A New Method to Optimize the Satellite Broadcasting Schedules Using the Mean Field Annealing of a Hopfield Neural Network

Nirwan Ansari, *Senior Member, IEEE*, Edwin S. H. Hou, *Member, IEEE*, and Youyi Yu

Abstract—**This paper reports a new method for optimizing satellite broadcasting schedules based on the Hopfield neural model in combination with the mean field annealing theory. A clamping technique is used with an associative matrix, thus reducing the dimensions of the solution space. A formula for estimating the critical temperature for the mean field annealing procedure is derived, hence enabling the updating of the mean field theory equations to be more economical. Several factors on the numerical implementation of the mean field equations using a straightforward iteration method that may cause divergence are discussed; methods to avoid this kind of divergence are also proposed. Excellent results are consistently found for problems of various sizes.**

I. Introduction

OPTIMIZATION of large connectionist problems is a long-standing topic in various disciplines, with many different approaches and applications. The problem discussed here, optimization of the broadcasting time from a set of satellites to a set of ground terminals (the satellite broadcast scheduling (SBS) problem), is one of these categories that must be solved for satellite communication systems. In their papers [1], [2], Bourret *et al.* solved this problem by using a neural network in which neurons are connected in a three-layer model. To find the optimum, a sequential search is used. The search is controlled by a competitive activation mechanism based on a dynamic prioritization of satellites. The sequential search, which is local in scope, is also very time-consuming. In addition, two additional premises (a set of distinct priorities of satellites and a set of suitable requests which are very difficult to determine for large problems) are also needed. Therefore, alternative efficient optimization methods are explored to solve this problem.

In this paper, a new method is presented to solve the SBS problem. The work is based on a Hopfield neural network [3], [4], where all neurons are completely connected, in combination with the mean field annealing theory (MFT) which was recently found to be an efficient method in solving large connectionist problems [5], [6]. The main advantage of using the MFT method lies in the fact that the search for optima is parallel in the global sense, and hence the execution time is shorter than other stochastic hill-climbing methods [7]–[9]. In contrast to the method mentioned in [1], [2] which requires the two premises mentioned above, they are not required for our method. Using our method, excellent solutions are consistently found for problems of various sizes.

Manuscript received January 22, 1993; revised July 18, 1993.

The authors are with the Center for Communications and Signal Processing, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, University Heights, Newark, NJ 07102 USA.

IEEE Log Number 9214273.

Instead of using a special neuron model (graded neuron) [10] to reduce the solution space and to avoid a destructive redundancy, a conventional neuron model clamped by an "associative matrix" is used in this work. This clamping technique is often applied in learning algorithms [5], [11], [12], resulting in a large decrease of the solution space.

Due to the nonlinearity of the sigmoid function, a so-called critical temperature T_c exists. Instead of using the "trial and error" approach to determine T_c, a formula for estimating T_c is derived. Experiments show that the estimated values using this formula are within 10% from the experimental (trial & error approach) results.

In this work, a type of divergence caused by the numerical implementation of the mean field equations is analyzed, and some schemes are suggested to avoid this kind of divergence.

This paper is organized as follows. In Section II, we briefly describe the satellite broadcasting problem, and map it onto a neural network framework. This is followed, in Section III, by a brief review on some recently proposed optimization methods with emphasis on the MFT. A set of mean field equations are also derived in this section to solve the SBS problem. In the next section, the determination of the Lagrange parameters and the derivation of the formula for T_c are presented. The numerical calculations and simulation results are discussed in Section V. Finally, conclusions are presented in Section VI.

II. Mapping the SBS Problem onto a Neural Network

Since the successful launch of the first commercial satellite Telstar in the 1960's, satellite communications has grown into a multibillion dollar industry. Most commercial systems are launched onto the geostationary orbit in spite of the disadvantages such as the inability to cover the far northern latitudes [13], the high costs for launching, and the requirement of very large antennas. The bulk traffic required for international telephone service over satellite systems is the key factor for adopting geostationary orbit where the earth stations and satellites appear stationary, i.e., no "hand-over" from a satellite to another is needed. As the demand for various telecommunications applications becomes increasingly sophisticated,

1045-9227/95$04.00

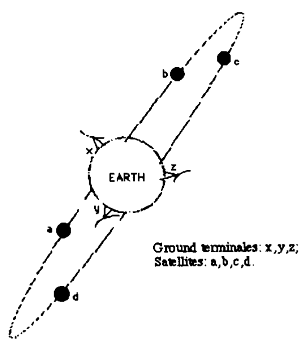

Fig. 1. An orbiting satellite communication system.

there are situations where orbits other than geostationary orbit become desirable. In this case, it is necessary to schedule the "hand-over" of operations from one satellite to another [14]—this is the problem addressed in this paper. This is a relatively new problem that few have addressed because 1) there were only a few low-altitude satellites in place, and 2) most were used for surveillance and data collection that may have proprietary and classified constraints. It emerges as an important problem, however, since potential advantages of a low altitude system such as reduced satellite power requirement and antennas, smaller propagation delay and high resolution images have prompted the industry to build such a system. For example, ORBCOMM proposes to operate a system of about 20 low altitude satellites, and GLOBALSTAR is considering a larger system of about 40 satellites [15]. Other players are following suit. Thus, the satellite broadcast scheduling is becoming a crucial problem. An European group has recently attempted to address this problem [1], [2].

The problem discussed here is related to a low altitude satellite communication system. As shown in Fig. 1, this system consists of a set of satellites and a set of ground terminals. Unlike the geostationary communication systems, the satellites here are usually located in a polar orbit with a rather low altitude and they always orbit around the earth. Hence, the ground terminals only need to employ low-power transmitters and portable antennas. The system can provide global communications coverage including the two polar regions which still cannot be achieved by the geostationary systems [13].

Our work is to maximize the broadcasting time for each satellite such that all the following constraints are met:

1) A satellite cannot broadcast to more than one ground terminal at a time;
2) A ground terminal cannot receive information from more than one satellite at a time;
3) A satellite must broadcast as much as possibly close to its requested time, and the system cannot allocate more time than requested unless the requested time for the rest of the satellites are completely satisfied;
4) A satellite only broadcasts when it is visible from a ground terminal.

To solve this problem, we adopt the following notation.

$\boldsymbol{S}$ is the set of satellites consisting of N_S elements (satellites)

$$\boldsymbol{S} = \{a, b, c, d, \cdots\} = \{1, 2, 3, \cdots, i, \cdots, N_S\}.$$

Here, $a, b, c, d, \cdots$ denote the different satellites each of which can be indexed by an integer number, i, ranging from 1 to N_S.

$\boldsymbol{A}$ is the set of ground terminals consisting of N_A elements (terminals)

$$\boldsymbol{A} = \{z, y, x, w, \cdots\} = \{1, 2, 3, \cdots, j, \cdots, N_A\}.$$

Here, $z, y, x, w, \cdots$ are different ground terminals; each of which can also be indexed by an integer number, j, ranging from 1 to N_A;

$\boldsymbol{T}$ is the set of time slots consisting of N_T elements (time slots); each of which can be indexed by an integer number, k, ranging from 1 to N_T;

$\boldsymbol{R}$ is a vector denoting the set of requested number of time slots given by the problem. It consists of N_s elements (time slots)

$$\boldsymbol{R} = [r_1, r_2, r_3, \cdots r_{N_S}]^t.$$

Here, $r_1, r_2, r_3, \cdots, r_{N_S}$, are the requested time slots for satellite $1, 2, 3, \cdots, N_s$, respectively.

$\boldsymbol{U}$ is a vector denoting the set of maximum number of time slots for each satellite allocated by the system. It consists of N_s elements

$$\boldsymbol{U} = [u_1, u_2, u_3, \cdots, u_{N_S}]^t.$$

Here, $u_1, u_2, u_3, \cdots, u_{N_S}$ are the number of time slots allocated for satellites $1, 2, 3, \cdots, N_s$, respectively.

Our goal is to find the optimal schedule satisfying the following two criteria simultaneously:

1) The schedule must be legal, that is, all constraints are fulfilled;
2) The distance between the vectors, $\boldsymbol{U}$ and $\boldsymbol{R}$, must be minimized.

A. Neuron Encoding

In this paper, we denote a neuron by S_{ijk}. Each neuron is turned "on" or "off" depending on whether or not satellite i is assigned to transmit to terminal j during time slot k. Thus, S_{ijk} is mathematically defined by

$$S_{ijk} = \begin{cases} 1 & \text{if satellite } i \text{ is assigned to} \\ & \quad \text{terminal } j \text{ during time slot } k; \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

B. Associative Matrix A

From the above definition of neurons, it is clear that some neurons are always fixed to zero because of Constraint (4) mentioned earlier. This is due to the fact that no ground terminal is visible to the satellite even when all ground terminals are idle. Usually, the number of neurons which are nulled owing to Constraint (4) is large. We should reflect this constraint into the neural network by clamping those neurons which do not meet Constraint (4) to zero throughout the optimization. To do so, we define an associative matrix $\boldsymbol{A}$ with $N_S \times N_A$ rows and N_T columns

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_2 \\ \vdots \\ \boldsymbol{A}_{N_S} \end{bmatrix} = \begin{bmatrix} a_{111} & a_{112} & \cdots & \cdots & a_{11N_T} \\ \cdot & & \cdots & \cdots & \cdot \\ a_{1N_A1} & a_{1N_A2} & \cdots & \cdots & a_{1N_AN_T} \\ \cdot & \cdot & \cdots & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdots & \cdot \\ a_{N_SN_A1} & \cdot & \cdots & \cdots & a_{N_SN_AN_T} \end{bmatrix} \tag{2}$$

where A_i is the submatrix ($N_A \times N_T$) associated with Constraint (4) imposed on satellite i; a_{ijk} takes on zero or one according to

$$a_{ijk} = \begin{cases} 1 & \text{if satellite } i \text{ is visible to terminal } j \text{ at time slot } k; \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

From the definition of $\boldsymbol{A}$ and the problem constraints, two important relations are observed:

1) The maximum number of requested time slots $r_{\max(i)}$ for satellite i must be less than or equal to the number of nonzero columns of the submatrix $\boldsymbol{A}_i$

$$r_{\max(i)} \le \sum_{i=1}^{N_T} (\text{“nonzero columns in } \boldsymbol{A}_i\text{”}). \tag{4}$$

2) The usable time slots u_i must be less than or equal to $r_{\max(i)}$.

$$u_i = \sum_j^A \sum_k^T S_{ijk} \le r_{\max(i)}. \tag{5}$$

The above relations are useful since they can be used to check for the illegality of a solution.

C. Formulation of the Energy Function

In optimization problems, one needs to formulate a particular objective function which is to be optimized. Our problem is a constrained optimization problem. To map a constrained optimization problem onto a Hopfield neural network [3], [4], we have to embed the constraints onto one function known as the energy function which consists of two terms: the cost term and the constraint term. The cost term is the optimization cost (objective) function that is independent of the constraint term. This constraint term is the penalty imposed on for violating the constraints.

$$E = w_c \times \text{“cost”} + w_p \times \text{“penalty”} \tag{6}$$

where w_c and w_p are the Lagrange parameters [16]. These two terms must counteract each other. In our case, the cost term is negative and the constraint term is positive. The optimization is then achieved by minimizing the energy function. Here, the cost term or the energy due to the cost, E_0, is defined by

$$E_0 = -\tfrac{1}{2}\sum_i^S \sum_j^A \sum_k^T (S_{ijk} \cdot S_{ijk}) \tag{7}$$

which reflects the idea of maximizing the total broadcasting time. The negative sign implies that minimization is to be applied.

The following penalty terms will be defined according to the four constraints:

1) A satellite cannot broadcast to more than one ground terminal at a time. The statement implies that all of the following equations must be satisfied simultaneously because they represent all possible violations.

$$\sum_i^S \sum_k^T \sum_j^A \sum_{j_1 \neq j}^A (S_{ij_k} \cdot S_{ij_1k}) = 0 \tag{8}$$

$$\sum_i^S \sum_k^T \sum_j^A \sum_{j_1 \neq j}^A \sum_{j_2 \neq j_1 \neq j}^A (S_{ijk} \cdot S_{ij_1k} \cdot S_{ij_2k}) = 0 \tag{9}$$

$$\vdots$$

$$\sum_i^S \sum_k^T \sum_j^A \sum_{j_1 \neq j}^A \sum_{j_2 \neq j_1 \neq j}^A \cdots \sum_{j_m \neq \cdots \neq j_2 \neq j_1 \neq j}^A \cdot (S_{ijk} \cdot S_{ij_1k} \cdot S_{ij_2k} \cdot \cdots \cdot S_{ij_mk}) = 0. \tag{10}$$

Obviously, when the total number of ground terminals increases, the number of equations required to impose this constraint increases. Fortunately, however, it can be shown that if the first equation is satisfied, the remaining equations are also satisfied simultaneously.

Lemma 1: If (8) is satisfied, Constraint (1) is met.

Proof: Consider (9)

$$\sum_i^S \sum_k^T \sum_j^A \sum_{j_1 \neq j}^A \sum_{j_2 \neq j_1 \neq j}^A (S_{ijk} \cdot S_{ij_1k} \cdot S_{ij_2k}) = \sum_i^S \sum_k^T \sum_{j_2 \neq j_1 \neq j}^A S_{ij_2k} \sum_j^A \sum_{j_1 \neq j}^A (S_{ijk} \cdot S_{ij_1k}).$$

By definition, each neuron takes on either one or zero. Thus, if (8) is true, then every term in (8) must be zero

$$S_{ijk} \cdot S_{ij_1k} = 0 \qquad \forall i, k, j_1 \neq j. \tag{11}$$

Substituting (11) into (9), we obtain the following

$$\sum_i^S \sum_k^T \sum_j^A \sum_{j_1 \neq j}^A \sum_{j_2 \neq j_1 \neq j}^A (S_{ijk} \cdot S_{ij_1k} \cdot S_{ij_2k}) = 0.$$

By deduction, if (8) is true, the remaining equations required to impose Constraint (1) are all satisfied simultaneously. Thus, only (8) is needed to impose Constraint (1). Q.E.D.

Hence, the penalty term for this constraint is

$$E_1 = \sum_i^S \sum_k^T \sum_j^A \sum_{j_1 \neq j}^A (S_{ijk} \cdot S_{ij_1k}). \tag{12}$$

2) A ground terminal cannot receive information from more than one satellite at a time. Constraint (2) is a dual to Constraint (1). This can be seen by simply replacing the z, y, x, w, with a, b, c, d in (8) through (10), respectively. Thus, the penalty term for this constraint is similarly defined by

$$E_2 = \sum_{j}^{A} \sum_{k}^{T} \sum_{i}^{S} \sum_{i_1 \neq i}^{S} (S_{ijk} \cdot S_{i_1 jk}). \tag{13}$$

3) A satellite must broadcast as much as possibly close to its requested time slots, and the system cannot allocate more time than requested unless the requests for the rest of the satellites are completely satisfied.

The first part of the statement implies that the distance between $\boldsymbol{U}$ and $\boldsymbol{R}$ should be minimized. Thus, the penalty term, E_3, corresponding to this statement is

$$E_3 = \sum_{i}^{S} \left(\sum_{j}^{A} \sum_{k}^{T} S_{ijk} - r_i \right)^2 = \sum_{i}^{S} (u_i - r_i)^2 \tag{14}$$

where

$$u_i = \sum_{j}^{A} \sum_{k}^{T} S_{ijk}.$$

The second part of the statement implies that $u_i - r_i \begin{cases} \geq 0 & \forall i \\ < 0 & \forall i \end{cases}$. Note that this has been incorporated in the cost term E_0.

4) A satellite broadcasts only when it is visible from a ground terminal. This constraint is imposed by the clamping technique which will be discussed in Section III. That is, neurons are forced to 0 (complied to Constraint (4)) by using the associative matrix. The total energy function for the SBS problem defined in the Hopfield framework becomes

$$E = w_0 \cdot E_0 + w_1 \cdot E_1 + w_2 \cdot E_2 + w_3 \cdot E_3 \tag{15}$$

where w_0, w_1, w_2, w_3 are the Lagrange parameters used to weigh the significance of E_0, E_1, E_2 and E_3, respectively.

III. The MFT Framework for the SBS Problems

In the previous section, the SBS problem has been mapped onto the neural network framework, and the energy function has been formulated. The remaining task is to employ a robust method to minimize the energy function. Our problem is a large scale combinatorial optimization problem in which the energy function to be optimized is a function of discrete variables. A search for the optimal configuration is computationally expensive, if not impossible.

Conventional methods such as gradient-based methods which are local in scope are not applicable. Recently, several robust methods such as simulated annealing [7], [17] and genetic algorithm [18] have been proposed to solve large scale problems; however, not every problem can be mapped onto the framework suitable for these methods. MFT [5], [6], [19], [20] is derived from the Stochasticly Simulated Annealing (SSA) by incorporating the SSA mechanism with the Hopfield Energy function. It has been shown to be robust in solving large scale problems, and more efficient than SSA.

The main difference between SSA and conventional methods is that SSA searches for the global minimum by using the gradient descent method in a stochastic manner. It allows, under certain conditions, the search to climb uphill, thus providing the SSA a mechanism to escape from local minima.

In SSA, there are two conceptual operations involved: a thermostatic operation which schedules the decrease of the temperature (an algorithm parameter), and a random relaxation process which searches for the equilibrium solution at each temperature.

In MFT the two operations are still needed. The thermostatic operation is the same as in SSA; however, the relaxation process in searching for the equilibrium solution has been replaced by searching for the average (mean) value of the solutions. Equilibrium can be reached faster by using the mean [20], and thus the MFT speeds up by several tens to hundreds times over the SSA.

The remaining question is whether the two solutions obtained from the two respective relaxations are approximately equal to each other. It has been proved by Peterson [5], [6] that for large size problems which are really what we are interested in and also, by experiments, even for small size problems, the answer is true.

A. Mean Field Equations for the SBS Problem

In this section, we first briefly review the general mean field equations. The notations are adopted from [5], [6], [10] in which the detailed derivation can be found. The relaxation in both SSA and MFT are made according to the Boltzmann distribution [7]

$$\boldsymbol{P}(\boldsymbol{S}') = e^{-E(\boldsymbol{S}')/T}/Z \tag{16}$$

where $\boldsymbol{S}'$ is any one of the possible configurations specified by the corresponding neuron set

$E(\boldsymbol{S}')$ is the energy of the corresponding configuration;

T is the parameter called temperature;

Z is the partition function given by

$$Z = \sum_{\boldsymbol{S}'} e^{-E(S')/T} \tag{17}$$

and the summation covers all possible neuron configurations.

In the mean field theory, instead of concerning the neuron variables directly, we shall investigate their means (average) by defining

$$V_i = \langle S_i \rangle = 1 \cdot P_T(S_i = 1) + 0 \cdot P_T(S_i = 0) = P_T(S = 1) \tag{18}$$

and

$$\boldsymbol{V}' = \langle \boldsymbol{S}' \rangle \tag{19}$$

where S_i is a neuron; V_i is the mean of neuron S_i; $P_T(S_i = 1)$ and $P_T(S_i = 0)$ are the probabilities for $S_i = 1$ or $S_i = 0$, respectively; $\boldsymbol{S}'$ is any one of possible configurations; $\boldsymbol{V}'$ is the mean configuration corresponding to $\boldsymbol{S}'$. Thus, in the mean field, (16) becomes

$$\boldsymbol{P}(\boldsymbol{V}') = e^{-E(V')/T}/Z \tag{20}$$

and the discrete sum in (17) can be replaced by multiple nested integrals over the continuous variables V_i and U_i [5], [6]

$$Z = c\prod_{i=1}^{N}\int_{-\infty}^{\infty} dV_i' \int_{-j\infty}^{j\infty} dU_i' e^{-F(V',U',T)} \tag{21}$$

where c is a complex constant, and $\boldsymbol{U}' = \{U_1, U_2, U_3, \cdots, U_N\}$; F is called the effective energy given by

$$F(\boldsymbol{V}',\boldsymbol{U}',T) = E(\boldsymbol{V}')/T + \sum_{i=1}^{N}\cdot\left(U_iV_i - \log\left(\sum_{S} e^{SU_i}\right)\right). \tag{22}$$

As has been indicated by Peterson [5], [6], by using a saddle point expansion of F, one could see that the partition function Z is actually dominated by the saddle point, i.e.,

$$Z \simeq Ce^{F(V_0',U_0',T)} \tag{23}$$

where C is a constant, and (V_0, U_0) is the saddle point of (21). Thus, the statistical mechanism of the MFT governed by (20) is likewise determined by the mechanism of the saddle point. The saddle point can be obtained as follows

$$\frac{\partial(F)}{\partial(U_i)} = 0, \tag{24}$$

$$\frac{\partial(F)}{\partial(V_i)} = 0. \tag{25}$$

Substituting (22) into (24)

$$\frac{\partial(F)}{\partial(U_i)} = V_i - \left(\sum_{S=\{0,1\}} Se^{su_i} \Big/ \sum_{S} e^{SU_i}\right) = 0$$

we obtain

$$V_i = \left(\sum_{S} Se^{SU_i}\right) \Big/ \sum_{S} e^{SU_i}, \qquad \text{where } s = \{0, 1\}. \tag{26}$$

Substituting (22) into (25)

$$\frac{\partial(F)}{\partial(V_i)} = \frac{\partial(E)}{\partial(V_i)}\frac{1}{T} + U_i = 0$$

we have

$$U_i = -\frac{\partial(E)}{\partial(V_i)}\frac{1}{T}. \tag{27}$$

Equations (26) and (27) are known as the general MFT equations.

For our SBS problem, replacing S_{ijk} defined in (1) by V_{ijk} and substituting it into (27) and then into (26), we now get

$$\begin{aligned} V_{ijk} &= \frac{1\cdot e^{1\cdot Ui} + 0\cdot e^{0\cdot Ui}}{e^{1\cdot Ui} + e^{0\cdot Ui}} = \frac{e^{Ui}}{1+e^{Ui}} \\ &= \frac{1}{2} + \frac{e^{Ui}}{1+e^{Ui}} - \frac{1}{2} = \frac{1}{2} + \frac{2e^{Ui} - 1 - e^{Ui}}{2(1+e^{Ui})} \\ &= \frac{1}{2} + \frac{1}{2}\frac{e^{Ui}-1}{1+e^{Ui}} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{U_i}{2}\right) \\ &= 0.5 + 0.5\tanh\left(-\frac{\partial(E)}{\partial(V_{ijk})}\frac{1}{2T}\right). \end{aligned} \tag{28}$$

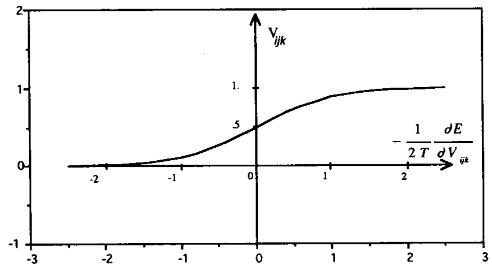

Fig. 2. Depicting the function of ν_{ijk}.

By incorporating the clamping technique discussed in Section II–B, we obtain the MFT equations for the SBS problem as follows

$$V_{ijk} = a_{ijk}\left(0.5 + 0.5\tanh\left(-\frac{\partial(E)}{\partial(V_{ijk})}\frac{1}{2T}a_{ijk}\right)\right). \tag{29}$$

Equation (29) is depicted below in Fig. 2.

IV. Algorithm Parameters

Before solving the MFA equations, several parameters must be specified. They are the Lagrange parameters, w_0, w_1, w_2, w_3, the critical temperature T_c, the saturation temperature and the annealing schedule. These are discussed below.

A. The Lagrange Parameters—w_0, w_1, w_3

In general, good solutions can be obtained for a reasonably wide domain in the space of w_0, w_1, w_2, w_3. Some guidelines are suggested here, however, to assure that our choices of the parameters lie within this domain.

In the mean field domain, all the energy functions E_0, E_1, E_2, E_3 become the functions of mean field variables as follows

$$E_0 = -\tfrac{1}{2}\sum_{i}^{S}\sum_{j}^{A}\sum_{k}^{T}(V_{ijk}\cdot V_{ijk}) \tag{30}$$

$$E_1 = \sum_{i}^{S}\sum_{k}^{T}\sum_{j}^{A}\sum_{j_1\neq j}^{A}(V_{ijk}\cdot V_{ij_1k}) \tag{31}$$

$$E_2 = \sum_{j}^{A}\sum_{k}^{T}\sum_{i}^{S}\sum_{i_1\neq i}^{S}(V_{ijk}\cdot V_{i_1jk}) \tag{32}$$

$$E_3 = \sum_{i}^{S}\left(\sum_{j}^{A}\sum_{k}^{T}V_{ijk} - r_i\right)^2. \tag{33}$$

Consider the derivative of the total energy function in the mean field domain

$$\frac{\partial(E)}{\partial(V_{ijk})} = w_0\frac{\partial(E_0)}{\partial(V_{ijk})} + w_1\frac{\partial(E_1)}{\partial(V_{ijk})} + w_2\frac{\partial(E_2)}{\partial(V_{ijk})} + w_3\frac{\partial(E_3)}{\partial(V_{ijk})}. \tag{34}$$

The parameter w_0 governs the relative balance between the "cost" and "constraint" terms. w_1, w_2, w_3 reflect the relative importance among Constraints (1) through (3). Since

$\partial(E_1)/\partial(V_{ijk})$ and $\partial(E_2)/\partial(V_{ijk})$ are similar in nature and much more important than the others, they are thus weighed equally, and are weighed heavier than the others. For example, we may choose: $w_0 = 0.4, w_1 = 2.0, w_2 = 2.0$. Consider the effect of each individual parameter on any neuron. Note that, from (31) and (32), $\partial(E_1)/\partial(V_{ijk})$ and $\partial(E_2)/\partial(V_{ijk})$ are always positive, and thus by (29), the value of neuron V_{ijk} due to E_1 and E_2 approaches "0." From (30) $\partial(E_0)/\partial(V_{ijk})$ is always negative, making neuron V_{ijk} approach "1." $\partial(E_3)/\partial(V_{ijk})$ may be positive or negative depending on whether the requested time slots have been satisfied, thus making the neuron approach "0" or "1," respectively.

Since we have already determined w_1 and w_2, we are left to determine the relationship between w_0 and w_3. In other words, we may now assume Constraints (1) and (2) are already satisfied, i.e., $E_1 = E_2 = 0 \Rightarrow \partial(E_1)/\partial(V_{ijk}) = \partial(E_2)/\partial(V_{ijk}) = 0$. Consider the extreme case in which V_{ijk} takes on either 0 or 1. In this case, for each fixed i, if the number of neurons having values "1" are more than the requested time slots (see (33)), this implies that the system tries to allocate more time slots than requested. Thus, we should try to force the system to turn "off" a neuron. Note that the neuron V_{ijk} that is "on" has $\partial(E_0)/\partial(V_{ijk}) = -1$ (see (30)), and $\partial(E_3)/\partial(V_{ijk}) = 2(\Sigma_j^A \Sigma_k^T v_{ijk} - r_i) \geq 2$ (see (33)) because the system has allocated more time slots than requested, i.e., $\Sigma_j^A \Sigma_k^T V_{ijk} - r_i \geq 1$. To turn off this neuron (see (34))

$$\begin{aligned}\frac{\partial(E)}{\partial(V_{ijk})} > 0 &\Rightarrow w_0 \frac{\partial(E_0)}{\partial(V_{ijk})} + w_3 \frac{\partial(E_3)}{\partial(V_{ijk})} > 0,\\ &\Rightarrow w_0(-1) + 2w_3\left(\sum_j^A \sum_k^T V_{ijk} - r_i\right) > 0,\\ &\Rightarrow w_0(-1) + 2w_3 > 0,\\ &\Rightarrow w_3 > 0.5w_0.\end{aligned}$$

Now consider the other extreme case in which the number of time slots allocated by the network is less than the requested time slots. In this case, the network should try to turn on a neuron. Note that the neuron V_{ijk} that is "off" has $\partial(E_0)/\partial(V_{ijk}) = 0$ and $\partial(E_3)/\partial(V_{ijk}) < 0$. Thus, $\partial(E)/\partial(V_{ijk}) < 0$, the neuron is turned "on" as long as $w_3 > 0$.

In conclusion, we may use the following rule of thumb

$$w_0 = 0.4, w_1 = w_2 = 2, w_1 > w_3 > 0.5w_0. \quad (35)$$

B. The Critical Temperature, T_c

In MFT, our task is to solve for the neuron value V_{ijk} at different temperatures through a set of nonlinear equations (29). For convenience, this equation is rewritten as follows

$$V_{ijk} - 0.5a_{ijk} = 0.5\tanh\left(-\frac{\partial(E)}{\partial(V_{ijk})}\frac{1}{2T}a_{ijk}\right). \quad (36)$$

To gain insight on the dynamics in obtaining a solution, consider Fig. 3. In this figure, while the abscissa represents the neuron V_{ijk}, the ordinate represents various functions of V_{ijk}. On the other hand, while the ordinate represents an arbitrary

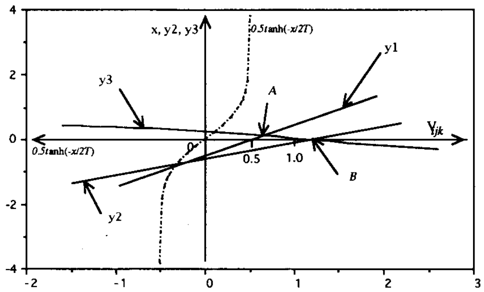

Fig. 3. The dynamics in obtaining a solution.

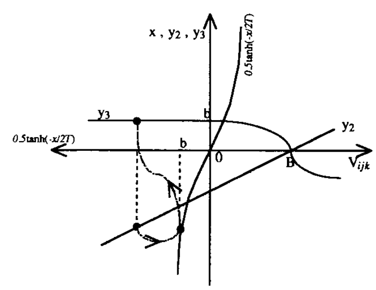

Fig. 4. Mapping a point from y_2 to y_3.

variable X, the abscissa represents $0.5\tanh(-X/2T)$. The straight line labeled by $y1$ represents the left-hand side of (36). Here we only need to consider the case when $a_{ijk} = 1$, otherwise, $V_{ijk} = 0$. The straight line labeled by $y2$ represents the function $\partial(E)/\partial(V_{ijk})$. Note that in our SBS problem, from (30) through (33) , we obtain

$$\frac{\partial(E)}{\partial(V_{ijk})} = m(V_{ijk} - B) \quad (37)$$

where m and B are constants, which is a straight line. The discussion here, however, is also applicable to the case when $\partial(E)/\partial(V_{ijk})$ is no longer a straight line. As mentioned above, the dashed curve represents the function $(0.5\tanh(-X/2T))$ in which the ordinate is X, and the abscissa is $0.5\tanh(X/2T)$. It is readily seen from (36) that we can map $y2$ through the hyperbolic tangent function [i.e., $0.5\tanh(-y2/2T)$] to obtain the curve labeled by $y3$. $y3$ corresponds to the right-side of (36). Fig. 4 shows an example of mapping a point on $y2$ to the corresponding point on $y3$ through the mapping, $0.5\tanh(-X/2T)$.

Note that Curve $y3$ and Line $y2$ intersect on the abscissa axis at a point labeled B. The value of B depends on the state of the network. Curve $y3$ and Line $y1$ intersect at A which is the solution for (36). The abscissa value of A is the neuron value of V_{ijk} at temperature T.

Fig. 5 shows the behavior of Curve $y3$ at different temperatures. Note that at high temperature, Curve $y3$ becomes a straight line with slope equal to approximately zero. Thus, the solution at high temperature is $V_{ijk} = 0.5$, i.e., the intersection point between $y1$ and $y3$ is (0.5, 0). We thus have the following Lemma.

Lemma 2: All neurons except those clamped by the associative matrix, have values of 0.5 at high temperature.

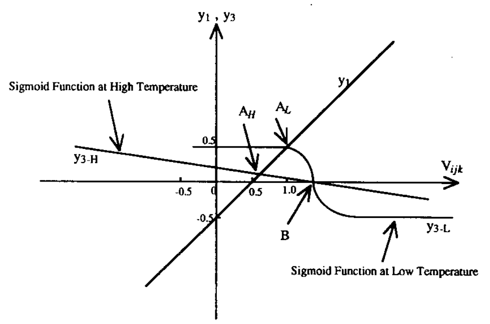

Fig. 5. The solutions at high and low temperatures.

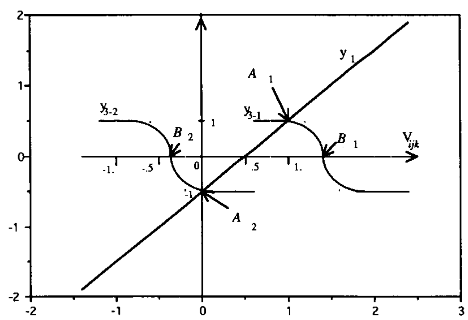

Fig. 6. The two possible solutions of (39).

As the temperature is decreasing, however, the dash curve and therefore $y3$ is becoming a signum function as shown in Fig. 5. If B is greater than 1, it can be seen that in this case, the intersection which is the solution is (1, 0.5), i.e., $V_{ijk} = 1$. Similarly, it can be shown that if B is less than 0, the solution is $V_{ijk} = 0$.

Our goal is to determine the temperature parameter known as the critical temperature at which a remarkable state transition takes place resulting in a deep drop of system energy. From Lemma 2 all neurons except those which are clamped have the same initial value of 0.5, and thus the remarkable state transition likely occurs when neurons start acquiring a value of 1 or 0, at which case the neurons start competing for 1 or 0. We thus propose the following definition for the critical temperature.

Definition: The critical temperature is the highest temperature at which at least one neuron V_{ijk} reaches 1 or 0 from its original trivial state, i.e., 0.5.

Lemma 3: The critical temperature, T_c, for our SBS problem is approximately equal to

$$T_c = m(2B - y - 1)/4y \tag{38}$$

where m, B and y are derived from the following system of equations

$$\begin{cases} 2V_{ijk} - 1 = y, \\ -\dfrac{1}{2T}m(V_{ijk} - B) = y. \end{cases} \tag{39}$$

Proof: From the above discussion and definition, the critical temperature corresponds to the critical value of T in (36) as the solution is making a transition from A_H to A_L as shown in Fig. 5. In other words, the critical temperature is obtained by solving for the parameter T in (36). Because of the nonlinear term, tanh(.), in (36), it is difficult, if not impossible, to obtain a closed form analytical solution. The following approximation is made by expanding $\tanh(-x/2T)$ by a Taylor series at $x = 0$, we have

$$\tanh\left(-\frac{x}{2T}\right) = -\frac{x}{2T} + \frac{1}{24}\left(\frac{x}{T}\right)^3 + \cdots.$$

If we use only the first order term to approximate $\tanh(-x/(2T))$, (36) becomes

$$2V_{ijk} - 1 = -\frac{\partial E}{\partial V_{ijk}}\frac{1}{2T}.$$

Here, the nontrivial case that the neuron is not clamped has been assumed, i.e., $a_{ijk} = 1$. Substituting (37) to the above equation, we have.

$$2V_{ijk} - 1 = -\frac{1}{2T}m(V_{ijk} - B).$$

Thus, the critical temperature can be obtained by solving the set of equations as stated in (39). Solving (39) results in the critical temperature as stated in (38). Here, y is $+1$ or -1 because the remarkable transition occurs when the neuron reaches 1 or 0 from its original trivial value; i.e., $V_{ijk} = 1$ or $0 \Rightarrow y = \pm 1$. Furthermore, as shown in Fig. 6, if $B > 1$ and $m > 0$, the solution $V_{ijk} = 1 (y = 1)$. Likewise, if $B < 0$ and $m > 0$, the solution $V_{ijk} = 0 (y = -1)$. Similarly, other conditions can be summarized below

$$\begin{array}{lll} & B > 0.5 & B < 0.5 \\ m > 0 & y = +1 & y = -1 \\ m < 0 & y = +1 & y = +1 \end{array}. \tag{40}$$

Strictly speaking, when $0 < B < 1, V_{ijk}$ does not take on 0 or 1, however, within a few iterations the particular V_{ijk} will converge to 1 or 0. Note that for a different neuron V_{ijk}, B may be different. As long as the estimated critical temperature is higher than the true critical temperature, however, the annealing theory guarantees that the system will converge to a (near) global optima. Thus, for a given associative matrix A, we shall solve for B using (37) for each neuron, and pick the largest B. Q.E.D.

The following example illustrates the computation of the critical temperature using Lemma 3.

Example 1: Let the associative matrix A and the vector R of the requested number of time slots be the same as those in Example 3 which will be described later in Section V–B, and let $w_0 = 0.3, w_1 = 2.0, w_2 = 2.0$, and $w_3 = 0.2$. Here, $m = 2w_3 - w_0 = 0.4 - .03 = 0.1 > 0$.

As discussed above, B is obtained by painstakingly checking every neuron that will yield the largest solution to (37). For this example, $B = 12$.

Since $m > 0$ and $B > 0.5$, then $y = 1$. Hence, $T_c = 0.5 \times m \times (B - 1) = 0.55$.

The critical temperature obtained through simulation results which will be presented later in Example 3 is 0.51. This agrees closely to the one computed above.

C. The Annealing Schedule

We adopt the following linear annealing schedule starting from the critical temperature

$$T(n+1) = 0.9T(n)$$

where

$$T(0) = T_c. \tag{41}$$

The stopping criterion for the annealing procedure is defined by the temperature at which the network is saturated. The network is saturated if the following conditions are met.

1) All neuron values are within the range [0.0, 0.2] or within the range of [0.8, 1.0] without any exception;
2) $\Sigma_i^S \Sigma_j^A \Sigma_k^T (V_{ijk})^2/N > 0.95$, where N is the number of neurons that have values within the range of [0.8, 1.0].

V. Numerical Implementation and Solutions

When implementing the MFA algorithm numerically, the straightforward iteration method below is used at each temperature to obtain the steady state neuron values

$$(V_{ijk})^{(n+1)} = a_{ijk}\left(0.5 + 0.5\tanh\left(-\frac{\partial(E)}{\partial(V_{ijk}^{(n)})}\frac{1}{2T}a_{ijk}\right)\right). \tag{42}$$

The superscript n indicates the iteration index. For each iteration, there are many neurons to be updated. We can either update all neurons synchronously or one after another asynchronously. In practice, it is found that asynchronous updating has a better performance. The procedure to schedule the satellite broadcasting times using MFT is summarized below:

1) For a given SBS problem, establish the associative matrix A described in Section II–B;
2) Establish the coefficients w_0, w_1, w_2, w_3 as discussed in Section IV;
3) Determine the critical temperature T_c according to Lemma 3;
4) Initialize neurons with random numbers as follows

$$V_{ijk} = \{0.5 + 0.2\text{rand}[-1,1]\}a_{ijk}; \tag{43}$$

5) Anneal the network until the network is saturated according to the saturation criterion defined in Section IV–C.
6) At each temperature, iterate the MFT equations until the following convergence criterion is met

$$\sum_i^S \sum_j^A \sum_k^T |V_{ijk}^{(n+1)} - V_{ijk}^{(n)}| < 10^{-3}N_n \tag{44}$$

where N_n is the number of nonzero neuron elements. That is, we require that the averaged difference of a neuron between two iterations to be within 10^{-3}.

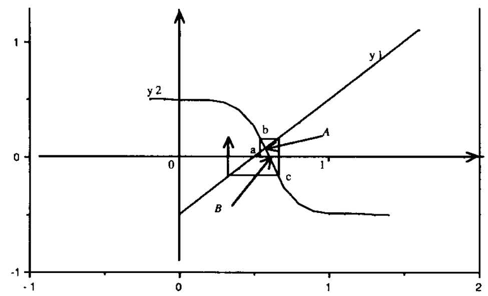

Fig. 7. Slope of y_2 in the neighborhood of the solution is greater than 1.

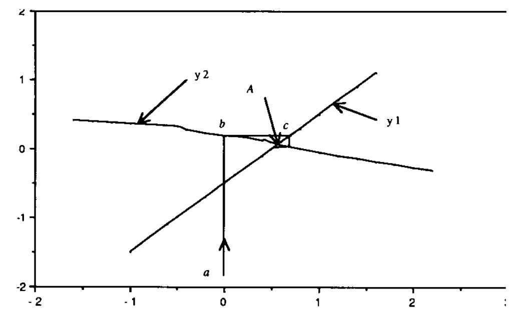

Fig. 8. Slope of y_2 in the neighborhood of the solution is less than 1.

A. Convergence

Divergence is one of the most difficult problems encountered in numerical calculations. There are several types of factors which lead the calculation to divergence. Next, we shall discuss the factors that will cause divergence in the straightforward iteration method used to solve the SBS problem. Rewriting the straightforward iteration described in (42) and ignoring the iteration index for the time being, we obtain

$$V_{ijk} - 0.5a_{ijk} = 0.5\tanh\left(-\frac{\partial(E)}{\partial(V_{ijk})}\frac{1}{2T}a_{ijk}\right). \tag{45}$$

Let the left-hand side and the right-hand side of (45) be y_1 and y_2, respectively

$$y_1 = V_{ijk} - 0.5a_{ijk} \tag{46}$$

$$y_2 = 0.5\tanh\left(-\frac{\partial(E)}{\partial(V_{ijk})}\frac{1}{2T}a_{ijk}\right). \tag{47}$$

The solution for (45) is the intersecting point between Line y_1 and Curve y_2 as shown in Fig. 7. Consider the following two cases in which the slope of y_2 in the neighborhood of the solution are less than and greater than 1

Case 1. $|\text{the slope of } y_2| > 1$;

Case 2. $|\text{the slope of } y_2| < 1$.

Fig. 7 depicts the condition corresponding to Case 1 and likewise, Fig. 8 to Case 2. These figures show how the solution evolves through the iteration procedure (45), indicated by the arrows. Here A is the solution. The sequence of arrow a-b-c represents one iteration. As shown in Fig. 7, the iteration procedure diverges from solution A, while the procedure converges to solution A in Fig. 8.

If $y2$ is moved such that the intersecting point B between $y2$ and the abscissa is outside the range [0, 1], it can be shown

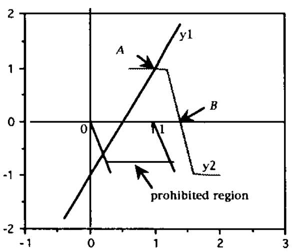

Fig. 9. The intersection Point B is outside the range [0, 1].

(see Fig. 9) that the iterating procedure will converge to the solution. Unfortunately, the exact location of the intersection point B is unknown because it will move dynamically during the iterating process. In Case 2, the divergence caused by one iteration is called local divergence. Local divergence may not be a fatal divergence because the intersection point B may move out of the range [0, 1] after a sweep of iterations (all neurons are updated once). If the intersection point B always lies within the range [0, 1], then the local divergence becomes global. We shall avoid global divergence.

In the previous section, we point out that asynchronous iteration is better than synchronous iteration. One reason is that the asynchronous method has more chances for the intersection point B to jump out of the [0, 1] range.

To avoid global divergence, one may adjust some parameters such that the intersection point B is out of the range [0, 1]. The following are a few suggestions:

1) Adjust w_0, w_1, w_2, w_3;
2) Simply increase r_i;
3) Use other iteration methods.

B. Solutions

We have implemented the proposed method to solve the SBS problem of various sizes. We consider cases when the requested broadcasting time is less than the maximum capacity the network can allocate, as well as cases when the requested broadcasting time exceeds the maximum capacity of the network. Cases of the first type are known as "small request" cases, and cases of the second type are known as "large request" cases. A network consisting of 108 neurons of which 44 neurons are active due to clamping is used to solve the following two examples. The last example shows a large size problem solved by a network of 864 neurons.

The capacity of a neural network is an important measure especially when the network is used as an associative memory or a classifier. The capacity allows one to quantify the amount of information the network can store or the number of patterns the network can distinguish. In a Hopfield net, information is stored as stable states. Hopfield net is known to have poor scaling properties. That is, its capacity for a network with binary neurons increases less than linearly [21], [22]. There has been a growing interest to increase the capacity such as using ternary neurons [23], [24], and subnetworking [25]. This interesting topic is, however, beyond the scope of this paper. Here, Hopfield net is used for optimization, and in combination with mean field annealing, allows fast convergence and escape from local minima in search for a global optima. Moreover, neurons take on sigmoidal nonlinearity rather than binary or ternary values. As mentioned earlier, problems of various sizes have been successfully solved by the proposed method. Here, a large example which is presentable within a page, consists of 864 neuron of which 400 are active. It corresponds to a system with 8 satellites and 6 antennas, a reasonably deployable system.

Example 2: Consider the SBS problem with four satellites, three ground terminals and nine time slots. Constraint (4) is defined by the Associate Matrix A and the requested broadcasting time for each satellite is defined by R as follows

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

$$R = [2 \quad 2 \quad 2 \quad 2]^t.$$

SOLUTION 1

$w_0 = 0.5, w_1 = 2.00, w_2 = 2.00, w_3 = 0.2.$

SATELLITE 1 (V_{1jk})
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000
0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000

SATELLITE 2 (V_{2jk})
0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000

SATELLITE 3 (V_{3jk})
0.000 0.000 0.000 0.000 0.000 1.000 1.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000

SATELLITE 4 (V_{4jk})
0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000
1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

$$U = [3 \quad 3 \quad 3 \quad 3]^t.$$

Here, U is the time slots allocated by the network.

SOLUTION 2

$w_0 = 0.3, w_1 = 2.00, w_2 = 2.00, w_3 = 0.2$

SATELLITE 1 (V_{1jk})
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.999 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.999 0.000 0.000

$$
\boldsymbol{A} = \begin{bmatrix}
1&1&0&1&0&1&1&0&1&1&0&0&1&1&1&0&1&0\\
0&0&1&0&0&0&0&1&0&0&1&0&0&1&1&1&0&1\\
1&0&0&1&1&1&1&1&0&0&1&0&1&0&0&1&1&1\\
1&1&1&1&0&0&0&0&1&1&1&0&0&0&1&1&0&1\\
0&0&1&0&0&0&1&1&0&0&1&1&0&1&0&1&0&0\\
0&0&0&1&0&1&0&0&1&1&0&1&1&0&0&0&1&1\\
1&0&1&0&0&1&1&0&1&0&1&1&1&1&0&0&1&0\\
0&0&0&0&0&0&0&0&1&1&0&0&1&0&1&1&0&0\\
0&0&0&0&1&1&1&1&0&1&1&0&0&1&0&0&1&1\\
0&1&1&0&0&0&0&0&1&0&0&1&1&0&0&0&1&1\\
0&0&0&0&0&0&0&1&1&0&0&0&1&1&0&0&0&0\\
1&0&0&1&0&0&0&1&1&1&1&0&1&0&1&1&0&1\\
0&1&0&0&1&1&1&0&0&1&1&0&0&1&1&0&1&0\\
0&0&1&0&1&0&0&1&1&0&1&1&0&1&0&0&1&1\\
1&0&0&1&1&1&1&1&0&0&1&0&0&1&0&0&1&1\\
1&1&1&1&0&0&0&0&0&1&0&0&1&1&1&0&0&1\\
0&0&1&0&0&0&0&1&1&0&0&1&1&0&1&0&0&0\\
0&0&0&1&0&1&1&0&0&1&0&1&1&1&0&1&1&1\\
1&0&1&0&1&0&1&0&1&0&1&1&0&1&0&0&1&0\\
0&0&0&0&0&1&0&0&1&1&0&0&1&0&1&1&0&0\\
0&0&0&1&1&1&1&1&0&1&1&0&0&1&0&0&1&1\\
0&1&1&0&0&0&1&0&1&0&1&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&1&1&0&0&0&1&1&0&0&0&0\\
1&0&0&1&0&0&0&1&1&1&1&0&1&0&1&1&0&1\\
1&1&0&1&0&1&1&0&1&1&0&0&1&1&1&0&1&0\\
0&0&1&0&0&0&0&1&0&0&1&0&0&1&1&1&0&1\\
1&0&0&1&1&1&1&1&0&0&1&0&1&0&0&1&1&1\\
1&1&1&1&0&0&0&0&1&1&1&0&0&0&1&1&0&1\\
0&0&1&0&0&0&1&1&0&0&1&1&0&1&0&1&0&0\\
0&0&0&1&0&1&0&0&1&1&0&1&1&0&0&0&1&1\\
1&0&1&0&0&1&1&0&1&0&1&1&1&1&0&0&1&0\\
0&0&0&0&0&0&0&0&1&1&0&0&1&0&1&1&0&0\\
0&0&0&0&1&1&1&1&0&1&1&0&0&1&0&0&1&1\\
0&1&1&0&0&0&0&0&1&0&0&1&1&0&0&0&1&1\\
0&0&0&0&0&0&0&1&1&0&0&0&1&1&0&0&0&0\\
1&0&0&1&0&0&0&1&1&1&1&0&1&0&1&1&0&1\\
0&1&0&0&1&1&1&0&0&1&1&0&0&1&1&0&1&0\\
0&0&1&0&1&0&0&1&1&0&1&1&0&1&0&0&1&1\\
1&0&0&1&1&1&1&1&0&0&1&0&0&1&0&0&1&1\\
1&1&1&1&0&0&0&0&0&1&0&0&1&1&1&0&0&1\\
0&0&1&0&0&0&0&1&1&0&0&1&1&0&1&0&0&0\\
0&0&0&1&0&1&1&0&0&1&0&1&1&1&0&1&1&1\\
1&0&1&0&1&0&1&0&1&0&1&1&0&1&0&0&1&0\\
0&0&0&0&0&1&0&0&1&1&0&0&1&0&1&1&0&0\\
0&0&0&1&1&1&1&1&0&1&1&0&0&1&0&0&1&1\\
0&1&1&0&0&0&1&0&1&0&1&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&1&1&0&0&0&1&1&0&0&0&0\\
1&0&0&1&0&0&0&1&1&1&1&0&1&0&1&1&0&1
\end{bmatrix} \qquad (x)
$$

$$\boldsymbol{R} = [4 \quad 7 \quad 4 \quad 10 \quad 12 \quad 1 \quad 13 \quad 2]^t$$

SATELLITE 2 (V_{2jk})
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.999 0.000 0.000
0.000 0.000 0.000 0.002 0.000 0.999 0.000 0.000 0.000

SATELLITE 3 (V_{3jk})
0.004 0.000 0.000 0.000 0.000 0.999 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.999
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

SATELLITE 4 (V_{4jk})
0.000 0.000 0.999 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.004 0.000
0.999 0.000 0.000 0.003 0.000 0.000 0.000 0.000 0.000

$$U = [2 \quad 2 \quad 2 \quad 2]^t.$$

Both solutions are legal, and sufficient to meet the requested time slots, i.e., $\boldsymbol{U} \geq \boldsymbol{R}$. Solution 1 allocates more time slots than requested because w_0 is larger, i.e., more emphasis is placed to maximize the capacity.

Example 3: Consider the SBS problem with four satellites, three ground terminals and nine time slots. Constraint (4) is defined by the Associate Matrix A and the requested broadcasting time for each satellite is defined by $\boldsymbol{R}$ as follows

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

$$\boldsymbol{R} = [9 \quad 8 \quad 7 \quad 6]^t$$

$$w_0 = 0.3, w_1 = 2.00, w_2 = 2.00, w_3 = 0.2$$

SOLUTION

SATELLITE 1 (V_{1jk})
0.000 1.000 0.000 1.000 0.000 1.000 1.000 0.000 1.000
0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.001 0.000 0.000 0.000 0.000

SATELLITE 2 (V_{2jk})
1.000 0.002 0.002 0.001 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 1.000 1.000 0.000
0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000 1.000

SATELLITE 3 (V_{3jk})
0.000 0.000 0.003 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000
0.000 0.000 0.000 0.000 1.000 0.000 1.000 1.000 0.000

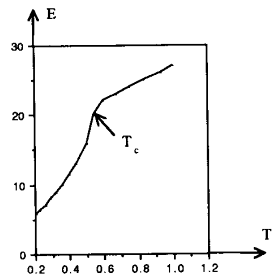

Fig. 10. System energy at different temperatures.

SATELLITE 4 (V_{4jk})
0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
1.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000
$U = [6 \quad 5 \quad 4 \quad 3]^t.$

In this example, the requested time slots are more than the system can allocate. Though the system cannot meet the request, it provides the maximum under the given constraints. Fig. 10 shows the system energy at different temperatures. The experiment shows that the critical temperature is 0.51 at which a remarkable transition of system (a deep drop of the system energy) takes place. This agrees quite well with the one computed in Section IV–B.

Example 4: A larger size problem: Consider the SBS problem with eight satellites, six ground terminals and eighteen time slots. $\boldsymbol{A}$ and $\boldsymbol{R}$ defined in (x) (see preceding page) correspond to the associative matrix and the requested broadcasting time for the satellites, respectively. Note that the network consists of 864 neurons of which 400 are active due to clamping.

With $w_0 = 0.3, w_1 = 2.00, w_2 = 2.00, w_3 = 0.4$, the system produces the following solution which meets all the constraints and the requested broadcasting time.

$$U = [4 \quad 7 \quad 4 \quad 11 \quad 12 \quad 1 \quad 13 \quad 2]^t$$

See satellites 1–8 at the bottom of this page and on the next page.

C. Summary of Results

Consistent results have been obtained for problems of larger sizes at a higher cost of computation. Table I summarizes the computational load of the above examples in terms of the average number of iterations required to reach the optimal solutions using the same weights but with varying initial conditions. Each example was run 200 times using random seeds for different initial conditions. Though the neurons may not converge to the exact values for different initial

SATELLITE 1 (V_{1jk})

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0

SATELLITE 2 (V_{2jk})

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

SATELLITE 3 (V_{3jk})

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

SATELLITE 4 (V_{4jk})

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.9	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.9	0.0	0.9	0.0	0.0	0.0	0.9	0.0	0.0	0.9

SATELLITE 5 (V_{5jk})

1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

SATELLITE 6 (V_{6jk})

0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

SATELLITE 7 (V_{7jk})

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

SATELLITE 8 (V_{8jk})

0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.9	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE I
SUMMARY OF RESULTS

Problems		Average number of iterations to reach steady state	Average number of sweeps per iteration
Example 2	Solution 1	23	21
	Solution 2	31	22
Example 3		3	27
Example 4		30	200

conditions, they are consistently either very close to 1 or 0. It is intuitive that a larger problem requires more computation as demonstrated in the table.

VI. CONCLUSION

In this paper, we have presented a new method to solve the satellite broadcast scheduling problem. The problem was first mapped onto a neural network from which an energy function is derived. Optimization is achieved by minimizing the energy by mean field annealing. Our key contributions include:

1) Formulate an appropriate energy function.
2) Introduce the clamping technique, and thus reduce the computation.
3) Derive the estimated critical temperature of the algorithm.
4) Discuss and suggest alternatives to avoid the divergence of the numerical implementation of the proposed method.
5) Demonstrate the robustness of our method by having achieved good solutions for problems of various sizes.

As compared to the previous method [1], [2], our method excels in the following ways:

1) Our method using MFT is parallel and global in scope, thus achieving good performance and computational efficiency.
2) Our method does not need to specify a set of distinct priorities for the satellites to broadcast, and no assumption is made on requiring a set of suitable requests.

In conclusion, MFT has been demonstrated to be an effective and robust optimization technique in solving the SBS problem.

REFERENCES

[1] P. Bourret, S. Goodall, and M. Samuelides, "Optimal scheduling competitive activation: Application to the satellite antennas scheduling problem," in *Proc. IJCNN '89*, 1989, pp. I565–572.

[2] P. Bourret, F. Rem, and S. Goodall, "A special purpose neural network for scheduling satellite broadcasting times," in *Proc. IJCNN '90*, 1990, pp. II535–538.

[3] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proc. Nat. Acad. Sci.*, vol. 81, 1984, pp. 3088–3092.

[4] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybernetics*, vol. 52, pp. 141–152, 1985.

[5] C. Peterson, "A mean field theory learning algorithm for neural networks," *Complex Syst.*, vol. 1, pp. 995–1019, 1987.

[6] C. Peterson, "Neural networks and NP-Complete optimization problem: a performances study on the graph bisection problem," *Complex Syst.*, vol. 2, pp. 59–89, 1988.

[7] S. Kirpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Sci.*, vol. 220, pp. 671–680, 1983.

[8] G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing, Vol. 1: Foundations*, Rumelhart and McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 282–317.

[9] P. J. M. Van Laarhoven and E. H. L. Arts, *Simulated Annealing: Theory and Applications*. Dordrecht, Holland: Reidel, 1987.

[10] C. Peterson, "A new method for mapping optimization problems onto neural networks," *Int. J. Neural Syst.*, vol. 1, no. 1, pp. 3–22, 1989.

[11] J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*. Cambridge, MA: MIT Press, 1988.

[12] N. Nilsson, *Learning Machines*. New York: McGraw-Hill, 1965.

[13] P. R. Staniforth, "Store-and-forward satellite communications system on UOSAT-2," *J. Institution of Electronic and Radio Eng.*, vol. 57, p. 43, Jan. 1987.

[14] W. L. Pritchard, H. G. Suyderhoud, and R. A. Nelson, *Satellite Communication Systems Engineering—2nd Edition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[15] P. A. Monte and A. E. Turner, "Constellation selection for GLOBALSTAR^{TM}, a global mobile communication system," *Proc. 14th Intl. Communication Satellite Systems Conference and Exhibit*, AIAA, Washington, D.C., pp. 1350–1360, Mar. 22–24, 1992.

[16] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison Wesley, 1984.

[17] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayessian restoration of images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 721–741, 1984.

[18] D. E. Goldberg, *Genetic Algorithm in Speech, Optimization & Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[19] C. Peterson and E. Hartman, "Explorations of the mean field theory learning algorithm," *Neural Networks*, vol. 2, pp. 475–494, 1989.

[20] G. Bilbro and R. Mann *et al.*, "Optimization by mean field annealing," in *Advances in Neural Information Processing System 1*, D. S. Touvetzky, Ed., 1989, pp. 91–98.

[21] Y. S. Abu-Mostafa and J. St. Jacques, "Information capacity of the Hopfield model," *IEEE Trans. Inform. Theory*, vol. 31, no. 4, pp. 461–464, July 1985.

[22] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. on Inform. Theory*, vol. 33, no. 4, pp. 461–482, July 1987.

[23] Y. Baram, "On the capacity of ternary Hebbian networks," *IEEE Trans. on Inform. Theory*, vol. 37, no. 3, pp. 528–534, May 1991.

[24] Y. Baram, "Encoding unique global minima in nested neural networks," *IEEE Trans. on Inform. Theory*, vol. 37, no. 4, pp. 1158–1162, July 1991.

[25] D. Y. Chao and D. T. Wang, "Enhancement of memory capacity of neural networks," in *Proc. 1992 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Raleigh, NC, July 7–10, 1992, pp. 519–526.

Nirwan Ansari (S'87–M'88–SM'94) received the B.S.E.E. degree from NJIT in 1982, the M.S.E.E. degree from the University of Michigan in 1983, and the Ph.D. degree from Purdue University in 1988.

In 1988, he joined the ECE Department at NJIT, where he is an Associate Professor and Assistant Chair for graduate studies. His current research interests include neural computing, pattern recognition, nonlinear signal processing, computer vision and intelligent networks.

Dr. Ansari serves as a referee, a session chair and/or organizer, and a technical representative for various major journals, conferences, and federal agencies. He publishes regularly in his areas of research and has co-edited a book *Neural Networks in Telecommunications* (Kluwer Academic Publishers, 1994).

Edwin S. H. Hou (S'88–M'89) received two B.S. degrees (magna cum laude) in electrical engineering and computer engineering from the University of Michigan, Ann Arbor, in 1982, the M.S. degree in computer science from Stanford University, Stanford, CA, in 1984, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1989.

He is an Assistant Professor in the Electrical and Computer Engineering Department and the Assistant Director in the Electronic Imaging Center at NJIT, Newark, NJ. His current research interests include infrared imaging, genetic algorithms, scheduling, and neural networks.

Dr. Hou was the Local Arrangement Chairman for the 1993 IEEE Regional Conference on Control Systems.

Youyi Yu received the B.S. degree in electrical engineering from Shanghai Science and Technology University, in 1983 and the M.S. degree in electrical engineering from NJIT, Newark, in 1992.

From 1972 to 1978 he worked as an Assistant Engineer at Shanghai No. 640 Institute, Shanghai P.R.C. He was appointed Assistant Professor in the Engineering College of Shanghai University from 1983 to 1990. In January 1993, he became a Software Engineer with Opencom Systems Inc., Piscataway, NJ. His current research interests include digital communications and signal processing.