Neural Network Approach to Zero-One Optimal Covering Problem

A.R. Khan, A.R. Marudarajan and C.A. Goben

Department of Electrical Engineering
Southern Illinois University, Carbondale, IL 62901

ABSTRACT

This work reports the investigation of the neural network solution to the Zero-One Optimal Covering Problem via computer simulation. The key idea used in this exploration is that for every covering problem there exists an equivalent integer linear programming problem which can be solved by modifying the linear programming neural net circuit proposed by Tank and Hopfield. Simulation results indicate that this method works very well.

I. INTRODUCTION

In a sense, a zero-one optimal covering problem (ZOOC) is an allocation problem because it is concerned with allocating limited resources among competing activities in the best possible (i.e., optimal) way. Problems of this kind will arise whenever one must select the level of certain activities that compete for scarce resources necessary to perform those activities. The variety of situations to which this description applies is diverse indeed [1]. However, the one common ingredient in each of these situations is the necessity for allocating resources to activities.

The main objective of this work is to explore the potential of a neural network in handling the ZOOC. The motivating factors for this study are: 1) the broad spectrum of practical applications spanned by the chosen problem and 2) the incapability of the conventional solution techniques with respect to a large size problem. Typically the ZOOC finds its application in areas such as Boolean function minimization, state table reduction [2], VLSI testing (fault analysis) [2,4,5], and operations research [1,7]. Traditionally, a covering problem is first transformed into an equivalent zero-one integer programming problem and is then solved by using some clever search procedure such as the Branch and Bound (Balas' Additive Algorithm) [1]. However, the Branch and Bound approach has two problems. First of all, this technique does not always converge to the best solution. Secondly, it does not lend itself to a highly parallel VLSI implementation that is economical. Another conventional method is the Petrick's method. This method provides a straightforward solution to the ZOOC by making use of the Boolean identities. However, the amount of computations required by this approach limits
its use to only small problems [3].

Recently, it has been shown that a certain class of optimization problems can be solved by using a neural network [8-11]. A neural network is a densely connected network of simple processing elements or neurons [16]. Such a network is an alternative to the traditional Von Neumann computer which is essentially a serial processor. A neural network is truly a parallel processor and this paradigm is proven to be successful for solving difficult problems such as pattern recognition, speech synthesis and translation, and robotic hand-eye coordination [18-21]. The superior problem solving capability of a neural network can be primarily attributed to: 1) Speed advantage of a massively parallel system; and 2) the ability to learn without an explicit algorithm. In a typical neural network, a neuron receives its inputs from all other neurons, and then it decides its subsequent output by using some propagation rule. This output is broadcasted to all neurons in the network [21]. Since all neurons proceed in this fashion asynchronously, the entire system as a whole will quickly converge to a stable equilibrium state that represents a solution to the problem that is presented to the system.

The goal of this work is to propose a neural network solution to the ZOOCP and study the characteristics of such a solution via computer simulation. Since this investigation attempts to devise an efficient method for an optimization problem, one can employ this idea in order to solve other related problems such as scheduling, transportation and assignment problems [1,12].

II. A TUTORIAL INTRODUCTION TO THE ZOOCP

A covering problem one is provided with two sets $F = \{f_1, f_2, \ldots, f_m\}$ and $T = \{t_1, t_2, \ldots, t_n\}$ where $F$ represents the set of items to be covered by using the items of the set $T$. Usually, a specific member of the set $T$, say $t_i$, will typically cover more than one member of the set $F$. In a real situation, the set $F$ may be the set of faults in a digital circuit while the set $T$ represents a set of tests to detect these faults [2]. Similarly, the set $F$ may represent a set of microoperations while the set $T$ could be a set of maximal compatible classes [12]. The covering status of a specific item $t_i$ is indicated by the covering matrix $A$ whose size is $n \times m$. In particular, each element $(a_{ij})$ of this matrix is either 0 or 1. In this situation, when $t_i$ covers $f_j$ then $a_{ij}$ will be equal to 1; otherwise $a_{ij}$ will be equal to zero. For example, consider the $A$ matrix shown in Figure 1.

In this case $n = 6$ and $m = 4$. Also, $a_{63} = a_{64} = 1$ because $t_6$ covers $f_3$ and $f_4$. Further, $a_{22} = a_{23} = a_{24} = 0$ implies that $t_2$ cannot cover $f_2$, $f_3$ and $f_4$.

The subset $\gamma' = \{t_1, t_5, t_6\}$ is said to be a cover for the set $F$ because by using $t_1$, $t_5$ and $t_6$ all elements of $F$ can be covered. The cost associated with this cover $C(\gamma')$ is 3 units because the subset $\gamma'$ contains 3 elements ($t_1$, $t_5$, and $t_6$).
Figure 1. A 6 X 4 covering matrix.

A close inspection reveals that in order to cover \( F \) there is no need to include \( t_5 \). Rather, the subset \( Y = \{t_1, t_6\} \) is a cover for the set \( F \). The cost associated with this cover is \( C(Y) = 2 \). In fact, it can be shown that \( Y \) is indeed an optimal cover because \( C(Y') \geq C(Y') \) for all \( i \) such that \( Y' \) is a cover for the set \( F \).

This optimization problem earns the name zero-one optimal covering problem because the decision to include a specific \( t_j \) into the subset \( Y' \) is binary in nature. That is, either \( t_j \) is included to the subset \( Y' \) or is not included; and no other possibility is allowed.

III. INTEGER LINEAR PROGRAMMING FORMULATION OF THE ZOOCP

This section describes how a ZOOCSP can be transformed into an equivalent integer linear programming (ILP) problem. As mentioned earlier, the main objective is to cover all the elements the set \( F = \{f_1, f_2, \ldots, f_m\} \) using the minimum number of items drawn from the set \( T = \{t_1, t_2, \ldots, t_n\} \). Again, the covering status of all \( t_i \)s are indicated the by the \( n \times m \) covering matrix \( A \). Now, one can define a set of \( n \) integer variables \( v_1, v_2, \ldots, v_n \) as described in equation (1).

\[
v_i = \begin{cases} 
1 & \text{if } t_i \text{ has to be included in the final cover} \\
0 & \text{otherwise}
\end{cases} \quad (1)
\]

where \( i = 1, 2, \ldots, n \).

In this situation, the required ILP problem will have the objective function as indicated in equation (2).

\[
\text{Minimize } Z = \sum_{i=1}^{n} v_i \quad (2)
\]

It is easy to see that this objective function makes sure that the final cover will contain minimum number of elements drawn from the set \( T \) provided each \( v_i \) assumes a value either 0 or 1. This condition can be satisfied by incorporating the \( 2n \) constraints as described in equation (3).

\[
v_i \geq 0 \text{ and } v_i \leq 1 \\
\text{for } i = 1, 2, \ldots, n \quad (3)
\]

Finally, the requirement that each \( f_i \) has to be covered by at least one \( t_j \) can be satisfied by adding \( m \) more constraints as shown in equation (4).
The whole process can be best understood via an example. For this purpose, provided below is the ILP problem that is equivalent to the covering problem described in the previous section (see Figure 1).

Minimize
\[ Z = v_1 + v_2 + v_3 + v_4 + v_5 + v_6 \]

Subject to
\[ v_1 \geq 0 \text{ and } v_1 \leq 1 \]
\[ v_2 \geq 0 \text{ and } v_2 \leq 1 \]
\[ v_3 \geq 0 \text{ and } v_3 \leq 1 \]
\[ v_4 \geq 0 \text{ and } v_4 \leq 1 \]
\[ v_5 \geq 0 \text{ and } v_5 \leq 1 \]
\[ v_6 \geq 0 \text{ and } v_6 \leq 1 \]

\[ v_1 + v_2 \geq 1 \]
(For column 1 of the A matrix)

\[ v_1 + v_3 + v_5 \geq 1 \]
(For column 2 of the A matrix)

\[ v_6 \geq 1 \]
(For column 3 of the A matrix)

\[ v_4 + v_6 \geq 1 \]
(For column 4 of the A matrix)

IV. NEURAL NETWORK APPROACH TO THE ZOOCP

This section is devoted to describe the neural network approach to the ZOOCP. In [11], Tank and Hopfield have proposed a neural network solution to the classical linear programming (LP) problem. Also, in the previous section it is shown that for every ZOOCP there exists an equivalent ILP problem. Therefore, it is possible to obtain a neural network solution to this equivalent ILP problem by modifying the original Tank and Hopfield LP circuit. This is the central idea of this section.

A general n-variable ILP problem attempts to minimize an objective function of the form shown in equation (5).

\[ Z = LV \]

In this equation, \( L \) is an n-dimensional row vector of coefficients while \( V \) is an n-dimensional column vector of solution variables. For the ZOOCP of the previous section, the components of \( L \) and \( V \) vectors degenerate to simple numbers as shown in equation (6).

\[ l_i = 1 \text{ and } v_i = 0 \text{ or } 1 \]
for \( i = 1, 2, \ldots, n \)

The minimization indicated in equation 6 is subject to \( m \) linear constraints and they can be expressed in matrix form as indicated in equation (7).

\[ DV \geq P \]

In particular, \( D \) is an \( m \times n \) matrix of constraint coefficients and \( P \) is an \( m \)-dimensional column vector that contains bounds for these \( m \)-constraints. For the ZOOCP, the \( D \) matrix degenerates to the transpose of the covering matrix \( A \) while the
vector \( \mathbf{P} \) degenerates to a column vector of \((1)\)s. That is,

\[ p_j = 1 \text{ for } j = 1, 2, \ldots, m \quad (8) \]

In this discussion it is assumed that \( D_i \) indicates the \( i \)th row of the \( D \) matrix.

The organization of a neural net hardware that solves an \( n \)-variable \( m \)-constraint ZOOCP is shown in Figure 2. The entire hardware is divided into two major sections: 1) Objective function section and 2) Constraint section. The first section contains \( n \) amplifiers one for each solution variable. For example, the output of the \( k \)th objective function section amplifier (OA) represents the value of the solution variable \( v_k \). A constant current proportional to \( k \) is fed as the input to this amplifier.

The constraint section entails \( m \) amplifiers one for each constraint equation. Each of these \( m \) constraints section amplifiers (CA) is fed a constant current proportional to the corresponding component of the bound vector \( \mathbf{P} \). For example, the \( k \)th CA receives a constant bias current of \( p_k \) as its input. In addition, each CA also receives an input current from OAs. For example, the \( i \)th CA receives as its input a current proportional to \( d_{ij} \) from the \( j \)th OA. In this setup, each CA indicates the constraint satisfaction status. In particular, each CA obeys the non-linear input-output relation as shown in equation (9).

\[ C_j = g(u_j) \]

where \( u_j = D_j \mathbf{v} - p_j \quad (9) \)

and \( g(x) = 0 \text{ for } x \geq 0 \)

\[ = 0 \text{ for } x < 0 \]

This equation indicates that when the \( j \)th constraint is violated, the \( j \)th CA injects a current proportional to \( d_{ij} \) into the \( i \)th OA. No current will be injected when this constraint is satisfied.
In this system, the $i^{th}$ OA has an input capacitance $C_i$ and an equivalent resistance $R$. Also, these amplifiers are assumed to satisfy the hard-limiting non-linear input-output relation as shown in equation (10).

$$ v_i = f(u_i) $$
where $f_i(x) = 1$ when $x \geq 0$
$= 0$ otherwise (10)

Thus it is clear that $v_i$ is either 0 or 1 as required.

The entire hardware operates in a push-pull fashion. Each OA basically attempts to minimize the corresponding variable by pushing its output to zero. However, each CA enforces the corresponding constraint by pulling the output of the OA via injecting a current of opposite polarity. Thus when the system reaches equilibrium, the vector $V$ gives the optimum value subject to all constraints being satisfied.

The equation of motion for the $i^{th}$ OA can be written as shown in equation (11).

$$ C_i \frac{du_i}{dt} = -l_i - u_i/R - \sum_j d_{ji} g(D_j V - p_j) $$

It has been proved by Tank and Hopfield [11] that these dynamics will minimize the Liapunov function shown in equation (12) provided the transfer function of the OA is a bounded monotonically increasing function.

$$ E = DV + \sum_i G (D_i V - p_i) $$
where
$$ g(Z) = \frac{dG(Z)}{dZ} $$

Now, an example is provided in order to illustrate how to arrive at a neural net hardware for a given ZOOCP. Diagramed in Figure 3 is the neural network circuit corresponding to the covering problem of Figure 1. In this hardware there are six OAs and four CAs because the original problem has six variables and four constraints respectively. In this Figure all $d_{ij}$ connections are indicated via thick diagonal lines.

Figure 3. Neural net for handling the ZOOCP of Figure 1.

The neural network circuit presented in this section can be implemented in IC technology.
provided the simulation results are satisfactory. In order to make simulation realistic, each OA is assumed to be in some arbitrary random initial excitation. The key calculation in the simulation is the evaluation of a time derivative of \( u_i \). In particular, the simulation algorithm follows the sequence described below:

1. Randomly initialize all \( u_i \)s.
2. Calculate each \( u_i \) by numerically integrating the equation 11.
3. Figure out \( v_i \) (by using equation (10))
4. Go to step 2

In order to expedite the simulation process the integration is carried out using the Euler's Method. The simulation program is coded in Pascal and it runs on the IBM PC/AT computer.

VI. SIMULATION RESULTS

The neural network solution to the covering problem is shown in Figure 4. From this figure one can notice that the system arrives at the final answer after two iterations. For a second example, consider the neural network solution for the covering problem shown in Figure 5. In this case there are 6 variables and 7 constraints. The simulation results indicate that the neural network convergence to the final answer after 6 iterations. For a final example, consider depicted in Figure 6. In this case, the neural network converges to the optimum result after 10 iterations.

From Figures 4, 5 and 6 one can notice that the neural network does exhibit a strange but consistent behavior with respect to arriving at the final answer. That is, starting from the arbitrary initial state the system first moves to the desired solution state, remains in this state for \( P-1 \) more iterations and then resets itself. It repeats this pattern of behavior for many number times during the course of its evolution. This situation leads to the following definition for convergence:

The ZOOCP neural net is said to have converged to the final answer \( V^* \) if the present solution vector \( V \) is a null vector and the value of the solution vector \( V \) in the last \( P \) iterations are identically equal to \( V^* \). From the simulation results it is conjectured that for a given ZOOCP the value of \( P \) is related to the number of constraints \( m \) as indicated in equation (13)

\[
P = \left\lfloor \frac{m}{2} \right\rfloor - 1 \quad (13)
\]

The above formula agrees with respect to all simulation results. For example the value of \( P \) for the simulation result of Figure 4 is 2 because in this in this problem there are 2 constraints, that is \( m = 2 \). Similarly, for the simulation results of Figure 6 the value \( P \) is 7 because for this problem \( m = 15 \).
Figure 4. Neural Network solution to ZOOCP-case-study I

1 1 0 0 1 1
1 0 0 0 0 0
0 1 0 1 0 0 0
0 0 0 1 0 0 0
0 1 0 0 1 0 0
0 0 1 1 1 1 0
0 0 0 0 0 0 0
Status after 1 iterations
0 0 0 0 0 0

Status after 2 iterations
1 0 0 0 0 1

Status after 3 iterations
0 0 0 0 0 0

Status after 4 iterations
1 0 0 0 0 1

Status after 5 iterations
0 0 0 0 0 0

Final Answer is: 1 0 0 0 0 1

Status after 6 iterations
1 0 0 0 0 1

Status after 7 iterations
1 0 0 0 0 1

Status after 8 iterations
1 0 0 0 0 1

Status after 9 iterations
0 0 0 0 0 0

Final Answer is: 1 0 0 0 0 1

Figure 5. Neural Network solution to ZOOCP-case-study II

0 1 0 0 0 0 0 1 1 1 1 0 0 1
0 0 1 1 0 0 1 1 0 0 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 1 1 1 0 0 0
0 0 1 1 1 1 1 1 0 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Status after 9 iterations
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Status after 10 iterations
0 0 1 0 0 0 0 0 0 0 1 0 0

Status after 11 iterations
0 0 1 0 0 0 0 0 0 0 0 1 0 0

Status after 12 iterations
0 0 1 0 0 0 0 0 0 0 0 1 0 0

Status after 13 iterations
0 0 1 0 0 0 0 0 0 0 0 1 0 0

Status after 14 iterations
0 0 1 0 0 0 0 0 0 0 0 1 0 0

Status after 15 iterations
0 0 1 0 0 0 0 0 0 0 0 1 0 0

Status after 16 iterations
0 0 1 0 0 0 0 0 0 0 0 1 0 0

Status after 17 iterations
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Final answer is:
0 0 1 0 0 0 0 0 0 0 0 1 0 0

Figure 6. Neural Network solution to ZOOCP-case-study III
VII. CONCLUSION

In this work, the neural network approach to the ZOOCP is discussed. Simulation results indicate that this approach provides more efficient solution to the ZOOCP as compared to the conventional techniques such as Petrick's method or Bala's Additive Algorithm [1]. Also conjectured in this work is a new definition for convergence. The hardware structure proposed in this investigation can be realized in VLSI by using MOS devices operating in the subthreshold region. This mode of operation guarantees small feature size as well as low power consumption. It has been reported by Mead [13] that VLSI chips that are built using subthreshold MOS devices have the ability to mimic biological functions. So, building a special purpose hardware for handling the ZOOCP can be deemed as a plausible extension to this work. Also, there are several difficult but important problems of switching theory can be recasted into an optimization problem. For instance, finding a closed cover for an incompletely specified sequential machine and finding race free state assignment for an asynchronous state assignment are the typical problems that fall into this category. At the moment, the authors are looking at these problems and the results will be reported elsewhere. Also, there is a similarity between the Hopfield neural net and asynchronous circuit built around threshold logic gates [2]. One may attempt to come up with a unified theory in order to explain the salient characteristics of the above mentioned paradigms.

REFERENCES


