, 2, 1 for four tasks). The two methods are simulated also for this condition and the results are shown in Figure 14. As it is seen in the Figure, in this situation, the Hopfield and Tank method is better for a few number of tasks, but for more tasks the branch and bound network responds more appropriately.

In the case of a real time problem, when the calculations may stop before considering all the $\boldsymbol{m}!$ permutations, the results obtained by the branch and bound method are preferable. When all the $\boldsymbol{m}!$ permutations are considered, the problem of reliability arises in the case of Hopfield and Tank net where in the branch and bound net all the permutations are considered in a relatively short calculation time.

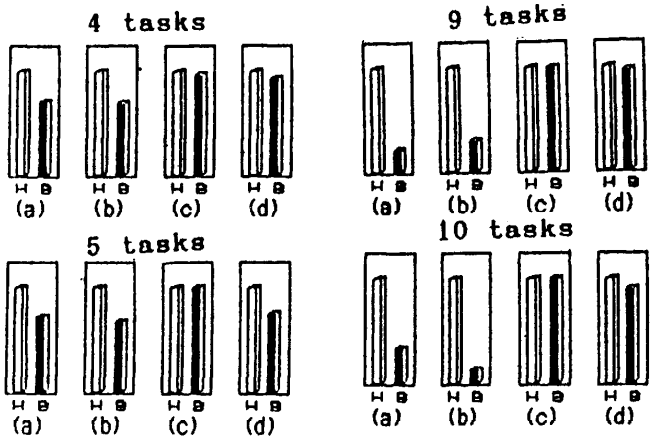

Figure 13. Simulation results for different commitments of 4, 5, 9 and 10 tasks. The clear bars in the boxes represent the values of Hopfield and Tank net while the dark bars are used for the representation of branch and bound net. The values are normalized such that the bigger value is set to unity. **(a)** boxes show the total number of iterations (changing the state of the system) to find the optimal sequence. **(b)** boxes compare the iteration in which the optimal (best) solution is obtained. **(c)** boxes show the optimal cost function obtained by each method. **(d)** boxes compare the cost function of the two methods when the optimal value is found by the branch and bound net. The results obtained to commitments of 6, 7 and 8 tasks (not shown in this figure) are similar to the bars of 9 and 10 tasks.

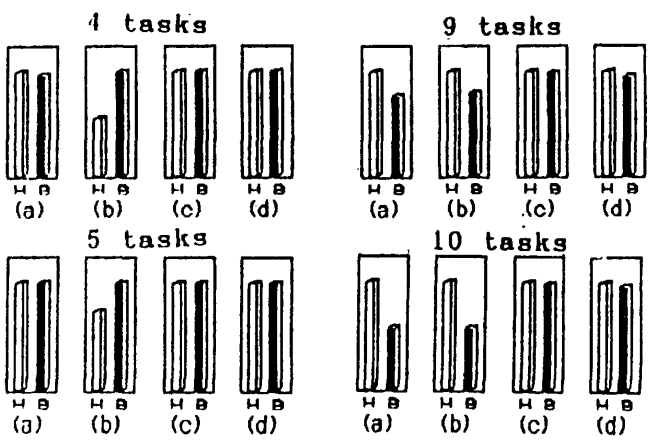

Figure 14. The same results as Figure 13, when the optimal sequencing is in descending order and the branch and bound net operates in critical conditions. The results obtained for the commitments of 6, 7 and 8 tasks (not shown in this figure) are similar to the bars of 9 and 10 tasks.

CONCLUSION

An artificial neural network for calculation of optimal sequencing of robot arm movement is introduced while exploiting the computational feature of the neural networks. This problem is in TSP category, which is classified as an NP complete problem. some special proposed neurons, cyclic competition and the shock phenomena are introduced in the architecture of this network. A case study with different commitments of tasks is simulated. The results show that the system is relatively reliable and the calculation time is satisfying. The proposed network may be a good suggestion for further studies of neural network utilization in branch and bound calculations.

REFERENCES

1. Judith E. Dayhoff, "Neural Network Architecture", Van Nostrand Reinbold, (1990).
2. Erol Belenbe, "Neural Networks Advanced and Applications", Elsevier Science Publishers B. V., (1991).
3. Igore Aleksander and Helen Marton, "An Introduction to Neural Computing", Chapman and Hall, (1990).
4. J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems": *Biol. Cybernetics.* 52, (1985).
5. G. V. Wilson and G. S. Pawely, "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank": *Biol. Cybernetics.* 58, (1988).
6. S. Khanmohammadi and R. Kenarangui, "Petri-Net Design of Real-Time Programming of Expert Manufacturing Systems", in proceedings of ICARCV' 90 Int. Conf. on Automation, Robotics and Computer Vision, (1990).
7. Patrick K. Simpson, "Artificial Neural Systems Foundations, Paradigm, Applications and Implementations", Pergamon press Inc., (1990).
8. F. Crick and C. Asanuma, "Certain Aspects of the Anatomy and Physiology of the Cerebral Cortex in Parallel Distributed Processing", D. E. Rumelhart and J. L. McClelland, (Eds.), MIT Press, 1986 (1988).
9. D. E. Rumelhart and D. Zipser, "Feature Discovery

by Competitive Learning": *Cognitive Science*, 11, (1987).

10. S. Grossberg, "Competitive Learning: From Interactive Activation to Adaptive Resonance": *Cognitive science*, 11, (1987).
11. M. Takeda and J. W. Goodman, "Neural Network for Computation, Number Representations and Programming Complexity": *Appl. Optics*, 25, 18, (1986).
12. Nakatsuji Takashi and Kaku Terutoshi, "Application of Neural Network Model to Traffic Engineering Problem Self-Organizing Approach to Traffic Management System", Elsevier Science Publishing Co. Inc. Transportation and Traffic Theory, M. Koshi (Ed.), (1990).