

In Section 5 we introduce an alternate NN model in which a neuron is defined for each link of each path between every SD pair. Our studies demonstrate the ability of this NN model to provide the excitatory connections that are needed to activate complete paths.

We then turn our attention to the problem of link-activation scheduling. In Section 6 we define the two versions of the scheduling problem that are addressed in this report, i.e., the nonsequential- and sequential-activation problems. Scheduling conflicts are defined, and the multihop radio network that has served as the testing ground for our simulations is introduced.

In Section 7 we present lower bounds on the length of schedules that satisfy communication requirements for both versions of the problem, as well as heuristics for the determination of short (although not necessarily minimum-length) schedules. These bounds and heuristics are helpful in assessing the performance of the NN model.

In Section 8 we define our NN model for the scheduling problem. Constraints are established that reflect the desired behavior of the NN, i.e., the generation of schedules of minimum length. As in the routing problem, these constraints are expressed in the form of energy terms in the Lyapunov energy function, thus permitting the determination of the corresponding connection weights and bias currents, which in turn leads to the equations of motion that characterize system evolution. Again, a key feature of our model, which has been crucial to its high degree of success, is the use of the method of Lagrange multipliers to dynamically vary the connection weights as the system state evolves. Several variations of the NN model are discussed.

In Section 9 we discuss issues associated with the simulation process. These include the generation of initial NN states, the interpretation of system state, termination criteria, and methods to evaluate performance.

In Section 10 we discuss simulation results for the nonsequential-activation scheduling (NAS) problem. The ability of our NN model to find optimum schedules for a number of diverse problem instances, including highly constrained ones, is demonstrated by simulating several variations of the model. These simulations also reveal that the NN model is relatively insensitive to variations in the parameter values and the communication requirements. Modifications that further enhance the NN performance are developed and evaluated.

In Section 11 we discuss simulation results for the sequential-activation scheduling (SAS) problem. Simulations of our SAS NN model indicate that the SAS problem is significantly more difficult than the NAS problem. Modifications to the model, based on conventional heuristic methods, are implemented to give a model that generates optimal, or nearly optimal, schedules for a number of different problem instances. The length of the optimum schedule is difficult to calculate, and, for several problem instances, has only been determined by the NN's generation of a schedule whose length matches a known lower bound.

In Section 12 we address the joint routing-scheduling problem. We demonstrate that the problems of routing and scheduling are not separable. Therefore, it would be desirable to develop a NN model (or other heuristic approach) that incorporates the interactions between routing and scheduling. We outline the components of a NN model for this problem; however, the resulting model is extremely complex, except for small problems, and has not been implemented.

Finally, in Section 13 we present our conclusions from this study.

2.0 ROUTING AND SCHEDULING PROBLEMS IN PACKET RADIO NETWORKS

The basic problem that motivated the work described in this report is the rather fundamental, and yet quite neglected, property of multihop radio networks that couples the problems of channel access and routing. Here, we are interested in networks with point-to-point

communication requirements, i.e., networks that support the delivery of traffic between specific source-destination (SD) pairs over multihop paths.* In all types of communication networks, it has been a common practice to break up the enormous total network design problem into subproblems, each of which can be studied in isolation. This partitioning usually follows the "layered" structure of the Open Systems Interconnection (OSI) architecture (see e.g., Ref. 21). Thus, even though it is recognized that problems, issues, and design choices that reside in separate layers are, in fact, interdependent, they are addressed separately; only at the final "integration" stage is there an occasional attempt to recognize their influence on each other.

However, it is increasingly being recognized that in certain cases the interaction between two or more factors from different layers may be so fundamental and strong that their joint effects must be studied simultaneously. One such case arises in multihop radio networks. In such networks there is a clear need to maintain and update routing tables for point-to-point traffic (a layer-3 OSI issue) and, at the same time, to resolve the multiple-access contention for the channel resource among neighboring node terminals (a sublayer of layer-2 issue). If the routing tables direct a lot of traffic through a portion of the network that is shared by many nodes, the broadcast nature of the radio medium will force the use of a channel-access mechanism. This could introduce substantially more delay in the overall end-to-end transmission process than an alternate set of routes through sparser portions of the network would, even though those routes might be longer. Thus what is best for a given radio network, as far as routing is concerned, depends on the channel-access protocol that is used; it need not be the same as for a nonradio, "wire"-linked network of the same topology in which there is no issue of channel access.

It is important to distinguish between the two major philosophies of channel access mechanisms. These are (i) contention-based, and (ii) scheduled transmissions. In the first category we have all variants of ALOHA, Carrier-Sense Multiple Access (CSMA), Conflict Resolution Algorithms, etc.; in the second category we have basically contention-free schemes such as Time-Division Multiple Access (TDMA), Frequency-Division Multiple Access (FDMA), orthogonal Code-Division Multiple Access (CDMA), and reservation-based methods. Also, many hybrids of these schemes have been developed in recent years. Studies have shown that in single-hop applications, contention-based schemes perform well when traffic is bursty and traffic rates are low. Scheduled-access schemes perform well when traffic patterns are regular, e.g., periodic. However, it is difficult to make definitive conclusions on the performance of channel-access schemes in multihop environments.

Contention-based schemes suffer from the possibility of excessively long and unpredictable delays, especially during surges in the volume of traffic. To some degree, by using strictly controlled forms of contention protocols these disadvantages can be mitigated. However in multihop radio environments, the effects of contention can multiply rapidly across the network, and may be difficult to control. In fact, the only versions of controlled contention protocols that have been studied concern single-hop networks. For this reason, and although we do not rule them out for later consideration, we exclude such protocols from our further investigations in this report. Many of the issues associated with channel-access methods in multihop radio networks were discussed in greater detail in Ref. 8, where it was concluded that contention-free channel access methods are best for most broadcast networks. It was not possible to reach a definitive conclusion for point-to-point networks, such as those being considered here, and the question of which approach is preferable remains controversial. However, based on the considerations discussed above, the use of scheduled channel access appears to be a reasonable approach to this problem.

Therefore, we choose to focus on scheduled transmissions as the channel-access mechanism in this report. In particular, we consider a time-division implementation (although it is possible to introduce a limited degree of nonorthogonal CDMA in our approach to permit the simultaneous activation of neighboring links). The main virtue of time-division-based scheduled

*This report does not address broadcast networks in which the same information is to be delivered to all network members.

transmission protocols is that they are conceptually more attractive than the equivalent forms of their frequency- or code-based counterparts. Also, although they are not necessarily better than the contention-based ones in terms of performance (the unresolved issue just discussed), their performance *can* be assessed. The conceptual attractiveness of the time domain lies mainly in the natural and fundamental features of time and the sequential form of transmission control.

Thus we consider now the fact that the choice of routes for point-to-point traffic in multihop radio networks interacts with the choice of transmission schedules in each portion of the network where neighboring nodes must multiplex their transmissions in time. For example, we may consider a system in which the routes are specified in advance. In this case, the choice of routes determines the amount of traffic that must be carried over each of the network's links, and thus determines the communication requirements that must be satisfied by the channel-access mechanism. Alternatively, we may consider a system in which the schedules are specified in advance; in this case the problem becomes the determination of routes that use the predefined schedules. The capacity of a link is then proportional to the number of times the link is activated in one complete cycle of the schedule. The resulting set of link capacities can then be used as the basis for the determination of an optimal set of routes. Both of these approaches assume that something (i.e., either the routes or a transmission schedule) is specified in advance. In general, these problems are not separable. Thus nonoptimal solutions are obtained by attempting to solve them separately. In the true routing-scheduling problem, neither is specified a priori; both are to be determined by the optimization process.

Despite the intimate relationships that exist between these network control mechanisms, they are almost invariably addressed separately, resulting in network operation that may be far from optimal. As stated earlier, the recognition of the importance of the coupling between these two problems is relatively recent. In fact, the problem of best-schedule determination alone (without considering the effect of routing) was only recently studied by a number of authors [1, 7], and it was determined that, in almost all of its forms, it is a highly complex combinatorial-optimization problem. It has been referred to as the pure scheduling problem. If, for example, each node has a single transceiver and if a single frequency is used across the network (or alternatively if nonorthogonal CDMA codes are used), the pure scheduling problem is indeed NP-complete. In complexity theory, the term NP-complete describes the property that there is no known algorithm of polynomial complexity that can solve the problem. Reference 2 provides more details on the complexity of other forms of the pure scheduling problem.

Some instances of the pure scheduling problem are equivalent to the well-known problem of graph coloring or finding matching sets of nodes (or links) in a graph. Both are well understood and have been extensively studied in complexity theory. For example, a pure scheduling problem can be posed as follows: let f_i be the average traffic flow rate on link i , which is given as a result of a separate solution of the routing problem. We would like to find a schedule, i.e., a set of pairs (T_j, τ_j) , $j = 1, \dots, N$, where T_j is a set of links that can be activated simultaneously without violating interference constraints* and τ_j is the number of slots (or, simply, the total amount of time) for which the links in T_j are allowed to transmit. The length of the schedule is defined as

$$\ell = \sum_{j=1}^N \tau_j.$$

The problem is to find the schedule with minimum length such that the given flows f_i can be accommodated in the network. A solution to this problem was provided by Hajek and Sasaki [1], and further studied by Tassiulas and Ephremides [3]. The solution in Ref. 1 is not practical and

*Interference constraints can be variously defined, depending on the number of transceivers at each node, the form of signaling used (e.g., CDMA), etc. In their plainest form they require simply that no two links adjacent to the same node be allowed to transmit simultaneously.

has only academic value (although it is crucially important to the subsequent development of the joint routing-scheduling problem).

Thus, an effort was made to develop a heuristic solution that is based on a Hopfield neural network (NN) [9]. The inspiration to do so came from the observation by Hopfield and Tank [22] that combinatorial-optimization problems similar to the famous Traveling Salesman Problem (see Appendix A) could be solved efficiently by means of NNs of a certain special form. The results in Ref. 9 were encouraging, and pointed us in the direction of using NNs for the overall problem of joint routing and scheduling. In addition to the reasonably satisfactory performance results reported for the pure scheduling problem, the nature of the algorithm that the Hopfield NN implemented in that instance was such that it could be amenable to distributed implementation. We especially note that distributed implementations of algorithms for routing and scheduling in radio nets are very important and desirable. Of course, not all algorithms can be implemented distributedly. In fact, the ones that are described in this report are not. Distributed implementation remains an important goal for scheduling problems. Although distributed algorithms have been developed for scheduling, the development of an optimal distributed algorithm remains an elusive goal.

The next step in considering the joint optimization problems of choosing both the schedule of transmissions and the routes for the traffic was defining the network evacuation problem. This problem considers an initial amount of information residing at each node of the network that needs to be delivered to a single, common destination. The desire is again to find a schedule (as defined earlier) that accomplishes this delivery in minimum time. Implicit in the definition of this problem is the selection of routes and their interaction with the transmission schedules. This problem was studied in detail by Tassiulas and Ephremides [3].

It turns out that this problem is not as restricted as it may seem at first; it is equivalent to the problem of sustained operation under steady (nonrandom) traffic flow generation at the source nodes. In other words, the specified traffic levels may be interpreted as periodic communication requirements (instead of simply quantities of traffic that must be eliminated from the network), in which case the schedule developed for the evacuation problem could simply be repeated periodically. It also turns out that the joint optimization decomposes into two separate problems that are weakly coupled, one of schedule optimization and one of flow optimization (i.e., routing). Specifically, the value of the length of the optimal schedule is known to be equal to the maximum nodal degree in the network, where the degree of a node is defined as the total amount of flow into that node plus the total amount of flow out of that node. Thus, minimizing the maximum degree by choice of the flows solves the routing part of the problem in a way that couples it to the scheduling problem.

This nice and encouraging result was based on a pivotal graph-theoretic observation first made by Hajek and Sasaki [1] and then put to use by Tassiulas and Ephremides [3]. However, the solution to the scheduling component of the problem remained as impractical as that in the original pure scheduling problem. A somewhat further discouraging observation is that extending the result to multiple destinations, although certainly possible, seems to be substantially more difficult and has not been accomplished to date. Also, this result is based on the assumption that all links have equal capacity. If they do not, new complications arise.

The formulation of the routing-scheduling problem that gave rise to even these limited results also contains another important simplification, namely that the traffic flow is a continuous variable and that the schedule lengths have arbitrary real values from a continuum rather than integer values (multiples of a slot length). The practical version of the problem in which the units of transmission are fixed-length packets has not been directly simplified in a manner analogous to the methods used by Hajek and Sasaki and Tassiulas and Ephremides for the nonquantized case.

Finally, the ultimate joint routing-scheduling problem requires consideration of random traffic generation patterns and not simply constant deterministic flows. Although this case lies

beyond the scope of our report, it has been considered in some of its simpler forms (single tandem topology, immediate neighbor destination, and some others) in Ref. 2 and gives rise to two types of questions. The first type relates to stability and simply asks whether a schedule exists such that the queues in the network do not increase without bound (for given average rates of exogenous [i.e., externally generated] traffic injected into the network). The second type relates to optimality in the sense of determining the schedule for which the usual weighted average delay performance measure for end-to-end delivery is minimized. Preliminary results have been obtained for both questions by L. Tassiulas in his recently completed Ph.D. dissertation at the University of Maryland. By and large, however, the case of random traffic inputs remains a very complex problem that is a subject of additional future research.

The conclusions that can be drawn at this point for the joint routing-scheduling problem are that the problem remains basically unsolved, although it has been "dented" appreciably at various corners. In this report we outline our effort to effect another dent at a corner that, so far, has remained untouched. Namely, instead of approaching the joint problem by first determining the schedule of transmissions and then introducing the routing component in it (which is the way the problem has been approached so far [1, 3] and has not yielded a successful resolution to the joint problem), we considered doing the reverse. So we proposed to look at the route selection problem first, and to bring in the scheduling aspect next. Of course, the plain routing problem has been extensively studied in the literature, and is considered basically solved. A plethora of algorithms and variants of them exist for determining good (or, indeed, optimal) routes under various conditions of changing environments and limited information. However in considering the routing problem here, we propose a version that already incorporates in some measure (albeit only implicitly) the role and effect of the scheduling component. Namely, we assume that routes that result in nodes with a high degree (i.e., routes that include nodes that are multiply shared by other routes) are aggravating the scheduling problem, and are bound to introduce longer scheduling delays. Therefore, we would like to strike a balance between routes that are "short," in the traditional sense of route quality,* and "disjoint" to the extent possible (i.e., sharing as few nodes as possible). We consider this to be a first step toward approaching the combined optimization problem from the "other end," namely that of routing.

As is seen in the next section in which the specifics of our routing model are introduced, we generally adopt the viewpoint that, for a given graph that represents a multihop radio network, each source-destination pair is assigned a prespecified set of possible routes. These routes mesh with each other in the graph, and each choice we make results in different numbers of shared nodes amongst them. Clearly, we prefer to choose routes that are short (the pure-routing component of the problem) *and* such that the number of shared nodes is small (which relates to the scheduling aspect). In this formulation the problem becomes one of combinatorial optimization. For a network with J source-destination pairs and K routes per pair, an exhaustive search would entail scanning through K^J possible solutions; e.g., for $K = 5$ and $J = 20$, a problem of moderate size, there are 9.5×10^{13} solutions. Although we have not formally proven it, we suspect strongly that this problem is NP-complete. Thus we are naturally led to considering a Hopfield neural network for its solution. Indeed, we have developed NN models for several versions of both the routing and the scheduling components of our problem. In fact, we believe we have exploited the full strength of the Hopfield NN approach by using several variations and improvements of the basic technique. In particular, we have obtained excellent results in large, heavily congested networks by using the method of Lagrange multipliers to dynamically determine system parameters. We have also investigated the use of simulated annealing and a variety of heuristic methods to improve performance. Appendix A provides background material on these methods. In the following sections we describe in detail the models we have developed and their performance analysis and evaluation.

*Usually, the "length" of a link in routing problems reflects a measure of message delay on that link that incorporates propagation, transmission, processing, and waiting times. Frequently it is taken to be a constant, in which case we refer to the problem as "minimum number of hops" routing.

3.0 A HOPFIELD NETWORK TO MINIMIZE CONGESTION

We have found it advantageous to consider separately the routing and scheduling problems before implementing a neural network (NN) for the complete joint scheduling-routing problem. Although these problems are not independent, addressing them separately is expected to provide reasonably good performance and to provide insight into an eventual design of a NN for the combined problem. Toward this end, we have implemented a NN simulation that chooses routes based on the criterion of minimizing congestion, and that indirectly takes into account the factor of link scheduling effects. Once these routes are chosen, schedules can be generated either by using NN methods that are applicable to pure scheduling problems as are discussed in Sections 8 - 11, or by means of some (preferably distributed) heuristic such as those developed in Ref. 23.

In this section, we present a Hopfield NN model for the selection of paths between several source-destination (SD) pairs in a packet radio network; in Section 4 we demonstrate its effectiveness in large, heavily-congested networks. We start by presenting the basic energy function and equations of motion that we have developed for the NN model of this problem. Then, in the course of discussing the simulation process, we explain many of the design issues that have been encountered and the techniques we have used to improve performance. For the reader who is not familiar with Hopfield NNs, we strongly recommend that Appendix A, which provides a discussion of the use of Hopfield NNs for combinatorial-optimization problems, be read at this point to provide the necessary background material for this section.

3.1 The Problem

Given the connectivity graph of a radio communication network, a set of N_{sd} SD pairs, and a set of $N_p(i)$ paths connecting SD pair i ($1 \leq i \leq N_{sd}$), select a single path between each SD pair so that network congestion, as defined in Section 3.3, is minimized. Minimizing congestion encourages the activation of paths that do not share many common nodes, and thus reduces the delay effect that the interference-free scheduling of the link activation induces. For simplicity, we first consider a network in which equal traffic is specified between each SD pair. The assumption that paths are predefined is often reasonable and corresponds to the prespecification of virtual circuits that may be activated as needed.

The number of admissible solutions (i.e., unique sets consisting of one path between each of the N_{sd} SD pairs) is easily seen to be

$$\prod_{i=1}^{N_{sd}} N_p(i).$$

For the special case in which $N_p(i) = N_p$ for all i (i.e., the same number of paths are defined for each SD pair), the number of unique admissible solutions is $(N_p)^{N_{sd}}$. Thus, in this case, the number of admissible solutions grows exponentially with the number of SD pairs, making exhaustive search impractical in large examples. We strongly suspect that this problem is NP-complete, although we have not formally verified this property.

3.2 Neural Network Model

The first step in formulating a Hopfield NN model is defining neurons that correspond to binary variables in the system that is being modeled. In this section, we consider a Hopfield NN in which one neuron is defined for each path between every SD pair.* For example, Fig. 1(a)

* Alternate approaches are possible. In fact, in Section 5 we discuss a Hopfield NN formulation in which a neuron is defined for every link of each path between all SD pairs.

shows a very simple six-node network with two paths between each of two SD pairs, and Fig. 1(b) shows the corresponding path-neuron model. A double index is used to specify the neurons, i.e., neuron ij represents the j th path between SD pair i . The neurons are analog devices, which are characterized by an input-output relation that has the sigmoidal form

$$V_{ij} = \frac{1}{2} \left[1 + \tanh \left(\frac{u_{ij}}{u_o} \right) \right], \quad (1)$$

where u_{ij} and V_{ij} are the input and output voltages, respectively, of neuron ij , and u_o is a parameter that governs the slope of the nonlinearity. Since in every admissible solution one path is chosen for each SD pair, exactly one of the V_{ij} 's is equal to 1 for every value of i (which means that the corresponding path is chosen), and the others are 0. In practice, since analog neurons are used, an admissible solution will have one neuron per SD pair with an output voltage value close to 1, while the others will be close to 0.

A key feature of these networks is, in fact, the analog nature of these processing elements, which permits embedding discrete problems in a continuous solution space. Permitting the search for an optimal solution to proceed through the interior of a continuous region yields better solutions than are possible with strictly digital processing elements and determines them very rapidly when the NN is implemented in hardware [22].

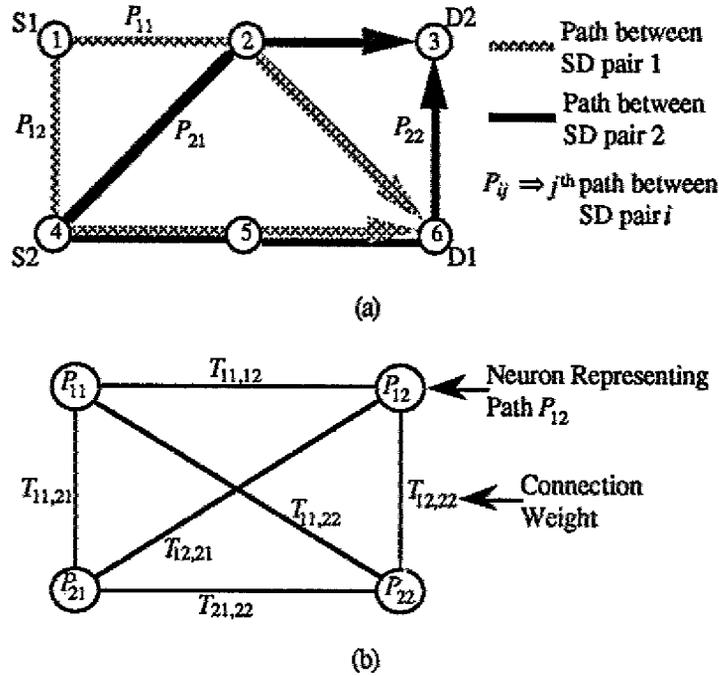


Fig. 1 — An example network: (a) shows a six-node communication network; (b) shows the corresponding path-neuron NN model

Connections are established between all pairs of neurons. The NN evolves from some initial state to a final state that represents a local (but not necessarily global) minimum of the Lyapunov energy function, which may be written in terms of the connection weights, bias currents, and neuron output voltages as

$$E_{total} = -\frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{k=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{l=1}^{N_p(k)} T_{ij,kl} V_{ij} V_{kl} - \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} V_{ij} I_{ij}, \quad (2)$$

where $N_p(i)$ is the number of paths between SD pair i . System evolution follows a trajectory of monotonically decreasing energy, E_{total} . In our model, the connection weights are symmetric (i.e., $T_{ij,kl} = T_{kl,ij}$); thus, in Fig. 1(b) it is sufficient to show a single connection weight to represent the interaction between each pair of neurons. The total number of neurons N is given by

$$N = \sum_{i=1}^{N_{sd}} N_p(i).$$

Thus an $N \times N$ connectivity matrix T can be defined, whose elements are the connection weights $T_{ij,kl}$. Convergence to a stable state is guaranteed as long as the connections are symmetric [22], a condition satisfied by our model. Except for some of our early examples, reliable convergence to admissible states (i.e., states in which the constraints are satisfied) has, in fact, been achieved. In our problem, as will be discussed later, the strengths of these connections are chosen to discourage the sharing of nodes by many paths (i.e., limit congestion), while encouraging the correct number of path activations (i.e., exactly one neuron turned on per source-destination pair).

The NN is "programmed" by implementing the set of connection weights and bias currents that correspond to the function that is to be minimized. An analog hardware implementation of a Hopfield NN will normally converge to its final state within at most a few RC time constants, thus providing an extremely rapid solution to a complex optimization problem. In our studies (as in most studies of this technique) we have simulated the system dynamics in software. Although such software solutions are extremely time consuming, they verify the soundness of the use of the Hopfield NN approach for optimization problems of this type and suggest that hardware implementations may be worthwhile.

3.3 Congestion Energy

The class of objective functions that can be modeled by Hopfield NNs is normally limited to those that can be expressed in the form of Eq. (2). This class includes weighted sums of the products of pairs of neuron output voltages as well as output voltages taken individually.

For the case of continuous traffic and for a certain class of network topologies,* Hajek and Sasaki [1] have shown that the selection of paths that minimize the maximum nodal degree (where the degree of a node is defined to be the sum of all flows into the node plus all flows out of the node) in the network permits the generation of schedules of minimum length. Clearly, this performance measure cannot be put in the form of the desired Lyapunov energy function. Instead, we have chosen to minimize the following measure of congestion, which is in the form of Eq. (2), as required:

$$E_b = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{k=1}^{N_{sd}} \sum_{\substack{j=1 \\ k \neq i}}^{N_p(i)} \sum_{l=1}^{N_p(k)} |P_{ij} \cap P_{kl}| V_{ij} V_{kl},$$

where

P_{ij} is the j th path between SD pair i , and

$|P_{ij} \cap P_{kl}|$ is the number of nodes shared by paths P_{ij} and P_{kl} .

To facilitate the interpretation of this "congestion-energy" term, we first assume that an admissible state has been reached. In this case, the congestion energy corresponds to the sum of the number

* Although the class of topologies for which this property holds initially appears restrictive, Tassiulas and Ephremides [3, 24] have shown that it can easily be generalized to a broader class of networks.

of common nodes of all selected paths (one for each SD pair), taken on a pairwise basis. Minimizing this congestion energy loosely corresponds to selecting a set of paths that may be scheduled in a minimal number of slots. Before convergence is reached, the neuron output voltages take on values in the continuum $[0,1]$. The congestion energy is the weighted sum of the number of common nodes of all pairs of paths for different SD pairs in the network, where the weights are the products of the corresponding neuron pair output voltages. As the system converges to an admissible state, the output voltage of exactly one neuron per SD pair approaches 1 while that of the others approaches 0; thus the congestion energy approaches that of an admissible state in which one path is chosen per SD pair, as described above.

Note that E_b is actually minimized when all neuron output voltages are zero, which corresponds to a state in which no paths are activated. The tendency of the E_b term to turn off all of the neurons is manifested by contributions to the connection weight matrix that are purely inhibitory and whose strengths are proportional to the number of shared nodes in the two paths. Thus constraints, which are discussed in the next subsection, are needed to ensure that the correct number of neurons is activated.

By incorporating the constraints into the desired objective function, the energy function for the minimization of congestion assumes the form

$$E_{total} = bE_b + \sum_{c=1}^C \lambda_c E_c - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} V_{ij}. \quad (3)$$

The first term represents the network congestion, and is the function that we would like to minimize, as just discussed. The second term represents the impact of the C system constraints, each of which adds zero energy when the corresponding equality constraint is satisfied and a positive state-dependent energy when it is not. These equality constraints are incorporated into the objective function by using the classical approach of Lagrange multipliers.

The most critical issue in the design and simulation of a Hopfield NN model is the choice of the coefficients used in the connection weights. In most studies of Hopfield NNs, the values of the λ_c 's have been assumed to be constants, whose best values are typically determined by trial and error in software simulations. In Section 4.7 we exploit the full power of the Lagrange multiplier method by permitting the λ_c 's to evolve along with the system state. The last term of Eq. (3) represents the impact of additional bias currents, which in our problem formulation are applied equally to all neurons to help satisfy system constraints. The coefficients b , λ_c , and I are all positive.

Connection weights and bias currents are determined by transforming this problem-specific form of the energy function into the generic form given in Eq. (2). This problem formulation is quite similar to the one developed by Hopfield and Tank for the Traveling Salesman Problem (TSP) [22]. Implementation of the system parameters defined in this manner results in a NN that follows a trajectory over which the energy function E_{total} decreases monotonically until a local minimum is found. We have observed, as have other studies of Hopfield NN models, that such equilibrium points often correspond to low-energy values of the desired objective function E_b as well.

3.4 Incorporation of Constraints into the Energy Function

The problem constraints and the corresponding terms in the energy equation (which must each be equal to zero when the constraints are satisfied) are summarized as follows.

1. *Activate (select) no more than one path per SD pair:*

$$E_1 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{\substack{l=1 \\ l \neq j}}^{N_p(i)} V_{ij} V_{il} = 0.$$

This term provides a positive contribution to the energy function whenever two or more paths between the same SD pair have nonzero voltage. It represents purely inhibitory contributions to the connection weights.

2. *Activate a total of exactly N_{sd} paths in the network:*

$$E_2 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} V_{ij} - N_{sd} \right)^2 = 0.$$

This term is zero when exactly N_{sd} neurons have output voltage values of 1. Its effect is excitatory if an insufficient number of neurons is active and inhibitory if too many are active.

3. *Activate exactly one path per SD pair:*

$$E_3 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \left(\sum_{j=1}^{N_p(i)} V_{ij} - 1 \right)^2 = 0.$$

This term vanishes whenever exactly one path is chosen for each SD pair. Like constraint 2, it can be either excitatory or inhibitory. Although this constraint appears to be redundant (because satisfying the first two constraints would guarantee that it is satisfied as well), its inclusion in the energy equation is helpful in achieving convergence to admissible solutions. The use of such seemingly redundant constraints is common in Hopfield network models. Satisfying constraint 3 alone (along with a mechanism to guarantee that all nodes take on binary values) would actually be sufficient for our problem. However, constraint 2 is useful because it imposes a greater penalty when an incorrect number of neurons in the entire NN are set to 1; this is because it is a quadratic form centered about N_{sd} , whereas constraint 3 contains N_{sd} quadratic forms each centered about 1.

Since the neurons are analog devices whose output voltages take on values in the continuum between 0 and 1, these constraints cannot be satisfied simultaneously until, and unless, a state is reached in which all output voltages take on binary values. It is possible for constraints 2 and 3 to be satisfied by a state in which more than N_{sd} neurons are partially active (i.e., have output values less than 1). This is why constraint 1 is needed to discourage the (even partial) activation of more than one neuron per SD pair; it is also the reason that, in the neuron input/output relationship, a relatively steep nonlinearity is used to force the neuron output voltages toward binary values. Thus, although the system evolves through the interior of an N -dimensional hypercube, the incorporation of these constraints into the energy function encourages the system to evolve to "legal" or "admissible states," which are binary states in which the constraints are, in fact, satisfied. Whether or not convergence to admissible states is achieved depends on factors such as the λ_c coefficient values, the initial state of the system, the slope of the input-output nonlinearity, the time constants used in the iteration, etc. All of these factors are discussed later in this section and, in more detail, in Section 4.



NRL/FR/5521-91-9366

NRL 9366

The Problems of Routing and Scheduling in Multihop Radio Networks — A Hopfield Neural Network Approach

JEFFREY E. WIESELTHIER AND CRAIG M. BARNHART

*Communication Systems Branch
Information Technology Division*

ANTHONY EPHREMIDES

*Locus, Inc.
Alexandria, Virginia*

and

*University of Maryland
College Park, Maryland*

December 31, 1991

3.5 Determination of Connection Weights and Bias Currents

Substitution of the expressions for E_b and the E_c 's into Eq. (3) yields:

$$\begin{aligned}
 E_{total} = & \frac{b}{2} \sum_{i=1}^{N_{sd}} \sum_{\substack{k=1 \\ k \neq i}}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{l=1}^{N_p(k)} |P_{ij} \cap P_{kl}| V_{ij} V_{kl} + \frac{\lambda_1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{\substack{l=1 \\ l \neq j}}^{N_p(i)} V_{ij} V_{il} \\
 & + \frac{\lambda_2}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} V_{ij} - N_{sd} \right)^2 + \frac{\lambda_3}{2} \sum_{i=1}^{N_{sd}} \left(\sum_{j=1}^{N_p(i)} V_{ij} - 1 \right)^2 - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} V_{ij}. \quad (4)
 \end{aligned}$$

To determine the connection weights, we compare Eq. (4) with the generic form given in Eq. (2). The energy function contains both quadratic and linear terms. The coefficients of the quadratic terms, which involve products of the form $V_{ij}V_{kl}$, correspond to connection weights of the form $T_{ij,kl}$. Thus the connection weight $T_{ij,kl}$ is the sum of all coefficients that multiply the product $V_{ij}V_{kl}$ in Eq. (4):

$$T_{ij,kl} = -b |P_{ij} \cap P_{kl}| (1 - \delta_{ik}) - \lambda_1 \delta_{ik} (1 - \delta_{jl}) - \lambda_2 - \lambda_3 \delta_{ik},$$

where δ_{ik} (which equals 1 if $i = k$ and 0 otherwise) is the Kronecker delta symbol.

Similarly, the coefficients of the linear terms, which involve the V_{ij} 's one at a time, correspond to the bias currents. Thus I_{ij} is the sum of the coefficients that multiply V_{ij} .

In our simulation studies we have observed that an insufficient number of neurons are typically activated, a problem that can be mitigated by increasing the bias currents. Hopfield and Tank observed the same behavior in their studies of the TSP [22]. The need for additional bias currents stems, at least in part, from the purely inhibitory nature of the congestion energy's contribution to the connection matrix T . Thus additional bias current is needed to, in effect, adjust the neutral positions of the amplifiers. We have incorporated the additional bias current into constraint 2, which can now be expressed as

$$E_2' = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} V_{ij} - \alpha N_{sd} \right)^2 = 0.$$

Setting the parameter α to a value greater than 1 provides the additional excitation bias. A typical value that we have used is $\alpha = 1.5$, which is equal to the one used by Hopfield and Tank [22] in their solution of the TSP. Incorporating the additional bias currents in this manner permits us to set $I = 0$ in Eq. (3). The resultant expression for bias currents is

$$I_{ij} = \lambda_2 \alpha N_{sd} + \lambda_3.$$

3.6 Equations of Motion

An equation characterizing the evolution of the input voltage at each neuron can be obtained by examining the circuit diagram shown in Fig. A2. We have

$$\frac{du_{ij}}{dt} = -\frac{u_{ij}}{\tau} + \sum_{k=1}^{N_{sd}} \sum_{l=1}^{N_p(k)} T_{ij,kl} V_{kl} + I_{ij},$$

where $\tau = RC$ (which may be set equal to 1 without loss of generality) is the time constant of the RC circuit connected to the neuron. This relationship can be expressed in terms of the energy function as follows:

$$\frac{du_{ij}}{dt} = -\frac{\partial E_{total}}{\partial V_{ij}} - \frac{u_{ij}}{\tau}. \quad (5)$$

As the system evolves from an initial state, the energy function E_{total} decreases monotonically until equilibrium at a (local) minimum is reached. Since only a local minimum can be guaranteed, the final state depends on the initial state at which the system evolution is started. The equations of motion may be expressed in iterative form as

$$u_{ij}(t+\Delta t) = u_{ij}(t) - (\Delta t)u_{ij}(t) - (\Delta t)b \sum_{\substack{k=1 \\ k \neq i}}^{N_{sd}} \sum_{l=1}^{N_p(k)} |P_{ij} \cap P_{kl}| V_{kl} - (\Delta t)\lambda_1 \sum_{\substack{l=1 \\ l \neq j}}^{N_p(i)} V_{il} \\ - (\Delta t)\lambda_2 \left(\sum_{k=1}^{N_{sd}} \sum_{l=1}^{N_p(k)} V_{kl} - \alpha N_{sd} \right) - (\Delta t)\lambda_3 \left(\sum_{l=1}^{N_p(i)} V_{il} - 1 \right). \quad (6)$$

This form is customary and appropriate for computation. A number of issues arise when these equations are simulated in software. The most apparent is the need to choose the coefficients in the weight matrix, i.e., b and the λ_c 's. Also important is the bias current parameter α . Somewhat more subtle is the impact of the step size Δt and the nonlinearity parameter u_o . The ability of the state to converge to a good solution, or even to an admissible solution, depends strongly on these parameters.

4.0 PERFORMANCE EVALUATION USING THE PATH-NEURON MODEL

In this section we discuss the major issues associated with the simulation of our NN models, and we demonstrate the ability of this approach to find good sets of routes in large, heavily congested networks.

4.1 Basic Simulation Issues

The NN model described in Section 3 was simulated by using a program written in C++, which was run on Sun-3 and Sun-4 workstations. The program reads a file that lists the nodes traversed in each of the predefined paths. This information is used to build the NN model. The system parameters used in determining the coefficients in T and the bias currents are contained in a separate file to facilitate their modification as necessary. The process of "building" the NN is completely automated; the operator is only required to provide the file that lists the paths and, if desired, to edit the parameter file to adjust the system parameters. Thus, different problem instances or networks can be quickly and easily analyzed.

The initial input voltage to each neuron ij is set so that the output voltage is equal to the inverse of the number of paths between SD pair i . That is, to obtain an initial output voltage of $V_{ij}(0) = 1/N_p(i)$, we set

$$u_{ij}(0) = u_o \tanh^{-1} \left(\frac{2}{N_p(i)} - 1 \right).$$

A small perturbation is then added to the initial input voltage of each neuron; addition of this perturbation avoids the undesirable condition of a totally symmetric initial state [22]. The perturbation quantity is a random deviate drawn from a uniform distribution on $[-0.1u_o, 0.1u_o]$ and is different for each neuron.

The equations of motion are iterated, allowing the NN to “relax” to a minimal energy state. It is important to note that system evolution is deterministic. The only randomness in the model is that which is associated with the choice of the initial state.* For a particular set of parameter values, the NN was typically run from 100 different initial states. To obtain different initial states for a given problem instance,† different seeds are passed to a random number generator so that a unique sequence of random deviates is used to perturb the initial input voltages in each simulation. A particular initial condition can be reproduced by, once again, passing to the random number generator the seed that produced the original state. Since the system evolution follows a trajectory of monotonically decreasing energy, the initial condition determines which portion of the solution space is actually searched, and thus which final state is reached.

The iteration is terminated when all neuron output voltages are within some specified value ϵ of the output voltage limits 0 and 1 (event of convergence) or when a “time-out” is reached (event of no convergence by a specified number of iterations). If convergence is achieved, those neurons that have output voltages within ϵ of 1 are declared to be “on,” which means that their corresponding paths are activated. The remaining neurons are declared to be “off,” and their corresponding paths are not used. A convergence is “admissible” if the constraints are satisfied, i.e., if exactly one path is activated for every SD pair.

4.2 Algorithm for the Determination of Path Sets

The paths for each SD pair were generated via an algorithm that finds highly node-disjoint paths. This algorithm is based on the repeated application of Dijkstra’s shortest-path algorithm [25] as follows:

```

For each SD pair  $i$ , DO
{  Assign each node a weight of 1.
  Repeat  $iterations$  times:
    {  Use Dijkstra's shortest-path algorithm to select a minimum-weight path between
      SD pair  $i$ , where the weight of a path is the sum of the weights of the nodes
      in the path.
      If the selected path is redundant,**
      then discard it;
      else, list it.
    }
  Increase the weights of the nodes in the selected path by  $N_{nodes}$ , the number of
  nodes in the network.
}
}

```

Thus, the first path selected is a shortest-hop path, and all subsequent paths share the minimum possible number of nodes with previously chosen paths. In our examples we have used $iterations = 100$. Because duplicate paths are discarded, as are paths that contain previously discovered paths, the number of paths actually found may vary for each SD pair. The rationale for selecting highly node-disjoint paths is that their use tends to provide a large number of options for spreading the traffic among the network’s nodes without requiring the study of all paths between every SD pair. Clearly, there is no guarantee that all paths necessary for minimum-congestion

*Except for the case of using simulated annealing in conjunction with our NN model, as discussed in Sections 4.8.4 and 8.3.4, which does result in nondeterministic system evolution.

†The *problem* is routing to minimize congestion. An *instance* of the problem (a problem instance) is a completely specified example of the problem in which the network connectivity, the set of SD pairs, and the sets of path connecting the SD pairs are given.

**We define a redundant path to be a path that is either a duplicate of or contains a previously listed path. Note that the shortest-path algorithm will not select a path that contains a previously listed path because this type of redundant path will always have a larger weight than the shorter path it contains.

solutions will be found, even for arbitrarily large values of *iterations*. Also, our algorithm does not guarantee that a maximally node-disjoint set of paths is found.* However, our experience has shown that this approach provides an effective heuristic for the determination of a good set of candidate paths; this is discussed in Section 4.6.

4.3 A Binary Interpretation of the Analog State: An Instantaneous State Description

A special feature of our problem is that exactly one path (neuron) must be selected for each SD pair. This property can be exploited by declaring the neuron with the largest output voltage in each SD pair to be "on," regardless of its actual output voltage.† Ties (i.e., equal output voltages) can be broken arbitrarily, e.g., by choosing the lowest numbered neuron in such a set. Thus, at any time in the NN evolution, an admissible solution may be obtained from the analog system state by picking a binary state in this manner. We refer to this state as the "instantaneous" state of the system. Tracking the instantaneous state permits the observation of the set of chosen paths as it evolves over time. (The actual system evolution proceeds in the interior of the hypercube, however. This mapping from an analog to a binary state at each step of the iteration is simply for the purpose of assigning a binary interpretation to the state before convergence has been reached.)

We have found it advantageous to use this binary interpretation of the analog state to define a "binary-instantaneous congestion energy," which is the value of E_b that would correspond to the selection of the neurons chosen in this manner. Thus it is possible to track the evolution of the energy of the binary state chosen by the NN from the initial state until convergence is reached. Recall that although E_{total} decreases monotonically throughout system evolution, neither the actual nor the binary-instantaneous values of E_b are necessarily monotonically decreasing.

The evolution of these instantaneously admissible solutions reveals that the final state is usually reached relatively early in the simulation. Typically, many more iterations are then required for all neurons to reach values within ϵ of their binary values, and thereby allow termination based on the criterion specified earlier. This observation permits the simulation to be stopped when the set of chosen neurons does not change for a sufficient number of iterations (typically several hundred). Our experiments using Lagrange multipliers that evolve along with the system state (to be discussed in Section 4.7) have confirmed that, although the output voltage of a selected neuron may be significantly less than 0.5 when this criterion is satisfied, continued iteration beyond this point will eventually lead to a network in which that voltage very closely approaches 1. This early termination criterion has been used in all of the path-neuron simulation examples presented in this report.

4.4 Alternative Metrics

Although E_b is a reasonable metric to choose to minimize, there are also other metrics that reflect a measure of congestion. It is desirable to track some of these metrics as well. Unfortunately, certain types of desirable metrics cannot be implemented directly by using Hopfield NNs. For example, we noted in the previous section that under certain conditions, for which the routing and scheduling problems are separable, a minimum-length schedule can be achieved if the maximum nodal degree in the network is minimized. It is not possible to incorporate such a performance criterion into the Lyapunov energy function framework, which consists of weighted sums of the product of pairs of neuron output voltages and single neuron output voltages. A node-

*For example, the fact that the choice of the first path is limited to the set of shortest paths may make it impossible to find a pair of node-disjoint paths, whereas an algorithm that chooses them simultaneously (or that modifies the first path chosen as necessary) may be able to do so (see e.g., Ref. 26).

†This criterion was also used by Kamgar-Parsi et al. in their studies of the "clustering" problem [27, 28]. However, it can only be applied to certain types of problems. For example, it cannot be applied to the Traveling Salesman Problem, which requires that the set of neurons satisfy the permutation matrix condition, as discussed in Appendix A.

based performance measure cannot be directly mapped to the individual neurons in the network because each neuron corresponds to a complete path.

An alternate node-based metric that also penalizes heavily congested nodes is

$$\psi = \sum_{i=1}^N (\Omega(i) - 1)^2,$$

where

N is the number of nodes, and

$$\Omega(i) = \max \{ \text{number of paths that use node } i, 1 \}.$$

Thus nodes that do not belong to any chosen paths do not contribute to ψ . Our simulation results show that performance under the ψ criterion tracks quite well that based on the NN's natural criterion of E_b . Clearly, the values of these performance measures are not the same, and minimizing one does not necessarily minimize the other. However, there appears to be a nearly monotonic relationship between them.

4.5 Exhaustive Search as a Means to Assess System Performance

Since we do not possess an algorithm that guarantees the discovery of solutions with minimum congestion (under any of the metrics we have discussed here), the only way to determine whether a solution generated by the NN is, in fact, the minimum-congestion solution (or even a *good* solution) is to perform an exhaustive search over the entire binary solution space. Such an exhaustive search is possible only for relatively small problems. For example, it is possible for the first 24-node network example we have studied (discussed in Section 4.6), which has approximately four million different admissible solutions, but it is clearly not practical for the 100-node network (discussed in Section 4.9), which has approximately 5×10^{35} different solutions.

4.5.1 A Shortest-Path Heuristic

Since it is generally not possible to determine the minimum possible value of congestion, it is difficult to assess the quality of the NN solutions. Thus we have also considered a "shortest-path" heuristic, which appears to provide reasonably good solutions to our problem. The quality of the NN solutions can then be compared to those produced by this scheme. The shortest-path heuristic is simply an exhaustive search that includes only those paths between each SD pair that are of shortest length. For example, if a particular SD pair is connected by three paths of length 5, two paths of length 6, and two paths of length 7, only the paths of length 5 are included in the search. Although the use of only shortest paths does not guarantee optimum solutions, our studies have shown that the shortest-path heuristic performs reasonably well in many examples where a comparison with the exhaustive search is possible. More importantly, as we show in this paper, our NN solutions are generally somewhat better than the shortest-path solutions, thus demonstrating the ability of our NN approach to provide good solutions to complex problems.

4.6 An Example 24-Node Network

The first network of significant size that we studied in detail was the 24-node network shown in Fig. 2. This network has a diameter of 6, mean path length (i.e., the expected distance between an arbitrarily selected pair of nodes) of 3.1 hops, and a mean nodal degree of 3.8 (assuming unit flow on each link). A set of 10 SD pairs that require the use of multihop paths was selected from this network. The SD pairs are enumerated in Table 1. There are 62 nonredundant paths between these SD pairs, from which 16,329,600 different admissible solutions can be obtained. An exhaustive search of these solutions found that there are 48 different optimum

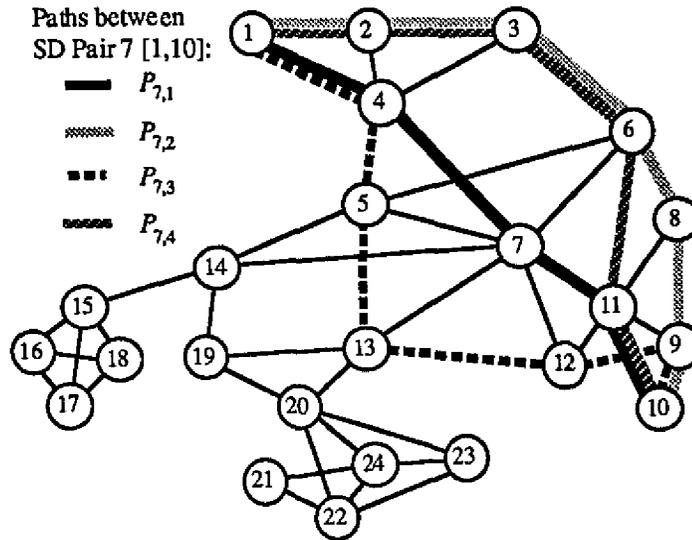


Fig. 2 — A 24-node communication network

Table 1. SD Pairs Associated with the Network of Fig. 2

Pair	Source	Dest.	Pair	Source	Dest.
1	4	24	6	21	6
2	7	17	7	1	10
3	9	16	8	3	18
4	1	19	9	2	12
5	5	11	10	14	8

solutions, which all have congestion energy $E_b = 45$. The use of the modified-Dijkstra's algorithm with 100 iterations for each SD pair, as discussed in Section 4.2, produced a total of 51 paths between the 10 SD pairs; the set of paths selected for SD pair 7 is shown in Fig. 2. A total of 1,990,656 different admissible solutions are possible for this example.* An exhaustive search of these solutions also found that there are 48 different optimum solutions, which all have congestion energy $E_b = 45$. Thus, for this example, the use of the modified-Dijkstra's algorithm reduces the solution space by more than a factor of eight without reducing the number of optimum solutions. The best solution found by the shortest-path heuristic has $E_b = 47$. Thus the shortest-path heuristic produces good (although nonoptimal) results in this example. However, our latest NN simulations (see Section 4.8) have been able to produce optimal solutions for this network in almost all runs from different random seeds. Typically, and especially for larger, heavily congested networks, we show that NN solutions can be significantly better than the best solution obtained by using the shortest-path heuristic.

*For one of the SD pairs, only one path was found. In our NN simulations we have added an additional path between that SD pair, which had not been selected by our path-selection algorithm because it contains the previously discovered path. The purpose for its inclusion, thereby raising the total number of paths to 52 and the total number of admissible solutions to 3,981,312, was to make the problem somewhat harder by requiring our NN model to make a decision on the path selection for that SD pair. It is significant to note that the NN always chose the shorter path. Here, the length of the path is not of primary importance, but rather the fact that the choice of the longer path cannot provide a lower measure of congestion energy than the choice of the shorter path because the longer path contains all of the nodes in the shorter path plus additional ones.

4.6.1 The Use of Mean-Field Annealing

In our preliminary studies, the coefficients used in the connection weights were determined by trial and error, as is common in studies of Hopfield NN models [22]. It was difficult to find good parameter values that would lead to reliable convergence to low-energy solutions. Improved results were obtained by using a form of mean-field annealing (MFA), similar to the method described in Ref. 29. This method involved gradually steepening the slope of the nonlinear neural input-output relationship by decreasing the parameter u_0 with time according to $u_0 = 10/(c+t/\tau_u)$ if $u_0 < u_0'$, and $u_0 = u_0'$ otherwise. The parameter c sets the value of u_0 at time $t = 0$, τ_u controls the rate at which the nonlinearity is steepened, and u_0' is the minimum allowed value of u_0 (which results in the maximum slope of the nonlinearity). The initial use of a smaller slope in effect permits one to make preliminary approximate decisions on the states of the neurons, thus postponing the final decision until a more thorough search of the interior of the search region has been performed. The use of a steeper nonlinearity in the later stages results in neuron output voltages that are closer to binary values. Thus it is more likely to reach an admissible low-energy (although not generally globally optimal) solution. However, a solution with a minimum value of E_b was found only once by using this approach. We then focused our attention on the use of the method of Lagrange multipliers, which has proven to be a powerful method to obtain reliable convergence to low-energy solutions. In Section 4.8.2 we compare the results obtained by using MFA with those obtained by using two versions of the Lagrange multiplier method. Performance results are shown in Figs. 5, 6, and 7.

4.7 Use of the Method of Lagrange Multipliers to Determine Connection Weights

Wacholder et al. [30] and Platt and Barr [31] observed that the energy expression corresponding to each equality constraint can be used to dynamically update the corresponding connection coefficients by the method of Lagrange multipliers (LM). With this method, each of the λ_c 's becomes a variable coefficient (Lagrange multiplier) that, at each iteration, is increased by an amount proportional to its corresponding constraint energy evaluated in the previous iteration. That is, at the $(n+1)$ st iteration, we have

$$\lambda_c(n+1) = \lambda_c(n) + (\Delta t)_\lambda E_c(n),$$

where $(\Delta t)_\lambda$ is the LM step size, which is chosen independently of the step size Δt in Eq. (6). Note that, since $E_c \geq 0$, the quantities λ_c are monotonically nondecreasing. Typically, the Lagrange multipliers are assigned initial values of 1.

The use of time-varying LM makes it inappropriate to include the additional bias current term in the second equality constraint.* Thus we set $\alpha = 1$ and include an additional bias current term I , which is positive, in each of the equations of motion.

The most obvious advantage of the method of Lagrange multipliers is that it eliminates the need to perform a trial-and-error search for the best system parameters. Such a search is especially time-consuming in large networks because many (e.g., 100) runs with different random initial conditions are typically needed to assess the performance achievable when a particular set of parameters is used. In addition, based on our application of this method to routing problems, we suspect that the dynamic nature of the λ_c 's provides better performance than the use of the best set of coefficients with constant values. This is because the relatively small initial values of the λ_c 's permits the search to emphasize somewhat the desire to minimize the performance index (network congestion) during the early part of the iteration. Toward the latter part of the iteration, the increased values of the λ_c 's more heavily penalize system states in which the constraints are not

*Recall that the parameter α was introduced for this purpose. Since the Lagrange multipliers change (they are monotonically nondecreasing), a value of α greater than 1 would correspond to an increase in the additional bias current that is proportional to the corresponding Lagrange multiplier.

satisfied; thus the neuron voltages move closer to binary values, and yield admissible equilibrium states.

4.7.1 An Alternate Formulation with Multiple Lagrange Multipliers

Examination of the third constraint energy term E_3 suggests that it may be advantageous to define a separate Lagrange multiplier for the constraint applied to each SD pair. Doing so would increase the LMs associated with those SD pairs that were unsuccessful in turning on exactly one neuron. We call this the method of "multiple Lagrange multipliers" (MLM).

The third constraint formulation can be rewritten as

$$E_3 = \sum_{i=1}^{N_{sd}} e_{3i} = 0,$$

where each of the terms of the form

$$e_{3i} = \frac{1}{2} \left(\sum_{j=1}^{N_p(i)} V_{ij} - 1 \right)^2 = 0$$

is an equality constraint specifically for SD pair i . Now Lagrange multipliers are defined to correspond to each of the e_{3i} 's, and they evolve as

$$\lambda_{3i}(n+1) = \lambda_{3i}(n) + (\Delta t)_\lambda e_{3i}(n).$$

4.8 Simulation Results of a 24-Node Network Using the Method of Lagrange Multipliers

In the following subsections, the results of simulations of the 24-node network described in Section 4.6 are used to demonstrate the effectiveness of the use of LM (Section 4.7) and MLM (Section 4.7.1) in conjunction with the path-neuron model. In the simulations, optimum solutions* were found at least 96% of the time, and the remaining solutions were nearly optimum.

4.8.1 Results Obtained Using the LM Method

The 24-node network was examined by using LM and the following parameters [$\lambda_c(0)$ is used to denote the initial Lagrange multiplier value]:

$\lambda_c(0)$	b	α	I	Δt	$(\Delta t)_\lambda$	u_o	ϵ
1.0	0.5	1.0	2.1	0.005	0.01	0.1	0.01

With the use of the method of Lagrange multipliers, we have found that, if given sufficient time, we are virtually guaranteed convergence to an admissible state (without having to rely on the instantaneous state description). Furthermore, the instantaneous state usually reaches its final value relatively early in the simulation, although the convergence criterion based on the quantity ϵ is usually not satisfied at that time. Thus, use of the early termination criterion is indicated. In this and the following simulations, we terminate a run after 500 iterations without a change in the instantaneous state, or when all the neuron output voltages reach values that are within ϵ of binary values. In conjunction with this criterion, the use of the small ϵ value listed in the table prevents premature termination of the simulation, which might result in a poor-quality solution.

*Optimum solutions are admissible solutions that have the minimum congestion-energy (E_b) value.

All of the solutions resulting from simulations from 100 different initial states were better than the best one found by using the shortest-path heuristic (for which $E_b = 47$); 97% of the solutions were optimum in terms of congestion energy ($E_b = 45$), and the remaining 3% were worse by only one unit. In the 97 optimum solutions, 6 different states (sets of selected paths) were found. Two different states were found that had $E_b = 46$.

The fact that several different optimum solutions were found further confirms that the use of different random seeds results in the search of different regions of the state space. We noted in Section 4.6 that there are actually 48 different optimal solutions out of the approximately 4 million total admissible solutions for this problem.

The evolution of the constraint energies and the Lagrange multipliers from one of the simulations is shown in Figs. 3 and 4.* These results are typical in that they show that the instantaneous state converges relatively rapidly, although convergence to a state in which the output voltage of all neurons is within some ϵ -value of binary values may take a very large number of iterations, as noted in Section 4.3. In Fig. 3, all three constraint energies initially decrease rapidly, with E_1 (which corresponds to the activation of no more than 1 path per SD pair) reaching a value very close to zero in about 250 iterations.† After 250 iterations, the NN state is relatively stable, although the values of E_2 and E_3 are still relatively large because many of the neuron output voltages are far from binary values. As the iteration continues, the rate of decrease in the values of E_2 and E_3 slows to an asymptotic approach toward zero. Figure 4 shows that the high initial constraint energies cause rapid initial growth of the Lagrange multipliers. The growth of λ_1 virtually ceases when the corresponding energy reaches a value close to 0. The asymptotic decay of E_2 and E_3 continues to cause λ_2 and λ_3 to grow slowly throughout the duration of the simulation.

4.8.2 Results Obtained Using the MLM Method

The simulations of Section 4.8.1 were repeated by using MLM with the initial value of each of the λ_{3i} set to 1.0. Figures 5, 6, and 7 present the results of this simulation compared with those from Section 4.8.1 and the best results from early studies using MFA and constant coefficients** as discussed in Section 4.6.1. In Figs. 5 and 6 the comparison is on the basis of congestion energy. These figures show that the use of MLM produces results that are slightly inferior to those produced by the use of a single LM for the third constraint. The histograms of Fig. 5 show that optimum solutions (solutions with $E_b = 45$) were found 97 times using the LM method and 96 times using MLM. In contrast, only one of the solutions found when using constant coefficients in conjunction with MFA was optimal.

We acknowledge that the relative lack of success using MFA with constant connection weights does not necessarily prove the ineffectiveness of this method; it is certainly possible that better results might have been obtained by using different values of the coefficients or MFA parameters. However, our experience has shown that a significant amount of trial and error is needed to determine the best set of system parameters and that they appear to depend strongly on the particular problem. In contrast, the method of Lagrange multipliers requires relatively little parameter adjustment when considering networks of highly varying size and connectivity, as is shown later.

*This simulation was forced to run through 10,000 iterations so that the evolution of the constraint energies and the Lagrange multipliers could be observed. The use of the early termination criterion would have allowed the termination of the run after 775 iterations.

†Note that a very small value of E_1 , by itself, does not guarantee that the instantaneous state has reached its final value. Therefore, we base the termination criterion on an unchanging instantaneous state.

**The parameter values for the simulations using MFA were the same as those listed in Section 4.8.1. In addition, the following MFA parameters were used: $c = 1$, $\tau_u = 1$, and $u_o' = 0.1$.

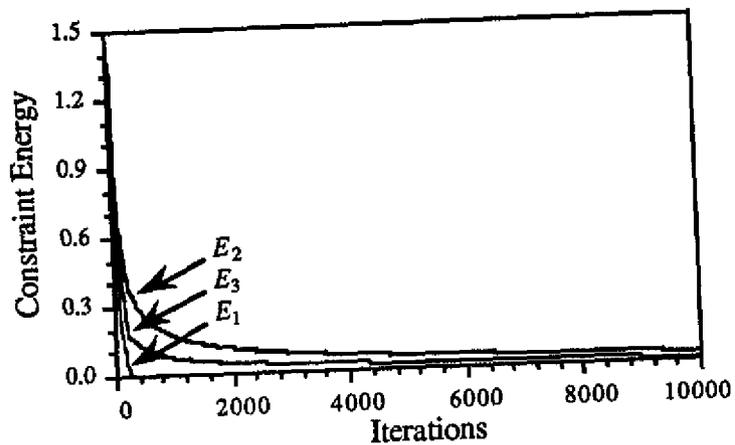


Fig. 3 — Constraint energy evolution in a sample run

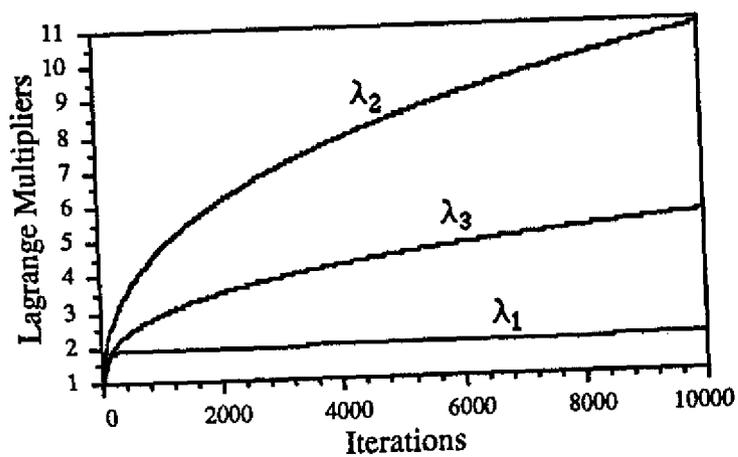


Fig. 4 — Lagrange multiplier evolution in a sample run

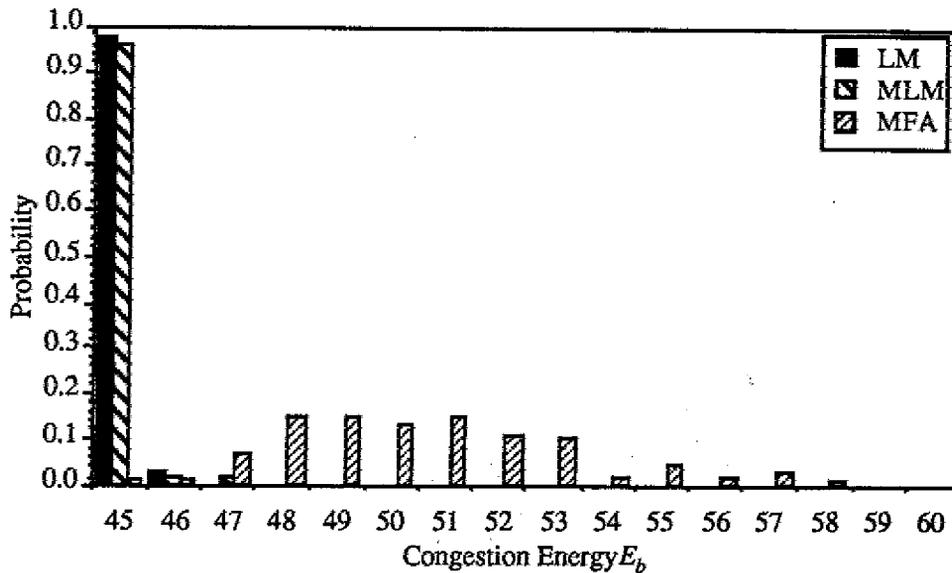


Fig. 5 — Histograms of the results of simulations of the 24-node network in terms of congestion energy

Figure 6 shows the cumulative mass functions of the results shown in Fig. 5. In our simulation runs that used LM or MLM, all of the solutions have congestion energy $E_b \leq 47$; with the use of constant coefficients and MFA, all of the solutions have congestion energy values that are less than 59.

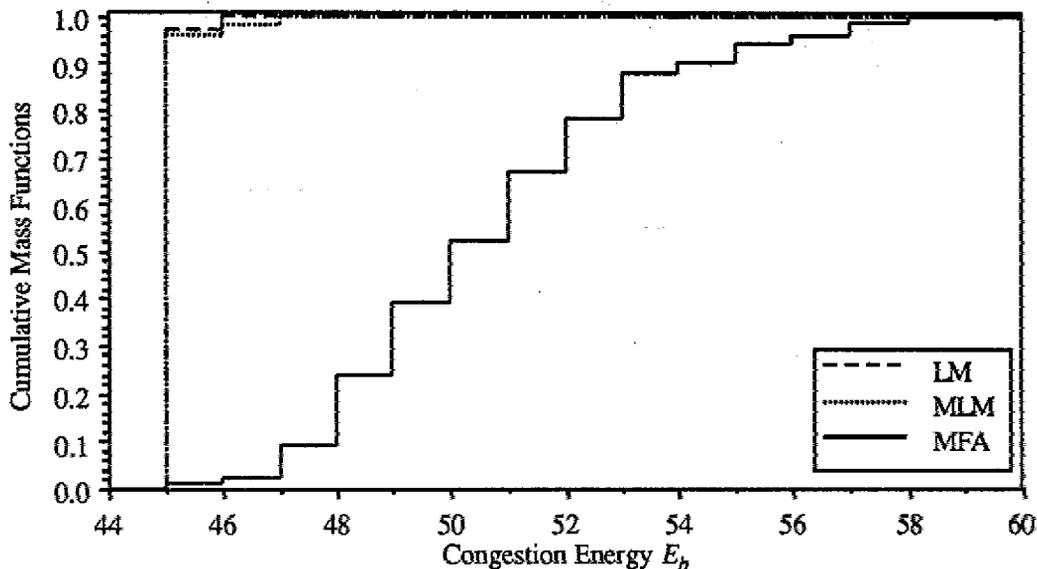


Fig. 6 — Cumulative mass functions of the results shown in Fig. 5

Figure 7 compares the quality of the same results measured with the ψ metric discussed in Section 4.4. Under this metric, the use of MLM provides the best results, and, again, the use of either form of Lagrange multipliers gives much better results than those found by using MFA and constant constraint coefficients. The most important conclusion from these results is that the use of LM or MLM, besides giving 100% admissible convergences, yields significantly better solutions than those obtained with MFA in conjunction with constant constraint coefficients.

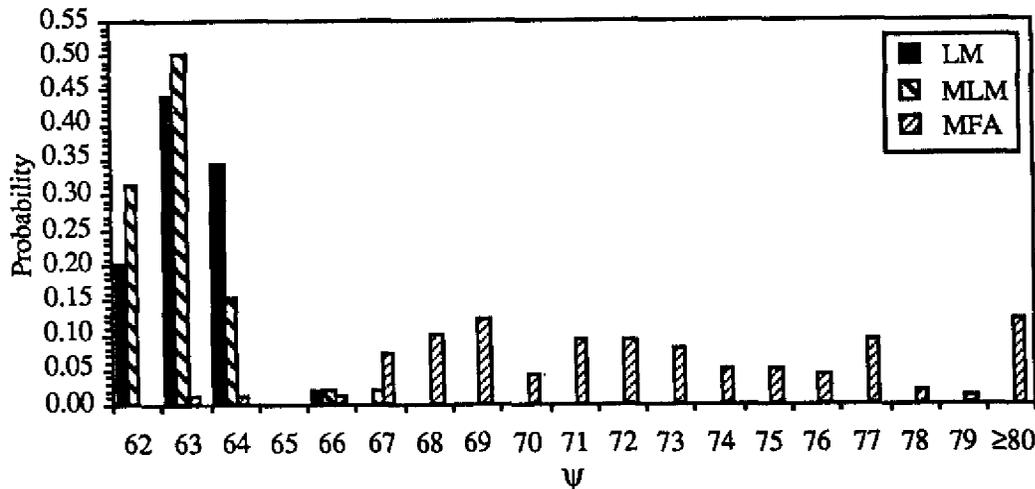


Fig. 7 — Histograms of the results of simulations of the 24-node network in terms of the ψ metric (optimum $\psi = 62$)

These results show that solutions with low values of congestion energy also tend to have low values of ψ . However, minimization of E_b does not guarantee a minimum value of ψ , and the relationship between E_b and ψ is not monotonic.

4.8.3 Combined Use of MFA and LM Techniques

Our attempts to combine the LM approach with mean-field annealing have shown that these two methods do not work well together. The apparent reason for this lack of synergism is that the LM technique works well only when a relatively steep input-output nonlinearity function is used. This is because use of a small slope results in a large region of input voltage values that do not produce (nearly) binary output values; consequently it is more difficult to satisfy the system constraints.

4.8.4 Use of Simulated Annealing to Search for the Global Minimum

Simulated annealing is a technique in which noise is added to the system throughout its evolution to permit the escape from local minima, with the goal of eventually finding the global minimum. Appendix A and Refs. 32 and 33 describe its use in conjunction with Hopfield NN models. Our use of this method produced highly mixed results. Although optimal solutions were occasionally found by using simulated annealing in conjunction with LM, most solutions were not as good as those obtained by using the method of LM alone. Reference 15 provides further details.

4.9 Simulation Results of a 100-Node Network Using the Method of Lagrange Multipliers

The 100-node network shown in Fig. 8 was used to evaluate the NN's ability to handle larger networks. The network, which was randomly generated, has a mean nodal degree of 2.9 (assuming unit flow on each link), mean path length of 4.6 hops, and a diameter of 10. A set of 40 SD pairs was arbitrarily selected (and are enumerated in Appendix B of Ref. 15), and sets of maximally node-disjoint paths were found for each SD pair. A total of 327 paths were found, yielding a network with approximately 5×10^{35} different admissible solutions. Since an exhaustive search of this network is prohibitive, a random search of 2×10^6 samples was performed to give a reference performance level for NN evaluation. The lowest congestion-energy value found by the random search was $E_b = 567$. The shortest-path heuristic solution had $E_b = 313$.

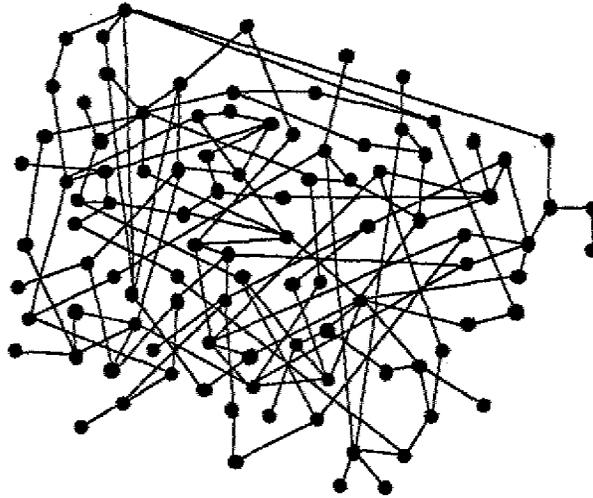


Fig. 8 — A 100-node communication network

Simulations of both the LM and the MLM models were performed from 50 different initial states with the following parameter values:

$\lambda_c(0)$	$\lambda_{3i}(0)$	b	α	I	Δt	$(\Delta t)_\lambda$	u_0	ϵ
1.0	1.0	0.5	1.0	5.0	0.0009	0.01	0.1	0.01

The results of the simulations are compared in Fig. 9. The largest congestion-energy value found using either LM or MLM was 303, which is 10 units less than the shortest-path heuristic solution. The best solution found by any method had $E_b = 291$ and was found using the LM NN. Although the LM and the MLM NNs yield results in the same range, Fig. 9 shows that the use of a single LM for the third constraint yields slightly better results than the MLM formulation.

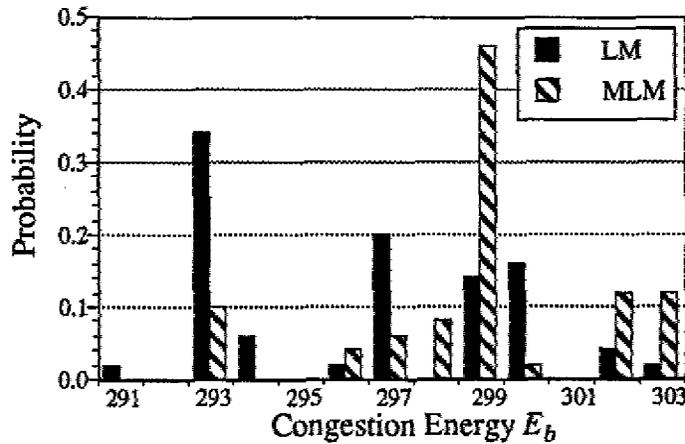


Fig. 9 — Results of simulations of a 100-node network using the LM and MLM models

4.10 Nonunit Traffic and Alternate Routing

Thus far it has been assumed that a single route is selected to carry the entire amount of traffic between each SD pair, and that the traffic requirements are uniform, i.e., the same between each SD pair. Thus, in our problem specification, it has been sufficient to require the delivery of a single unit of traffic between each SD pair. It is known, however, that alternate routing, i.e., splitting the traffic over two or more routes, can be advantageous. To permit alternate routing, we can divide the basic unit of traffic into m subunits, which can then be divided between two or more paths.

In an effort to demonstrate the benefits of alternate routing, the traffic requirements between each SD pair were increased. To accommodate m units of traffic on the i th SD pair, m replicates of each of the neurons $ij, j = 1, \dots, N_p(i)$, which represent paths between SD pair i , were created and treated as distinct paths between new, independent clones of the same SD pair. With the additional neurons installed, the problem has been transformed back to the unit-traffic routing problem. The same constraints and optimization connections used for the unit-traffic example are used here as well.

4.10.1 Triplicate 100-Node Network

Three units of traffic were placed on each of the 40 SD pairs of the 100-node network (Fig. 8), resulting in a NN model consisting of 981 neurons. Again, it would be desirable to determine the shortest-path heuristic solution, which could serve as a benchmark against which to compare our NN solutions. Unfortunately, the number of shortest-path solutions in this example makes an exhaustive search of them prohibitive; there are 15552 shortest-path solutions for the unit-traffic problem, which results in $15552^3 = 3.76 \times 10^{12}$ shortest-path solutions to the triple-traffic problem. However, a useful benchmark is the performance achieved by sending all three units of traffic on the set of paths determined by the shortest-path heuristic solution to the unit-traffic problem. The congestion energy of this solution is $E_b = 3543$.

Simulations using LM and the following parameter values were run from 20 different initial states.

$\lambda_i(0)$	b	α	I	Δt	$(\Delta t)\lambda$	u_o	ϵ
1.0	0.5	1.0	5.0	0.0001	0.01	0.1	0.01

The triple-traffic solutions had congestion-energy values E_b that ranged from 3349 to 3417. In the best triple-traffic solution, the NN split the traffic at seven SD pairs. To determine whether any benefit was actually obtained by the triplication of neurons, these results were compared with those obtained by using the unit-traffic formulation. First, we considered the simple triplication of the best unit-traffic NN solution. This resulted in a value of $E_b = 3381$. Eight of the 20 solutions obtained by the triplicate network had lower values of E_b , once again demonstrating the benefits of alternate routing.

The results of the triple-traffic NN model were also compared to those obtained by combining three of the unit-traffic solutions. A search of all possible combinations of three solutions obtained from the NN with unit traffic (only solutions with the lowest value of E_b were considered, and duplicates were permitted) resulted in a best solution that had a congestion energy of $E_b = 3368$. Although this was better than the solution obtained by simply triplicating the best solution for the unit-traffic model, it was still not as good as that obtained by the NN that was programmed for three units of traffic.

Although the improvement obtained by triplicating the neurons in the network was not dramatic, it is significant in that we are unaware of other methods to find better sets of paths. Perhaps even more noteworthy is the capability of the NN model to handle such a large network and to provide good solutions for a reasonably large percentage of the initial states. It is significant

to note that, although some adjustment of system parameters was necessary when studying networks of different sizes under the LM method, such adjustments were typically limited to time constants and bias currents, for which acceptable values were determined relatively quickly.

The capability to determine good solutions to large examples and the robustness of the model when presented with problems of varying size indicate that the solutions scale well with increasing problem sizes. Good scaling properties were also observed by Kamgar-Parsi et al. [27, 28] in their NN studies of the clustering problem. They attributed this behavior to the nature of the constraints in their problem, which permitted an instantaneous state description similar to the one we have used in this study. An example of a problem that does not scale well is a Hopfield NN implementation of the Traveling Salesman Problem. The source of difficulty in that problem is apparently the fact that the neurons must satisfy the permutation matrix constraint, which does not readily admit such an instantaneous state description. The link-activation scheduling problem, which we discuss in Sections 6-11, appears to fall somewhere between these two extremes in terms of scalability properties.

4.10.2 *Nonuniform Traffic in the 100-Node Network*

To assess the ability of the NN model to route nonuniform traffic so that congestion is minimized, an arbitrary number of units of traffic were placed on each of the 40 SD pairs in the 100-node network (Fig. 8). Two different nonuniform network loadings were created. Both had traffic levels of 1 to 4 units on each SD pair. In the first example, a total of 100 units of traffic was specified, resulting in a total of 823 neurons. In the second example, the total traffic was 105 units, resulting in 872 neurons. Both of the loadings were briefly analyzed by using the NN model with LM and the parameter values listed in Section 4.10.1. The solutions did contain traffic splitting, and gave lower congestion energy than would have been obtained by placing all traffic on the best unit-traffic solution paths.

The case of nonuniform traffic is considerably more interesting than that of uniform traffic. First, this case is more likely to arise in practice. More importantly, it does not appear that good results are often obtained by simply using the single paths obtained for unit-traffic requirements, although this approach worked fairly well for the uniform-traffic example. Since the demands placed on the network by nonuniform traffic are quite different from those of uniform traffic, it appears that they cannot be satisfied well unless these traffic requirements are programmed into the NN formulation.

4.11 A Modified Form of the Congestion-Energy Function

The development of a NN model that defines a neuron for each link of every path, which is discussed in Section 5, has motivated our examination of a minor modification to the method used for calculating E_b . It was observed that it is not possible to make a direct comparison between the results obtained under the path-neuron and link-neuron models when the original formulation of E_b is used because the measures used to define congestion energy in these two cases are not totally consistent with each other. Thus a modified performance measure, which is denoted as E_b' , has been developed. When E_b' is used as the performance measure, the resultant congestion energy for any particular solution under the path-neuron formulation is directly proportional to that under the link-neuron formulation, thus permitting a direct comparison of the solutions obtained by using these two approaches.

The modified calculation uses a slightly more involved method of counting shared nodes that weights each occurrence of a shared node according to the "node type." A node that is shared by two paths is one of three types:

- A type-1 shared node is an intermediate node (i.e., neither a source nor a destination) in both paths and receives a weight of 1.

- A type-2 node is an end-node (a source or destination) for one path and an intermediate node in the other path and receives a weight of 0.5.
- A type-3 node is an end-node for both paths and receives a weight of 0.25.

Thus E_b' has the same form as E_b , except that $|P_{ij} \cap P_{kl}|$, which was originally defined in Section 3.3 to be the number of nodes shared by paths P_{ij} and P_{kl} , is now replaced by

$$|P_{ij} \cap P_{kl}'| = \begin{aligned} & \text{the number of type-1 nodes shared by paths } P_{ij} \text{ and } P_{kl} \\ & + 0.5 \text{ (the number of type-2 nodes shared by paths } P_{ij} \text{ and } P_{kl}) \\ & + 0.25 \text{ (the number of type-3 nodes shared by paths } P_{ij} \text{ and } P_{kl}). \end{aligned}$$

This weighting scheme is somewhat arbitrary, but it appears to be reasonable because it assigns a heavier weight for nodes that require more slots. In particular, it reflects the fact that intermediate nodes must support both input and output flows, whereas the source and destination nodes support only one or the other.

An exhaustive search of the 24-node network (Fig. 2) based on the modified congestion energy metric (E_b') found that the best solutions have $E_b' = 33.25$.^{*} In a series of 100 runs of the NN model with the modified E_b' formulation, all of the solutions were one of two different states that both had $E_b' = 33.75$, which is greater than the optimum value by 0.5.

4.12 Conclusions on the Path-Neuron Model

We have demonstrated the power of our path-neuron Hopfield NN model to choose sets of paths that provide low levels of congestion in relatively large, heavily congested networks. Optimal or nearly-optimal solutions were found in many of our examples. In this concluding subsection, we summarize our results qualitatively, and we attempt to put our studies of the path-neuron NN model into perspective.

The most critical issue in the design and simulation of a Hopfield NN model is the choice of the coefficients used in the connection weights. In our early studies, they were determined by using a tedious trial-and-error approach. It was found that the use of mean-field annealing (gradually increasing the slope of the neurons' input-output nonlinearity) in conjunction with constant coefficients provided greatly improved performance over the performance achieved by using an unvarying nonlinearity. However, the solutions found in most of the NN runs were not as good as the solutions found by the shortest-path heuristic. Furthermore, the need to determine a new set of coefficients for each network configuration, and our inability to find a method to automate the procedure for determining these coefficients, limit the general applicability of this method.

Dramatic performance improvement was obtained by using the method of Lagrange multipliers, which permits the coefficients to vary dynamically along with the evolution of the system state. In the early stages of a simulation run, the congestion-limiting component dominates the connection weights, guiding the search toward a region of low congestion. As the run progresses, the impact of the constraints guides the solution toward an admissible state with binary neuron values. The use of dynamically varying LMs provides better performance than would be possible through the use of an optimal set of constant connection-weight coefficients because of this ability to emphasize congestion control in the early stages and constraint satisfaction later on. Use of this method has provided optimal or near-optimal solutions in many of our examples. We have noted that the model scales well with increasing problem size, with relatively little need for change in system parameters.

^{*}Congestion energy is lower under the new metric because type-2 and type-3 nodes are weighted less than type-1 nodes.

The fact that global minima are not always found, a common characteristic of Hopfield NNs, is typical of heuristic algorithms for solving difficult combinatorial-optimization problems; in many such problems, optimal solutions cannot be guaranteed without exhaustive search. However, the inability to guarantee a global optimum is mitigated by the fact that repeated runs are possible from different initial conditions; thus the best solution that is found can be chosen as the solution to the problem. Although the simulation runs begin in random initial states, this method is not simply one of random search; system evolution is guided by the equations of motion, which are derived from the energy function, which in turn is based on the objective function and the system constraints. The fact that most of our solutions are so close to the optimum value in such a large fraction of the cases studied demonstrates the robustness of our models, and suggests that they may perform well in considerably larger examples as well.

In conclusion, we have demonstrated the effectiveness of our Hopfield NN model for minimizing congestion in large, heavily congested networks. In particular, the use of the method of Lagrange multipliers, under which the coefficients in the connection weights evolve dynamically along with the system state, provides highly-robust operation. Ultimately, our goal is to develop NN models for the joint routing-scheduling problem. As a step toward this goal, in Section 5 we discuss an alternate NN model in which a neuron is defined for each link along every path in the network.

5.0 A LINK-NEURON NN FORMULATION FOR THE MINIMIZATION OF CONGESTION

Ultimately, it would be desirable to develop a NN model for the complete joint routing-scheduling problem. Such a NN would not only choose paths between each SD pair, but would also determine the particular time interval in which each link along a selected path is to be activated so that destructive interference does not occur. Thus system modeling must reflect the behavior of each individual link, a level of detail that the path-neuron formulation discussed in Sections 3 and 4 does not provide. In a step toward this goal, we have developed an alternate formulation of the congestion-minimization problem, in which neurons are defined for each link along every path, rather than one for each complete path. We also note that the link-neuron formulation may be viewed as a first effort toward the solution of the more general routing problem in which paths between each SD pair are not specified in advance; in this case the NN must piece together complete paths from individual links.

The link-neuron formulation of this problem is similar to the path-neuron formulation in that the same basic constraints hold and the optimization goal is again to minimize congestion. However system modeling and simulation is somewhat more difficult, because the interactions among individual links rather than among complete paths must be considered. The most obvious complication is the much greater number of neurons and interconnections that are needed to model the system. A further complication is the need to ensure that complete paths are formed.

In this section, we reformulate the congestion-minimization problem by developing a link-neuron Hopfield NN model. The development follows the same basic procedure as that for the path-neuron model. An energy function is derived that incorporates both the minimization of congestion and the problem constraints. The method of Lagrange multipliers is again used to dynamically determine the connection weights. Performance results demonstrate the soundness of our approach. Studies of the same 24-node network considered in Section 4 show that complete paths are formed reliably and that good, although not optimal, solutions are usually found.

5.1 The Basic Link-Neuron Model

In the link-neuron model, a neuron is defined for each link of every path. A triple index is used to specify the neurons, e.g., neuron ijk represents the k th link in the j th path between SD pair i . The input and output voltages of neuron ijk are denoted u_{ijk} and V_{ijk} respectively. Figure 10(a) shows the same example six-node network with two paths between each of two SD pairs used in

Section 3.2 to describe the path-neuron model. Figure 10(b) shows the corresponding link-neuron NN model. In Fig. 10(b), the upper horizontal plane defined by neurons $1jk$ ($j \in \{1, 2\}, k \in \{1, 2, 3\}$) contains all the neurons that represent links between SD pair 1, and the lower plane contains all the neurons $2jk$ that represent links between SD pair 2. Several neurons may correspond to the same physical link in the NN model. For example, neurons 221 and 122 in Fig. 10(b) both represent the physical link connecting nodes 4 and 5. The connection weights shown in the figure are discussed in Section 5.3.

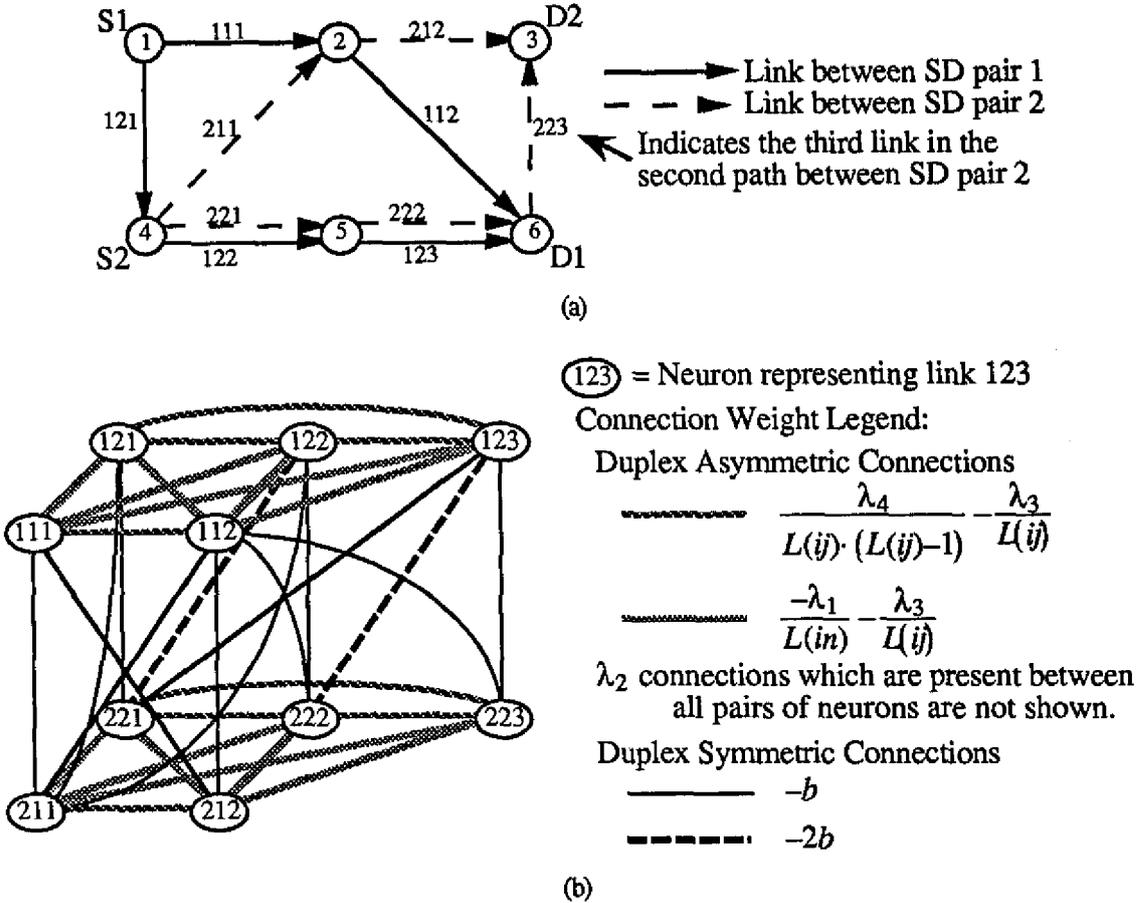


Fig. 10 — An example network: (a) shows a six-node communication network; (b) is the corresponding link-neuron model

Following the same basic procedure used in the development of the path-neuron model in Section 3, the generic form of the Lyapunov energy function can be rewritten as

$$E_{total} = -\frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{m=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{n=1}^{N_p(m)} \sum_{k=1}^{L(ij)} \sum_{o=1}^{L(mn)} T_{ijk,mno} V_{ijk} V_{mno} - \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} I_{ijk} V_{ijk} \quad (7)$$

Here, $N_p(i)$ is the number of paths between SD pair i ; $L(ij)$ is the length in hops of the j th path between SD pair i ; $T_{ijk,mno}$ is the connection weight between neurons ijk and mno ; and I_{ijk} is the bias current applied to neuron ijk .

As in the path-neuron model, the link neurons are interconnected in a manner that enforces the problem constraints and that tends to reduce congestion. The connections are derived from an energy formulation in a manner similar to that of Section 3. However, the particular characteristics of the link neurons are incorporated into the model. The total energy can again be expressed as the

weighted sum of the congestion energy, the energies associated with the constraints, and energy resulting from additional bias currents as follows:

$$E_{total} = bE_b + \sum_{c=1}^C \lambda_c E_c - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} V_{ijk}. \quad (8)$$

Again, b is a constant coefficient that weights the relative priority given to optimization, as compared to constraint satisfaction. The λ_c 's are the connection coefficients for the constraint terms. The additional bias (I) term, as in the path-neuron formulation, provides a neutralizing shift in the activation level.

After a discussion of congestion energy and the system constraints, we present the resulting expressions for the connection weights and bias currents. Then we present our simulation results. Again, the method of Lagrange multipliers is used to permit the λ_c 's to vary dynamically along with the system state.

5.2 Congestion Energy

Recall that in our studies of the path-neuron model we considered two congestion energy functions, E_b and E_b' . Under the E_b criterion, the strength of the component of the inhibitory connections resulting from the congestion-energy term is proportional to the number of common links in the two paths. Under the E_b' criterion, a distinction is made between intermediate and terminal nodes along the paths to reflect their different requirements for transmission slots.

In the link-neuron NN model, interaction takes place between individual links on a pairwise basis rather than between entire paths (as was the case in the path-neuron model); thus congestion enters the system dynamics in terms of the pairwise interaction of individual links.

The expression for the congestion energy in the link-neuron formulation is essentially the same as that in the path-neuron formulation, i.e.,

$$E_b' = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{m=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{n=1}^{N_p(m)} \sum_{k=1}^{L(ij)} \sum_{o=1}^{L(mn)} |A_{ijk} \cap A_{mno}| V_{ijk} V_{mno},$$

$m \neq i$

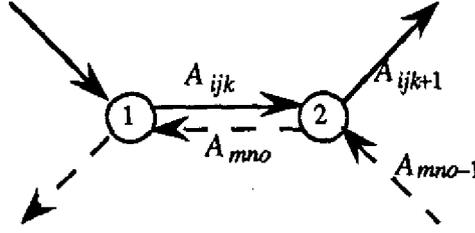
where

A_{ijk} is the k th link in the j th path between SD pair i , and

$|A_{ijk} \cap A_{mno}|$ is the number of nodes shared by links A_{ijk} and A_{mno} .

Note that $|A_{ijk} \cap A_{mno}|$ can take on only the values 0, 1 and 2. This quantity is 2 if the links A_{ijk} and A_{mno} share the same physical link (in which case they share two nodes), and it is 1 if the links share a single common node. Figure 11 illustrates these two situations. When the two links have no common nodes, $|A_{ijk} \cap A_{mno}| = 0$. Based on this model, it is straightforward to determine the contributions to the connection weights that are associated with the congestion-energy term (which again represent inhibitory connections), where the coefficient b is again used to weight the congestion-energy term. If two neurons from different SD pairs share the same physical link, an inhibitory connection of strength $2b$ is established between them because they have two nodes in common. If two neurons from different SD pairs represent links that have one node in common, an inhibitory connection of strength b is established between them. Note that since the system constraints, which are discussed in Section 5.3, discourage the selection of links in different paths

between a common SD pair, the congestion-limiting connections need only be created between links in paths between different SD pairs.



$$\text{If } m \neq i, |A_{ijk} \cap A_{mno}| = 2, \text{ and } |A_{ijk+1} \cap A_{mno}| = |A_{ijk+1} \cap A_{mno-1}| = 1.$$

Fig. 11 — Illustration of adjacent links and links that share the same physical link

The basic difference between the expression for E'_b and that given earlier for E_b under the path-neuron formulation is that, since interactions between neurons now correspond to the interactions of individual links, the summation must be taken over all possible such interactions. The corresponding contribution to the equation of motion term is found by taking the partial derivative of E'_b with respect to V_{ijk} :

$$\frac{-\partial E'_b}{\partial V_{ijk}} = - \sum_{\substack{m=1 \\ m \neq i}}^{N_d} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} |A_{ijk} \cap A_{mno}| V_{mno}.$$

Reference 15 demonstrated that

$$E'_b = 4E'_b'.$$

This proportionality relationship permits a straightforward comparison between the solutions obtained by the path-neuron and link-neuron NN models; e.g., performance measures of $E'_b' = 36$ and $E'_b = 144$ represent solutions of identical quality.

5.3 Incorporation of Constraints into the Energy Function

We have established four constraints for the link-neuron NN model. The first three correspond directly to the constraints used in the path-neuron model. The fourth has been added to ensure the activation of complete paths. Again, all are equality constraints, and the corresponding energies are zero when the constraints are satisfied. We next discuss each of these constraints and its corresponding contribution to the equations of motion.

1. Activate (select) links from no more than one path per SD pair:

$$E_1 = \frac{1}{2} \sum_{i=1}^{N_d} \sum_{\substack{j=1 \\ m \neq j}}^{N_p(i)} \sum_{m=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \sum_{n=1}^{L(im)} \frac{V_{ijk} V_{imn}}{L(im)} = 0 \quad (9)$$

$$\Rightarrow \frac{-\partial E_1}{\partial V_{ijk}} = - \sum_{\substack{m=1 \\ m \neq j}}^{N_p(i)} \sum_{n=1}^{L(im)} \frac{V_{imn}}{L(im)}.$$

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 31, 1991	3. REPORT TYPE AND DATES COVERED Interim-10/89-4/91		
4. TITLE AND SUBTITLE The Problems of Routing and Scheduling in Multihop Radio Networks — A Hopfield Neural Network Approach			5. FUNDING NUMBERS PE: 61153N PR: RR021-0542 WU: DN480-557 DN159-036	
6. AUTHOR(S) Jeffrey E. Wieselthier, Craig M. Barnhart, and Anthony Ephremides*				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5000			8. PERFORMING ORGANIZATION REPORT NUMBER NRL/FR/5521—91-9366	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of the Chief of Naval Research Arlington, VA 22217			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES *University of Maryland and Locus, Inc.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Although the issues of routing and scheduling in packet radio networks are highly interdependent, few studies have addressed their interactions. After reviewing the major issues associated with the joint study of these problems, we address both components individually. Each is formulated as a combinatorial-optimization problem, and Hopfield neural network (NN) models are developed for solving several versions of these problems. A key feature of our models is the use of the method of Lagrange multipliers, which permits the coefficients in the connection weights to vary dynamically with the evolution of the system state. Extensive software simulation results demonstrate the capability of our approach to determine good sets of routes and link activation schedules. These results also demonstrate that these problems are not separable, i.e., that a model that jointly optimizes routes and link-activation schedules is needed. Issues associated with the extension of this approach to the joint routing/scheduling problem are discussed, and a preliminary description of such a model is provided.				
14. SUBJECT TERMS Communications network Hopfield network Routing Multiple access Neural network			15. NUMBER OF PAGES 93	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

This constraint supplies an inhibitory connection between all pairs of neurons representing links in different paths that correspond to the same SD pair. The normalization with respect to path length has been introduced to remove the tendency of this type of constraint term to favor long paths; this is explained in detail in Ref. 15.

We remark that the normalization associated with this constraint, as well as constraints 2 and 3, results in asymmetric connection weights. Although symmetric connection weights may be needed to guarantee convergence (see Ref. 22), the lack of symmetry in our problem formulation has not prevented convergence, as will be discussed later. However, it is possible that the asymmetry is a factor in our inability to find globally optimal solutions.

2. *Activate a total of exactly N_{sd} paths in the network:*

$$E_2 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \frac{V_{ijk}}{L(ij)} - N_{sd} \right)^2 = 0 \quad (10)$$

$$\Rightarrow \frac{-\partial E_2}{\partial V_{ijk}} = \frac{-1}{L(ij)} \left(\sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} \frac{V_{mno}}{L(mn)} - N_{sd} \right).$$

This constraint specifies that exactly N_{sd} paths shall be completely activated. In this term, the normalization is required to maintain the equality constraint. Any path that is completely activated will contribute the value 1 to the triple sum. The triple sum may also be viewed as the sum of the mean path output voltages, where the mean path output voltage is defined to be the mean voltage of all neurons that belong to that path. Again, any path that is completely activated will have a mean path output voltage of 1. Note that, as in the case of path neurons, it is possible for this constraint to be satisfied when a larger number of paths have their mean neuron output voltages somewhat less than 1. In the present case of link neurons, we may also have situations in which portions of paths are fully activated; e.g., for a given SD pair, half of the nodes in each of two different paths may have output voltages of 1 and the remainder may have output voltages of 0. Thus a separate constraint (i.e., constraint 4, to be discussed shortly) is needed to ensure that complete paths are formed.

3. *Activate exactly one path per SD pair:*

$$E_3 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \left(\sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \frac{V_{ijk}}{L(ij)} - 1 \right)^2 = 0 \quad (11)$$

$$\Rightarrow \frac{-\partial E_3}{\partial V_{ijk}} = \frac{-1}{L(ij)} \left(\sum_{m=1}^{N_p(i)} \sum_{n=1}^{L(im)} \frac{V_{imn}}{L(im)} - 1 \right).$$

This constraint is essentially a reformulation of constraint 2. As in the case of the path-neuron formulation, it is helpful although it is redundant. It specifies that exactly one path shall be completely activated between each SD pair. (The above discussion of partially activated paths applies here as well.) As with the path-neuron model, E_3 can be broken up to create multiple Lagrange multipliers. The squared expression in parentheses gives the energy that drives a Lagrange multiplier for each SD pair.

4. Activate complete paths:*

$$E_4' = \frac{1}{2} \left(N_{sd} - \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \sum_{\substack{m=1 \\ m \neq k}}^{L(ij)} \frac{V_{ijk} V_{ijm}}{L(ij) \cdot (L(ij) - 1)} \right)^2 = 0. \quad (12)$$

The corresponding equation of motion term is given by

$$\frac{-\partial E_4'}{\partial V_{ijk}} = \left(N_{sd} - \sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} \sum_{\substack{p=1 \\ p \neq o}}^{L(mn)} \frac{V_{mno} V_{mnp}}{L(mn) \cdot (L(mn) - 1)} \right) \left(\sum_{\substack{h=1 \\ h \neq k}}^{L(ij)} \frac{V_{ijh}}{L(ij) \cdot (L(ij) - 1)} \right).$$

Although the squaring operation produces fourth-order terms in the constraint term and third-order terms in the equations of motion, it does not introduce any new problems into our simulations. Despite the form of the expression, fourth-order neuron interconnections are *not* needed since the original energy term in effect becomes a coefficient (with the same value for all neurons at any step of the iteration) in the new expression. When this constraint is satisfied, $E_4' = 0$ and the effect of the corresponding equation of motion term vanishes because the first factor goes to zero. When the constraint is far from satisfied, the first factor of the equation of motion term enhances the effect of the E_4' term, resulting in faster movement toward an admissible solution.

This constraint is unique to the link-neuron model. It specifies that if any neurons in a path are active, all neurons in that path shall be active. The constraint is enforced by creating excitatory connections between all neurons that form a path. Again, normalization is required to express this requirement in terms of an equality constraint. Since the sum takes the pairwise product of the output voltages of all neurons on a path, there are $\binom{L(ij)}{2} = L(ij) \cdot (L(ij) - 1) / 2$ products added for each path ij . Thus, normalization by $L(ij) \cdot (L(ij) - 1) / 2$ allows each active path to contribute unity to the sum.

5.4 Connection Weights and the Equations of Motion

The total energy is obtained by taking a weighted sum of the congestion-energy and constraint-energy terms, which were given in Sections 5.2 and 5.3, as specified by Eq. (8). The connection weights and bias currents are then determined by transforming the resulting expression into the form of Eq. (7), as was done for the path-neuron model in Section 3.5. Reference 15 provides complete expressions for the connection weights, bias currents, and equations of motion.

5.5 Simulation Procedure

The link-neuron model was simulated by using a computer program written in C++ and run on Sun workstations. A listing of the paths in the network to be analyzed, similar to that shown in Table B.1 in Appendix B of Ref. 15, gives the information needed for the program to create the NN model. The initial input voltage to each neuron ijk is set so that the output voltage is equal to the inverse of the number of paths between SD pair i . A small random perturbation (independently chosen for each neuron) is again added to the input to avoid the effects of a totally symmetric initial state. The perturbation is uniformly distributed on $[-0.1u_o, 0.1u_o]$. An iteration of the equations of motion is then performed until one of three termination criteria is met. The iteration is terminated if (1) the NN reaches stable convergence, (2) the NN state (the set of selected paths) remains unchanged for a specified number of iterations, or (3) a time-out is reached.

*The notation of E_4' is used to maintain consistency with Ref. 15 in which the original, and less successful, formulation of this constraint energy was denoted as E_4 .

5.5.1 Termination Criteria

Unlike the path-neuron model, the link-neuron model does not allow an instantaneous solution to be obtained by simply interpreting the neuron with the largest output voltage of all neurons associated with a SD pair as the chosen neuron.* In the link-neuron model, sets of neurons are used to represent individual paths. For a path to be activated, all of the neurons in the set representing that path must be active. To apply the second termination criterion, we use the following method to determine the instantaneous NN state (the set of prematurely chosen paths at any instant in time). We declare the path ij to be the only active path between SD pair i if

$$\sum_{k=1}^{L(ij)} \frac{V_{ijk}}{L(ij)} = \max_{n=1}^{N_p(n)} \left(\sum_{o=1}^{L(in)} \frac{V_{ino}}{L(in)} \right).$$

That is, the path with the largest average neuron output voltage is declared the active path between its associated SD pair. With this instantaneous interpretation of the NN state, the second termination criterion can be restated as: Terminate the iteration if the instantaneous state remains unchanged for a specified number (typically several hundred) of iterations. Reference 15 discusses termination criteria in greater detail.

5.5.2 Network Description

All of the simulation results presented here are based on the 24-node network shown in Figure 2, which we studied earlier using the path-neuron NN model. Table B.1 in Appendix B of Ref. 15 lists the 10 SD pairs and 52 paths in the network. An exhaustive search of the approximately 4 million admissible solutions has shown that the lowest possible value of E_b^c is 133; this is, of course, exactly four times the value of $E_b' = 33.25$ evaluated under the path-neuron formulation. To provide a measure of the quality of our NN solutions, we note that the best solution found by applying the shortest-path heuristic to this network yielded $E_b^c = 147$.

5.6 Miscellaneous Considerations

5.6.1 The Need for Additional Bias Currents

It was originally hypothesized that including the excitatory connections provided by the E_4 formulation would eliminate the need for any additional bias currents. In efforts to verify this hypothesis, a set of three simulations from 100 different initial states† were run with three different levels of additional bias I : $I = 0$, $I = 3.0$, and $I = 5.0$. The other parameter values were as follows:

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	$\lambda_4(0)$	b	α	Δt	$(\Delta t)_\lambda$	u_a	ϵ
1.0	1.0	1.0	1.0	0.5	1.0	0.001	0.01	0.1	0.01

Performance results, discussed in detail in Ref. 15, demonstrate that considerable improvement is obtained by using bias values of $I = 3.0$ or 5.0 . The best solutions found in simulations without additional bias had $E_b^c = 151$. With additional bias of 3.0 or 5.0, solutions were found with $E_b^c = 135$ **.

*The instantaneous state of the system is defined in Section 4.3.

†The same set of 100 different initial random seeds was used for each of the three values of I .

**As a result of the topology of this network and the method of calculating congestion energy, all E_b^c values will be odd integers.

As a result of these simulations, the additional bias current term was used in all subsequent simulations of the link-neuron model. Although much improvement has been obtained in the quality of the solutions by using the additional bias, the results remain inferior to those obtained by using the path-neuron model. With the path-neuron model, we were able to find solutions with $E_b' = 33.75$ (which corresponds to $E_b' = 135$) virtually 100% of the time. A reasonable explanation for the inability to find optimal solutions is that the link-neuron model contains a much larger number of neurons and interconnections, and thus a much larger solution space must be searched. Another possible factor may be the use of asymmetric connection weights that result from the first three constraints, as was discussed in Section 5.3.

5.6.2 The Use of MLM for the "Complete-Path" Excitatory Connections

As was done in the path-neuron formulation, we again considered the use of multiple Lagrange multipliers (MLM) to implement the fourth constraint, which encourages the activation of complete paths. Separate Lagrange multipliers were defined for each SD pair, as is discussed in Ref. 15. Although the use of MLM improved performance for the path-neuron model, as is discussed in Section 4.8.2, performance results for the link-neuron model indicate that the use of MLM is detrimental to NN performance in this application

5.7 Methods to Overcome a Detrimental Preference for Short Paths

A careful evaluation of the link-neuron model reveals that the energy expression corresponding to the fourth constraint (E_4') contains a subtle bias in favor of shorter paths, which makes it difficult to find minimum-energy paths. Although the solutions produced by the shortest-path heuristic are usually of reasonably good quality, as discussed in Section 4.5.1, a built-in preference for short paths in the NN model is not desirable because it limits the search space. Consequently, better solutions that use longer paths may be overlooked.

Two different methods were devised to eliminate, or at least reduce, the innate short-path preference of the E_4' formulation. The first method artificially forces all paths between a SD pair to have the same length by adding "dummy neurons" to the shorter paths. The second method adds a "compensatory bias" to longer paths to minimize the short-path preference. Both methods are fully discussed in Ref. 15, and simulation results are presented in the following subsection.

Simulation Results Using the Dummy-Neuron Model

The use of the dummy-neuron formulation results in a marked improvement, as compared to the original E_4' formulation (Fig. 12). As usual, simulations were run from the 100 different initial states used in Section 5.6. In 61 of these runs, $E_b' \leq 147$ (the congestion energy of the solution found by the shortest-path heuristic); a total of 49 runs had $E_b' < 147$. Seventeen solutions were found that had congestion energy of $E_b' = 135$. The parameter values used in the dummy-neuron simulations were

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	$\lambda_4(0)$	b	I	α	Δt	$(\Delta t)\lambda$	u_o	ϵ
1.0	1.0	1.0	1.0	0.5	1.0	1.0	0.001	0.005	0.1	0.01

Simulation Results with Compensatory Bias

Simulations using the compensatory bias described above in conjunction with the E_4' formulation were run from the same set of 100 different initial states used previously. The parameter values were the same as used in the dummy-neuron simulations with the exception that $I = 5.0$. The results are compared with those from the dummy-neuron simulations and simulations with no effort to compensate for the short-path preference in Fig. 12. The use of compensatory

bias yields improved performance, but not by as much as the use of the dummy-neuron model. Twelve of the compensatory-bias solutions had congestion energy values of $E_b^l = 135$, and 41 solutions had $E_b^l \leq 147$, the congestion energy of the solution obtained by using the shortest-path heuristic.

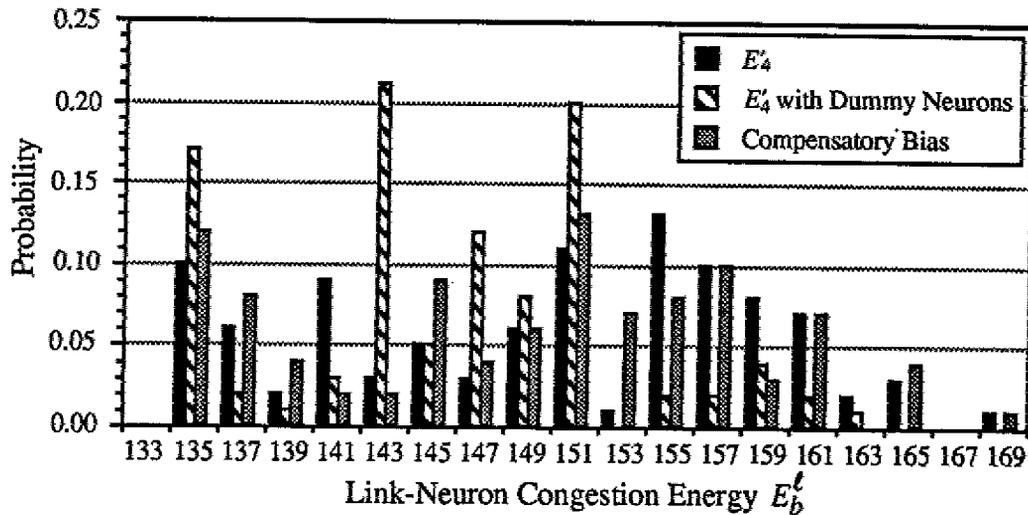


Fig. 12 — Results of simulations using the dummy-neuron and the compensatory-bias models compared with best previous results

5.8 Use of Simulated Annealing with the Link-Neuron Model

In limited experimentation with Gaussian simulated annealing using different cooling schedules, solutions were found with congestion energy values ranging from $E_b^l = 155$ to 217. These poor results, combined with the highly mixed results of GSA when applied to the path-neuron model, prompted us not to pursue this approach any further.

5.9 A Distributed Implementation of the Link-Neuron Model

In Ref. 15 we describe a “distributed” NN model and present results of its simulation. Typically, in a distributed network protocol there is no central controller with access to global network information. Decisions are made at each node based solely on local information, i.e., information obtained solely from one- or two-hop neighbors. A NN model with neural interconnections between only those neurons that represent adjacent links bases its decisions solely on information from no further than two hops away, and therefore a distributed implementation of its function is perhaps possible. However, the practicality of such a distributed implementation is questionable. In simulations of the algorithm, the lack of a central controller with global knowledge rules out the use of the early termination criterion, resulting in the need for many more iterations of the equations of motion. Furthermore, the large number of iterations, each of which requires information to be passed between neighboring nodes, generates such a large quantity of communication overhead that a purely distributed implementation of the NN algorithm in a communication network is virtually precluded. Another disadvantage of a distributed model arises because low-energy solutions cannot be guaranteed, a property that has required the simulation of a large number of runs for each set of system parameters. In a distributed setting, it is not possible to determine the quality of a solution based on only local state information.

Nonetheless, we have further addressed the question of distributed operation to decrease the network complexity by reducing the number of neuron interconnections, while maintaining reasonably good results. Reference 15 restates the constraints in a distributed form and presents the results of simulations of the distributed model. It was found that although the “distributed”

model does not produce low congestion solutions as consistently as the centralized model, it is the only model that has found a minimum congestion-energy solution ($E_b^* = 133$) for the problem instance described in Section 5.5.2. In a Monte-Carlo simulation from 100 different states, the distributed model found three optimal solutions.

5.10 Conclusions on the Link-Neuron Model

In this section, we have presented a link-neuron NN model for solving the congestion-minimization problem, and we have shown that it is capable of determining reasonably good solutions to this problem. This model is quite similar to the path-neuron model discussed in Section 4, except that interactions between individual links are taken into account. This results in a much larger number of neurons and interconnections than were needed in the path-neuron model. A further complication is the need to add excitatory connections between neurons corresponding to links on the same path, to ensure that complete paths are formed.

Again, we used the method of Lagrange multipliers to dynamically determine the coefficients in the connection weights. After discovering a bias toward the selection of shortest paths, which interfered somewhat with the selection of minimum-congestion sets of paths, two methods were implemented in an attempt to improve performance. In the first, dummy neurons were added so that the resulting lengths of all paths between a given SD pair would be equal. In the second, additional compensatory bias was added to the neurons corresponding to links on non-shortest paths, so that the bias in favor of the shortest-path solutions would be eliminated. Both of these methods improved results; the dummy-neuron model performed somewhat better than the compensatory-bias model.

Although the solutions obtained by the link-neuron NN model are typically not as good as those produced by the path-neuron model, the link-neuron model does, in fact, represent a significant advance in our study of NN models of network problems. In particular, the ability of this model to generate complete paths by means of excitatory connections between neurons on the same path can be viewed as a first step toward the more general, and more difficult, routing problem in which the paths between each SD pair are not specified in advance; in this case, the NN must piece together complete paths from individual links. It may also be viewed as a first step toward solving the joint routing-scheduling problem, in which the time slot for the activation of each individual link along every path is to be determined. These problems are the subject of future research.

6.0 THE LINK-ACTIVATION SCHEDULING PROBLEM

The second class of problems we have studied concerns "link activation" or "scheduling" in multihop packet radio networks. Briefly stated, this problem is the determination of conflict-free transmission schedules that satisfy the specified communication requirements. In particular, given the connectivity graph of a radio communication network, a set of N_{sd} SD pairs, and a multihop path connecting each SD pair, determine a link-activation schedule of minimum length that will deliver one packet* between each source and the corresponding destination such that no scheduling conflicts occur. Before addressing the nature of scheduling conflicts, we define two versions of the scheduling problem.

6.1 Two Scheduling Problems: Nonsequential and Sequential Activation

In the case of nonsequential-activation scheduling (NAS), the goal is to schedule each link the specified number of times in each activation cycle, without regard to the order in which the links of any path are activated. In the more-difficult case of sequential-activation scheduling (SAS), the sequence of link activations along any multihop path must be preserved; i.e., for each

*The model can easily be extended to incorporate nonunit traffic requirements.

path, the link emanating from the source must be activated first, the next link second, and so on. The same total traffic (in terms of the number of activations of each link per cycle) is supported under both models; however, the NAS model may result in greater end-to-end packet delay because several cycles may be needed to transport a packet from source to destination. Actually, in most cases throughput (measured in terms of packets per slot) is greater when using NAS than when using SAS because removal of the sequentiality constraint often permits satisfaction of communication requirements in fewer slots.

6.2 Scheduling Conflicts

We distinguish three types of scheduling conflicts—primary, secondary, and sequence. These are described below.

Primary Conflicts

In this report we say that a *primary conflict* [5] occurs if:

1. A node has been scheduled to transmit and receive in the same slot; or
2. A node has been scheduled to receive from two or more nodes in the same slot; or
3. A node has been scheduled to transmit to two or more nodes in the same slot.

For rather restricted systems in which each node has a single transmitter and a single receiver, such conflicts prevent the correct reception of a packet. In systems with multiple receivers available at each platform, and/or a capability for successful simultaneous transmission and reception, the notion of primary conflict can be redefined easily to incorporate such less-restrictive constraints.

Secondary Conflicts

We say that *secondary conflicts* occur when additional signals are transmitted in the same neighborhood as the desired receiver, although not directed to that receiver. Whether or not they are destructive depends on the nature of the signaling (i.e., coding and modulation) scheme that is being used. For example, single-channel, narrowband systems normally cannot tolerate any secondary conflicts, unless the interfering signals are of considerably lower power than the desired signal. However, in spread-spectrum code-division multiple-access (CDMA) systems, several interfering signals transmitted on codes that are quasiorthogonal to that of the desired signal can typically be tolerated; the probability of packet error in frequency-hopping systems depends on the number of frequency bins over which the signal is hopped and on the properties of the error-control coding that is used [8]. In spread-spectrum systems that use orthogonal CDMA codes this is not a problem; any number of simultaneous transmissions can be tolerated. Since our main objective is to demonstrate the capability of the NN approach rather than the precise modeling of the interference, we assume here that such orthogonal spread-spectrum signaling is used, and thus the problem of secondary conflicts is not addressed.

Sequence Conflicts

The sequential scheduling requirement is a further restriction of the problem. We declare that a *sequence conflict* occurs if, within the schedule of activation, two or more links are activated out of order. As mentioned before, under the SAS model we require that along a SD path the links are activated in the order in which they appear in the path, starting with the link that emanates from the source node.

Thus, overall, we declare the occurrence of a scheduling conflict if there is a primary conflict, or if there is a sequence conflict. However, sequence conflicts are not addressed in the formulation of the NAS problem.

6.3 Communication Requirements

The testing ground of our approach to this problem has been the 24-node network shown in Fig. 2, which was used in some of the routing studies discussed in Sections 4 and 5. All of our scheduling simulations are based on scheduling the delivery of one unit of traffic from each of a specified set of source nodes to a specified set of destination nodes in this network. In each case a single path between each of the SD pairs is prespecified. Table 2 gives one example of a set of paths between 10 specified SD pairs. One packet is to be delivered from each source node to each destination node in the duration of every activation cycle. The resulting communication requirements are shown in Fig. 13, e.g., each link represented by a single arrow corresponds to a communication requirement of one packet, each double arrow to two packets, and each triple arrow to three packets.

Table 2. A Set of Paths Connecting 10 SD Pairs in the Network of Fig. 2

SD pair	[S, D]	Path (nodes traversed)						
1	[4, 24]	4	5	13	20	24		
2	[7, 17]	7	14	15	17			
3	[9, 16]	9	12	13	19	14	15	16
4	[1, 19]	1	4	5	13	19		
5	[5, 11]	5	6	11				
6	[21, 6]	21	22	20	13	5	6	
7	[1, 10]	1	2	3	6	8	9	10
8	[3, 18]	3	4	7	14	15	18	
9	[2, 12]	2	4	7	12			
10	[14, 8]	14	7	11	8			

Examples of primary conflicts that may occur in the network shown in Fig. 13 include the following:

- If node 1 is scheduled to transmit to node 2 in the same slot in which node 2 is scheduled to transmit to node 3, node 2 is scheduled to both transmit and receive in the same slot.
- If nodes 2 and 3 are both scheduled to transmit to node 4 at the same time, node 4 is scheduled to receive from two nodes in the same slot.
- Node 1 is scheduled to transmit to two nodes in the same slot if it is scheduled to transmit to both nodes 2 and 4 at the same time.

An example of secondary conflict arises if there are simultaneous transmissions from node 2 to 3 and from node 1 to 4. Although the message transmitted by node 2 is intended for node 3, it also collides with node 1's transmission because node 4 is within range of node 2. Whether or not this interference is destructive depends on the type of signaling that is used. In a narrowband system, it is generally destructive. In a system with quasiorthogonal CDMA codes, it will most likely not be. However, if nodes 5 and 7 (which are also neighbors of node 4) also transmit at the same time, the combined effect of their interference may raise the packet error probability significantly.

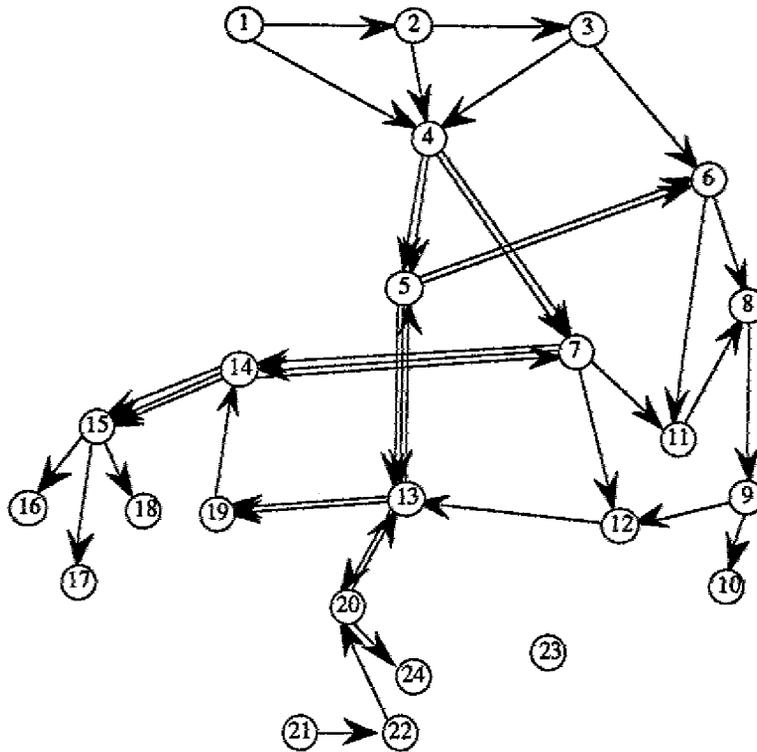


Fig. 13 — Link communication requirements for the set of paths listed in Table 2

6.4 Complexity Issues

It is easily recognized that the link-activation scheduling problem is equivalent to the graph edge-coloring problem, which is known to be NP-complete [34]; time slots in the former correspond to colors in the latter. Thus our problem is NP-complete, a property that suggests that the Hopfield NN approach is reasonable. We point out that versions of the scheduling problem exist that are solvable by polynomial algorithms [1, 35]. In such problems, fractions of packets are permitted to be transmitted in shorter slots, unlike our problem in which we require complete packets of fixed length to be transmitted in full-length slots. Such polynomial scheduling problems are related to the determination of the fractional chromatic index of a graph (known to be a polynomial problem), whereas the discrete packet scheduling problem corresponds to the determination of the chromatic index of a graph (which is NP-complete).

7.0 BOUNDS AND HEURISTICS FOR MINIMUM-LENGTH SCHEDULING

We have noted that the problem of determining a minimum-length schedule that satisfies a specified end-to-end communication demand, in almost all of its forms, is NP-complete [1, 2, 7]. Therefore, unless a schedule's length is equal to a known lower bound on the schedule length, there is no way to determine whether the schedule is optimal (other than exhaustive search, which is practical only for relatively small problems). Thus, a reasonably tight lower bound on the length of the minimum-length schedule is needed to aid in determining whether a schedule meets the desired objective of being (at least) nearly minimum in length. The lower bounds on the minimum length of nonsequential-activation schedules that are established in Refs. 1 and 6 are summarized in Section 7.1.1. The lower bound on the minimum length of sequential-activation schedules, which is introduced in Section 7.1.2, is a bound that we have developed to address the restrictions introduced by the sequential scheduling requirement.

7.1 A Lower Bound on the Schedule Length

7.1.1 A Lower Bound on the Minimum Length of a Nonsequential-Activation Schedule

For the case of nonsequential-activation scheduling (NAS), we use the lower bound on schedule length developed by Post et al. [6]. This is given by

$$B_{nas} = \max\{B_d, B_\Delta\},$$

where

B_{nas} is a lower bound on the schedule length without the sequence constraint,

$$B_d = \max_{\forall \text{nodes } n} \{\text{deg}(n)\},$$

$$B_\Delta = \max_{\forall \text{nodes } n, o, p} \{f(n, o) + f(n, p) + f(o, p)\},$$

$f(n, o)$ = the number of packets that must traverse
the physical link (n, o) that connects nodes n and o .

The degree of a node n (denoted $\text{deg}(n)$) is defined to be the sum of the number of packets that flow into it plus the number of packets that flow out of it. The inclusion of the expression B_Δ tightens the lower bound B_{nas} by detecting the presence of three-node cycles, which may cause the minimum schedule length Λ^* to be greater than B_d . For example, the three-node cycle shown in Fig. 14 clearly has $B_d = 2$, $B_\Delta = 3$, and $\Lambda^* = 3$.

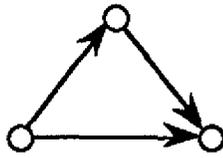


Fig. 14 — A three-node cycle

7.1.2 A Lower Bound on the Minimum Length of a Sequential-Activation Schedule

For the sequential-activation scheduling (SAS) problem we have developed a lower bound on the schedule length, denoted B_{sas} , that addresses the additional restrictions introduced by the SAS requirement. The NAS bound B_{nas} is a lower bound on the minimum sequential-activation schedule length, which can be tightened by noting that the length of a conflict-free sequential-activation schedule can be no shorter than the length of the longest path in the network. The observation that the positions that moderate- and high-degree nodes hold on the paths can increase the minimum schedule length is also used to tighten B_{sas} . The new bound is given by

$$B_{sas} = \max\left\{B'_d, B_\Delta, \max_{i=1}^{N_{sd}} (L(i))\right\},$$

CONTENTS

1.0	INTRODUCTION.....	1
1.1	A Hopfield NN for the Minimization of Congestion	2
1.2	A Hopfield NN for Link Activation Scheduling	3
1.3	Outline of the Report	3
2.0	ROUTING AND SCHEDULING PROBLEMS IN PACKET RADIO NETWORKS	4
3.0	A HOPFIELD NETWORK TO MINIMIZE CONGESTION.....	9
3.1	The Problem.....	9
3.2	Neural Network Model	9
3.3	Congestion Energy	11
3.4	Incorporation of Constraints into the Energy Function	12
3.5	Determination of Connection Weights and Bias Currents	14
3.6	Equations of Motion.....	14
4.0	PERFORMANCE EVALUATION USING THE PATH-NEURON MODEL.....	15
4.1	Basic Simulation Issues.....	15
4.2	Algorithm for the Determination of Path Sets	16
4.3	A Binary Interpretation of the Analog State: An Instantaneous State Description	17
4.4	Alternative Metrics.....	17
4.5	Exhaustive Search as a Means to Assess System Performance	18
4.5.1	A Shortest-Path Heuristic	18
4.6	An Example 24-Node Network.....	18
4.6.1	The Use of Mean-Field Annealing	20
4.7	Use of the Method of Lagrange Multipliers to Determine Connection Weights	20
4.7.1	An Alternate Formulation with Multiple Lagrange Multipliers.....	21
4.8	Simulation Results of a 24-Node Network Using the Method of Lagrange Multipliers	21
4.8.1	Results Obtained Using the LM Method.....	21
4.8.2	Results Obtained Using the MLM Method	22
4.8.3	Combined use of MFA and LM Techniques	25
4.8.4	Use of Simulated Annealing to Search for the Global Minimum	25
4.9	Simulation Results of a 100-Node Network Using the Method of Lagrange Multipliers	25
4.10	Nonunit Traffic and Alternate Routing.....	27
4.10.1	Triplicate 100-Node Network.....	27
4.10.2	Nonuniform Traffic in the 100-Node Network.....	28
4.11	A Modified Form of the Congestion-Energy Function.....	28
4.12	Conclusions on the Path-Neuron Model.....	29
5.0	A LINK-NEURON NN FORMULATION FOR THE MINIMIZATION OF CONGESTION.....	30
5.1	The Basic Link-Neuron Model.....	30
5.2	Congestion Energy	32
5.3	Incorporation of Constraints into the Energy Function	33
5.4	Connection Weights and the Equations of Motion.....	35
5.5	Simulation Procedure	35

where

$$B'_d = \max_{\forall \text{ nodes } n} \{ \deg(n) + F_a(n) + L_a(n) \},$$

$F_a(n)$ = The minimum number of slots required prior to the first legal activation of node n ,

$L_a(n)$ = The minimum number of slots required to complete the activation of all paths after the last activation of node n .

Since B_Δ and the maximum path length are clearly bounds for Λ^* , it suffices to show that B'_d is also a lower bound. The proof that B'_d is indeed a bound for Λ^* proceeds simply as follows.

For each node n , let \mathcal{L}_n denote the set of (unit-) traffic carrying links that include node n ; let us also refer to the j th link of the path between SD pair i as the ij -link. Note that

$$F_a(n) = \left\{ \min_{ij \in \mathcal{L}_n} (j) \right\} - 1,$$

$$L_a(n) = \min_{ij \in \mathcal{L}_n} (L(i) - j),$$

where $L(i)$ is the length, in hops, of the path between SD pair i .

Clearly, to activate links sequentially, for each node n a minimum of $F_a(n)$ slots must be used prior to activating any of the links in \mathcal{L}_n . Additionally, a minimum of $\deg(n)$ slots are required to complete the activation of the links in \mathcal{L}_n without generating a primary conflict. Finally, an additional number of slots are needed to complete the path corresponding to the link in \mathcal{L}_n that was activated last, and this is at least $L_a(n)$. By maximizing over all nodes we get the expression for B'_d above.

Thus, B_{sas} includes some of the effects of the additional SAS constraint. Besides tightening the bound by capturing the length of the longest path, it also tightens the bound by considering those nodes of moderate and high degree that cannot be legally activated in the early and/or later slots because of their position in the paths. For example, node 13 in Fig. 13 has $\deg(13) = 8$. Because its earliest appearance in any path is as a receiver in the second link in the paths between SD pairs 1 and 3 (see Table 2), $F_a(13) = 1$, i.e., its first legal activation can occur no earlier than the second slot. Its latest appearance is as the transmitter in the last link of the path between SD pair 4; therefore, $L_a(13) = 0$. For the network shown in the figure, the value of $B_{sas} = B'_d = \deg(13) + F_a(13) + L_a(13) = 9$ is, in fact, a tight bound on the sequential schedule length, i.e., $B_{sas} = \Lambda^*$.

7.2 Heuristics for Scheduling

To further assess the quality of NN solutions, we have also considered two forms of a "biased-greedy" heuristic similar to that developed by Post et al. [36]. The NAS heuristic provides optimal or near-optimal nonsequential-activation schedules (schedules with length equal to or slightly greater than the NAS bound B_{nas} , respectively) for most instances of the NAS problem. The SAS heuristic generally provides sequential schedules with lengths one or two slots greater than the SAS bound B_{sas} . Section 7.3 summarizes the performance of these heuristics. Reference 16 provides a more complete discussion of performance. Note that both heuristics are deterministic; thus each produces a unique set of link activations for a specific set of paths.

7.2.1 The NAS Heuristic

For the nonsequential-activation scheduling problem, the heuristic first creates a list of all the links in the network and assigns to each link a bias equal to the sum of the nodal degrees of the two nodes on which the link is incident. In this setting, a link corresponds to one unit of traffic that must traverse one hop. Thus, if four units of traffic must be passed between adjacent nodes i and j , four parallel links connect the nodes. The list of links is then sorted in descending order based on the bias. The algorithm attempts to schedule each link in the first slot by descending through the list and activating and removing each link from the list that does not share a node with a previously activated link. When the bottom of the list is reached, the slot number is incremented, and the process is repeated; the algorithm descends through the remaining list, activating and removing each link that does not share a node with a link that was previously activated in the present slot. The process is repeated until every link has been assigned a slot and the list is empty.

7.2.2 The SAS Heuristic

The SAS heuristic, which we introduce in this report, is the first algorithm we know of for sequential link activation in radio networks.* This algorithm is essentially the same as the NAS heuristic, except that the list of links for each slot is restricted to those links that are eligible for activation in that slot, and the basis on which the links are sorted is slightly different. In the first slot, only the first links (the links emanating from a source) from each path are included in the link list because they are the only links eligible for activation. The list of links for the second slot includes only the first links that were not scheduled in the first slot and the second links in paths whose first links were scheduled in the first slot. In general, the list of links that are eligible for activation in the k th slot contains no more than N_{sd} links, where N_{sd} denotes the number of SD pairs; each link in the list is less than or equal to the k th link in its path, and all of the links in the path that precede a listed link have been activated in an earlier slot.

In the SAS heuristic, the bias assigned to each link is slightly altered from that used in the NAS heuristic to reflect the emphasis on sequential scheduling. Link ij (the j th link in the path between SD pair i) between nodes t and r is assigned a bias given by

$$bias_{ij} = \max\{\deg(t), \deg(r)\} + \frac{L(i) - j}{\phi},$$

where ϕ , which was arbitrarily set equal to 10, can be used to shift the priority from activating nodes of high degree to activating those links furthest from the destination. The eligible links at each slot are sorted based on their bias and are "greedily" activated as in the NAS heuristic.

7.3 Performance of Scheduling Heuristics

7.3.1 Performance of the NAS Heuristic

We again consider the 24-node network shown in Fig. 2, which was discussed earlier in conjunction with the routing-to-minimize-congestion problem. As discussed in Section 4.6, a total of 52 maximally node-disjoint paths between 10 SD pairs were found by using the path-selection algorithm discussed in Section 4.2. There are 3,981,312 different sets of 10 single paths between each of the 10 SD pairs that can be extracted from the 52 paths listed. Applying the NAS heuristic to each of these path sets has demonstrated that route selection greatly impacts the minimum obtainable schedule length. The minimum value of the bound for the nonsequential-activation schedule length ($B_{nas} = 7$ slots) was obtained for 858 path sets; the NAS heuristic was able to find a minimum-length schedule for 481 of these 858 path sets. As noted earlier, it is not known a priori whether a schedule of length B_{nas} actually exists; however, our NAS NN model has, in fact, found seven-slot schedules for all of the path sets with $B_{nas} = 7$ (discussed in Section 10.5). The

*Mukherji [20] presents an algorithm for a similar problem in wire-line networks.

heuristic schedules have lengths which are generally near B_{nas} ; 97.86% of the heuristically determined schedules were easily verified to be optimal because they have lengths equal to B_{nas} . The heuristic schedules that have lengths greater than B_{nas} exceed the bound by an average of 1.12 slots.

The set of 858 path sets that have $B_{nas} = 7$ was divided into two disjoint subsets. The first subset consists of the 377 path sets that the NAS heuristic was unable to schedule in seven slots. For future reference, this set is labeled “(7, >7)” ($B_{nas} = 7$, heuristic schedule length > 7). The second set consists of the 481 path sets that the NAS heuristic was able to schedule in seven slots, and is labeled “(7, 7).” Our NN model was able to find schedules of length seven for all of the path sets with $B_{nas} = 7$, i.e., for all path sets in the union of (7, >7) and (7, 7). Partial listings of sets (7, >7) and (7, 7) are given in Tables A5 and A6, respectively, in Appendix A of Ref. 16.

7.3.2 Performance of the SAS Heuristic

The SAS heuristic was similarly applied to each of the nearly 4 million different path sets. It was observed that it does not produce schedules that can easily be verified to be optimal (i.e., schedules whose length matches the bound B_{sas}) as frequently as the NAS heuristic; 31.34% of the schedules found by the SAS heuristic had lengths equal to B_{sas} , whereas 97.86% of the NAS heuristic schedule lengths matched their corresponding B_{nas} value. The SAS heuristic schedules whose length exceeded B_{sas} were an average of 1.96 hops longer than the bound. A total of 1862 sets of paths were found that had $B_{sas} = 8$, but the SAS heuristic was able to schedule only eight of these path sets in eight slots. These eight path sets are listed in Table A4 in Appendix A of Ref. 16. However, the apparently poor performance of the heuristic results at least partially from the looseness of the bound B_{sas} ; that is, some (perhaps many) of the heuristically found schedules with length greater than B_{sas} may actually be minimum in length. Simulations of the SAS NN model, which are discussed in detail in Section 11, have typically yielded sequential-activation schedules with lengths between B_{sas} and the value of the heuristic schedule length. This indicates that the disparity between B_{sas} and the heuristic schedule lengths is the result of a combination of the loose bound and the inability of the SAS heuristic to consistently find minimum-length schedules.

8.0 A NEURAL NETWORK MODEL FOR LINK-ACTIVATION SCHEDULING

As in the case of the routing model discussed in Section 3, the first step in formulating a Hopfield NN model is defining neurons that correspond to binary variables in the system that is being modeled. In this section, we consider a Hopfield NN in which, for every link in the predetermined paths connecting the N_{sd} SD pairs, one neuron is defined for each slot.* For example, Fig. 15(a) shows a very simple six-node network with one path between each of two SD pairs, and Fig. 15(b) shows a possible configuration (depending on the implementation of the constraints) of the corresponding four-slot NN model designed for the SAS problem. A triple index is now used to specify the neurons, i.e., neuron ijk represents time slot k for the j th link in the path connecting SD pair i . The lower horizontal plane defined by neurons 211, 231, and 234 contains all of the neurons that represent links in the path connecting SD pair 2. The parallel upper horizontal plane contains the neurons that represent the two links in the path connecting SD pair 1. Each of the parallel vertical planes defined by neurons with the same last digit corresponds to a time slot, e.g., the plane defined by neurons 211, 231, and 121 corresponds to time slot 1. The connections shown are mutually inhibitory; thus any two neurons that are directly connected to each other cannot both be “on” (i.e., have a value of 1) in a valid solution. The solid lines are present in both the NAS and SAS models, whereas the dotted lines are present in only the SAS model.†

*Alternate formulations are also possible. Some of the other possibilities are discussed in Section 8.3.

†Additional connections are needed to help satisfy system constraints. This figure is meant to provide a schematic representation of the principles involved in the development of a NN model; it is not meant to be complete.

The Lyapunov energy function can be written in terms of connection weights and bias currents as

$$E_{total} = -\frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{l=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{m=1}^{L(l)} \sum_{k=1}^{\Lambda} \sum_{n=1}^{\Lambda} T_{ijk,lmn} V_{ijk} V_{lmn} - \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk} I_{ijk} \quad (13)$$

where $T_{ijk,lmn}$ is the strength of the connection between neurons ijk and lmn (it is positive if the connection is excitatory, and negative if it is inhibitory). I_{ijk} is the bias current applied to neuron ijk , Λ is the number of slots, and $L(i)$ is the length of the path between SD pair i in number of hops. The total number of neurons N is given by

$$N = \Lambda \sum_{i=1}^{N_{sd}} L(i).$$

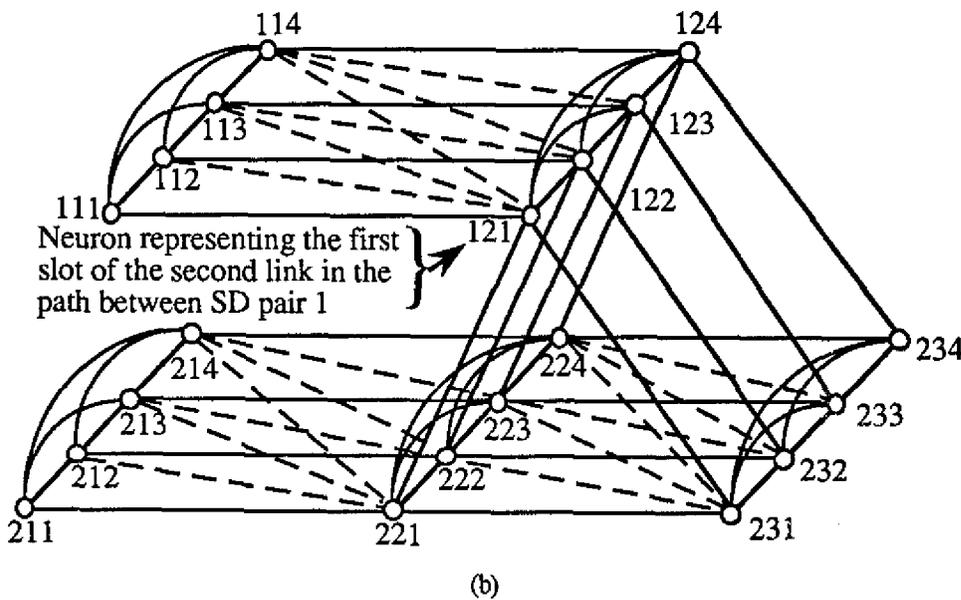
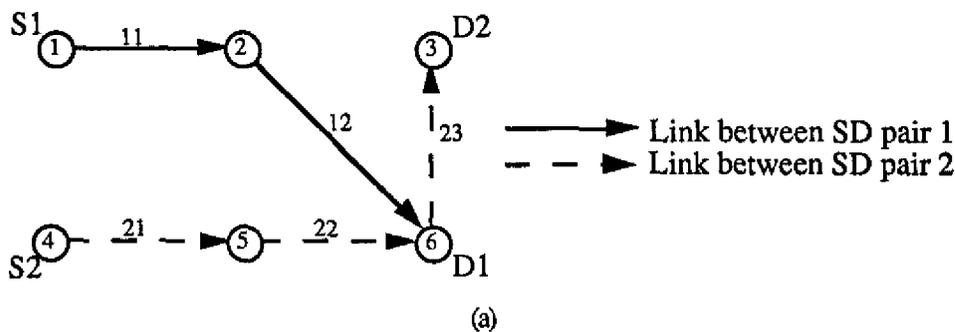


Fig. 15 — An example network: (a) shows a six-node communication network; (b) shows the corresponding four-slot NN model

It is customary and advantageous in certain cases to alter somewhat the approach to the minimization problem. Instead of trying to minimize the schedule length directly, for example, we may ask a series of binary questions like: "Is there a schedule of length Λ that satisfies all constraints (of conflict-free transmissions, here)?" for $\Lambda = \Lambda_o, \Lambda_o + 1, \dots$, where Λ_o is set equal to the appropriate lower bound on the schedule length, i.e., B_{sas} for sequential-, or B_{nas} for nonsequential-activation scheduling.

Typically, a number of runs are performed from different initial states of the NN. If a schedule of Λ length cannot be found, Λ is incremented by one and the process is repeated. In this formulation of the link-activation problem, for each value of Λ there is no objective function E_{obj} to be minimized. Since there is no objective function to be minimized directly in the modified, constraint-based NN model, the energy function given by Eq. (13) can now be rewritten as

$$E_{total} = \sum_{c=1}^4 \lambda_c E_c - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk}. \quad (14)$$

where the E_c 's are the constraint-energy terms, discussed in Section 8.1, and the λ_c 's are the corresponding Lagrange multipliers. The goal is simply to determine the existence of a schedule of given length that does not violate a number of constraints. These constraints are established to prevent transmissions that would result in collisions, and, in the case of sequential-activation scheduling (SAS), to prevent scheduling the transmission of a packet before it is received. Whether or not the minimum length is achieved depends on how successful the NN is in satisfying these constraints for the different values of Λ .

As in our studies of the routing model, we have found that, by allowing the λ_c 's to vary dynamically along with the system state as in the classical method of Lagrange multipliers, we obtain significantly better NN performance. Therefore, this method is used in all of our NN models for the scheduling problem.

8.1 Formulation of the Constraint-Energy Terms

We have studied several versions of the constraints for this problem. In this section we present a "basic" version of the set of constraint formulations; in Section 8.3 we examine the variations of the basic version that have yielded improved performance.

Each of the constraints generates a term in Eq. (14) that must be equal to zero when the constraint is satisfied. This is simply the usual Lagrange multiplier method for constrained optimization.

Constraint 1 — *Activate no links that cause primary conflicts:*

$$E_1 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{l=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{m=1}^{L(l)} \sum_{k=1}^{\Lambda} |A_{ij} \cap A_{lm}| V_{ijk} V_{lmk} = 0, \quad (15)$$

where A_{ij} denotes the j th link in the path between SD pair i , and

$$|A_{ij} \cap A_{lm}| = \begin{cases} 1, & \text{if } ij \neq lm, \text{ and links } A_{ij} \text{ and } A_{lm} \text{ share one or two nodes} \\ 0, & \text{if links } A_{ij} \text{ and } A_{lm} \text{ are disjoint or } ij = lm \end{cases}$$

The implementation of this constraint provides strictly inhibitory contributions to the connection weights: Two conflicting neurons (i.e., two neurons that represent adjacent links in a common slot) mutually exert on each other a negative force that is proportional to the product of their output voltages.

Constraint 2 — *Activate each link once and only once:*

$$E_2 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \left(\sum_{k=1}^{\Lambda} V_{ijk} - 1 \right)^2 = 0. \quad (16)$$

This term is zero when exactly one slot is chosen for each link. In other words, it is zero when, for every link in the network, exactly one neuron out of the set of neurons that represent different slots for that link has an output voltage of 1, and the neurons representing all of the other slots have output voltages of 0. This constraint can be either excitatory or inhibitory. Loosely speaking, the effect of this term is excitatory if the majority of links have less than one active neuron and inhibitory if the majority of links have more than one active neuron.

Constraint 3 — *Activate a total of N_x neurons:*

$$E_3 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk} - N_x \right)^2 = 0, \quad (17)$$

where $N_x = \sum_{i=1}^{N_{sd}} L(i)$ (assuming one unit of traffic is to be delivered between each SD pair) is the total number of transmissions required to satisfy the communication requirements. This term vanishes when exactly one slot has been selected for each link. Like constraint 2, it can be either excitatory or inhibitory. Although this constraint appears to be redundant (because satisfaction of the second constraint guarantees that it is satisfied as well), its inclusion in the energy equation is helpful in achieving convergence to valid solutions. As noted in our discussion of the routing model in Section 3, the use of such seemingly redundant constraints is common in Hopfield network models. Satisfaction of constraint 2 alone (along with a mechanism to guarantee that all neurons take on binary values) actually suffices to constrain the number of activations. However, constraint 3 is useful because it imposes a greater penalty when an incorrect number of neurons in the entire NN are set to 1. This is because it is a quadratic form centered about N_x , whereas constraint 2 contains N_x quadratic forms each centered about 1.

Constraint 4 — *Sequentially activate the links in each path (i.e., link ij must be activated before link im , for $m > j$):*

$$E_4 = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)-1} \sum_{m=j+1}^{L(i)} \sum_{k=1}^{\Lambda} \sum_{n=1}^{m-j+k-1} V_{ijk} V_{imn} = 0. \quad (18)$$

This term provides a positive contribution to the energy function when two neurons that represent an out-of-sequence activation of the links in a path have nonzero output voltages. It turns out that it represents purely inhibitory contributions to the connection weights. In applications where it is not necessary to maintain the sequential order of link activation, i.e., in the NAS problem, the E_4 term is not included in the energy equation.

8.2 Determination of Connection Weights, Bias Currents, and Equations of Motion

The connection weights and bias currents are determined in a manner similar to that used for the routing problem, which was discussed in Section 3.5. Substituting the constraint-energy expressions into Eq. (14) results in an expression in which the quadratic terms correspond to the connection weights and the linear terms correspond to the bias currents. The equations of motion are then easily determined. Reference 16 provides complete expressions.

Use of the Method of Lagrange Multipliers to Determine Connection Weights

As was done in the routing model, we again permit the Lagrange multipliers (LM) λ_c to vary dynamically as follows:

$$\lambda_c(n+1) = \lambda_c(n) + (\Delta t)_{\lambda_c} E_c(n),$$

where the time constant $(\Delta t)_{\lambda_c}$ may be a different value for each of the λ_c 's. Note that, since $E_c \geq 0$, the quantities λ_c are monotonically nondecreasing. Typically, the Lagrange multipliers are assigned initial values of 1.

Multiple Lagrange Multipliers

Examination of the second constraint energy term E_2 , which requires that each link be activated once and only once, suggests that it may again be advantageous to use the method of multiple Lagrange multipliers, which was introduced in Section 4.7.1. We define a separate Lagrange multiplier for the constraint applied to each link. Doing so would increase the LMs associated with those links that were unsuccessful in activating exactly one neuron.

The second constraint formulation may be rewritten as

$$E_2 = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} e_{2ij} = 0,$$

where each of the terms of the form

$$e_{2ij} = \frac{1}{2} \left(\sum_{k=1}^{\Lambda} V_{ijk} - 1 \right)^2 = 0$$

is an equality constraint specifically for the j th link between SD pair i . Now Lagrange multipliers are defined to correspond to each of the e_{2ij} 's, and they evolve as

$$\lambda_{2ij}(n+1) = \lambda_{2ij}(n) + (\Delta t)_{\lambda_{2ij}} e_{2ij}(n).$$

8.3 Variations of the Basic Scheduling NN Model

8.3.1 *The Condensed NAS Model*

Although the NAS model attempts to schedule the communication requirements for a set of multihop paths, this formulation of the problem permits a decomposition that essentially transforms it to a one-hop scheduling problem similar to that considered in Ref. 2. As such, it is no longer necessary to associate each link activation with a particular path or SD pair. Therefore, the size of the NN model can be reduced by creating Λ neurons for each *physical* link, rather than creating Λ neurons for each unit of traffic on each physical link as was done in the basic model. Then constraint 2 is altered to require that each physical link p be activated $N_x(p)$ times, where $N_x(p)$ is the number of units of traffic that must traverse physical link p . We refer to this as the condensed NAS model. By denoting the modified formulation of the condensed NAS model with a superscript c , the resulting constraint can be expressed as

$$E_2^c = \frac{1}{2} \sum_{p=1}^{N_{pl}} \left(\sum_{k=1}^{\Lambda} V_{pk} - N_x(p) \right)^2 = 0,$$

where N_{pl} is the number of physical links in the network.* Differentiating E_2^c with respect to V_{pk} , as suggested by Eq. (5), yields the corresponding equation of motion term:

*This expression can easily be converted to the MLM format by applying an approach similar to that used in Section 8.2. That is, for each physical link p , a Lagrange multiplier is defined to correspond to an equality constraint that requires link p to be activated exactly $N_x(p)$ times.

$$\frac{-\partial E_2^c}{\partial V_{pk}} = N_x(p) - \sum_{n=1}^{\Lambda} V_{pn}.$$

Note that now the neurons are double indexed (a triple index was needed in the basic model) so that neuron pk represents the p th physical link at the k th time slot. Although the other constraints remain virtually unchanged, the modified structure of the condensed NAS NN model requires that each of the constraint-energy terms be rewritten to conform to the new doubly indexed neuron notation. This gives the following modified total energy equation:

$$E_{total}^c = \frac{\lambda_1}{2} \sum_{p=1}^{N_{pl}} \sum_{\substack{q=1 \\ q \neq p}}^{N_{pl}} \sum_{k=1}^{\Lambda} |A_p \cap A_q| V_{pk} V_{qk} + \frac{\lambda_2}{2} \sum_{p=1}^{N_{pl}} \left(\sum_{k=1}^{\Lambda} V_{pk} - N_x(p) \right)^2 + \frac{\lambda_3}{2} \left(\sum_{p=1}^{N_{pl}} \sum_{k=1}^{\Lambda} V_{pk} - N_x \right)^2 - I \sum_{p=1}^{N_{pl}} \sum_{k=1}^{\Lambda} V_{pk},$$

where we now have

$$|A_p \cap A_q| = \begin{cases} 1, & \text{if physical links } A_p \text{ and } A_q \text{ share one or two nodes} \\ 0, & \text{if physical links } A_p \text{ and } A_q \text{ are node disjoint} \end{cases}$$

Because this is strictly a NAS model, the sequentiality constraint E_4 has been omitted.

The communication requirements of Table 2 represent a typical example considered in this report. Scheduling these requirements with the basic NAS model requires a NN consisting of (41 required transmissions \times 8 slots $=$) 328 neurons. The use of the condensed NAS model results in a reduction in the number of neurons to (30 physical links \times 8 slots $=$) 240. Since the number of computations required at each iteration is approximately proportional to the square of the number of neurons, this reduction in the number of neurons markedly reduces the NN complexity. Furthermore, extensive simulation results, which are discussed in Section 10, have shown that the condensed NAS model consistently delivers better performance than the basic model.

8.3.2 The Reduced SAS Model

The condensing process just discussed for the NAS model cannot be applied to the SAS model. Under the SAS operation, it is necessary to keep track of the SD pair for which each link is activated, so that the sequential order of link activations over every SD pair can be maintained. However, the number of neurons can be reduced by eliminating from consideration those neurons whose activation would not be consistent with the sequential-activation constraint. For example, the second link of a path cannot be activated in the first slot of the schedule, etc. We refer to the resulting model as the reduced SAS model.

When the sequential-activation constraint is enforced, the set of slots in which link ij can be legally activated is a subset of the set of Λ slots that form the schedule. If, for example, the path between SD pair x has length $L(x) = 5$, and the sequential schedule length is $\Lambda = 6$, the first link of the path, link x_1 , can be legally activated only in one of the first two slots; activation in a later slot must result in either a primary conflict, a sequence conflict, or both, or the failure to schedule the activation of every link. Similarly, the second link in path x , link x_2 , can be legally activated only in one of slots 2 or 3, and link x_j can be legally activated only in one of slots j or $j+1$. In general, link ij can be legally activated only in one of slots j through $j + X_i$, where $X_i = \Lambda - L(i)$. Therefore, the neurons that represent illegal slots (i.e., neurons ijk , $k < j$ or $k > j + X_i$) can be

eliminated without reducing the admissible* solution space. The elimination of these neurons, besides reducing the complexity of the NN, also removes a significant number of potential local minima that may trap the NN in an inadmissible solution.

Thus in the reduced SAS model, only $X_i + 1$ (rather than Λ) neurons are created for each link in the path between SD pair i . The constraint formulations of the basic model are virtually unchanged; only the indices of the summations over the number of slots must be changed to reflect the absence of neurons that correspond to illegal slots. (Alternatively, the removed neurons may be considered to have output voltage values of zero, in which case the indices of the basic model equations need not be altered.)

Figure 16 shows the reduced SAS model for the network of Fig. 15(a). Comparing the model in Fig. 16 with that shown in Fig. 15(b) illustrates that, even in this very simple problem, a significant reduction in both the number of neurons, and the number of connections, is obtained by using the reduced SAS model. The solid lines in Fig. 16 represent the neuron interconnections that enforce the first two constraints. Those that are parallel to the y axis enforce the first constraint, which prohibits primary conflicts. The interconnections that are parallel to the *time* axis enforce the second constraint (activate each link exactly once). The dashed lines represent the interconnections that enforce the sequentiality constraint. The interconnections between every pair of neurons that enforce the third constraint (activate a total of N_x neurons) are not shown.

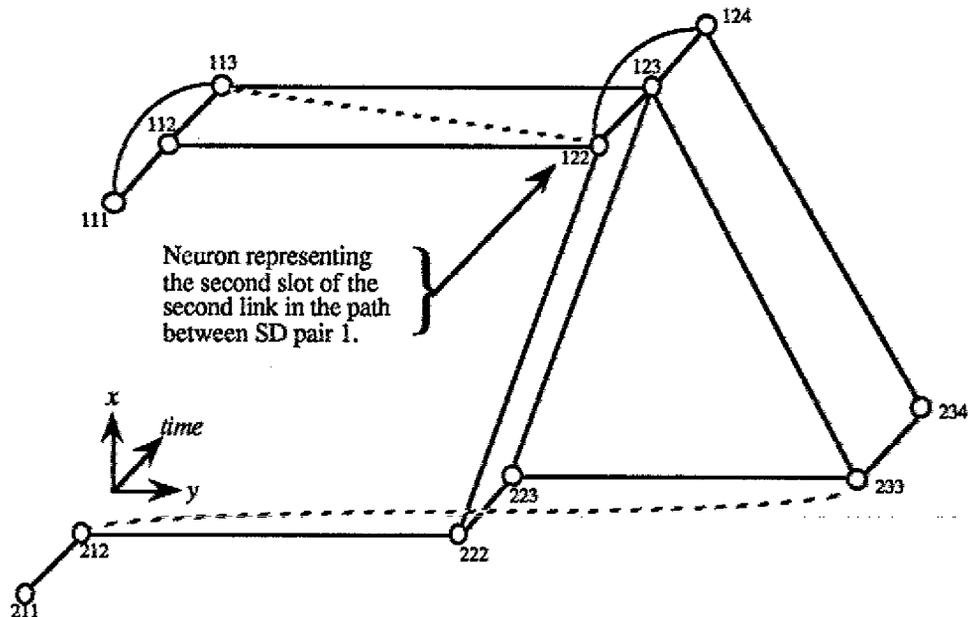


Fig. 16 — The reduced SAS model for the network shown in Fig. 15(a)

The minimum length of a sequential-activation schedule that satisfies the communication requirements of Fig. 15(a) is four slots. Table 3 lists one such schedule. In the table, entries are shown only for the slots that are represented by a neuron in Fig. 16. The blank cells represent the slots in which the link can never be activated in an admissible sequential-activation schedule. Thus, the table also aids in understanding the structure of the reduced SAS NN model. Since each of the cells in the table corresponds to a neuron, the cells are addressed by a triple index in the same manner as the neurons; i.e., cell i,j,k represents slot k of the j th link in the path between SD pair i . A cell entry of X denotes a link activation, o denotes an open slot (i.e., the link can be activated without conflict), b denotes a blocked slot in which activation will result in a primary

* An admissible solution or schedule is one that satisfies all of the constraints.

conflict, and *i* denotes an ineligible slot (i.e., a slot in which activation of the link must cause a sequence conflict). For example, the *b* entry in cell 1,2,4 indicates that activation of link 1,2 (the second link in the path connecting SD pair 1) is blocked in the fourth slot by the scheduled activation of link 2,3 in this slot. Therefore, activation of link 1,2 in the fourth slot would cause a primary conflict. A close examination of the table reveals that link 1,1 can be activated in slot 3 without causing a primary conflict. However, because link 1,2 is blocked in slot 4, there is no way that a unit of traffic received in slot 3 can be relayed by link 1,2 in this four-slot cycle; i.e., as a result of the blockage of link 1,2 in slot 4, activation of link 1,1 in slot 3 *must* result in a sequence conflict. Therefore, we declare link 1,1 ineligible for activation in slot 3, and enter an *i* in cell 1,1,3.

Table 3. An optimal schedule for the network of Fig. 15(a) (X = link activation, o = open for activation, b = link blockage, i = ineligible for activation owing to the blockage of an adjacent link)

Path	Link	Indices (SD, link)	Slot			
			1	2	3	4
1	(1->2)	1,1	X	b	i	
	(2->6)	1,2		X	b	b
2	(4->5)	2,1	X	o		
	(5->6)	2,2		b	X	
	(6->3)	2,3			b	X

8.3.3 The Adjustable-Length Model

Our approach with the basic, the condensed NAS, and the reduced SAS NN models, like the approach in Ref. 2, has been to attempt to solve the problem of determining the minimum-length admissible schedule that satisfies a given set of end-to-end communication requirements by repeatedly posing the binary question "Can a schedule be found that satisfies the given end-to-end demand in Λ slots?" for different values of Λ . Starting with Λ equal to a known lower bound on the schedule length, the question is repeated as Λ is incremented until an admissible schedule is found. Since the NN model only guarantees convergence to local minima of the energy function, the failure of any NN simulation to find a Λ -slot admissible schedule does not preclude the existence of such a schedule. Therefore multiple NN runs from different initial conditions must be run for a value of Λ that is too small before it may be concluded with any degree of confidence that a larger value is required. In the NAS problem a schedule can usually be found for the first value of Λ because the NAS bound B_{nas} is tight for many networks and communication requirements (the bound is tight for all the topologies we have examined). In the SAS problem, however, multiple runs for several values of Λ are usually required because the SAS bound B_{sas} is generally not tight. These considerations, coupled with the frustration of multiple inadmissible solutions, led to the development of the "adjustable-length" model.

The adjustable-length model attempts to solve the scheduling problem without resorting to the binary-question approach. Ideally, every run of the adjustable-length NN model will deliver a conflict-free schedule of short, but not necessarily optimum, length. This is achieved by implementing the basic model (or the condensed NAS, or the reduced SAS variants) with an obviously excessive number of slots (e.g., use $\Lambda = 1.5 B_d$,* a known upper bound on the minimum schedule length for NAS [1]), and penalizing activations that occur in later slots by using an additional energy equation term, which we denote E_5 . Thus, link activations are encouraged in the early slots, and unused slots are discarded to yield a nearly minimum, if not minimum, length admissible schedule. Reference 16 provides a complete description and discussion of performance results.

*The bound B_d is the maximum nodal degree in the network, as discussed in Section 7.1.1.

5.5.1	Termination Criteria.....	36
5.5.2	Network Description.....	36
5.6	Miscellaneous Considerations:.....	36
5.6.1	The Need for Additional Bias Currents.....	36
5.6.2	The Use of MLM for the "Complete-Path" Excitatory Connections.....	37
5.7	Methods to Overcome a Detrimental Preference for Short Paths.....	37
5.8	Use of Simulated Annealing with the Link-Neuron Model.....	38
5.9	A Distributed Implementation of the Link-Neuron Model.....	38
5.10	Conclusions on the Link-Neuron Model.....	39
6.0	THE LINK-ACTIVATION SCHEDULING PROBLEM.....	39
6.1	Two Scheduling Problems: Nonsequential and Sequential Activation.....	39
6.2	Scheduling Conflicts.....	40
6.3	Communication Requirements.....	41
6.4	Complexity Issues.....	42
7.0	BOUNDS AND HEURISTICS FOR MINIMUM-LENGTH SCHEDULING.....	42
7.1	A Lower Bound on the Schedule Length.....	43
7.1.1	A Lower Bound on the Minimum Length of a Nonsequential-Activation Schedule.....	43
7.1.2	A Lower Bound on the Minimum Length of a Sequential- Activation Schedule.....	43
7.2	Heuristics for Scheduling.....	44
7.2.1	The NAS Heuristic.....	45
7.2.2	The SAS Heuristic.....	45
7.3	Performance of Scheduling Heuristics.....	45
7.3.1	Performance of the NAS Heuristic.....	45
7.3.2	Performance of the SAS Heuristic.....	46
8.0	A NEURAL NETWORK MODEL FOR LINK-ACTIVATION SCHEDULING.....	46
8.1	Formulation of the Constraint-Energy Terms.....	48
8.2	Determination of Connection Weights, Bias Currents, and Equations of Motion.....	49
8.3	Variations of the Basic Scheduling NN Model.....	50
8.3.1	The Condensed NAS Model.....	50
8.3.2	The Reduced SAS Model.....	51
8.3.3	The Adjustable-Length Model.....	53
8.3.4	Gaussian Simulated Annealing.....	54
9.0	SIMULATION ISSUES — THE SCHEDULING PROBLEM.....	54
9.1	A Binary Interpretation of the Analog State.....	56
9.2	Termination Criteria.....	56
9.3	Two Methods of Evaluating NN Performance.....	57
9.4	A Modification that has Improved Simulation Results.....	57
10.0	NAS SIMULATION RESULTS.....	58
10.1	The Condensed NAS Model Using the Monte-Carlo Approach.....	58
10.2	Parameter Sensitivity.....	59
10.3	A More Difficult NAS Problem Instance.....	59
10.4	Some Improvements to the NN Model.....	60
10.4.1	Time-Varying β	60

8.3.4 Gaussian Simulated Annealing

Simulated annealing (SA) [37-39] is a probabilistic minimization algorithm that facilitates the escape from local minima so that the chances of finding the global minimum are enhanced. Under this technique, the energy function normally follows a gradient descent; however, random perturbations are applied to permit occasional transitions to states with higher energy. If these perturbations are large enough, it is possible to escape the local minimum, thereby permitting the search by gradient descent to resume in a new location in the search space.

The Gaussian Machine of Ref. 32 used constant valued connection weights and combined a form of mean field annealing (MFA) [29] with additive Gaussian noise (AGN) (an overview of MFA is given in Appendix A); this approach is known as Gaussian simulated annealing (GSA). However, we have found that the combined use of the methods of Lagrange multipliers and AGN generally provides better results than those obtained by using MFA with AGN. Efforts to combine the use of MFA with the method of Lagrange multipliers led to the conclusion that no synergistic relationship exists between the two methods [15]. Therefore, our form of GSA uses the method of LM with AGN and an unchanging nonlinear neuron input/output voltage relationship. Our best results for the routing problem have been obtained without the use of GSA, and so we did not present those results in this report. However, the use of GSA has been essential to the performance of our scheduling NN models.

The use of GSA has no direct effect on the NN energy formulation. Therefore, it can be used in conjunction with any of the NN models or their variants. The use of GSA is reflected in the equations of motion only by the additive noise term as follows:

$$u'_{ijk} = u_{ijk} + \eta,$$

where u_{ijk} is the input voltage in the absence of noise. This may also be written as

$$u'_{ijk}(t + \Delta t) = u'_{ijk}(t) - (\Delta t) \left(u'_{ijk}(t) - \sum_{l=1}^{N_{sd}} \sum_{m=1}^{L(t)} \sum_{n=1}^{\Lambda} T_{ijk,lmn} V_{lmn} - I_{ijk} \right) + \eta,$$

where u'_{ijk} is the input voltage in the presence of AGN, and the noise term η has a zero-mean Gaussian distribution with variance σ^2 . The variance is decreased according to a "cooling" schedule given by

$$\sigma = \frac{kT_o}{1 + t / \tau_T},$$

where $k = \sqrt{8/\pi}$, T_o is a parameter that controls the initial value (temperature) of the variance, and τ_T is the annealing time constant that controls the rate of cooling. After a specified number of iterations, G_{end} , η is set to zero so that noise is no longer added to the system. The use of this form of GSA has yielded improved performance in several of the scheduling NN models, as discussed in Sections 10 and 11.

9.0 SIMULATION ISSUES — THE SCHEDULING PROBLEM

Extensive simulation results have demonstrated the capability of our NN formulation to generate optimum or near-optimum schedules. Several variants of the NN model, which have

Since all of the constraints are essentially satisfied, continued iteration only serves to drive the output voltages of those neurons that are declared to be "on" (on the basis of the instantaneous state interpretation) nearer to their assumed output voltage value of 1, and force the output voltages of the remaining neurons toward 0. Thus a conflict-free schedule of length Λ has been found, and continued iteration can yield no improvements or new information.

In runs that are terminated because of convergence, stalemate, or time-out, the NN has failed to find a conflict-free schedule of length Λ . Termination resulting from convergence or stalemate generally indicates that the system is trapped in a high-energy local minimum of the energy function, and further iteration (without the aid of simulated annealing or some other mechanism to escape local minima) generally is futile. Termination as a result of a time-out may also indicate that the system is trapped in a high-energy local minimum of the energy function, or, more likely, that the system is waffling between two (or more) different inadmissible states. For example, an insufficient number of activations may provide sufficient excitation to activate neuron ijk , resulting in a primary conflict. This constraint violation in turn provides sufficient inhibition to deactivate neuron ijk , thereby returning the system to the original condition of an insufficient number of activations.

9.3 Two Methods of Evaluating NN Performance

We have taken two different approaches in evaluating the performance of the NN models—the *Monte-Carlo approach* and the *multiple-instance approach*. With the Monte-Carlo approach for a particular problem instance, a given value of Λ , and set of parameter values, the NN is run from a number of different initial states (typically 100). The fraction of runs that yield admissible schedules in the series of simulations is then used as a measure of the NN model's performance.

With the multiple-instance approach, a problem instance is simulated from different initial states until an admissible schedule is found. If an admissible schedule is not found within a specified number of different initial states N_{s-max} , Λ is incremented and the process is repeated. By applying this approach to a sequence of problem instances with similar characteristics, e.g., a sequence of problem instances that all have the same lower bound on schedule length, the performance of the NN model can be characterized by the average schedule length and the average number of runs required to find a conflict-free schedule.

As discussed in Section 7.3.1, the 858 path sets that have $B_{nas} = 7$ were divided into two communication specification sets, set $(7, >7)$ and set $(7, 7)$. Set $(7, >7)$ consists of the 377 path sets that the NAS heuristic was unable to schedule in seven slots, and set $(7, 7)$ consists of the 481 path sets that the NAS heuristic was able to schedule in seven slots. In our simulation studies using the multiple-instance approach, we have compared these two sets in terms of the ability of the NN to generate minimum-length schedules.

9.4 A Modification That Has Improved Simulation Results

We observed in our studies of the routing NN model (see Section 3.5) that an insufficient number of neurons were typically activated, a problem that was mitigated by setting the parameter α to a value greater than one, which corresponds to increasing the bias currents. Hopfield and Tank [22] observed the same behavior in their studies of the TSP, as is discussed in Appendix A. The typical value of α that was used in the routing problem was $\alpha = 1.5$, which provided the additional excitation required to activate the correct number of neurons. This is also the value of α that was used by Hopfield and Tank in their solution of the TSP.

In our studies of the scheduling NN model, insufficient neuron activation has not been a problem. Nonetheless, we have found that adjusting the neutral positions of the amplifiers with additional bias currents has helped in satisfying the system constraints. We have incorporated the additional bias current into constraint 3 (see Section 8.1, Eq. (17)), which can now be expressed as

$$E'_3 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk} - \beta N_x \right)^2 = 0.$$

In the scheduling NN model, where the underactivation problem is minimal, we have found that setting the parameter β , which corresponds to the parameter α that was used in the routing NN, to approximately 0.61 generally provides the best results. The resultant expression for bias currents is

$$I_{ijk} = \lambda_2 + \lambda_3 \beta N_x + I.$$

Methods to update the value of β dynamically in conjunction with either the NAS or the SAS model are discussed in Section 10.4.1.

10.0 NAS SIMULATION RESULTS

In this section we present the results of simulation of the somewhat-easier NAS problem in which the sequential-activation requirement is not present. In this case, the goal is to schedule each link the correct number of times in each activation cycle, without regard to the order in which the links of any path are activated. This approach supports the same total traffic (in terms of the number of activations of each link per cycle) as the sequential-activation model, but it may result in greater end-to-end packet delay because several cycles may be needed to transport a packet from source to destination. Actually, in most cases, throughput (measured in terms of packets per slot) is greater using nonsequential-activation scheduling because removal of the sequentiality constraint often permits communication requirements to be satisfied in fewer slots.

In Sections 10.1 - 10.3, the performance of the condensed NAS NN model is evaluated on the basis of Monte-Carlo simulations of two problem instances. The model that was found to give the best performance is the condensed NAS model with $\beta = 0.61$. The sensitivity of this model to parameter variations is also evaluated. In Section 10.4, heuristic improvements for the NN model are developed, and their effects are evaluated by multiple-instance simulations in Section 10.5. The results of simulations of the basic and the adjustable-length NAS NN models are presented in Section 10.6. Although, in general, the performance of these models is inferior to that of the condensed model with $\beta = 0.61$, each of the models has certain attributes that make its evaluation worthwhile.

10.1 The Condensed NAS Model Using the Monte-Carlo Approach

We have studied in detail the scheduling of the links corresponding to the communication requirements shown in Table 2 and Fig. 13. As can be seen in Fig. 13, the maximum nodal degree in the network is 8 at node 13. Therefore, a lower bound on the nonsequential-activation schedule length for this problem is eight slots. We know that this is the minimum schedule length for this example because the NAS NN model has, in fact, found a number of admissible eight-slot schedules for it.

We first considered the condensed NAS model with $\beta = 1$ and MLM. Starting from 100 different initial states, an optimal schedule (conflict-free eight-slot schedule) was found in 84 runs. When GSA was used in conjunction with this model, simulations from the same 100 different states found 89 optimal schedules. However, the best results were obtained by using the condensed NAS model with $\beta = 0.61$ and MLM. Optimal solutions were found using this value of β , both with and without GSA, in all of the simulations from 100 initial states. Out of the 200 optimal schedules found in these two simulations, no two were the same. This verifies that the NN is, in fact, searching different portions of the solution space and successfully converging to local minima of the energy function that correspond to optimal schedules. Despite the fact that both simulations used the same initial states, the application of GSA caused a completely different set of

optimal schedules to be found. Thus in this case, AGN alters the search trajectory without compromising (nor enhancing) the final solution quality.

The parameter values used in the simulations of the condensed NAS models are shown in Table 4. In the table, $\lambda_c(0)$ denotes the initial value of all of the Lagrange multipliers λ_c 's (this includes the initial value of each of the MLM λ_{2p} 's), and the parameter ζ is the limiting value of the neuron input voltages, i.e., $-\zeta \leq u_{pk} \leq \zeta$.* The GSA parameters apply only to the runs that used simulated annealing.

Table 4. Condensed NAS NN Parameters

$\lambda_c(0)$	$(\Delta t)_{\lambda_1}$	$(\Delta t)_{\lambda_2}$	$(\Delta t)_{\lambda_3}$	Δt	I	β	N_{i-max}	N_c	ζ	Λ	MLM	T_o	τ_T	G_{end}
1	0.02	0.1	0.1	10^{-4}	10	1.0, 0.61	2×10^4	10^4	0.75	8	yes	0.01	50	10^4

10.2 Parameter Sensitivity

An important issue in measuring the performance of a NN model is its parameter sensitivity. If *good* solutions are obtained using a wide range of parameter values, the time required to determine appropriate parameter values by multiple trial-and-error simulations is greatly reduced. The sensitivity of the condensed NAS model to variations in the values of the bias parameters I and β , and the third constraint LM time constant $(\Delta t)_{\lambda_3}$ has been evaluated through simulations. It was found that the model is relatively insensitive to variations in any of these parameters. The most critical parameter is β , which, if set too small, prevents the discovery of admissible schedules by causing an oscillatory NN state. It was found that, for the problem instance given by Table 2, the use of any value of β in the range [0.488, 0.854] yields 100% optimal solutions, and the use of a value of β in [0.366, 1.122] yields greater than 70% optimal solutions.

10.3 A More Difficult NAS Problem Instance

Monte-Carlo simulations of a more difficult problem instance were run by using the condensed NAS model with $\beta = 0.61$. The communication requirements of this problem instance (referred to as the "augmented problem" in this subsection), which are listed in Table A2 of Appendix A in Ref. 16, were generated simply by increasing the load on several of the physical links of Fig. 13. Here we refer to the problem instance defined by the communication requirements of Fig. 13 as the "original problem." In this example, the transformation from the original to the augmented problem results in an increase in the value of N_x (the total number of link activations) from 41 to 63, while the number of physical links in the network remains constant at 30. The minimum schedule length for this problem is eight slots. The NAS heuristic scheduled the problem in nine slots.

A Monte-Carlo simulation of the augmented problem was run from 100 different initial states by using the condensed NAS model with MLM, GSA, and the parameter values shown in Table 4 ($\beta = 0.61$). In the simulations, 21 optimum schedules were found. In a typical optimal schedule found by the NN model, only one more admissible link activation is possible; 99.6% of the slots are either activated or blocked. The ability of our NN model to produce a significant number of optimal solutions to such a highly constrained problem illustrates the power of our method.

*Our studies have shown that it is helpful to place limits on the neuron input voltages (which otherwise could range from $-\infty$ to ∞); such limits help to prevent the neurons from being irrevocably locked into an "on" or "off" state.

10.4 Some Improvements to the NN Model

10.4.1 Time-Varying β

The results of the Monte-Carlo simulations, which were presented above, have shown that improved NN performance is obtained by using the condensed model with β less than one. However as discussed previously, the use of a constant, less-than-one value of β , presents two problems: First, continued iteration after the discovery of an admissible schedule must cause the NN state to change to an inadmissible schedule. Second, continued iteration as a result of the failure to discover an admissible schedule eventually leads to an oscillatory NN state. Therefore, a function that initially causes β to be approximately equal to 0.61 and to approach 1.0 as the NN converges, appears to be appropriate. Functions of this type are discussed in Refs. 16 and 14.

10.4.2 A Traffic-Based Heuristic

Additionally, methods of extending conventional heuristic concepts to the NN model for nonsequential-activation scheduling have been examined. In the biased-greedy heuristic [6, 36], a bias, proportional to a node's degree, is applied to every node so that high-degree nodes are given a higher scheduling priority than lower-degree nodes. Thus, lower-degree nodes are scheduled in slots that are not blocked by the "prescheduled" high-degree nodes. This concept of scheduling high-degree nodes first may be extended to the condensed NAS NN model by applying a "traffic-based bias" to each physical link. Details of this method are discussed in Refs. 16 and 14.

10.5 Evaluation of the NN Improvements via the Multiple-Instance Approach

The concepts developed in Section 10.4, i.e., the use of time-varying β , or a traffic-based bias to emphasize early scheduling of high-degree nodes (links), were evaluated by means of five multiple-instance simulations of the condensed NAS model. In each of these simulations, the goal was to schedule to the problem instances in set (7, >7) in seven slots. These are the path sets that have $B_{nas} = 7$ but that the NAS heuristic was unable to schedule in seven slots (see Table A5 in Appendix A of Ref. 16 for a partial listing). All five of the simulations were able to find optimum schedules (i.e., admissible schedules of length seven) for each of the 377 problem instances in set (7, >7), thus verifying that the lower bound on all of these path sets is tight, and more importantly, that the NN model is capable of delivering better schedules than the NAS heuristic.

The results of these simulations (labeled A - E) are shown in Fig. 17. The figure shows the average number of different initial NN states per problem instance required to find an optimal schedule. The parameter values that make each of the simulations unique are shown in Fig. 17, and the parameter values that were common to all of the simulations are listed in Table 5.

Figure 17 shows that the use of the time-varying β in simulation B produces much better performance than the use of a constant value of β in simulation A. Simulation A is essentially a benchmark simulation that uses the methods and parameter values (i.e., the condensed model with a constant β value of 0.61) that had been found effective in the Monte-Carlo simulations as described in Section 10.1. Each of the simulations C, D, and E uses the traffic-based bias in addition to the use of time-varying β . In these simulations, separate time constants $(\Delta t)_{\lambda_{2p}}$ were introduced to correspond to each of the Lagrange multipliers of the form λ_{2p} . These time constants were assigned values proportional to the degrees of their associated links,* with the constant of proportionality for each of the simulations shown in the figure. Figure 17 shows that the value of the constant of proportionality greatly impacts the NN performance. With this constant set to 0.05 (simulation C), an average of 2.13 runs are required to find a schedule. This is the least-efficient performance of any of the multiple-instance simulations. However, with the constant of proportionality set to 0.01 (simulation D), the most efficient performance of any of the simulations

*We define the degree of a link to be the sum of the nodal degrees of the two nodes on which the link is incident.

was obtained. This simulation required an average of 1.28 different initial states to find a schedule.

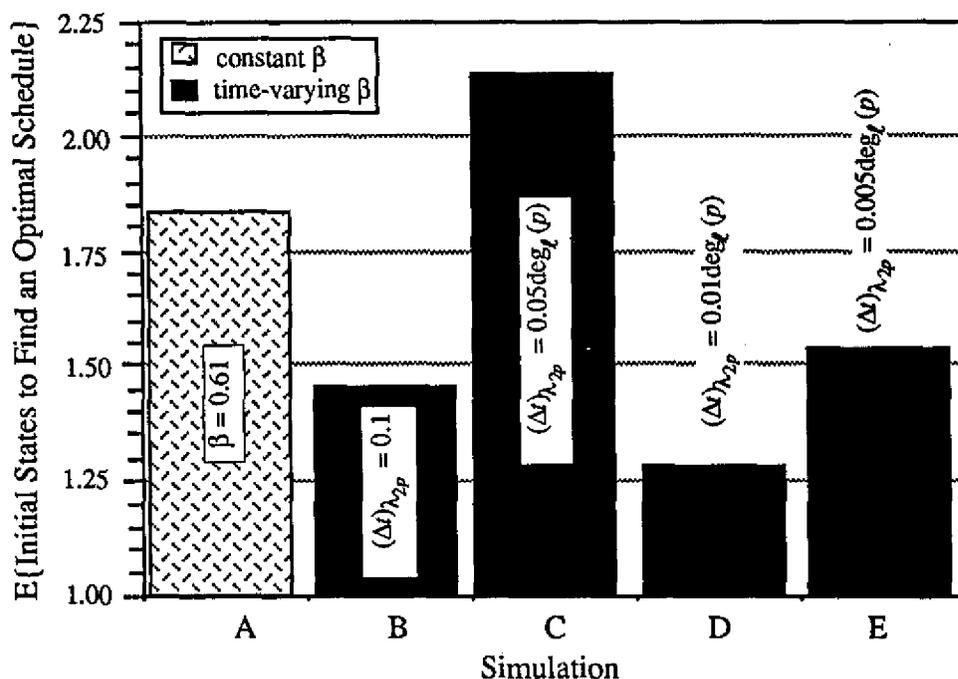


Fig. 17 — Average number of runs required to find an optimum schedule by using time-varying β and/or a traffic-based bias in multiple-instance simulations

Table 5. Common Parameters for Runs A - E

$\lambda_c(0)$	$(\Delta t)\lambda_1$	$(\Delta t)\lambda_3$	Δt	N_{i-max}	N_c	ζ	A	MLM	GSA
1	0.02	0.1	10^{-4}	11000	5000	0.75	7	yes	off

The results of these simulations indicate that the use of a time-varying β can significantly improve the NN performance. Additionally applying a traffic-based bias beneficially impacts the NN performance only when the constant of proportionality is set to an appropriate value.

10.6 An Evaluation of the Basic and the Adjustable-Length NAS Models

The basic and the adjustable-length NAS NN models offer certain advantages over the condensed NAS model and its variants, even though they do not provide comparable performance. The main advantage of the basic model is its simple formulation, which may be relatively easily implemented. The adjustable-length model, on the other hand, is a more complex formulation. It adds one constraint, and requires approximately 1.5 times the number of neurons required in the basic model. Nonetheless, the adjustable-length model potentially yields an admissible, if not optimal, schedule from every run. It also provides an important step toward the development of a joint routing-scheduling NN model, which is discussed in Section 12.

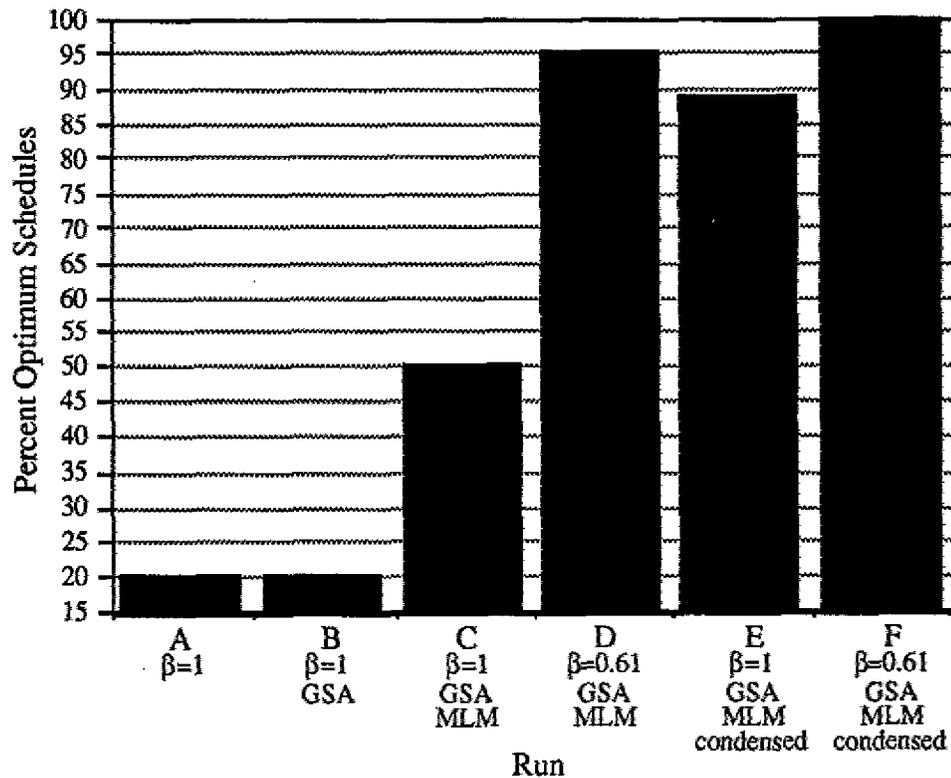
We have performed 12 Monte-Carlo simulations to evaluate the performance of the basic and the adjustable-length NAS models. The simulations are labeled A through F (no relationship to the simulations A through E that were presented in Section 10.5), and A' through F'. Simulations A through F used the basic model, and simulations A' through F' used the adjustable-length model, to schedule the communication requirements of Table 2. Recall that $B_{nas} = 8$ for this example; thus the minimum schedule length that can be found is eight slots. The parameter values used in these simulations are shown in Table 6.

Table 6. NAS NN Parameters Used in Simulations of the Basic and the Adjustable-Length Models

Run	$\lambda_c(0)$	$(\Delta t)\lambda_1$	$(\Delta t)\lambda_2$	$(\Delta t)\lambda_3$	$(\Delta t)\lambda_5$	Δt	I	N_{i-max}	N_c	ζ	T_o	τ_T	G_{end}
A, A'	1	0.01	0.1	0.1	0.01	10^{-4}	10	2×10^4	10^4	∞	off	off	off
B-F, B'-F'	1	0.02	0.01	0.01	0.01	10^{-4}	10	2×10^4	10^4	0.75	.01	50	10^4

The results of Runs A through F are shown in Fig. 18, and the results of Runs A' through F' are shown in Fig. 19. The similarities and differences between each of the runs are also summarized in the figures. The data shown in these two figures illustrate four important points:

- Point 1:* The use of $\beta = 0.61$ yields significantly better results than the use of a unit-valued β in all of the NAS models.
- Point 2:* The use of the condensed model in conjunction with the use of $\beta = 0.61$ further improves the performance of both the basic and the adjustable-length models.
- Point 3:* Both GSA and MLM make important contributions to the NN performance.
- Point 4:* All variants of the adjustable-length model deliver a reasonably high percentage of admissible schedules.

Fig. 18 — Results of simulations of variations of the basic NAS NN model ($\Lambda^* = 8$ slots)

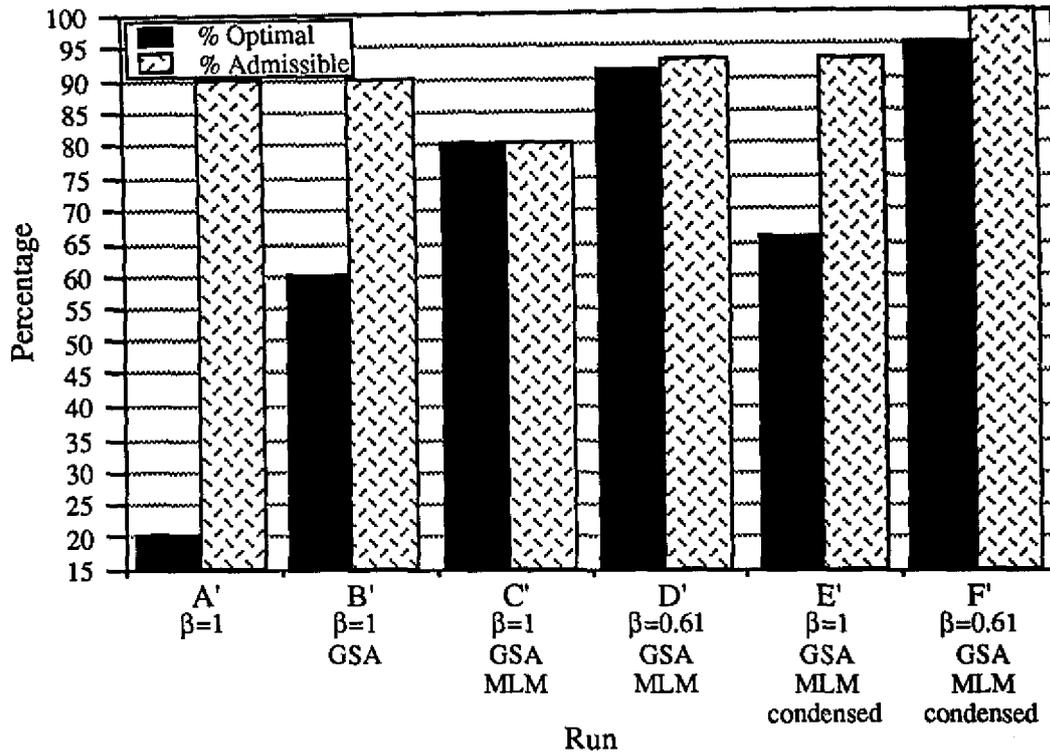


Fig. 19 — Results of simulations of the adjustable-length NAS NN model ($\Lambda^* = 8$ slots)

10.7 Conclusions on the NAS NN Model

In this section we have presented the results from simulations of three different NAS NN models, the basic, the condensed, and the adjustable-length models. In both Monte-Carlo and multiple-instance simulations, very satisfactory results were obtained by using the condensed NAS model with $\beta = 0.61$ and MLM. In Monte-Carlo simulations of the problem instance given by Table 2, 100% of the schedules found by this model were optimal. Simulations of the heavily congested problem instance given by Table A2 of Ref. 16, which required near-maximal scheduling, resulted in 21% optimal solutions, thus demonstrating the ability of our method to determine optimal schedules for highly constrained problems. The model was also able to optimally schedule all of the problem instances in sets (7, 7) and (7, >7), thereby demonstrating the ability of the NN model to produce optimal schedules for many instances for which the NAS heuristic was unable to do so. The efficiency of the condensed NAS model was increased by the introduction of a traffic-based heuristic and the use of a time-varying β . Multiple-instance simulations of set (7, >7) with this more-efficient model found optimal solutions in an average of 30% fewer runs than were required by the condensed NN model that used $\beta = 0.61$ and MLM. These simulations of the condensed NAS NN model have shown that the model is capable of finding minimum-length schedules in a large fraction of the runs. They also indicate that the model is fairly robust to variations in the parameter values and in the communication requirements. Thus we feel that the model is applicable to and will perform well in a broad class of scheduling problems.

The basic and the adjustable-length NAS NN models offer certain advantages over the condensed NAS model and its variants, even though they do not provide comparable performance. The basic model provides the foundation on which all of the other NAS NN models are built. The adjustable-length model (usually) provides an increased percentage of admissible schedules. In addition, because this model does not require an accurate estimate of the minimum schedule length, it provides an important step toward the development of a joint routing-scheduling NN model for which such an estimate may be difficult. Simulations of both the basic and the adjustable-length models have demonstrated their ability to deliver reasonably good performance.

11.0 SAS SIMULATION RESULTS

All of the SAS simulations have used some form of the reduced SAS model. The use of this model, besides reducing the dimensionality of the solution space by eliminating a number of inadmissible solutions, also significantly reduces the number of neurons in the NN. Since the number of computations required at each iteration of the NN model is approximately proportional to the square of the number of neurons, reducing the number of neurons markedly reduces the time required to complete a simulation.

Preliminary simulations of the reduced SAS model, without the use of any of the heuristic aids that were developed for the NAS NN in Section 10.4, yielded disappointing results. Monte-Carlo simulations that attempted to schedule the sequential activation of the communication requirements of Table 2 in nine slots (the minimum length) produced only 7% optimal schedules.

Because of these disappointing results, which are markedly inferior to the results obtained for the NAS model (without the heuristics that were subsequently developed for it), it has been necessary to develop heuristics to improve NN performance. As was done in modifying the NAS NN model, the concepts used in developing a heuristic SAS algorithm were applied to the SAS NN model. These modifications are described in the next two subsections.

11.1 Skewed Initialization and Skewed Randomization

Much improved SAS performance has been obtained by, in essence, setting the initial neuron output voltages so that the initial instantaneous state was free of sequence conflicts. This is done by setting the initial output voltage value of neuron ijk to

$$V_{ijk}(0) = \frac{1}{X_i + 1} + m_i \left(k - j - \frac{X_i}{2} \right), \quad |m_i| < \frac{2}{X_i(X_i + 1)}, \quad (19)$$

where $X_i + 1 = \Lambda - L(i) + 1$ is the number of neurons that represent each link in the path connecting SD pair i , and m_i is the slope of the initial output voltage $V_{ijk}(0)$ as a function of the time slot k that neuron ijk represents. This function is plotted in Fig. 20. We call this type of neuron initialization a "skewed initialization." We have used $m_i < 0$; however, positive values of m_i can also be used, as discussed below. Setting $m_i = 0$ results in the original form of initialization, as discussed in Section 9.0. As in Section 9.0, a random perturbation is added to each neuron following skewed initialization of the NN so that different random seeds cause different portions of the solution space to be explored.

For $m_i < 0$, the j th link in the path is activated in the j th slot of the cycle, which is the first slot in which it could possibly be activated without violating the sequential-activation requirement. Alternatively, for a positive slope ($m_i > 0$), each link is activated in the last possible slot. Either choice, if used consistently for all the links on the same path, will provide an initial state (based on the instantaneous state description) that is free of sequence conflicts. However, this initial state is not necessarily free from primary conflicts. Typically after this type of initialization, the NN must remove a large number of primary conflicts. As the number of primary conflicts are reduced, temporary sequence conflicts are often generated. However, simulation results have shown that the discovery of a conflict-free schedule is much more likely when the initial instantaneous state is free of sequence conflicts.

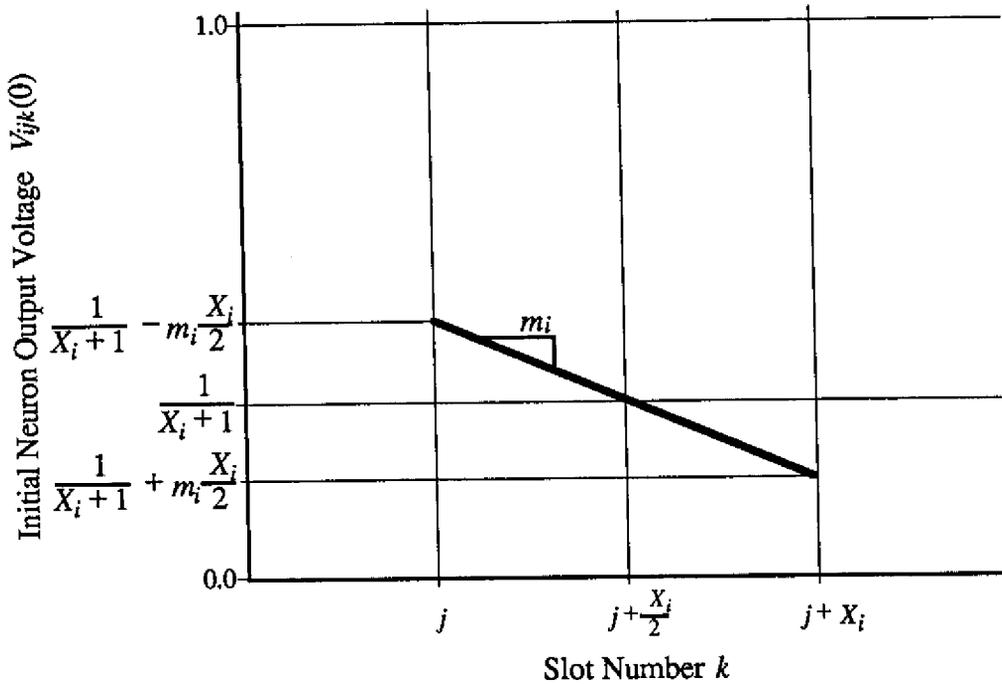


Fig. 20 — Initial neuron output voltage as a result of skewed initialization

We have developed two different approaches for assigning a value to m_i , which we have named the “path-length dependent” approach and the “hybrid” approach. A different approach would be to randomly distribute the tentative activations over the entire set of legal slots for each link, rather than clustering them in the early and/or late slots. We have called this the “skewed-randomization” approach. All of these schemes are described in detail in Ref. 16.

11.2 Evaluation of Skewed Initialization and/or Randomization via Monte-Carlo Simulation

The effects of skewed initialization and/or skewed randomization on the performance of the reduced SAS NN model were evaluated by Monte-Carlo simulations. Each of the runs attempted to find nine-slot sequential-activation schedules (optimum schedules) that satisfy the requirements of Table 2 and Fig. 13 from 100 different initial states.

The use of hybrid skewed initialization, with appropriate parameter values, produced 64 optimal schedules in 100 attempts. Thus, the introduction of hybrid skewed initialization yields an order of magnitude increase in the percentage of optimum solutions. (Recall that only 7% of the schedules found prior to the introduction of this method were optimal.) However, multiple-instance simulations, which are discussed further in Section 11.3, have indicated that the hybrid form of initialization does not provide the consistently good performance that is obtained when the path-length-dependent form of initialization is used to schedule a number of different problem instances.

In another series of runs, the path-length-dependent approach to initialization was used in conjunction with skewed randomization and the time-varying β , which was discussed in Section 10.4.1. Optimal solutions were found in 21 of 100 runs. Although these results do not match the 64 optimal solutions found by the use of hybrid skewed initialization, it is noteworthy that the number of optimal schedules generated is three times the number found without the aid of any of the heuristic modifications. This simulation demonstrates that path-length-dependent initialization can be used to obtain reasonably good results for this problem instance.

10.4.2	A Traffic-Based Heuristic	60
10.5	Evaluation of the NN Improvements via the Multiple-Instance Approach	60
10.6	An Evaluation of the Basic and the Adjustable-Length NAS Models	61
10.7	Conclusions on the NAS NN Model.....	63
11.0	SAS SIMULATION RESULTS.....	64
11.1	Skewed Initialization and Skewed Randomization	64
11.2	Evaluation of Skewed Initialization and/or Randomization via Monte-Carlo Simulation.....	65
11.3	Evaluation of Skewed Initialization and/or Randomization via Multiple-Instance Simulation.....	66
11.3.1	SAS Multiple-Instance Simulation Issues	66
11.3.2	A Simulation Using Hybrid Initialization.....	66
11.3.3	Evaluation of the Path-Length-Dependent Form of Initialization	67
11.3.4	SAS of Highly Constrained Problem Instances	67
11.4	Conclusions on the SAS NN Model	68
12.0	THE JOINT ROUTING-SCHEDULING PROBLEM	68
12.1	Problem Formulation.....	70
12.2	A Joint Routing-Scheduling NN Model	70
13.0	CONCLUSIONS.....	71
	REFERENCES.....	73
	APPENDIX A – Hopfield Neural Networks and Their Application to Optimization Problems.....	77

11.3 Evaluation of Skewed Initialization and/or Randomization via Multiple-Instance Simulation

Multiple-instance simulations of the reduced SAS NN model were performed to evaluate the use of skewed initialization and/or randomization. These simulations scheduled the 50 problem instances in set (7, >7) that are listed in Table A5 in Appendix A of Ref. 16. Recall that each of the problem instances in set (7, >7) has $B_{nas} = 7$ as a lower bound on its nonsequential-activation schedule length and a NAS heuristic schedule length greater than seven slots. Although these problem instances were selected on the basis of a nonsequential-activation scheduling analysis of the network of Fig. 2 and of the paths listed in Table A1 in Appendix A of Ref. 16, they represent a set of problem instances that offers a degree of diversity while maintaining some common characteristics. Of the 377 path sets that are elements of the set (7, >7), 233 have lower bounds on the sequential schedule length of $B_{sas} = 8$ slots, 114 have $B_{sas} = 9$ slots, and the remaining path sets have $B_{sas} = 10$ slots. The SAS heuristic found schedules for these specifications with lengths ranging from 9 to 14 slots. The majority of the schedules generated by the heuristic were 10 or 11 slots long; 25 were 9-slot schedules, and five were 14-slot schedules. None of the heuristic schedules were as short as their corresponding bound.

Before presenting the results of these simulations in Sections 11.3.2 and 11.3.3, concerns that are unique to SAS multiple-instance simulation are discussed in Section 11.3.1.

11.3.1 SAS Multiple-Instance Simulation Issues

In multiple-instance simulations, the parameter N_{s-max} is critical to both the quality of the solution and to computational efficiency. Since Λ^* (the shortest possible schedule length for a particular problem instance) is not known a priori, an attempt is first made to generate a schedule of length $\Lambda_o = B_{sas}$. If an admissible schedule has not been found after N_{s-max} attempts, the value of Λ is incremented by one, and up to N_{s-max} attempts are made again. This process is repeated until an admissible solution is found. Clearly, none of the runs for which $\Lambda < \Lambda^*$ can possibly produce an admissible schedule. Thus, use of an excessively large value of N_{s-max} results in a large number of futile runs. On the other hand, use of too small a value of N_{s-max} can result in the failure to find a schedule of optimal length. For example, increasing the value of N_{s-max} from 5 to 20 in a pair of otherwise identical multiple-instance SAS simulations permitted the generation of shorter schedules for a significant number of problem instances; the percentage of schedules that were no longer than those generated by the SAS heuristic was increased from 57.4% to 90%. Thus, although the NN is able to find an admissible schedule fairly rapidly if $N_{s-max} = 5$ (because the schedule length is increased after only five unsuccessful attempts), it is also likely to overlook admissible schedules with shorter length. Increasing the value of N_{s-max} to 20 significantly increases the probability of discovering a "short" schedule.

We have not been able to determine the length of an optimum sequential-activation schedule for many of the problem instances in the set (7, >7). When a minimum schedule length has been ascertained for a problem instance, confirmation has been achieved almost exclusively by the discovery of a NN schedule with length that matches the tightened bound (i.e., for a few problem instances, we have been able to increment B_{sas} by examining the network, e.g., see Appendix B of Ref. 16). To assess the quality of all of the admissible schedules generated in a simulation, including those with unknown minimum length, we compare their lengths to the lengths of the schedules found by the SAS heuristic.

11.3.2 A Simulation Using Hybrid Initialization

A multiple-instance simulation was performed by using the hybrid form of initialization (with $N_{s-max} = 20$) in the absence of simulated annealing. Although admissible schedules were found consistently in this run, only 30% of the schedules were shorter than those found by the SAS heuristic, and 36% were longer than the schedules generated by the heuristic. Seven of the 50 schedules produced in this simulation are known to be optimal (i.e., minimum in length), and

35 are known to be nonoptimal.* Thus, the good performance that was obtained by using hybrid initialization in the Monte-Carlo run discussed in Section 11.2 and the relatively poor performance obtained in this multiple-instance simulation, indicate that the performance of hybrid initialization is inconsistent. In the next subsection, simulations show that, in general, significantly better results are obtained by using the path-length-dependent form of initialization.

11.3.3 Evaluation of the Path-Length-Dependent Form of Initialization

We have evaluated the use of the path-length-dependent form of initialization in a number of variants of the reduced SAS model by multiple-instance simulations that scheduled the problem instances listed in Table A5 of Ref. 16. The best results were found by using the same NN model that was used in Section 11.2. In addition to using path-length-dependent initialization, this model used GSA, skewed randomization, and time-varying β . In this simulation, the value of N_{s-max} was set equal to 20. Only two of the 50 schedules found by this model were longer than the schedules found by the SAS heuristic. Eleven of the schedules have been shown to be optimal; thirteen of the schedules are known to be nonoptimal. Six of the certified nonoptimal schedules have length $\Lambda^* + 1$ (for these problem instances, we have been able to ascertain that $\Lambda^* = 9$). Another five were verified to be nonoptimal when shorter admissible schedules were generated by a second simulation of the same NN model with $N_{s-max} = 40$ (instead of 20). We know that the remaining two nonoptimal schedules have lengths greater than Λ^* because the schedules generated by the SAS heuristic are shorter than those found by the NN. Clearly, a schedule whose length matches the lower bound[†] is a schedule of minimum length; Λ^* is equal to the lower bound in such cases.

This simulation serves to confirm the conclusions drawn in Sections 11.2 and 11.3.2. The NN model yielded better performance than the SAS heuristic without the risk of incurring an oscillatory NN state. Whereas 22% of the NN schedules have been verified to be optimum and 74% *may* be optimum, only 26% of the heuristic schedules *may* be optimum but none have been verified; 64% of the NN schedules were shorter than the corresponding schedules generated by the SAS heuristic. This simulation also confirms, as did the NAS simulation of Section 10.5, that the use of time-varying β is beneficial to the NN performance. Furthermore, this simulation demonstrates the model's ability to reliably generate sequential-activation schedules of minimum or nearly minimum length for a diverse set of problem instances.

11.3.4 SAS of Highly Constrained Problem Instances

The SAS heuristic was able to find optimal eight-slot sequential-activation schedules for the eight path sets listed in Table A4 in Appendix A of Ref. 16. These problems are more highly constrained than the problems for which nine-slot schedules are needed (because approximately the same total number of links must be scheduled), and they present a significant challenge to our methodology. In a multiple-instance simulation of this set of problem instances, it was reasonable to set N_{s-max} to a large value ($N_{s-max} = 500$) because the minimum schedule length is known for these problem instances, i.e., because $B_{sas} = \Lambda^*$. The NN model that was used in this multiple-instance simulation is the same as the one used in Section 11.3.3; it used skewed randomization, the time-varying β , and GSA.

The large value of N_{s-max} allowed the NN to find optimal schedules for each of the problem instances. However, an average of 85.5 different initial states were required to find each of these schedules. Typical schedules for these problem instances are about 96% maximal, i.e., 96% of the slots are either used or blocked. The nearly maximal nature of these schedules verifies that these

*The schedules that have been verified to be optimal have lengths equal to a tightened lower bound on the schedule length (see Appendix B of Ref. 16). The schedules that have been verified to be nonoptimal are longer than an admissible schedule generated in a different simulation.

[†]Typically, this bound was $B_{sas} + 1$ slots, because, as discussed in Appendix B of Ref. 16, it was found by contradiction that the network could not be scheduled in B_{sas} slots. Therefore the bound was tightened to $B_{sas} + 1$ slots.

particular problem instances are, in fact, highly constrained. Thus we have again demonstrated the capability of our NN formulation to solve highly constrained problems.

11.4 Conclusions on the SAS NN Model

The results of this section have demonstrated the ability of the NN model to find minimum- or nearly minimum-length sequential-activation schedules for several difficult problem instances. They also indicate that the NN model is robust and can perform well in a broad class of scheduling problems.

For example, the minimum-length sequential-activation schedule that satisfies the communication requirements of Table 2 is nine slots. All of the SAS NN models have been able to find such a schedule, with varying degrees of efficiency. In Monte-Carlo simulations that scheduled the sequential activation of the links in this problem instance, the best performance was obtained in a run that produced 64% optimal solutions using hybrid initialization (see Section 11.2). However, this model performed relatively poorly in the multiple-instance simulation discussed in Section 11.3.2. On the other hand, the NN model using the path-length-dependent form of initialization performed extremely well in multiple-instance simulations (Section 11.3.3) but could not equal the performance of the hybrid initialization model in Monte-Carlo Simulations of the problem instance given by Table 2. From these observations, we conclude that the NN model using path-length-dependent initialization gives the best overall performance when examining diverse problems. However, one of the other models might be better suited for a given problem instance.

12.0 THE JOINT ROUTING-SCHEDULING PROBLEM

Our NN models discussed thus far have separately considered the problems of routing and link-activation scheduling. We have used a two-phase approach in which the routing model generates a set of communication requirements, which serves as the input for the scheduling NN model. The routing NN model attempts to pick those paths that result in "small" numbers of shared links among the chosen paths so that "congestion" at these links is reduced, permitting relatively short schedules to be generated. Minimization of congestion was chosen as the performance measure because it was hypothesized that doing so would permit the generation of short schedules. As we now discuss, this was not strictly true, but the schedule length required for path sets with low congestion tended to be lower than that for path sets with high congestion.

To evaluate the hypothesis that minimization of congestion would permit the generation of short schedules, the "congestion energy" (see Section 3) of each of the 858 path sets that admit seven-slot nonsequential-activation schedules (the problem instances in the union of sets (7, 7) and (7, >7)) was calculated. Figure 21 compares the cumulative mass function of these results to that obtained by considering all permutations of the paths in Table A1 of Ref. 16 (exhaustive search), and to that obtained by considering all permutations of the subset of paths in the table that consists of only the shortest paths between each SD pair (shortest-path heuristic). The figure shows that the path sets that admit minimum-length nonsequential-activation schedules do, indeed, tend to have low congestion energy. However, the relationship is not monotonic: The 12 path sets with minimum congestion energy (33.25) do, in fact, admit seven-slot schedules, but only 12 of the 30 path sets that have congestion energy values of 33.75 admit seven-slot schedules. The congestion energy of the 858 path sets that can be scheduled in seven slots ranges from 33.25 to 46.25. However, 498,975 path sets have congestion energy in this range; thus, many path sets cannot be scheduled in seven slots, even though they have lower congestion-energy values than some of the path sets that can be scheduled in seven slots. The existence of relatively high congestion-energy solutions that *do* admit optimum schedules and of relatively low congestion-energy solutions that *do not* admit optimum schedules indicates that the routing and the scheduling problems are not separable.

From a different viewpoint, we have found that the shortest-path heuristic generally yields a set of routes with reasonably low congestion energy. For the problem posed by Table A1 of Ref. 16, the set of paths chosen by the shortest-path heuristic (i.e., the path set with the lowest congestion energy of any of the path sets in the restricted subset that includes only the shortest paths between each SD pair) has a lower congestion-energy value than 56% of the path sets that can be scheduled in seven slots. However, it turns out that *none* of the path sets considered by the shortest-path heuristic can be scheduled in seven slots. In fact, the minimum-length nonsequential-activation schedule for any of these path sets is nine slots.

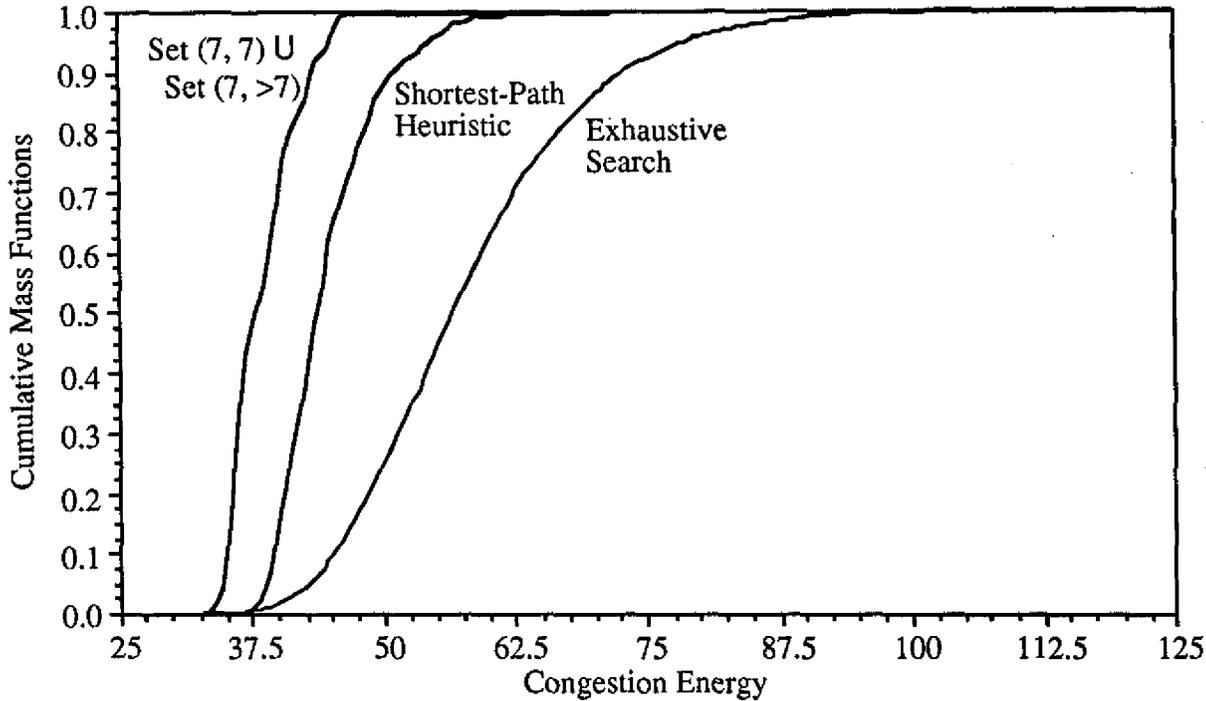


Fig. 21 — Congestion energy of the path sets that admit minimum length schedules

Although a path set that has a low congestion-energy value will probably admit a short nonsequential-activation schedule, the above observations lead to the conclusion that the use of congestion minimization as the sole criterion for route selection is inadequate to guarantee a solution that can be optimally scheduled. However, for the routing-scheduling problem given by Table A1 of Ref. 16, selecting routes that minimize the maximum nodal degree in the network does yield a set of paths that admits an optimal (seven-slot) schedule. This phenomenon results from the absence — in any of the 858 path sets that have maximum nodal degrees of seven — of odd-length cycles that require more than seven slots to schedule. As discussed in Section 7.1.1, odd-length cycles may require more slots in the schedule than do the maximum-degree nodes. Since verifying the presence or absence of odd-length cycles is, in general, excessively computation intensive, simply minimizing the maximum nodal degree is also inadequate to guarantee a solution that can be optimally scheduled because the absence of odd-length cycles cannot be readily verified.

Thus, selecting routes to minimize congestion energy or selecting routes to minimize the maximum nodal degree in the network are both reasonable heuristic methods that tend to yield solutions that admit short, if not optimum, schedules. However, neither approach is capable of delivering solutions that can guarantee minimal values of schedule length. In fact, it appears that any approach that separates the routing problem from the scheduling problem cannot do this. Hence, a joint formulation is needed to simultaneously select routes and schedule link activations optimally.

The SAS problem is a restricted case of the NAS problem that would also benefit from a joint routing-scheduling approach. Because link activations are restricted by the position of the links in the paths as well as by adjacent links, route selection is even more critical for the SAS problem than it is for the NAS problem. Therefore, a joint routing-scheduling formulation is necessary and appropriate for this problem as well. The joint formulation proposed in the next subsection addresses either NAS or SAS.

12.1 Problem Formulation

Our problem formulation combines the path selection problem addressed in Sections 3-5 with the scheduling problem discussed in Sections 6-11. Thus the problem becomes the simultaneous choice of one of these paths for each SD pair along with the determination of an activation schedule for each link along this path so that a schedule of minimum length with no scheduling conflicts is produced. Since we are considering discrete packets, a combinatorial-optimization formulation is again appropriate. In Ref. 16 we propose a NN model to address the joint routing-scheduling problem. However, the NN designed for this application is extremely complex and has not yet been implemented completely. Here we summarize some of the main features of this model.

Problem Statement

Given a set of N_{sd} source-destination (SD) pairs and an equal number of sets of paths, the i th of which contains $N_p(i)$ paths connecting SD pair i , and one unit* of traffic to be delivered between each SD pair, select one path from each set of $N_p(i)$ paths that satisfies the communication requirements and schedule the activation of the links in these paths in a minimum number of slots. As was done in addressing the scheduling problem, we can specify either sequential-activation scheduling (SAS) or nonsequential-activation scheduling (NAS).

12.2 A Joint Routing-Scheduling NN Model

We would like to satisfy the specified communication requirements in Λ slots.[†] For every link in each of the paths connecting the N_{sd} SD pairs, we define Λ neurons, each corresponding to one slot. We use four integers to index the neurons, i.e., neuron $ijkl$ represents slot l of the k th link in the j th path between SD pair i . As in the pure scheduling problems discussed earlier in this report, the Lyapunov energy function consists only of constraint terms; there is no objective function to be minimized. We have identified eight constraints that represent the desired objective of the NN for the joint routing-scheduling problem. These constraints, which are listed below, incorporate features of both the routing and scheduling constraints discussed earlier in this report. In Ref. 16, we also show the energy expression associated with each of the constraints as well as the corresponding equation-of-motion term.

- RS_1 . Activate (select) links from no more than one path per SD pair (a modification of Eq. (9)).
- RS_2 . Activate a total of exactly N_{sd} paths in the network (a modification of Eq. (10)).
- RS_3 . Activate exactly one path per SD pair (a modification of Eq. (11)).
- RS_4 . Activate complete paths (a modification of Eq. (12)).
- RS_5 . Activate no conflicting links (no primary conflicts; a modification of Eq. (15)).

*The model can easily be extended to handle nonunit traffic.

[†]Again, we can use either a fixed schedule length or the adjustable-length method (described in Section 8.3.3), under which a penalty is incurred for using the higher-numbered slots.

- RS*₆. Activate a total of N_x neurons (a modification of Eq. (17)).
- RS*₇. Schedule all activations in slots numbered less than or equal to some lower bound on the schedule length (a modification of the constraint generated by the adjustable-length NN model for scheduling, which was discussed in Section 8.3.3).
- RS*₈. Sequentially activate the links in each path (a modification of Eq. (18)).

13.0 CONCLUSIONS

Although the issues of routing and scheduling in packet radio networks are highly interdependent, few studies have addressed them jointly. In this report, we have posed the joint routing-scheduling problem as one of combinatorial optimization, whose objective is to determine the best set of paths between each of a number of source-destination pairs as well as the time slots in which each of the links along these paths should be activated. Our approach has been to use Hopfield NNs, which are known to provide good solutions to a wide variety of combinatorial-optimization problems. Because of the complexity of the joint routing-scheduling problem, we have first studied the two problems individually and have demonstrated the capability of Hopfield NNs to provide good solutions for both of these problems. We have made observations on the degree of separability that exists between these problems, and we have also presented a formulation for the joint problem.

The Hopfield NN methodology is different from more-traditional algorithmic or simulation approaches to combinatorial-optimization problems. For example, this approach involves embedding a discrete problem in a continuous solution space. The NN is "programmed" by implementing the set of connection weights and bias currents that correspond to the "energy" function that is to be minimized. This energy function is a linear combination of the desired objective function and energy components that are related to the constraints that must be satisfied by valid solutions.

An analog hardware implementation of a Hopfield NN will normally converge to its final state within at most a few RC time constants, thus providing an extremely rapid solution to a complex optimization problem. In our studies (as in most studies of this technique) we have simulated the system dynamics in software. Although such software solutions are extremely time-consuming, they verify the soundness of the use of the Hopfield NN approach for optimization problems of this type and suggest that hardware implementations may be worthwhile. In fact, hardware implementation may be feasible for problem sizes that exceed by far those that can be handled in software.

The most critical issue in the design and simulation of a Hopfield NN model is the choice of the coefficients used in the connection weights. Most studies of Hopfield NNs have used trial-and-error methods to determine acceptable values for these coefficients, a process that is tedious at best and often ineffective. We have used the method of Lagrange multipliers (LM) to determine good values for these coefficients. This method permits the coefficients to dynamically vary with the evolution of the system state. Although some experimentation is still needed to determine good values for system parameters such as bias currents and time constants, our studies have shown that acceptable values for these parameters are generally found rather quickly when the LM method is used. Furthermore, the results are generally better than those obtained when using constant valued coefficients.

To assess the performance of our NN models, a benchmark is needed against which to compare them. In our smaller routing examples, such as the 24-node network with 10 SD pairs, an exhaustive search of all possible solutions has been possible. In some of our studies of this network, globally optimal solutions were in fact found in almost all runs; in other examples, near-

optimal solutions were found most of the time. Although it has not been possible to determine the optimal set of paths for our 100-node network by exhaustive search (because the number of possible states makes doing so prohibitive), it is significant to note that all of the solutions produced by our NN model are better than the solutions found by our shortest-path heuristic. We are strongly encouraged by the capability of our simulation model to handle NNs with nearly 1000 neurons.

Although the solutions obtained by the link-neuron routing NN model are typically not as good as those produced by the path-neuron model, the link-neuron model does, in fact, represent a significant advance in our study of NN applications to network problems. In particular, the ability of this model to generate complete paths by using excitatory connections between neurons belonging to the same path can be viewed as a first step toward the more general, and more difficult, routing problem in which the paths between each SD pair are not specified in advance. In that case, the NN must piece together complete paths from individual links.

We have also addressed the use of Hopfield NN models to solve the problem of link activation, or scheduling, in multihop packet radio networks. Both nonsequential-activation scheduling (NAS) and sequential-activation scheduling (SAS) models have been studied. In addition to using Lagrange multipliers, other important aspects of our models include incorporating heuristics into the equations of motion and using Gaussian simulated annealing, both of which encourage the evolution of the NN to optimal solutions.

Extensive simulation results have demonstrated the capability of our models to find optimal, i.e., minimum-length, schedules in many cases for which our heuristic (i.e., non-NN) approach was unable to do so. The degree of success obtained by the NN models is related to the degree to which the problem is constrained. For example, in some Monte-Carlo simulations of NAS problems, all of the solutions obtained from 100 different initial states (random seeds) were optimal. For a very highly constrained problem, in which 99.6% of the slots were either activated or blocked in a typical schedule, 21% of the solutions were optimal. Although this may appear to be a small number, it is notable that the NN model was able to determine optimal schedules for such a highly constrained problem, whereas our purely-heuristic approach was unable to do so. In studies of the SAS model, as many as 79% of the solutions were optimal.

Analysis of the sensitivity of our scheduling NN models to variations in parameter values and communication requirements has shown that both models are fairly robust. Both the NAS and the SAS models have produced optimal or nearly optimal schedules for a number of diverse problem instances without the need to adjust the parameters to accommodate different communication requirements. Simulations have shown that the performance of the NN does depend on the set of parameter values, but good performance is achieved over a broad range of these values.

The fact that global minima are not always found (a common characteristic of Hopfield NNs) is typical of heuristic algorithms for the solution of difficult combinatorial-optimization problems. In many such problems, optimal solutions cannot be guaranteed without exhaustive search. However, the inability to guarantee a global optimum is mitigated by the fact that repeated runs are possible from different initial conditions; thus the best solution that is found can be chosen as the solution to the problem. Although the simulation runs begin in random initial states, this method is not simply one of random search; system evolution is guided by the equations of motion, which are derived from the energy function, which in turn is based on the objective function and the system constraints. The fact that most of our solutions are so close to the optimum value in such a large fraction of the cases studied demonstrates the robustness of our models and suggests that they may perform well in considerably larger examples as well.

Our studies of routing and scheduling problems have verified our hypothesis that these two network control functions are not independent and that schemes should be developed that jointly choose routes and link-activation schedules. We have characterized the joint routing-scheduling

problem as a combinatorial-optimization problem, and we have outlined the major components of a NN model for its solution. However, it would be difficult to simulate this model in software except for very small networks, because of the large number of neurons and interconnections that are involved. It is hoped that future developments in the hardware design of NN components will be able to incorporate the techniques developed in this study to permit the solution of complicated communication network control problems of this type.

REFERENCES

1. B. Hajek and G. Sasaki, "Link Scheduling in Polynomial Time," *IEEE Trans. Inf. Theory* **34**, 910-917 (1988).
2. L. Tassiulas, "Scheduling Problems in Multihop-Packet Radio Networks," Master's thesis, University of Maryland, 1989.
3. L. Tassiulas and A. Ephremides, "An Algorithm for Joint Routing and Scheduling in Radio Networks," *Proceedings of the American Control Conference*, pp. 697-702, June 1989.
4. D. J. Baker, A. Ephremides, and J. A. Flynn, "The Design and Simulation of a Mobile Radio Network with Distributed Control," *IEEE J. Selected Areas Commun. SAC-2*, 226-237 (1984).
5. A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," *Proc. IEEE* **75**, 56-73 (1987).
6. M. J. Post, A. S. Kershenbaum, and P. E. Sarachik, "Scheduling Multihop CDMA Networks in the Presence of Secondary Conflicts," *Algorithmica* **4**, 365-393 (1989).
7. E. Arıkan, "Some Complexity Results About Packet Radio Networks," *IEEE Trans. Inf. Theory* **IT-30**, 681-685 (1984).
8. J. E. Wieselthier, "Code-Division Multiple-Access Techniques and Their Application to the High-Frequency (HF) Intratask Force (ITF) Communication Network," NRL Report 9094, September 1988.
9. L. Tassiulas, A. Ephremides, and J. Gunn, "Solving Hard Optimization Problems Arising in Packet Radio Networks Using Hopfield's Net," *Proceedings of the 1989 Conference on Information Sciences and Systems*, pp. 603-608, March 1989.
10. Special Issue on "Neural Networks in Communication," *IEEE Communications Magazine*, **27**, November 1989.
11. Special Issue on "Neural Networks in Control Systems," *IEEE Control Systems Magazine*, **10**, April 1990.
12. H. E. Rauch and T. Winarske, "Neural Networks for Routing Communication Traffic," *IEEE Control Systems Magazine* **8**, 26-31, April 1988.
13. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "A Neural Network Approach to Routing in Multihop Radio Networks," *Proceedings of IEEE INFOCOM'91*, pp. 1074-1083, April 1991.
14. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Neural Network Techniques for Scheduling and Routing Problems in Multihop Radio Networks," *MILCOM'91 Conference Record*, pp. 407-413, November 1991.

15. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "The Application of Hopfield Neural Network Techniques to Problems of Routing and Scheduling in Packet Radio Networks," NRL Memorandum Report 6730, November 1990.
16. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Scheduling Link Activation in Multihop Radio Networks by Means of Hopfield Neural Network Techniques," NRL Memorandum Report 6885, September 1991.
17. J. E. Wieselthier, C. Barnhart, A. Ephremides, and W. Thoet, "Routing and Scheduling in Packet Radio Networks: A Hopfield Network Approach," *Proceedings of the 1990 Conference on Information Sciences and Systems*, p. 185, March 1990.
18. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "Sequential Link Activation in Multihop Radio Networks by Means of Hopfield Neural Network Techniques," *Proceedings of the 1991 International Symposium on Information Theory*, p. 155, June 1991.
19. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "The Use of Hopfield Neural Nets in Combinatorial-Optimization Problems Arising in Radio Communication Networks," *Proceedings of the Thirtieth International Meeting of the Institute of Management Sciences (TIMS)*, July 1991.
20. U. Mukherji, "A Periodic Scheduling Problem in Flow Control for Data Communication Networks," *IEEE Trans. Inf. Theory* **35**, 436-443 (1989).
21. A. S. Tanenbaum, *Computer Networks, Second Edition* (Prentice Hall, Englewood Cliffs, NJ, 1988).
22. J. J. Hopfield and D. W. Tank, "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics* **52**, 141-152 (1985).
23. A. Ephremides and T. V. Truong, "Scheduling Broadcasts in Multihop Radio Networks," *IEEE Trans. Commun.* **38**, 456-460 (1990).
24. L. Tassiulas and A. Ephremides, "Jointly Optimal Routing and Scheduling in Packet Radio Networks," to appear in *IEEE Trans. Inf. Theory*, January 1992.
25. D. Bertsekas and R. Gallager, *Data Networks* (Prentice Hall, Englewood Cliffs, NJ, 1987) pp. 322-324.
26. J. W. Suurballe, "Disjoint Paths in a Network," *Networks* **4**, 125-145 (1974).
27. B. Kamgar-Parsi and B. Kamgar-Parsi, "On Problem Solving with Hopfield Neural Networks," *Biological Cybernetics* **62**, 415-423 (1990).
28. B. Kamgar-Parsi, J. A. Gualtieri, J. E. Devaney, and B. Kamgar-Parsi, "Clustering with Neural Networks," *Biological Cybernetics* **63**, 201-208 (1990).
29. D. E. Van den Bout and T. K. Miller III, "Graph Partitioning Using Annealed Neural Networks," *IEEE Trans. Neural Networks* **1**, 192-203 (1990).
30. E. Wacholder, J. Han, and R. C. Mann, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem," *Biological Cybernetics* **61**, 11-19 (1989).
31. J. C. Platt and A. H. Barr, "Constrained Differential Optimization," *Proceedings of the IEEE 1987 Neural Information Processing Systems Conference*, pp. 612-621, 1987.

32. Y. Akiyama, A. Yamashita, M. Kajiura, Y. Anzai, and H. Aiso, "The Gaussian Machine: A Stochastic Neural Network for Solving Assignment Problems," *J. Neural Network Computing* **2**, 43-51 (1991).
33. B. C. Levy and M. B. Adams, "Global Optimization with Stochastic Neural Networks," *IEEE International Conference on Neural Networks*, pp. III-681 - III-689, 1987.
34. I. Holyer, "The NP-Completeness of Edge-Coloring," *SIAM J. Comput.* **10**, 718-720 (1981).
35. R. G. Ogier, "A Decomposition Method for Optimal Link Scheduling," *Proceedings of the 24th Allerton Conference*, October 1986.
36. M. J. Post, P. E. Sarachik, and A. S. Kershenbaum, "A 'Biased Greedy' Algorithm for Scheduling Multi-Hop Radio Networks," *Proceedings of the '85 Conference on Information Science and Systems*, pp. 564-572, March 1985.
37. P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, (D. Reidel Publishing Company, Dordrecht, 1987).
38. S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* **220**, 671-680 (1983).
39. B. Hajek, "A Tutorial Survey of Theory and Applications of Simulated Annealing," *Proceedings of the 24th IEEE Conference on Decision and Control*, pp. 755-760, December 1985.

THE PROBLEMS OF ROUTING AND SCHEDULING IN MULTIHOP RADIO NETWORKS — A HOPFIELD NEURAL NETWORK APPROACH

1.0 INTRODUCTION

Two of the most important facets of multihop packet radio network design and control are routing and channel access. Although these issues are highly interdependent, few studies have addressed them jointly (e.g., see Refs. 1-3). For example, the choice of routes determines the amount of traffic that must be carried over each of the network's links, and thus determines the communication requirements that must be satisfied by the channel-access mechanism. Despite the intimate relationships that exist between these network control mechanisms, they are almost invariably addressed separately, resulting in network operation that may be far from optimal. In this report we make a step toward the development of schemes for the joint control of routing and channel access by making use of the recently developed Hopfield neural network (NN) method for the solution of combinatorial-optimization problems.

We assume the use of a contention-free form of channel access, which is alternately known as "link activation" or "scheduling" [4-6]. Under this channel-access mechanism, the nodes are assigned noninterfering, periodically recurring time slots in which to transmit their packets. In generating these transmission schedules, it is possible to take advantage of the spatial separation of the nodes, thus permitting two nodes separated by a sufficiently large distance to transmit simultaneously. In spread-spectrum code-division multiple-access (CDMA) systems, it is also possible for several nodes in the same vicinity to transmit simultaneously, provided that they use different frequency-hopping patterns. The determination of optimal schedules, i.e., schedules that satisfy the traffic demand in the minimum number of time slots, is a difficult combinatorial-optimization problem. In fact, it is by now well known that this problem, in almost all of its forms, is NP-complete (i.e., cannot be solved by an algorithm of polynomial complexity) [1, 2, 7]. Thus heuristics are generally used to produce suboptimal link-activation schedules. Reference 8 presents a discussion of link-activation methods, with an emphasis on CDMA considerations.

An alternate approach to the link-activation problem is the use of a Hopfield NN to generate good, although not necessarily optimal, communication schedules [9]. Under this approach, the scheduling problem is transformed into a graph-coloring problem, which is known to be NP-complete. The objective of this problem is the determination of a coloring of the graph that requires the minimum number of colors, where each color corresponds to a time slot. A Hopfield NN is then designed to solve the corresponding coloring problem. As a result of the successful application of this method to the scheduling problem, we decided to extend it to the joint routing-scheduling problem. Our approach fits well into the currently widening interest of the research community in applying NN methods to communication and control problems (e.g., see Refs. 10, 11). Reference 12 discusses a NN approach for a different version of the routing problem.

Like the pure scheduling problem, the joint routing-scheduling problem can also be posed as a combinatorial-optimization problem. Our study is, in fact, the first formulation of the joint routing-scheduling problem as a problem of combinatorial optimization. The objective is now to

Appendix A

HOPFIELD NEURAL NETWORKS AND THEIR APPLICATION TO OPTIMIZATION PROBLEMS

Since the introduction of the use of neural networks (NN) for the solution of combinatorial-optimization problems by Hopfield and Tank [A1], there have been many applications of that idea to diverse optimization problems of high computational complexity. In this appendix we review the principles of Hopfield NNs, and we show why they are well-suited to such problems. Our discussion is based primarily on the Traveling Salesman Problem (TSP), which was the application originally considered by Hopfield and Tank, and which has been perhaps the most widely studied combinatorial-optimization problem. It is hoped that this appendix will provide enough background material to facilitate an understanding of the NN models we have developed for routing and scheduling problems.

A.1 Hopfield Neural Networks

A NN consists of a large number of elements that behave like simple analog amplifiers, and are highly interconnected in a manner that permits highly parallel and fault-tolerant computation. These amplifier elements are called "neurons" because their behavior is similar to that of biological neurons, which are also simple highly interconnected analog devices. Each neuron corresponds to a binary variable in the system that is being modeled by the NN. A Hopfield NN is a NN with a special structure that can achieve a very rapid solution to a specific optimization problem.* The generic combinatorial-optimization problem that is solved by a Hopfield NN consists of determining which of the neurons should be "on" (have a value of 1) and which should be "off" (have a value of 0) so that some cost function is minimized.

In a Hopfield NN, the strengths of the pairwise connections between neurons are chosen so that the desired objective function is minimized. Appropriate choice of connection strengths also ensures that any constraints that are present in the optimization problem are not violated. We note at the outset that the solutions provided by Hopfield NNs are not necessarily optimal because local rather than global minima may be found. Also, the class of objective functions that can be minimized by this method is somewhat limited. However, reliable convergence to good solutions has been demonstrated for a number of difficult and important combinatorial-optimization problems including those discussed in this report.

The neuron input-output relation typically has the sigmoidal form

$$V_i = g(u_i) = \frac{1}{2} \left[1 + \tanh \left(\frac{u_i}{u_0} \right) \right]$$

* Other types of NNs are well-suited for learning applications, including a wide range of pattern-recognition tasks. References A2 and A3 are general references on NNs.

as shown in Fig. A1. Here, V_i is the output voltage of neuron i (which can take on values between 0 and 1), u_i is the input voltage to neuron i (which can range from $-\infty$ to ∞), and u_0 is a parameter that characterizes the slope of the nonlinearity. A key feature of these networks is, in fact, the analog nature of these processing elements, which permits the embedding of discrete problems in a continuous solution space. As we discuss later in this appendix, permitting the search for an optimal solution to proceed through the interior of a continuous region yields better solutions than are possible with strictly digital processing elements, and determines them very rapidly when the NN is implemented in hardware.

Figure A2 shows a portion of a Hopfield NN. The output of each neuron is connected to the inputs of a number of other neurons through resistors whose values are chosen to control the level of interaction between the neurons. Each neuron has both normal and inverted outputs; thus it can provide either excitatory or inhibitory synaptic connections as needed. Neurons normally interact with each other on a pairwise basis. In particular, a synapse between neurons i and j is defined by a connection weight T_{ij} (implemented by using a resistor of value $1/|T_{ij}|$), whose value is positive if the connection is excitatory and negative if it is inhibitory. For example, assume that an inhibitory connection is present between neurons i and j (i.e., T_{ij} is negative). If neuron i is "on" it will discourage neuron j from turning "on," and conversely. Actually, since the output voltages take on analog values between 0 and 1, the degree of inhibition applied to neuron j is proportional to the output voltage of neuron i . In addition, bias currents can be applied directly to each neuron, e.g., I_i is applied to neuron i . These bias currents represent fixed inputs that are applied to the neurons, and they are independent of the state of the other neurons in the network. The combined effect of the connection weights and bias currents encourages the NN to find a solution that minimizes the desired function while satisfying a number of problem constraints, as we describe in the following.

Hopfield NNs evolve from some initial state to a final state that represents a local (but not necessarily global) minimum of the Lyapunov energy function

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N T_{ij} V_i V_j - \sum_{i=1}^N V_i I_i,$$

where N is the number of neurons. Thus an $N \times N$ connectivity matrix T can be defined whose elements are the connection weights T_{ij} . Convergence to a stable state is guaranteed as long as the connections are symmetric (i.e., $T_{ij} = T_{ji}$) [A1]. These conditions are satisfied in many problems of practical interest.

In the above expression, the double summation represents the pairwise contribution to the energy by all possible pairs of neurons (which are weighted by the connection weights T_{ij}); the single summation represents the contributions that the neurons make on an individual basis (which are weighted by the bias currents I_i). Note that the T_{ij} 's and I_i 's represent the combined impact of the function to be minimized and the constraints that are to be satisfied. More complicated forms of the energy function have also been considered in the literature, e.g., those that include connection weights for triplets of neurons (e.g., T_{ijk} connecting neurons i , j , and k , which results in a contribution of the form $T_{ijk}V_iV_jV_k$). However, they are generally much more difficult to implement, and it is not clear that they offer an advantage; thus we did not consider them in this study.

In the limit of high gain (i.e., a steep nonlinearity in the input-output relation of a neuron), the minima of the energy function occur only at the corners of the N -dimensional hypercube, i.e., for neuron output voltages of $V_i = 0$ or 1. Thus, although the system state evolves over the interior of an N -dimensional hypercube, the solution corresponds to a discrete system representation in which one of the 2^N corners is selected.

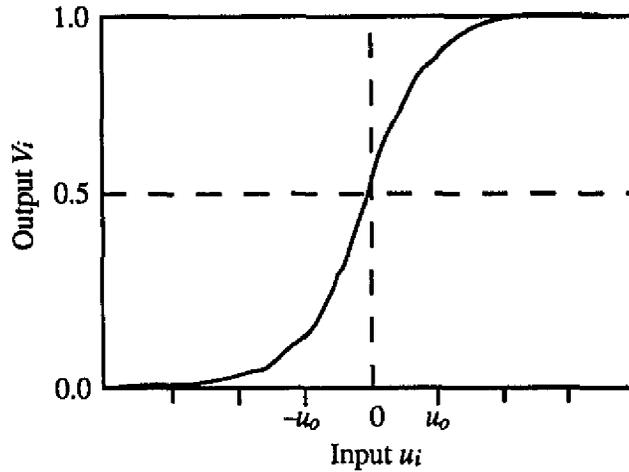


Fig. A1 — Input-output nonlinearity

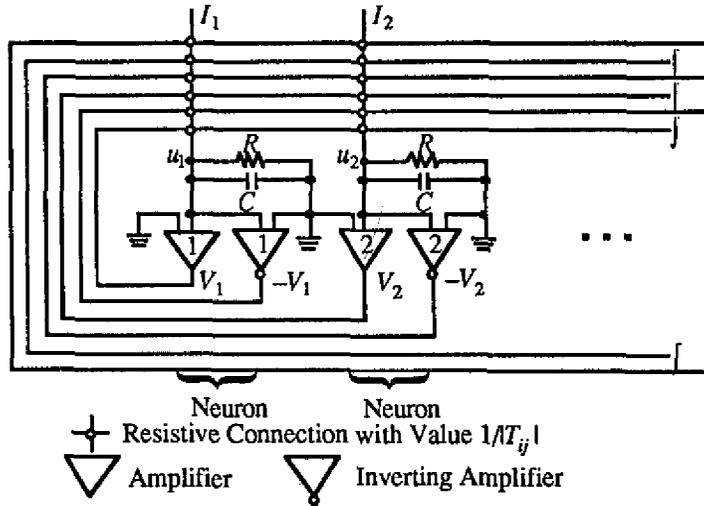


Fig. A2 — Portion of a Hopfield NN

Clearly, the form of the energy function presented above is not completely general, so it is not possible to define such a Lyapunov function for some minimization problems. However, a variety of interesting and diverse problems have been formulated by using Hopfield networks. For example, Ramanujam and Sadayappan studied several graph-partitioning problems [A4], Brandt et al. studied the list-matching problem [A5], and Foo and Takefuji studied job-shop scheduling [A6, A7]. We note that when the function to be minimized is not of the form shown in the above energy equation, it is often possible to define a related energy function that provides good, although not necessarily optimal, performance for the problem of interest. We have, in fact, done so in our studies of Hopfield NNs for minimizing congestion in networks.

Examination of Fig. A2 yields the following form for the equation of motion at neuron i :

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} - \frac{\partial E}{\partial V_i} = -\frac{u_i}{\tau} + \sum_{j=1}^N T_{ij}V_j + I_i,$$

where $\tau = RC$ (which may be set equal to 1 without loss of generality) is the time constant of the RC circuit connected to the neuron. As the system evolves, the energy function decreases

monotonically until equilibrium at a (local) minimum is reached. Since only a local minimum can be guaranteed, the final state depends on the initial state at which the system evolution is started.

The NN is “programmed” by implementing the set of connection weights and bias currents that correspond to the function that is to be minimized. An analog implementation of a Hopfield NN will normally converge within at most a few RC time constants, thus providing an extremely rapid solution to a complex optimization problem. In our studies (as in most studies of this technique) we have simulated the system dynamics in software. Although such software solutions are extremely time-consuming, they verify the soundness of the use of the Hopfield NN approach for optimization problems of this type and suggest that hardware implementations may be worthwhile.

To illustrate the considerations associated with the selection of these system parameters, we now discuss the application of Hopfield NNs to the Traveling Salesman Problem (TSP). Since some of the optimization considerations associated with our routing and scheduling problems are similar to those related to the TSP, a discussion of the TSP provides useful background material for the presentation of our model. It also permits us to highlight the differences between our problem and the TSP, and thus to illustrate some of the novel contributions of our research.

A.2 The Traveling Salesman Problem—A Hopfield Net Formulation

The Traveling Salesman Problem is one of the classic NP-complete problems of combinatorial optimization, and it provides a convenient vehicle to explain the use of the Hopfield NN methodology. Although practical complications arise when the number of cities exceeds values of around 50, recently developed techniques may extend the power of the Hopfield NN approach to considerably larger problems. Fortunately, such limitations are not always encountered; our methodology has permitted the simulation of rather large examples, which are discussed in this report.

The TSP is defined as follows. A salesman would like to visit each of a set of n cities exactly once and return to the city of origin, while minimizing the total distance traveled. He is given the pairwise distances of separation d_{ij} between cities i and j ($1 \leq i < n$, $1 \leq j < n$, $i \neq j$). It is easy to see that there are $n!$ possible solutions, i.e., orderings of the cities. By observing that each solution has a $2n$ -fold degeneracy (because the same tour can begin in any city and reversing the order of tour does not affect the cost function), the number of distinct paths is reduced to $n!/2n$. However, this number is still far too great to examine by exhaustive search when n is large, and heuristics are normally used to provide suboptimal solutions [A8]. Like these heuristics, the Hopfield NN approach cannot be expected to produce the optimal solution at every attempt. A reasonable expectation is the determination of *good* solutions a significant fraction of the time, as we explain in detail later.

The TSP requires that we specify the sequence in which the n cities are visited. A natural way to represent any particular solution is in the form of a permutation matrix, such as that shown below for a five-city problem. In this array, the letters represent the cities and the numbers represent the position on the tour. For example, city C is visited first because in row C a “1” appears in the first column. Similarly, city A is visited second because in row A a “1” appears in the second column. Continuing in this manner, the entire tour is specified by the sequence C, A, E, B, D, C . Clearly, since each city must be visited exactly once, there is exactly one “1” in each row. Similarly, since only one city is visited at a time, an admissible solution has exactly one “1” in each column. The length of the tour corresponding to this particular sequence is $d_{CA} + d_{AE} + d_{EB} + d_{BD} + d_{DC}$.

		position in tour				
		1	2	3	4	5
city	A	0	1	0	0	0
	B	0	0	0	1	0
	C	1	0	0	0	0
	D	0	0	0	0	1
	E	0	0	1	0	0

To formulate this problem as a Hopfield NN problem, a neuron is defined for each element of the array. In general, the notation V_{Xj} represents the output voltage of the neuron corresponding to city X being visited in the j th position on the tour. Thus $N = n^2$ neurons are needed to represent the state in a problem with n cities. The following energy function is defined to reflect the desire to minimize the total path length while satisfying the constraints just described:

$$E = \frac{a}{2} \sum_X \sum_i \sum_{j \neq i} V_{Xi} V_{Xj} + \frac{b}{2} \sum_i \sum_X \sum_{Y \neq X} V_{Xi} V_{Yi} + \frac{c}{2} \left(\sum_X \sum_i V_{Xi} - n \right)^2 + \frac{d}{2} \sum_X \sum_{Y \neq X} \sum_i d_{XY} V_{Xi} (V_{Y,i+1} + V_{Y,i-1}) .$$

The first three terms represent the effect of equality constraints, and each must be zero if these constraints are satisfied. In particular, the first term is zero if and only if each city row contains no more than one "1;" i.e., each city must be visited not more than once. The second term is zero if and only if each position-in-tour column contains not more than one "1;" i.e., only one city can be visited at a time. Finally, the third term is zero if and only if there are exactly n "1" entries in the entire matrix. The last term represents the total distance traveled, and is thus the performance index that we actually want to minimize. Note that the subscripts are defined modulo n , so that the city n is adjacent to both city $n-1$ and city 1. In all cases, the factor of $1/2$ is present because the summations include all terms twice. Later in this appendix we discuss some issues related to the choice of the coefficients a , b , and c .

We emphasize that the neuron output voltages take on values in the continuum (0,1). The permutation matrix condition is normally satisfied only when the system has reached equilibrium.

In the last few years, a number of alternative formulations of the energy function have been developed for the TSP under which different forms of the constraints are imposed on the system and for which better performance is claimed; e.g., see Ref. A5. However, we confine the present discussion to Hopfield and Tank's energy function because the objective of this appendix is to demonstrate the application of Hopfield NNs to constraint-satisfaction problems, and this can be done without an exhaustive discussion of the totality of NN methods that have been developed for this problem.

Given the above form of the energy equation, the form of the connection weights is easily determined as follows:

$$T_{Xi,Yj} = - a \delta_{XY} (1 - \delta_{ij}) - b \delta_{ij} (1 - \delta_{XY}) - c - d d_{XY} (\delta_{j,i+1} + \delta_{j,i-1}) ,$$

where δ_{ij} is the Kronecker delta (i.e., $\delta_{ij} = 1$ if $i = j$ and is 0 otherwise). The external input bias currents are simply

$$I_{Xi} = c n .$$

The corresponding equation of motion for the input voltage to neuron X_i is

$$\frac{du_{X_i}}{dt} = -\frac{u_{X_i}}{\tau} - a \sum_{j \neq i} V_{X_j} - b \sum_{Y \neq X} V_{Y_i} - c \left(\sum_X \sum_j V_{X_j} - n \right) - d \sum_{Y \neq X} d_{XY} (V_{Y,i+1} + V_{Y,i-1}).$$

Recall that $V_{X_i} = 1$ means that city X is visited in the i th position of the tour. Thus, a negative value of du_{X_i}/dt tends to turn neuron X_i "off" (discouraging the visiting of city X in the i th position), whereas a positive value tends to turn it "on." In numerical simulations, the input voltages are updated synchronously as follows:

$$u_{X_i}^{new} = u_{X_i}^{old} + \Delta t \left[\frac{du_{X_i}}{dt} \right]^{old}.$$

The new output voltages are obtained by passing the updated input values through the nonlinearity $V_i = g(u_i)$. In their simulations of a ten-city TSP, Hopfield and Tank used the following parameter values: $a = b = 500$; $c = 200$; $d = 500$; $u_o = 0.02$; and $\tau = 1$. The value of Δt was not mentioned in Ref. A1, but 10^{-5} appears to be a reasonable value for use with these system parameters (see e.g., Ref. A9). If Δt is too large, the system may evolve too rapidly, thereby missing optima and possibly resulting in oscillations; if it is too small, then convergence takes an excessively long time. The choice of system parameters for the connection weights and bias currents is a crucial aspect of the NN design problem. Later in this appendix we discuss the issues associated with the choice of these parameters and the consequent impact on NN design and performance.

The equations of motion have a satisfying intuitive interpretation. The first term simply represents the RC decay of the input voltage to the neuron. The next three terms represent the impact of the system constraints on system evolution. We refer to each of them here by the coefficient that multiplies them, i.e., as the a , b , and c terms. As discussed earlier, the a term represents the constraint that city X be visited only once during the tour. Thus, if any of the other V_{X_j} 's are nonzero (which correspond to visiting city X in position j), the input voltage to neuron X_i is reduced. Similarly, the b term represents the constraint that only one city be visited in position i . Thus, if any of the other V_{Y_i} 's are nonzero (which correspond to visiting city Y in position i), the input voltage to neuron X_i is reduced. Note that the a and b terms are purely inhibitory, i.e., they cannot be positive. The c term represents the constraint that exactly n neurons be turned "on" in the entire NN, and it is excitatory if an insufficient number are currently on and inhibitory if too many neurons are currently on. This term is the same for all neurons in the network.

The d term, which represents the impact of the length of the tour on the input to neuron X_i , is a purely inhibitory term. It is less inhibitory when the distances between consecutive cities are small and more inhibitory when they are large.

It is important to note that although the equations of motion reflect the constraints that must be satisfied by admissible TSP tours (which are characterized by neuron output voltages that are all 0's and 1's and that satisfy the permutation matrix constraint), they are applied to a continuous system in which the output voltages can take on values in the continuum (0,1). As should be clear from the equation of motion, the impact of an inhibitory connection is proportional to the output voltage of the neuron that supplies it. Normally, the TSP constraints are not satisfied until an equilibrium state is reached.

In the problem formulation, n is the number of cities. However, Hopfield and Tank observed that when this number was used in the equations of motion, it was difficult to ensure that a sufficient number of neurons were activated. A possible explanation is that it is difficult to overcome the inhibitory effect of the d term, which was discussed above. Thus additional bias current is needed to, in effect, adjust the neutral positions of the amplifiers. The bias current

appears in the equation of motion as the quantity cn . Without changing the size of the problem (i.e., the number of neurons in the model), one can increase n in that equation to effect the desired increase in bias current. This was done by choosing $n = 15$ for the 10-city problem.

An initial state must be chosen as the starting point for the iteration. Although we are searching for an admissible solution (i.e., one for which all neuron voltages are 0 or 1 and for which all constraints are satisfied), it is not advisable to start the search from such a state. This is because it is difficult to leave a state in which the constraints are satisfied, since doing so often results in an increase in system energy. To permit a search over a larger portion of the N -dimensional hypercube, a value in the interior of the search space is chosen as the starting point.

Nominal initial values of the neuron input voltages were chosen to be equal to the same constant u_{00} , so that no tour would be preferred above any other a priori. For the 10-city problem, u_{00} was chosen so that

$$\sum_X \sum_i V_{Xi} = 10.$$

This value is reasonable because it is the desired value of the summation when convergence has been reached. However, it is inadvisable to start the iteration with all voltages exactly equal because it is then difficult for the NN to break the symmetry and thus to choose a promising direction for the search. Therefore, the nominal initial values of the neuron input voltages were then perturbed by a small amount (a different random number, uniformly distributed between $-0.1u_0$ and $+0.1u_0$, for each neuron) before beginning the iteration. Typically, a large number of simulation runs (e.g., 100) are performed, each with a different random seed. Although not all solutions will necessarily be good (or even admissible), the best solution from a collection of runs can be chosen as the solution to the problem of interest. We note that system evolution from any given initial state is deterministic. The only randomness in this model arises from the use of a random initial state.*

A key feature of the system evolution, as discussed earlier, is that the system state evolves in the *interior* of the N -dimensional hypercube, whereas admissible tours can be described only on the corners. Thus the neuron voltages can be interpreted as solutions of the TSP only when convergence has been reached (i.e., the permutation matrix condition of exactly one "1" per row and one "1" per column is satisfied). In practice, convergence can be declared as soon as there is a clear "winner" in each row and column such that this condition is satisfied, although it is often advisable to continue the iteration until relatively tight thresholds are satisfied because it is possible for changes in system state to occur (e.g., voltages above 0.9 could correspond to "1" and voltages below 0.1 could correspond to "0").

Hopfield and Tank have, in fact, demonstrated that the use of analog neurons provides considerably better performance than the use of binary neurons. The same energy equation and equations of motion were used in the simulation of a discrete system. However, a simple threshold was used to determine whether each neuron's output voltage was 0 or 1. The solutions that were found were of considerably poorer quality (i.e., greater tour length) than those obtained by using analog neurons. This is true because the use of binary neurons forces the search to make a binary decision on the state of each neuron at each step of the iteration, thereby limiting the search to the corners of the hypercube. In contrast, the use of analog neurons enables the solution to follow trajectories of decreasing energy, thereby permitting decisions to be postponed until there is a clear winner. In general, use of too steep a nonlinearity (too small a value of u_0) confines the search to a region near the edges of the hypercube, and may thus prevent the finding of the best

* Later in this appendix we discuss the use of simulated annealing, which is a technique that applies noise of gradually decreasing power to help the search escape from local minima, and thereby find the global minimum.

paths. On the other hand, use of a linearity whose gain is too low will result in final states that are not sufficiently close to 0 or 1.

A.3 On the Selection of Parameters for the Hopfield Neural Network

A difficult aspect of the design of Hopfield NNs is the choice of the parameters used in the connection weights; for the case of the TSP formulation discussed here, we are referring to the parameters a , b , c , and d . We noted earlier that each of the three constraint terms in the energy function (which are multiplied by $a/2$, $b/2$, and $c/2$, respectively) vanish when the problem constraints are satisfied. Thus, in principle, it is possible to reach a minimum of the energy function for any values of these parameters. However, the relative magnitudes of these parameters have a profound effect on the direction of system evolution throughout the N -dimensional hypercube. Their choice is critical not only to the quality of the solutions that are obtained (i.e., the tour length), but also to whether convergence to admissible solutions (i.e., solutions for which all constraints are satisfied) is, in fact, achieved.

The coefficients a , b , c , and d reflect the relative importance of the corresponding terms in the energy equation. In Hopfield and Tank's studies, these parameters were chosen by trial and error to determine the values that result in the best performance. We make the observation here that use of an overly large value of d (in comparison to a , b , and c) may lead to solutions that do not visit all cities. Inadmissible states that may occur include those in which some cities are visited twice as well as those in which no neurons are activated in a particular tour position. Such behavior may arise because in this case the NN is concerned primarily with minimizing the length of the tour, not with the generation of complete tours. Similarly, if d is too small, overly long tours may be generated because not enough importance is given to the lengths of the tours. The values of the coefficients that represent the equality constraints, i.e., a , b , and c , must also be determined carefully.

Another important parameter is the bias current, which for all neurons is $I_i = cn$. We noted earlier that Hopfield and Tank used a value of $n = 15$ in a network corresponding to 10 cities because additional bias was needed to ensure that the correct number of neurons was activated. The two other system parameters are the slope of the nonlinearity (Hopfield and Tank used $u_0 = 0.02$) and the time step size Δt (the value of which was not specified).

Selecting system parameters by trial and error is a tedious process. Moreover, convergence to admissible low-energy solutions is not guaranteed. For example, Wilson and Pawley [A9], while acknowledging the fundamental importance of Hopfield and Tank's method, claimed that they were unable to reproduce the low-energy solutions that Hopfield and Tank claimed they found in their initial study. Most of Wilson and Pawley's runs did not converge to admissible tours, and those that did were only slightly better than randomly chosen tours. After examining the use of several heuristics to choose connection weights, they concluded that the basic method is unreliable.

A number of other researchers have also investigated the issues associated with the choice of parameters for Hopfield-Tank TSP networks. For example, Hegde, Sweet, and Levy [A10] claim to have developed a "cookbook" approach for the determination of these parameters, and they provide an explanation of why these networks seem to be of decreasing usefulness as the number of cities increases. Brandt et al. [A5] present an alternative energy function that they claim provides better results than the original formulation. Kahng [A11] examines the strengths and limitations of the Hopfield-Tank formulation, and summarizes a variety of heuristics that have been proposed for NN implementations for the TSP. Although his assessment of the Hopfield-Tank approach is more favorable than that of Wilson and Pawley, he notes that its lack of robustness may limit its applicability.

Few studies of the Hopfield NN model have gone deeper than a study of the behavior of the differential equations that determine the course of system evolution. One that has done so is a recent paper by Aiyer, Niranjan, and Fallside [A12], who analyze the eigenvalues of the

connection matrix for the TSP and the geometry of the corresponding subspaces. They provide an explanation of the dynamics of the Hopfield network, including a procedure for the determination of the coefficients in the connection matrix. They claim convergence to admissible solutions 100% of the time for problems with as many as 50 cities. Moreover, the average quality of the solutions for TSPs with up to 30 cities was at least as good as those produced by the nearest neighbor algorithm [A8], which is a well-known heuristic for the TSP. Although the 50-city problem is still small, in terms of the TSP problems that can be solved by other (non-NN) methods, the results of Ref. A12 indicate that the Hopfield NN is applicable to larger combinatorial-optimization problems than previously believed.

A.4 The Use of Lagrange Multipliers to Determine the Coefficients in the Connection Weights

All of the approaches discussed thus far have used constant values for the parameters a , b , and c . Wacholder, Han, and Mann [A13] made the crucial observation that, since these parameters are associated with equality constraints that have been incorporated into the energy function, they can be modeled as Lagrange multipliers. This approach permits the connection weights to evolve with the system state and adapt to problem-specific parameters. The problem they addressed was the Multiple Traveling Salesman Problem (MTSP), which is an extension of the standard TSP problem. Under the MTSP, M salesmen are to start from a specified city and cooperatively visit the remaining $N-1$ cities, such that each city is visited by exactly one salesman, all cities are to be visited, and the total tour length is to be short.

We illustrate this method for the standard TSP (the details of the MTSP are not critical here) by rewriting the energy function in the following form:

$$E = \sum_{i=1}^3 \lambda_i E_i + E_p,$$

where $\lambda_1 = a/2$, $\lambda_2 = b/2$, and $\lambda_3 = c/2$. The E_i 's represent the corresponding "constraint energy" terms (all of which are zero for an admissible solution), and E_p is the path length of the tour (the term associated with parameter d). The iterative equation for the input voltages is the same as before, except that the λ_i 's are used instead of the fixed values of a , b , and c . However, the Lagrange multipliers also evolve as

$$\frac{\partial \lambda_i}{\partial t} = |E_i|$$

The use of the absolute value here is based on a recommendation by Platt and Barr [A14] as a means to encourage the satisfaction of the constraints. In iterative form, this is expressed as

$$\lambda_i^{new} = \lambda_i^{old} + \Delta t E_i^{old},$$

where it is reasonable to set the initial values of the λ_i 's equal to 1. The λ_i 's thus increase in proportion to the corresponding constraint energy. Thus, when a constraint is not satisfied, the corresponding connection weights continue to increase, thereby encouraging movement in a direction that will ultimately satisfy the constraint. Finally, when the constraint on E_i is satisfied (in which case $E_i = 0$), λ_i stops increasing.

The coefficient multiplying the tour length (i.e., d) cannot be determined in this manner because this term in the energy equation is the quantity that is to be minimized; it does not represent an equality constraint. A typical value for d would be about 1 (± 0.5).

The primary advantage of this method is that it eliminates the need to perform a trial-and-error search for the best system parameters. Such a search is especially time-consuming in large networks because many (e.g., 100) runs with different random initial conditions are typically needed to assess the performance achievable when a particular set of parameters is used. In addition, based on our application of this method to routing problems, we suspect that the dynamic nature of the λ_i 's provides better performance than the use of the best set of coefficients with constant values. This is because the relatively small initial values of the λ_i 's permits the search to emphasize somewhat the desire to minimize the performance index (tour length in the TSP and network congestion in the routing problem) during the early part of the iteration. Toward the latter part of the iteration, the increased values of the λ_i 's more heavily penalize system states in which the constraints are not satisfied; thus the neuron voltages move closer to binary values, and equilibrium states are reached in which an admissible set of neurons is active.

Wacholder, Han, and Mann [A13] successfully tested this method on problems with up to 30 cities and five salesmen, and claimed that the algorithm always converged rapidly to admissible solutions. Like the standard implementations of Hopfield networks for the TSP, it should be possible to implement a system that incorporates the Lagrange multiplier method in hardware. Therefore, this approach represents an important advance in implementing Hopfield nets for combinatorial-optimization problems.

We have used the method of Lagrange multipliers in most of our NN studies and have concluded that this approach aids greatly in the reliable convergence to good solutions. Our studies have shown that despite the need for some parameter adjustments it is relatively fast and easy to determine good values for these parameters. Use of this method has provided considerable improvement as compared with the trial-and-error method for determining system coefficients, both in terms of the quality of solutions and the ease with which they have been obtained.

A.5 Simulated Annealing and the Search for the Global Minimum

We have noted that the equilibrium states reached by Hopfield NNs are generally local, rather than global, minima of the energy function. The reason that only local minima can be guaranteed is that the equations of motion force the system state to follow a trajectory of decreasing energy; thus it is normally not possible to escape from local minima.* Simulated annealing (SA) [A15-A17] is a probabilistic hill-climbing algorithm that facilitates the escape from local minima so that the global minimum can be found. Under this technique, the energy function normally follows a gradient descent; however, random perturbations are applied to permit occasional transitions to states with higher energy. If these perturbations are large enough, it is possible to escape the local minimum, thereby permitting the search by gradient descent to resume in a new location in the search space. We first discuss the basic SA method, and then discuss its use in conjunction with the Hopfield NN model.

The terminology of simulated annealing is based on an analogy with the physical annealing process, in which the material under study is first heated past the melting point and then cooled very slowly until it solidifies, to ensure that a minimum energy state is achieved when the final temperature is reached. If the material is cooled too rapidly, random fluctuations that occur during the cooling process will cause imperfections, e.g., a crystal grown with a large number of defects, which corresponds to a state other than that of minimum energy.

We first consider the application of SA to a purely discrete optimization problem, in which each of a number of binary variables is to be set to 0 or 1. At each instant in time the value of a variable that is chosen at random is switched. If this switch results in a lower energy, the switch is

* It may be possible to move to a state with higher energy in a system in which the connection weights are time varying, such as in the Lagrange-multiplier method discussed earlier.

coordinate the choice of routes and schedules so that communication requirements are satisfied in the minimum number of time slots. Because of the size of the NN needed for the joint routing-scheduling problem, we have found it advantageous to separately consider the routing and scheduling components before implementing a NN to solve the entire problem. Although these problems are not independent, addressing them separately is expected to provide reasonably good performance and to provide insight into the design of the NN for the combined problem.

Our study has consisted of two primary phases. In the first, we have implemented a NN model that chooses routes, based on the criterion of minimizing congestion. In the second, we have implemented a NN model that schedules packets over these routes. We have also addressed the issues in the design of a NN model for the joint routing-scheduling problem. This report summarizes these models and demonstrates the performance that has been achieved. Both areas are discussed individually in Refs. 13 and 14, and in considerably greater detail, in Refs. 15 and 16. Our NN models have also been presented at technical symposia [17-19].

1.1 A Hopfield NN for the Minimization of Congestion

The first problem we address is as follows. Given the connectivity graph of a radio communication network, a set of source-destination (SD) pairs, a specified level of traffic between each SD pair, and a set of paths connecting each SD pair, select a single path between each SD pair so that network congestion is minimized.

The first step in the development of a NN model is defining neurons that correspond to binary variables in the system that is being modeled. Most of our routing studies have focused on a path-neuron NN model in which one neuron is defined for each path between every SD pair. We define an energy function that reflects the desired goal of minimizing congestion and that incorporates the constraints that are associated with the activation of a single path per SD pair. Connections, which may be either inhibitory or excitatory and whose values are based on the energy function that is to be minimized, are established between all pairs of neurons. The NN evolves from some initial state to a final state that represents a local (but not necessarily global) minimum of the energy function. The evolution of the NN is simulated in software. Although such software solutions are extremely time-consuming, they verify the soundness of the use of the Hopfield NN approach for optimization problems of this type. They also suggest that hardware implementations (which provide convergence to a final state almost instantaneously) may be worthwhile.

The Hopfield NN methodology is different from more-traditional approaches to combinatorial-optimization problems. For example, this approach involves embedding a discrete problem in a continuous solution space. The search for an optimal solution proceeds through the interior of a continuous region, until the output voltages of the neurons ultimately converge to binary values that satisfy system constraints and that provide a good, although not necessarily optimal, solution. Appendix A provides background material on the Hopfield NN model.

The most critical issue in the design and simulation of a Hopfield NN model is the choice of the coefficients used in the connection weights. To determine good values for these coefficients, we have used the method of Lagrange multipliers. This technique permits the coefficients to vary dynamically with the evolution of the system state. This approach provides a great improvement over the trial-and-error methods used in most studies of Hopfield nets, a process that is tedious at best, and often ineffective. We provide extensive simulation results that demonstrate the effectiveness of our Hopfield path-neuron NN model in large, heavily-congested networks.

We also present an alternative link-neuron NN formulation, in which a set of paths between every SD pair is again defined but a neuron is defined for each link of every such path. Although the solutions obtained by this model are typically not as good as those produced by the path-neuron model, the link-neuron model does, in fact, represent a significant advance in our study of NN

accepted and system evolution proceeds from this new state. If the switch results in a higher energy, the switch is accepted with the Boltzmann probability:

$$\Pr\{\text{uphill move} = \Delta E\} = \exp(-\Delta E/T),$$

where T is a parameter that represents the current "temperature" of the system. Thus, when the temperature is high, switches that increase the energy are accepted with relatively high probability. As the temperature decreases, uphill moves are accepted with decreasing probability, until the algorithm becomes one of pure gradient descent at very low temperatures. Whenever the temperature is decreased, it must be held constant until thermal equilibrium is reached. Although it has been proven that under certain conditions SA methods of this type eventually converge to the global minimum, the time required to do so is often prohibitive. A binary NN model of this type, known as the Boltzmann Machine, has been proposed by Hinton and Sejnowski [A18]. Faster convergence is claimed for the Cauchy Machine, proposed by Szu and Hartley [A19], under which the system is perturbed by noise with a Cauchy distribution.

Simulated annealing can be used in conjunction with Hopfield NNs to permit the escape from local minima, as demonstrated by Levy and Adams [A20] and Akiyama et al. [A21]. Analog neurons with a sigmoidal input-output function, such as that shown earlier in Fig. A1, are again assumed. Randomness is incorporated into the model by adding Gaussian noise to each of the neuron input voltages. Initially, a relatively large noise variance is applied, corresponding to a high temperature. The noise variance is then decreased slowly, permitting equilibrium to be reached at each temperature, as described above. The slope of the nonlinearity is also varied during the annealing process. During the early stages, when the temperature is high, a relatively flat curve (corresponding to a large value of u_0) is used. As the temperature decreases, the sigmoidal curve is gradually made steeper. The use of a low-gain system in the early stages permits the search to spend more time in the interior of the hypercube, thus, in effect, postponing the final decision on the state of each neuron. In the latter stages it is desirable to have a sharp nonlinearity to ensure that the corners of the hypercube are, in fact, reached. The annealing schedule and sharpening schedule should be coordinated with each other, e.g., use of a high variance noise process with a steep nonlinearity can be expected to cause oscillatory behavior (and thus failure to reach equilibrium). On the other hand, noise of low variance will have very little effect on a system with a relatively flat nonlinearity.

A noiseless Hopfield NN with a nonlinearity that steepens gradually as time progresses can also be considered. Such an approach, which is essentially SA but without the noise, has been called mean field annealing (MFA) [A22]. As in systems with noise, since hard decisions on the neuron voltages do not have to be made at the early stages of the iteration, a better search can be performed in some cases. However, MFA does not permit gradient hill climbing. Thus global minima cannot be guaranteed.

We have applied both MFA and SA techniques to the Hopfield NN models used in the solution of the routing problem. The use of MFA has not been as successful as the use of the method of Lagrange multipliers with a fixed, relatively steep nonlinearity; thus MFA is not a part of any of our final NN models. SA is a critical component of our scheduling NN models, although our routing models generally perform better without it. The use of both MFA and SA in routing and scheduling problems is discussed in this report and in more detail in Refs. A23 and A24.

REFERENCES

- A1. J. J. Hopfield and D. W. Tank, "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics* **52**, 141-152 (1985).
- A2. P. D. Wasserman, *Neural Computing: Theory and Practice*, (Van Nostrand Reinhold, New York, 1989).

- A3. P. K. Simpson, *Artificial Neural Systems*, (Pergamon Press, New York, 1990).
- A4. J. Ramanujam and P. Sadayappan, "Optimization by Neural Networks," *Proceedings of the IEEE International Conference on Neural Networks*, pp. II-325 - II-332, July 1988.
- A5. R. D. Brandt, Y. Wang, A. J. Laub, and S. K. Mitra, "Alternative Networks for Solving the Traveling Salesman Problem and the List-Matching Problem," *Proceedings of the IEEE International Conference on Neural Networks*, pp. II-333 - II-340, July 1988.
- A6. Y. S. Foo and Y. Takefuji, "Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 1. Problem Representation," *Proceedings of the IEEE International Conference on Neural Networks*, pp. II-275 - II-282, July 1988.
- A7. Y. S. Foo and Y. Takefuji, "Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 2. Architecture and Simulations," *Proceedings of the IEEE International Conference on Neural Networks*, pp. II-283 - II-290, July 1988.
- A8. E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, ed., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, (John Wiley & Sons, Chichester, 1985).
- A9. G. W. Wilson and G. S. Pawley, "On the Stability of the Traveling Salesman Problem Algorithm of Hopfield and Tank," *Biological Cybernetics* **58**, 63-70 (1988).
- A10. S. U. Hegde, J. L. Sweet, and W. B. Levy, "Determination of Parameters in a Hopfield/Tank Computational Network," *Proceedings of the IEEE International Conference on Neural Networks*, pp. II-291 - II-298, July 1988.
- A11. A. B. Kahng, "Traveling Salesman Heuristics and Embedding Dimension in the Hopfield Model," *Proceedings of the International Joint Conference on Neural Networks*, pp. I-513 - I-520, 1989.
- A12. S. V. B. Aiyer, M. Niranjan, and F. Fallside, "A Theoretical Investigation into the Performance of the Hopfield Model," *IEEE Trans. Neural Networks* **1**, 204-215 (1990).
- A13. E. Wacholder, J. Han, and R. C. Mann, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem," *Biological Cybernetics* **61**, 11-19 (1989).
- A14. J. C. Platt and A. H. Barr, "Constrained Differential Optimization," *Proceedings of the IEEE 1987 Neural Information Processing Systems Conference*, pp. 612-621, 1987.
- A15. P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, (D. Reidel Publishing Company, Dordrecht, 1987).
- A16. S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* **220**, 671-680 (1983).
- A17. B. Hajek, "A Tutorial Survey of Theory and Applications of Simulated Annealing," *Proceedings of the 24th IEEE Conference on Decision and Control*, pp. 755-760, December 1985.
- A18. G. E. Hinton and T. J. Sejnowski, "Learning and Relearning in Boltzmann Machines," in *Parallel Distributed Processing*, ed. D. E. Rumelhart and J. L. McClelland (MIT Press, Cambridge, MA, 1986) pp. 282-317.

- A19. H. H. Szu and R. L. Hartley, "Nonconvex Optimization by Fast Simulated Annealing," *Proc. IEEE* **75**, 1538-1540 (1987).
- A20. B. C. Levy and M. B. Adams, "Global Optimization with Stochastic Neural Networks," *IEEE International Conference on Neural Networks*, pp. III-681 - III-689, 1987.
- A21. Y. Akiyama, A. Yamashita, M. Kajiura, Y. Anzai, and H. Aiso, "The Gaussian Machine: A Stochastic Neural Network for Solving Assignment Problems," *J. Neural Network Computing* **2**, 43-51 (1991).
- A22. D. E. Van den Bout and T. K. Miller III, "Graph Partitioning Using Annealed Neural Networks," *IEEE Trans. Neural Networks* **1**, 192-203 (1990).
- A23. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "The Application of Hopfield Neural Network Techniques to Problems of Routing and Scheduling in Packet Radio Networks," NRL Memorandum Report 6730, November 1990.
- A24. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Scheduling Link Activation in Multihop Radio Networks by Means of Hopfield Neural Network Techniques," NRL Memorandum Report 6885, September 1991.

models of network problems. In particular, the ability of this model to supply the excitatory connections that are needed to generate complete paths from individual links may be viewed as a first step toward the more general, and more difficult, routing problem in which candidate paths between each SD pair are not specified in advance. In that case, the NN must piece together complete paths from individual links that are not a priori associated with each other, a situation that makes it much more difficult to establish the excitatory connections that are needed to guarantee complete paths. The link-neuron model may also be viewed as a first step toward our ultimate goal of solving the joint routing-scheduling problem, in which the time slot for the activation of each individual link along every path must be determined.

1.2 A Hopfield NN for Link Activation Scheduling

The second problem we address concerns "link activation" or "scheduling" in multihop packet radio networks. We first consider the determination of schedules of minimum length that satisfy the specified end-to-end communication requirements between a number of source-destination (SD) pairs in the network. In problems of this type, the communication requirements are normally specified simply in terms of the number of packets that must be transmitted over each physical link* in the network. We also consider the more-difficult case in which the sequence of link activations along any multihop path in the schedule must be preserved, i.e., the case in which, for each path, the link emanating from the source must be activated first, the next link second, and so on. In Ref. 20 this problem is shown to be NP-complete, and a heuristic for a variation of the problem in wireline (i.e., nonradio) networks is provided. The advantage of sequential link activation is that it reduces end-to-end delay in the network. We also discuss extensions of our model to the joint routing-scheduling problem.

Tassiulas et al. [9] were the first to apply the Hopfield NN methodology to the scheduling problem. In this report, we present improved Hopfield NN formulations, which have been shown to be capable of producing optimal or nearly-optimal link activation schedules in a variety of applications. Many of the considerations of our model are similar to those discussed earlier for the routing problem. For example, we again use the method of Lagrange multipliers to dynamically vary the values of the coefficients used in the connection weights. Other important aspects of our models include the incorporation of heuristics into the equations of motion and the use of Gaussian simulated annealing, both of which encourage the evolution of the NN to optimal solutions. Extensive simulation results demonstrate the effectiveness of our models.

1.3 Outline of the Report

In Section 2 we address the fundamental issues associated with the joint routing-scheduling problem, and we discuss the few attempts that have been made to solve this problem.

In Section 3 we present our basic path-neuron Hopfield NN model for the minimization of congestion. We begin by defining the optimization problem, and we show how an energy function is derived that incorporates the objective function as well as the system constraints. We then show how the corresponding NN connection weights and bias currents are determined from the energy function. We conclude by presenting the resulting equations of motion in an iterative form that is appropriate for simulation in software.

In Section 4 we present simulation results for the path-neuron model. We discuss the many issues that have arisen in the simulation of the equations of motion and the methods we have developed to overcome a variety of problems. In particular, use of the method of Lagrange multipliers, under which the coefficients in the connection weights evolve dynamically with the system state, is shown to provide highly robust operation in large, heavily-congested networks.

*In this report, we refer to the communication channel between two nodes as a *physical link*. A *link* corresponds to a "virtual" link that corresponds to a single unit of traffic that must traverse a given physical link. Thus, if n units of traffic are to be delivered on the physical link that connects nodes i and j , there are n parallel "links" between nodes i and j .