OPTIMIZATION OF $N + 1$ QUEENS PROBLEM USING DISCRETE NEURAL NETWORK

M. Waqas[*], *A.A. Bhatti* [*]

Abstract: Combinatorial optimization problems are extensively solved by using neural networks. Hopfield-Tank model is used to solve Traveling Salesman problem and many NP-Hard problems. This paper describes a neural network optimizer/scheduler that optimizes a solution for a highly complicated version of N Queens problem (NQP), i.e. $N+1$ non-threatening Queens on a $N \times N$ chessboard with an intermediate pawn on it. Both synchronous and asynchronous methods of updating of the neurons have been applied for optimization of $N+1$ Queens problem. Computer simulations are used to confirm the results. The proposed neural network is attracted to optimized solution or finds the global minima in 90% of the trials. A new rule of initialization, i.e. *the proximity rule of initialization* has been proposed. Using the proximity rule of initialization the performance of the system is enhanced and the system converges to an optimal solution in much less time. Many novel applications like multiprocessor job scheduling, resource optimization, of the above mentioned algorithm have been proposed. N Queens problem has been solved by many techniques but no other algorithm exists to solve $N+1$ QP in the literature. Consequently, the performance of the network is compared with full space search algorithm.

Key words: *neural networks optimization problem, Hopfielf-Tank model, NP-Hard problem, $N+1$ Queens problem, the proximity rule of initialization, multiprocessor job scheduling*

Received: May 24, 2014 **DOI:** 10.14311/NNW.2017.27.016
Revised and accepted: July 7, 2017

1. Introduction

Hopfield and Tank proposed a neural network that seemed to be very promising in solving optimization problems. The network is quite handy in solving Combinatorial Optimization problems (COPs) that are non-polynomial in nature and are called NP problems. The model was used to solve Traveling salesman problem [3–5]. Bizzari investigated in detail the convergence of neural network [1]. Afterwards many dedicated neural networks were designed to optimize different real life problems and the approach proved to be a success. Mańdziuk [7, 8] investigated and proposed a dedicated neural network in order to solve a classical

[*]Muhammad Waqas – Corresponding author; Abdul Aziz Bhatti; School of Systems and Technology, University of Management and Technology, Lahore, Pakistan, E-mail: engr.waqas.sandhu@gmail.com; drabhatti@umt.edu.pk

and conceptually simple N Queens problem. Isao Tanka and Yoshifumi Nishio [10] proposed Chaos Neural Network for N Queen problem. The proposed network is attracted, in 90 % of the trials, to the correct solution.

This paper presents a novel neural network to find solution to $(N+1)$ QP which is a very complex version of NQP. N Queens problem is a classical COP and NP in nature. So it is a natural candidate to be solved by neural network. The quest to find an efficient solution for $N+1$ QP is very valuable as it is analogous to many scheduling problems. Some of the scheduling problems have been proposed in [6, 9, 11, 12], where the resources were optimized.

The NQP and $(N+1)$ QP have many applications for real world problems. The analogous applications can be found in job/shop scheduling, data routing, deadlock or blockage prevention, efficient resource management in computer systems, task assignment in multiprocessors, digital image processing and parallel memory storage schemes.

NQP is addressed by many approaches ranging from backtracking search, Genetic algorithms, Particle Swarm Optimization, Ant Colony Optimization, Simulated Annealing and Neural Networks, but no such techniques exist for $N+1$ QP.

NQP is actually a scheduling problem, where N Queens are to be placed on a chess board of dimensions $N \times N$ such that no two queens attack each other. As Queen can move along any row, column and diagonal therefore the constraints of the problem are the following:

- Only one Queen in a row,
- Only one Queen in a column,
- Only one Queen in a diagonal,
- Total number of Queens $= n$.

As the Hopfield model proposed a neural network with following Energy function:

$$E = -1/2 \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} T_{ij} V_i V_j, \tag{1}$$

where T_{ij} is the weight of connection from the j-th neuron to i-th neuron and V_i and V_j are the output of i-th and j-th neuron respectively.

The input of i-th can be given as

$$u_i = -\frac{\partial E}{\partial u_i}; (i = 1, 2, 3, \ldots, n). \tag{2}$$

By taking the derivative of E we shall get

$$u_i = \sum_{j=1}^{n} T_{ij} V_j; (i = 1, 2, 3, \ldots, n)\,. \tag{3}$$

The activation function for neurons is chosen as

$$V_i = 1/2\left[1 + \tanh\left(\alpha u\right)\right]. \tag{4}$$

For a very high value of α Eq. (4) will reduce to

$$\begin{aligned} V_i &= 1 \text{ for } u_i > 0, \\ V_j &= 0 \text{ otherwise.} \end{aligned}$$

The energy function for the NQP is defined by Jacek Mandiziuk [2] as

$$\begin{aligned} E = \frac{1}{2} A \sum_{i=1}^{n} \sum_{j=1}^{n} & \left[\left(\sum_{\substack{k=1 \\ k\neq j}}^{n} v_{ik} \right) v_{ij} + \left(\sum_{\substack{k=1 \\ k\neq i}}^{n} v_{kj} \right) v_{ij} \right] \\ + B \sum_{i=2}^{n} \sum_{j=1}^{i-1} & \left[\left(\sum_{\substack{k=i-j+1 \\ k\neq j}}^{n} v_{k,k-i+j} \right) v_{ij} \right] \\ + B \sum_{i=1}^{n} \sum_{j=1}^{n} & \left[\left(\sum_{\substack{k=1 \\ k\neq i}}^{n+i-j} v_{k,k-i+j} \right) v_{ij} \right] \\ + B \sum_{i=1}^{n} \sum_{j=n-i+1}^{n} & \left[\left(\sum_{\substack{k=i+j-n \\ k\neq i}}^{n} v_{k,i+j-k} \right) v_{ij} \right] \\ + B \sum_{i=1}^{n-1} \sum_{j=1}^{n-i} & \left[\left(\sum_{\substack{k=1 \\ k\neq i}}^{i+j-1} v_{k,i+j-k} \right) v_{ij} \right] \\ + C & \left(\sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij} - (n+\sigma) \right)^2, \end{aligned} \tag{5}$$

where A, B, C and σ are positive constraints.

According to Eq. 2 we have

$$-u_{ij} = A \left(\sum_{\substack{k=1 \\ k\neq i}}^{n} v_{ij} + \sum_{\substack{k=1 \\ k\neq i}}^{n} v_{kj} \right) + R_{ij} + S_{ij} + C \left(\sum_{k=1}^{n} \sum_{l=1}^{n} v_{kl} - (n+\sigma) \right), \tag{6}$$

where

$$R_{ij} = \left(\begin{array}{c} B \sum\limits_{\substack{k=j+1 \\ k\neq i}}^{n} v_{k,k-i+j}; \text{ if } i-j > 0 \\ B \sum\limits_{\substack{k=1 \\ k\neq i}}^{n+i-j} v_{k,k-i+j} \text{ if } i-j \leq 0 \end{array} \right), \tag{7}$$

$$S_{ij} = \begin{pmatrix} B\sum\limits_{\substack{k=i+j-n\\k\neq i}}^{n} v_{k,i+j-k};\ \text{if } i+j>n \\ B\sum\limits_{\substack{k=1\\k\neq i}}^{i+j-1} v_{k,i+j-k};\ \text{if } i+j\leq n \end{pmatrix}, \tag{8}$$

where $ij = 1, 2, 3, \ldots, n$.

2. $(N + 1)$ Queens with an intercepting pawn

2.1 Problem statement

The $N + 1$ Queens with an intercepting pawn on the chess board of $N \times N$ dimensions is a highly complex version of NQP problem. The algorithm that solves $N + 1$ Queens problem with intercepting Pawn or $(N + 1)$ QP has to search the feasible solution out of 2^{64} states possible solution for $N = 8$.

For the sake of completeness the $(N+1)$ QP has been defined as follows: "Place $N+1$ Queens and a Pawn on a chessboard of $N \times N$ dimensions such that no Queen threatens the other."

In other words the constraints of the problem are as follows:

1. If there is no pawn in a row, column or diagonal of the cell then there must be only one Queen in that corresponding row, column or diagonal respectively;
2. If there is a pawn in a row, column or a diagonal then
   a. There can be only one Queen in that corresponding row, column or diagonal respectively or
   b. There can be only two Queens, one on each side of the pawn in that corresponding row, column or diagonal;
3. Total number of Queens $= N + 1$.

A possible solution to the problem has been shown in the Fig. 1.

Problem representation A discrete Hopfield-type neural network of $N \times N$ dimensions has been used for $(N + 1)$ QP such that each neuron represents cell on the chessboard. A Queen is represented by 1, a pawn is represented by -1 and empty cell is denoted by 0. As no solution exist if we place the pawn at the boundary cells. For $N = 8$ there are 28 boundary cells out of 64. So a pawn has to be placed on non-boundary cells. A pawn is placed permanently near the center at coordinates $Pawn(p, q)$ and the system is allowed to converge. However, the pawn can also be placed at any other non-boundary cell.

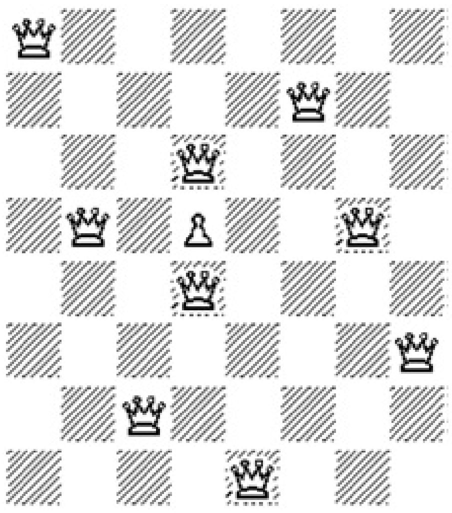

Fig. 1 *Solution of* $(N+1)$ *QP.*

Derivation of Energy function. The Energy function for the proposed neural network for $(N+1)$ QP has following constraints

$$E=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\begin{bmatrix} V_{pq}=-1 \\ \text{if } i\neq p \text{ and } j\neq q \\ \text{Sum of Row Terms } + \\ \text{Sum of Column Terms } + \\ \text{Sum of Diagonal 1 Terms } + \\ \text{Sum of Diagonal 2 Terms} \\ \text{if } i=p \text{ or } j=q \\ \{\text{Sum of Row Terms upto } (p,q) \\ \text{Sum of Column Terms upto } (p,q)\} \\ \text{if } i-j=0 \text{ or } i+j=0 \\ \text{Sum of Diagonal 1 Terms upto } (p,q) \\ \text{Sum of Diagonal 2 Terms upto } (p,q) \\ \text{for all } i=p \text{ and } j=q \\ \text{Total number of Queens } = n+1 \end{bmatrix} V_{ij}. \tag{9}$$

The detailed energy function for the problem is given as

$$E = \frac{1}{2}A\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\begin{array}{ccc}
\sum\limits_{\substack{k=1\\k\neq j\\i\neq p,j\neq q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=1\\k\neq i\\i\neq p;j\neq q}}^{n} V_{kj}+\\
\sum\limits_{\substack{k=1\\k\neq j\\i=p;j<q}}^{q-1} V_{pk} & + & \sum\limits_{\substack{k=1\\k\neq i\\i=p;j<q}}^{n} V_{kj}+\\
\sum\limits_{\substack{k=q+1\\k\neq j\\i=p;j>q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=1\\k\neq i\\i=p;j>q}}^{n} V_{kq}+\\
\sum\limits_{\substack{k=1\\k\neq j\\i<p;j=q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=1\\k\neq i\\i<p;j=q}}^{p-1} V_{kq}+\\
\sum\limits_{\substack{k=1\\k\neq j\\i>p;j=q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=p+1\\k\neq i\\i>p;j=q}}^{n} V_{kq}
\end{array}\right] V_{ij}$$

$$+\frac{1}{2}B\sum_{i=2}^{n}\sum_{j=1}^{i-1}\left[\begin{array}{c}
\sum\limits_{\substack{k=i-j+1\\k\neq i\\i-j\neq p-q}}^{n} V_{k,k-i+j}+\\
\sum\limits_{\substack{k=i-j+1\\k\neq i\\i-j=p-q\\i<p}}^{p-1} V_{k,k-i+j}+\\
\sum\limits_{\substack{k=p+1\\i-j=p-q\\i>p}}^{n} V_{k,k-i+j}
\end{array}\right] V_{ij}$$

$$+\frac{1}{2}B\sum_{i=2}^{n}\sum_{j=1}^{i-1}\left[\begin{array}{c}
\sum\limits_{\substack{k=i-j+1\\k\neq i\\i-j\neq p-q}}^{n} V_{k,k-i+j}\\
+\sum\limits_{\substack{k=i-j+1\\k\neq i\\i-j=p-q\\i<p}}^{p-1} V_{k,k-i+j}\\
+\sum\limits_{\substack{k=p+1\\i-j=p-q\\i>p}}^{n} V_{k,k-i+j}
\end{array}\right] V_{ij}$$

$$+\frac{1}{2}B\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\begin{array}{l} \displaystyle\sum_{\substack{k=1\\ k\neq i\\ i-j\neq p-q}}^{n+i-j} V_{k,k-i+j} \\ \displaystyle +\sum_{\substack{k=1\\ k\neq i\\ i-j=p-q\\ i<p}}^{p-1} V_{k,k-i+j} \\ \displaystyle +\sum_{\substack{k=p+1\\ k\neq i\\ i-j=p-q\\ i>p}}^{n+i-j} V_{k,k-i+j} \end{array}\right]V_{ij}$$

$$+\frac{1}{2}B\sum_{i=1}^{n}\sum_{j=n-i+1}^{n}\left[\begin{array}{l} \displaystyle\sum_{\substack{k=i+j-n\\ k\neq i\\ i-j\neq p-q}}^{n} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=i+j-n\\ k\neq i\\ i+j=p-q\\ i>p\\ k>p}}^{n} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=i+j-n\\ k\neq i\\ i+j=p-q\\ i<p\\ k<p}}^{n} V_{k,i+j-k} \end{array}\right]V_{ij}$$

$$+\frac{1}{2}B\sum_{i=1}^{n-1}\sum_{j=1}^{n-i}\left[\begin{array}{l} \displaystyle\sum_{\substack{k=1\\ k\neq i\\ i-j\neq p-q}}^{i+j-1} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=1\\ k\neq i\\ i+j=p-q\\ i>p\\ k>p}}^{i+j-1} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=1\\ k\neq i\\ i+j=p-q\\ i<p\\ k<p}}^{i+j-1} V_{k,i+j-k} \end{array}\right]V_{ij}$$

$$+\frac{1}{2}C\left[\sum_{i=1}^{n}\sum_{\substack{j=1\\ i\neq p,j\neq q}}^{n} V_{ij}-(n+1-\sigma)\right]^{2}+D\left(V_{pq}-(-1)\right). \qquad (10)$$

The motion equation can be derived by differentiating the above equation and is given as

$$-u_{ij} = A\left(\begin{array}{ccc} \sum\limits_{\substack{k=1\\k\neq j\\i\neq p;j\neq q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=1\\k\neq i\\i\neq p;j\neq q}}^{n} V_{kj}+ \\ \sum\limits_{\substack{k=1\\k\neq j\\i=p;j<q}}^{q-1} V_{pk} & + & \sum\limits_{\substack{k=1\\k\neq i\\i=p;j<q}}^{n} V_{kj}+ \\ \sum\limits_{\substack{k=q+1\\k\neq j\\i=p;j>q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=1\\k\neq i\\i=p;j>q}}^{n} V_{kq}+ \\ \sum\limits_{\substack{k=1\\k\neq j\\i<p;j=q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=1\\k\neq i\\i<p;j=q}}^{p-1} V_{kq}+ \\ \sum\limits_{\substack{k=1\\k\neq j\\i>p;j=q}}^{n} V_{ik} & + & \sum\limits_{\substack{k=p+1\\k\neq i\\i>p;j=q}}^{n} V_{kq} \end{array}\right) + R_{ij} + S_{ij}$$

$$+ C\left(\sum_{i=1}^{n}\sum_{\substack{j=1\\i\neq p,j\neq q}}^{n} V_{ij} - (n+1-\sigma)\right) + D\left(V_{pq} - (-1)\right), \tag{11}$$

where

$$R_{ij} = \left\{\begin{array}{cc} \text{for } i-j>0 & \text{for } i-j\leq 0 \\ B\left(\begin{array}{c} \sum\limits_{\substack{k=i-j+1\\k\neq i\\i-j\neq p-q}}^{n} V_{k,k-i+j}+ \\ \sum\limits_{\substack{k=i-j+1\\k\neq i\\i-j=p-q\\i<p}}^{p-1} V_{k,k-i+j}+ \\ \sum\limits_{\substack{k=p+1\\i-j=p-q\\i>p}}^{n} V_{k,k-i+j} \end{array}\right) & B\left(\begin{array}{c} \sum\limits_{\substack{k=1\\k\neq i\\i-j\neq p-q}}^{n+i-j} V_{k,k-i+j} \\ +\sum\limits_{\substack{k=1\\k\neq i\\i-j=p-q\\i<p}}^{p-1} V_{k,k-i+j} \\ +\sum\limits_{\substack{k=p+1\\k\neq i\\i-j=p-q\\i>p}}^{n+i-j} V_{k,k-i+j} \end{array}\right) \end{array}\right., \tag{12}$$

and

$$S_{ij}=\left\{\begin{array}{cc} \text{for } i+j>n & \text{for } i+j\leq n \\ B\left(\begin{array}{l} \displaystyle\sum_{\substack{k=i+j-n\\ k\neq i\\ i-j\neq p-q}}^{n} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=i+j-n\\ k\neq i\\ i+j=p-q\\ i>p\\ k>p}}^{n} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=i+j-n\\ k\neq i\\ i+j=p-q\\ i<p\\ k<p}}^{n} V_{k,i+j-k} \end{array}\right) & B\left(\begin{array}{l} \displaystyle\sum_{\substack{k=1\\ k\neq i\\ i-j\neq p-q}}^{i+j-1} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=1\\ k\neq i\\ i+j=p-q\\ i>p\\ k>p}}^{i+j-1} V_{k,i+j-k} \\ \displaystyle +\sum_{\substack{k=1\\ k\neq i\\ i+j=p-q\\ i<p\\ k<p}}^{i+j-1} V_{k,i+j-k} \end{array}\right) \end{array}\right. . \tag{13}$$

Explaination of terms In Eq. (11), the term V_{ik} calculates the row sum and the term V_{kj} column sum for the corresponding neuron. The different cases actually correspond to the different position of neuron with respect to the intercepting pawn. There are following different cases that exits:

i. if $i \neq p$ and $j \neq q$,

   a. calculate the row and column sum in the normal manner;

ii. if $i = p$ and $j \neq q$,

   a. if $j < q$,
      calculate Row sum up to $j = q - 1$,
      calculate Column sum normally,

   b. if $j > q$,
      calculate Row sum form $j = q + 1$ to $j = n$,
      calculate Column sum normally;

iii. if $i \neq p$ and $j = q$,

   a. if $i < p$,
      calculate Column sum up to $i = p - 1$,
      calculate Row sum normally,

   b. if $i > p$,
      calculate Column sum form $j = p + 1$ to $j = n$,
      calculate Row sum normally.

The terms R_{ij} and S_{ij} in Eq. (11) correspond to the two diagonals. The terms, R_{ij} and S_{ij}, were further explained in Eqs. 12 and 13. Their expression is the logical extension of their NQP expression's counterpart. We have identified the various regions that exist in $(N+1)$ QP. The term with C calculates the number of Queens in the system and the last term makes sure that the neuron corresponding to pawn is always at potential as -1.

The activation function for the network is given as

$$V_{ij} = \begin{cases} 0; & \text{if } i \neq p \text{ and } j \neq q \text{ and } u_{ij} \leq 0, \\ 1; & \text{if } i \neq p \text{ and } j \neq q \text{ and } u_{ij} > 0, \\ -1; & \text{if } i = p \text{ and } j = q. \end{cases} \quad (14)$$

Simulation results There are two modes of updating the neurons: synchronous and asynchronous. In synchronous updating, all the neurons are updated at the same time after specific intervals of time but in asynchronous updating, a neuron is randomly selected among the network and its value is updated. Both approaches, synchronous and asynchronous modes, are implemented and the result was almost identical. The performance of the network was not dependent upon the mode selected.

Initially the neurons are randomly assigned the value from $(0, 1)$ except for $V_{pq} = -1$, which corresponds to the intercepting pawn.

There are many ways to assign the value of A, B, C and D. As all the constraints involved are equally important the values of A, B, C and D must be identical. We have set $A = B = C = D = 100$ and $\sigma = 25$ as in [8]. However we can also set $A = B = C = D = 1$ and $\sigma = 0.25, 0.5, 0.75$. On average the network finds optimal solution in 38.9 iterations for $N = 8$. Some simulation results along with the number of iterations are shown in Fig. 2 and Tab. I. The time complexity of the simulations is $O(n^4)$.

No solution exists for $N \leq 5$. The solution exists for larger values of N.

N	$K = 1$	$K = 2$	$K = 3$	$K = 4$
6	16	0	0	0
7	20	4	0	0
8	128	44	8	0
9	396	280	44	8
10	2288	1304	528	88
11	11152	12452	5976	1688
12	65172	105012	77896	30936
13	437848	977664	1052884	627916
14	3118664	9239816	13666360	11546884
15	23387448	90776620	179787988	
16	183463680	897446082		
17	1474699536			

Tab. I *Number of solutions for $N + k$ Queens problem.*

The proximity rule of initialization The Proximity Rule of Initialization states that the Neural Network should be initialized in the vicinity of some expected outcome. If we place the pawn at the boundary cells of the board then no solution exists for $N+1$ QP as the pawn will not be able to block the attack between the queens. However if we place the pawn on cells other than the boundary cells

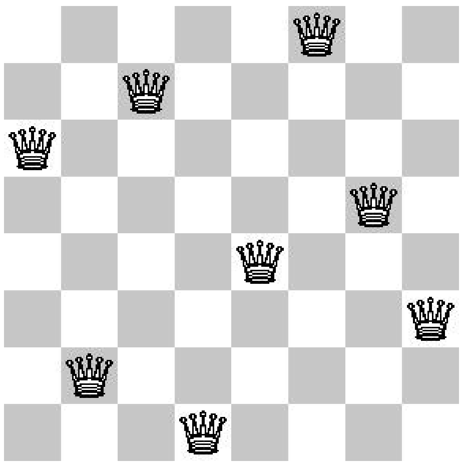

Fig. 2 *N Queens problem solution.*

the solution exists. Generally for every N there will be $4N-4$ boundary cells out of N^2 cells. Therefore for $N=8$ there will be 28 edge cells and if pawn is placed randomly then there is 43.75 % probability that the solution will fail. But if we place the pawns in the cells other than the boundary cells the system is observed to converge in 90 % of the trials. So if the Neural Network is initialized in the vicinity of the expected outcome (in this case placing the pawn on non-boundary cells) then the system converges to optimum solution in 90 % of the trials.

Comparison with full space search Mandzuik [8] estimated the computational complexity of $O(n^{4.47})$ and $O(n^{4.67})$ for asynchronous and synchronous model. However the time complexity of the proposed network $N+1$ QP is $O(n^4)$, which is an improvement upon Mandzuik model.

Chatham [2] presented number of possible solutions of $N+K$ QP for different values of N and K. The number of possible solutions to $N+k$ Queens problem for various values of N and k are given in the Tab. I [2].

The proposed network is found to be far better than full space search approach. In order to investigate full space search a different formulation is considered. A valid solution of NQP is presented in Fig. 2. A typical solution of NQP can be represented in the form of an array Q:

$$Q = [3, 7, 2, 8, 5, 1, 4, 6]. \tag{15}$$

The elements of Q represent the position of Queens in corresponding columns. For any given column there are N number of ways to place a queen. Hence for N columns there are N^N possible configurations. Similarly $N+1$ Queens problem can be represented as an array of N 3-tuples $q_i(q1, q2, p)$, where $q1, q2, p$ represent positions of pawn and queens in the i-th column. All but one of columns contain only one queen:

$$Q = [(8,0,0), (5,0,0), (2,0,0), (0,4,0), (1,7,4), (4,0,0), (6,0,0), (3,0,0)], \tag{16}$$

where first 3-tuple $(8, 0, 0)$ means that there is only one queen in first column at rown number 8. While fifth 3-tuple $(1, 7, 4)$ signify two queens at row number 1 and 7 and a pawn at 4-th row. Now in all but one of columns there will be one Queen. So there are N^{N-1} ways to place $N-1$ Queens in $N-1$ columns. On the remaining column there will be two queens and one pawn. So there are $N(N-1)(N-2)/2$ ways to place two queens and a pawn in a column. Therefore there are $[N^N(N-1)(N-2)/2]$ possible configurations. No solution of $N+1$ Queens problem for exist for $N \leq 5$ (see Tab. II). For $N = 6$ there are 446,560 possible configurations out of which only 16 yield correct solution. For $N = 8$ there are 352321536 possible configurations out of which 20 corresponds to correct solution. For $N = 10$ there are 7.2×10^{10} configurations and 2288 solutions. So the time complexity of brute force algorithm is $O(N^N)$. Clearly brute force or full space approach will not be fruitful for even the small possible value of N for $N+1$ QP. Considering the cost of evaluating the combinatons and keeping the track of past configurations, brute force approach would become too inefficient for even small values of N. Therefore the proposed algorithm is far better than full space search as ANN solves the problem in polynomial time while brute force has to search through $N^N(N-1)(N-2)/2$ configurations.

Queens(N)	Iterations	Queens(N)	Iterations
4	No.Solution	100	76.3
5	No.Solution	200	68.1
6	14.1	300	81.9
7	21.7	400	124.5
8	38.9	500	109.2
9	40.1	600	113.3
10	49.7	1000	107.4

Tab. II *Average no. of iterations for various values of N.*

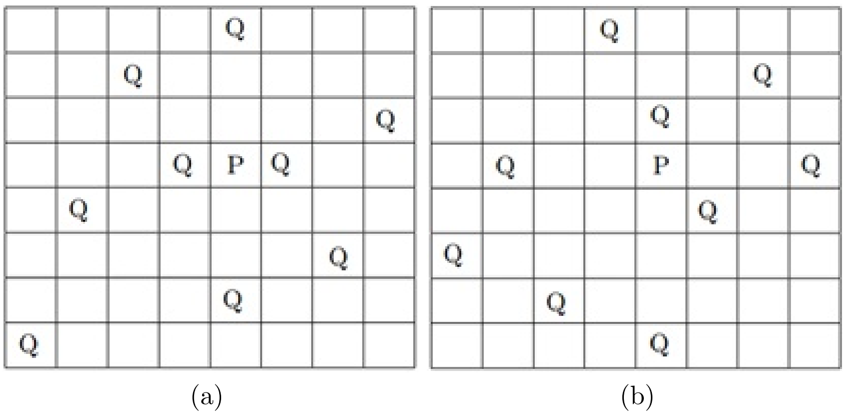

Fig. 3 *Two different solutions of $N+1$ Queens problem.*

3. Conclusion

$(N + 1)$ Queens problem is a challenging problem. As NQP is solved using many approaches ranging from backtracking search, Genetic algorithms, Particle Swarm Optimization, Ant Colony Optimization, to Simulated Annealing, but no such efficient algorithm exists to find the solution for $N + 1$ QP for all values of N. It is observed that the placement of an extra pawn introduces enormous complexity. The performance of the network is compared with full space search or brute force approach. The brute force approach has to search through $N^N\,(N{-}1)\,(N{-}2)\,/2$ configurations while the proposed ANN can solve the $N{+}1$ QP in $O(N^4)$.

A new initializing technique i.e. proximity rule of initialization has also been proposed, i.e. if we initialize the system in the proximity of the expected solution (place the pawn on non-boundary cells) then system will be attracted to the exact solution more rapidly. The results are identical for synchronous and asynchronous mode of updating neurons. The proposed network shows promising solution for large values of N. It is analogous in nature with scheduling/job shop problem and various scheduling. Therefore various job shop problems can be solved by the same methodology.

Acknowledgement

The author, Muhammad Waqas, thanks Mr. Mubshir Baigh, Assistant Professor CS Department, as without his encouragement the paper would not have been written.

References

[1] BIZZARRI A.R. Convergence properties of a modified Hopfield-Tank model. *Biological Cybernetics*. 1991, 64(4), pp. 293–300, doi: 10.1007/bf00199592.

[2] CHATHAM R.D. Reflections on the $N+k$ Queens problem. *The College Mathematics Journal*. 2009, 40(3), pp. 204–211, doi: 10.4169/193113409x469433.

[3] HOPFIELD J.J. Neural networks and physical systems with emergent collective computational abilities. In: *Proceedings of the National Academy of Sciences*. 1982, 79(8), pp. 2554–2558, doi: 10.1073/pnas.79.8.2554.

[4] HOPFIELD J.J. Neurons with graded response have collective computational properties like those of two-state neurons. In: *Proceedings of the national academy of sciences*. 1984, 81(10), pp. 3088–3092.

[5] HOPFIELD J.J., TANK D.W. "Neural" computation of decisions in optimization problems. *Biological cybernetics*. 1985, 52(3), pp. 141–152.

[6] HUANG Y.-M., CHEN R.-M. Scheduling multiprocessor job with resource and timing constraints using neural networks. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*. 1999, 29(4), pp. 490–502, doi: 10.1109/3477.775265.

[7] MANDZIUK J., MACUK B. A neural network designed to solve the N-Queens problem. *Biological Cybernetics*. 1992, 66(4), pp. 375–379, doi: 10.1007/bf00203674.

[8] MAŃDZIUK J. Solving the N-Queens problem with a binary Hopfield-type network. *Biological Cybernetics*. 1995, 72(5), pp. 439–445, doi: 10.1007/bf00201419.

[9] SIMON F.Y.-P., TAKEFUJI. Integer linear programming neural networks for job-shop scheduling. In: *IEEE International Conference on Neural Networks*, San Diego, CA. IEEE, 1988, doi: 10.1109/ICNN.1988.23946.

[10] TANAKA I., NISHIO Y., HASEGAWA M. An Approach to N-Queens Problem Using Chaos Neural Network-Challenge to Finding All Solutions of N-Queens Problem [online]. In: *International Symposium on Nonlinear Theory and its Applications*, Xi'an, PRC. 2002 [Accessed 7 July 2017]. Available from: https://www.researchgate.net/publication/265067059

[11] WILLEMS T.M., ROODA J.E. Neural networks for job-shop scheduling. *Control Engineering Practice*. 1994, 2(1), pp. 31–39, doi: 10.1016/0967-0661(94)90571-1.

[12] ZHANG C.-S., YAN P.-F., CHANG T. Solving job-shop scheduling problem with priority using neural network. In: *Proceedings of the IEEE International Joint Conference on Neural Networks*, Singapore, Singapore. IEEE, 1991, doi: 10.1109/IJCNN.1991.170586.