DTIC FILE COPY

Naval Research Laboratory

Washington, DC 20375-5000

NRL Memorandum Report 6730

AD-A229 039

The Application of Hopfield Neural Network Techniques to Problems of Routing and Scheduling in Packet Radio Networks

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

November 9, 1990

DTIC
ELECTE
NOV 21 1990
S B D

Approved for public release; distribution unlimited.

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	3. REPORT TYPE AND DATES COVERED
	1990 November 9	Interim Report 10/89 - 7/90

4. TITLE AND SUBTITLE
THE APPLICATION OF HOPFIELD NEURAL NETWORK TECHNIQUES TO PROBLEMS OF ROUTING AND SCHEDULING IN PACKET RADIO NETWORKS

5. FUNDING NUMBERS
PE: 61153N
PR: RR021-0542
WU: DN480-557

6. AUTHOR(S)
Jeffrey E. Wieselthier
Craig M. Barnhart
Anthony Ephremides

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
Washington, DC 20375-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NRL Memorandum Report 6730

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Arlington, VA 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
A. Ephremides is with the University of Maryland and Locus, Inc.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution 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. In this report, we review the major issues associated with the joint study of these problems, and we address the problem of routing for the minimization of congestion as a first step toward the solution of the joint routing-scheduling problem. We formulate this as a combinatorial optimization problem, and we develop a Hopfield neural network (NN) model for its solution. The issues associated with the development of the Hopfield NN model for this problem are discussed in detail. In particular, the determination of the coefficients in the connection weights is the most critical issue in the design and simulation of Hopfield NN models. In our studies, we use the method of Lagrange multipliers, which permits these coefficients to vary dynamically along with the evolution of the system state. Extensive software simulation results demonstrate the ability of our approach to determine good sets of routes in large, heavily-congested networks.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Communications network, Hopfield network, Routing, Neural network.	120
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z39-18
298-102

CONTENTS

THE APPLICATION OF HOPFIELD NEURAL NETWORK TECHNIQUES TO PROBLEMS OF ROUTING AND SCHEDULING IN PACKET RADIO NETWORKS

1.0 INTRODUCTION

Two of the most important facets of multihop packet radio network design and control are routing and channel access. These network functions are highly interdependent. 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 is 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." Under this channel-access mechanism, the nodes are assigned non-interfering, 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, some versions of it are NP-complete (i.e., cannot be solved by an algorithm of polynomial complexity). Thus heuristics are generally used to produce suboptimal link-activation schedules. A discussion of link-activation methods, with an emphasis on CDMA considerations, is presented in [1].

An alternate approach to the link-activation problem is the use of a Hopfield NN to generate good, although not necessarily optimal, communication schedules [2]. Under this approach, the scheduling problem is transformed into a graph-coloring problem, with the objective being 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

Manuscript approved August 10, 1990.

By
Distribution/
Availability Codes
Avail and/or
Dist Special
A-1

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 as well. Our approach fits well into the currently widening interest of the research community in the application of NN methods to communication and control problems (see e.g., [3] and [4]). We note that a NN approach for a different version of the routing problem is discussed in [5].

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 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 consider separately 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. Toward this end, we have implemented a NN that chooses routes, based on the criterion of minimizing congestion. The routes chosen by such a NN may then be used in conjunction with a NN that schedules packets over these routes. Ultimately, we hope to be able to design a NN for the complete problem.

1.1 A Hopfield NN for the Minimization of Congestion

The 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 the definition of neurons that correspond to binary variables in the system that is being modeled. Most of this report focuses 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, and suggest that hardware implementations (which provide convergence to a final state almost instantaneously) may be worthwhile.

The Hopfield NN methodology is rather different from more-traditional approaches to combinatorial-optimization problems. For example, this approach involves the embedding of 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. Background material on the Hopfield NN model is provided in Appendix A.

We take a rather "evolutionary" approach in the description of our model. In doing so, we explain the major issues that arise in the implementation of Hopfield NNs and the methods that we have developed to overcome a variety of problems. 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, a technique that permits the coefficients to vary dynamically along 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 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 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 extensive simulation results for the path-neuron model. In doing so, 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 along with the system state, is shown to provide highly-robust operation in large, heavily-congested networks.

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.

Finally, in Section 6, we discuss our conclusions from this study, and we indicate what we feel are promising research directions.

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.[1] In all types of communication networks, it

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

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 OSI architecture (see e.g., [6]). 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 sub-layer of layer-2 issue). It is quite clear that 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 that might 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, and 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., while 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, it is possible to mitigate these disadvantages. 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 [1], 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 transmission protocols is that they are conceptually more attractive than the equivalent forms of their frequency- or code-based counterparts and that, although not necessarily better than the contention-based ones in terms of performance (that is 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 which 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 is 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 [7, 8], and it was determined that, in almost all of its forms, it is a combinatorial-optimization problem of high complexity (NP-complete). It has been referred to as the pure scheduling problem. In complexity theory, the term NP-complete describes the property that there is no known algorithm of polynomial complexity that can solve the 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. For more details on the complexity of other forms of the pure scheduling problem, see [9].

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), $j = 1, ..., N$, where T_j is a set of links that can be activated simultaneously without violating interference constraints[2] 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 [8], and further studied by Tassiulas and Ephremides [10]. The solution in [8] is not practical and has only academic value (although it is crucially important to the subsequent development of the joint routing-scheduling problem).

[2] 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.

Thus, in [2], an effort was made to develop a heuristic solution that is based on a Hopfield neural network (NN). The inspiration to do so came from the observation by Hopfield and Tank [11] 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 [2] 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 in [2] 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 want to make note of the fact 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 goal of developing 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 the definition of 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 its interaction with the transmission schedules. This problem was studied in detail by Tassiulas and Ephremides in [10].

It turns out that, first of all, this problem is not as restricted as it may seem in the beginning, since 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. Secondly, it turns out that the joint optimization decomposes into two separate problems, one of schedule optimization and one of flow optimization (i.e., routing) that are weakly coupled. Specifically, it is known that the value of the length of the optimal schedule is 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 [8] and then put to use by Tassiulas and Ephremides [10]. 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 the extension of the result to multiple destinations, although certainly possible, seems to be substantially 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 [8] and Tassiulas and Ephremides [10] for the non-quantized 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, we wish to mention that it has been considered in some of its simpler forms (single tandem topology, immediate neighbor destination, and some others) in [9] and gives rise to two types of questions. The first type relates to stability and simply asks whether there exists a schedule 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 that 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 Ph.D. Dissertation that is underway 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 in [8, 10] 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. There exist a plethora of algorithms and variants of them 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 of it 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,[3] 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 will be seen in the next section, in which the specifics of our 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). It can be seen rather easily that 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 such a NN and applied it to several variations of the model described above. 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 determine system parameters dynamically. We have also investigated the use of simulated annealing and a variety of heuristic methods for performance improvement. Background material on these methods is provided in Appendix A. In Sections 3, 4, and 5 we describe in detail the models we have developed and their performance analysis and evaluation.

---

[3] 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

As noted in the previous section, 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 takes into account the factor of link scheduling effects indirectly. The development of this NN model and its simulation in software is the primary focus of this report. Once these routes are chosen, schedules can be generated either by using NN methods that are applicable to pure scheduling problems as developed in [2], or by means of some (preferably distributed) heuristic such as those developed in [12].

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, and 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 paths connecting each SD pair, select a single path between each SD pair so that network congestion, as defined below, 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.

In the future, we hope to investigate the more difficult routing problem in which paths between the SD pairs are not specified in advance. Here, we comment that the existence of predefined sets of paths is not unreasonable, and is actually similar in principle to the predefinition of virtual circuits that may be activated as needed in response to traffic demands.

3.2 Neural Network Model

The first step in the formulation of a Hopfield NN model is the definition of 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.[1] For example, Figure 3.1(a) shows a very simple six-node network with two paths between each of two SD pairs, and Figure 3.1(b) shows the corresponding path-neuron model. A double index is used to specify the neurons, e.g., 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

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

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 valid 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, a valid solution will have one neuron per SD pair with an output voltage value close to 1, while the others will be close to 0. We remark here that the main advantage of the use of analog neurons is that they permit the embedding of discrete optimization problems in a continuous solution space, which results in the capability of finding better solutions than are generally possible in a discrete solution space, as is discussed in Appendix A.

[1] 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.

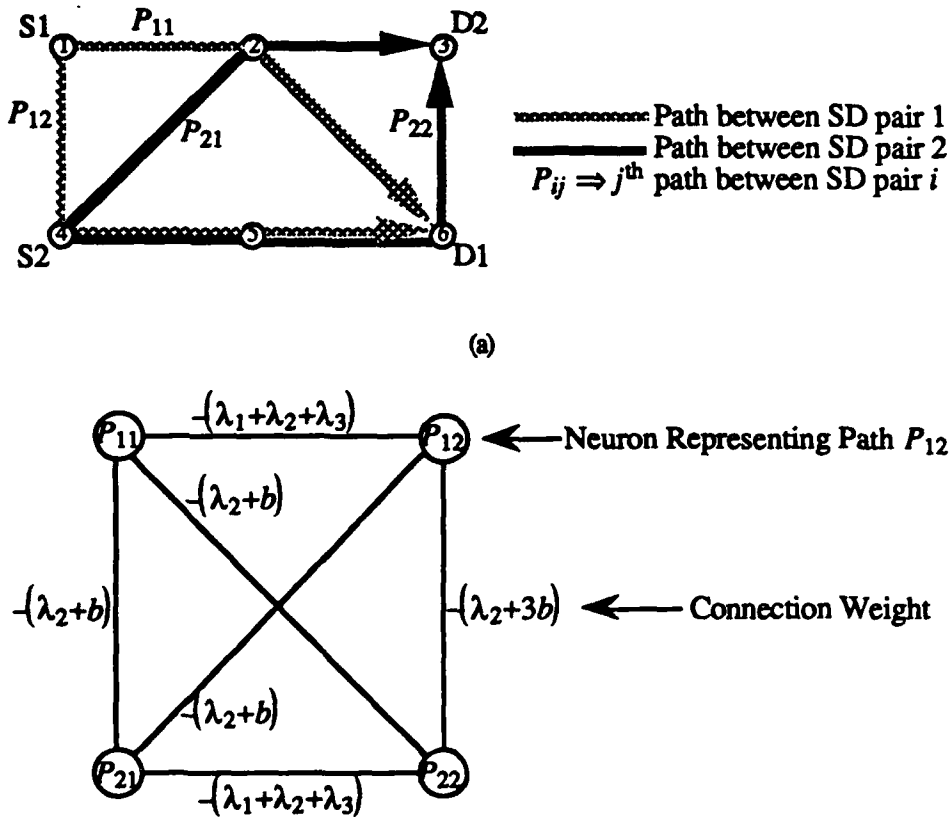


Figure 3.1. An example network, (a) shows a six-node communication network, and (b) shows the corresponding path-neuron NN model

Connections, which may be either inhibitory or excitatory, 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 the following generic form:

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

System evolution follows a trajectory of monotonically decreasing energy. In Eq. (*), $T_{ij,kl}$ is the strength of the connection between neurons ij and kl (it is positive if the connection is excitatory and negative if it is inhibitory), I_{ij} is the bias current applied to neuron ij, and $N_p(i)$ is the number of paths between SD pair i. 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 $T_{ij,kl}$, which specify the strength of the connection between neurons ij and kl. Convergence to a stable state is guaranteed as long as the connections are symmetric (i.e., $T_{ij,kl} = T_{kl,ij}$), and provided that $T_{ij,ij} = 0$ [11]. Although these conditions are not strictly satisfied in our problem formulation (nor, in fact, in Hopfield and Tank's NN formulation of the TSP), reliable convergence has, in fact, been achieved. In our problem, 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). We soon discuss how the connection weights and bias currents are chosen to solve our problem.

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 term 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.

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. In fact, hardware implementation may be feasible for sizes of the problem that exceed by far the ones that can be handled in software.

3.3 Congestion Energy

The class of objective functions that can be modeled by means of Hopfield NNs is normally limited to those that can be expressed in the form of Eq. (*). This class includes weighted sums of the products of pairs of neuron output voltages as well as output voltages taken individually. The nongeneral nature of the Lyapunov energy function [11] often necessitates the

use of creativity in determining a meaningful performance measure that can be modeled by the NN and that reflects the desired behavior of the system that is to be optimized.

We noted in the previous section that, for the case of continuous traffic, the selection of paths that minimize the maximum nodal degree[2] 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. (*), as required:

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

where

P_{ij} = the j^{th} path between SD pair i.

$\left| P_{ij} \cap P_{kl} \right|$ = 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 a legal state has been reached. In this case, the congestion energy corresponds to the sum of the number of common nodes of all selected paths (one for each SD pair), taken on a pairwise basis. Minimization of 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 a legal 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 a legal 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 strength is proportional to the number of shared nodes in the two paths.

[2] Recall that the degree of a node is defined to be the sum of all flows into the node plus all flows out of the node.

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 following form:

$$E_{total} = bE_b + \sum_{i=1}^{3} \lambda_i E_i - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} V_{ij} . \qquad (**)$$

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 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. In our initial discussions, the values of the λ_i's are assumed to be constants, whose best values are typically determined by trial-and-error in Hopfield NN simulations. In Section 4.5 we exploit the full power of this method by permitting the Lagrange multipliers to evolve along with the system state. The last term of Eq. (**) represents the impact of additional bias currents, which in our problem formulation are applied equally to all neurons to help with the satisfaction of system constraints. The coefficients b, λ_i, 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. (*). This problem formulation is quite similar to the one developed by Hopfield and Tank for the Traveling Salesman Problem (TSP), which is discussed in Appendix A. Note that although E_{total} follows a trajectory of monotonically decreasing energy, E_b does not necessarily do so.

Later, in the presentation of our simulation results, we discuss another performance measure for network congestion, which may appear to provide a more reasonable measure of system performance than the E_b function defined above, but which cannot be put into the form of the Lyapunov energy function. We show that the NN designed to minimize E_b, also provides nearly optimal values of the other performance measure 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 below. The effect of these constraints is perhaps best understood if the neurons are initially assumed to take on only binary

values. After examining the constraints from this viewpoint, we address their impact on a system with analog neurons.

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 would appear to be redundant (because satisfaction of the first two constraints would guarantee that it is satisfied as well), its inclusion in the energy equation is helpful in achieving convergence to valid solutions. The use of such seemingly redundant constraints is common in Hopfield network models. Satisfaction of 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. 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 states," which are binary states in which the constraints are, in fact, satisfied. Whether or not convergence to legal states is achieved, depends on factors such as the λ_i 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 will be discussed later in this section and, in more detail, in Section 4.

3.5 Determination of Connection Weights and Bias Currents

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

$$E_{total} = \frac{b}{2}\sum_{i=1}^{N_{sd}}\sum_{\substack{k=1\\k\neq i}}^{N_{sd}}\sum_{j=1}^{N_p(i)}\sum_{l=1}^{N_p(k)} \left|P_{ij}\cap P_{kl}\right| V_{ij}V_{kl} + \frac{\lambda_1}{2}\sum_{i=1}^{N_{sd}}\sum_{j=1}^{N_p(i)}\sum_{\substack{l=1\\l\neq j}}^{N_p(i)} V_{ij}V_{il}$$

$$+ \frac{\lambda_2}{2}\left(\sum_{i=1}^{N_{sd}}\sum_{j=1}^{N_p(i)} V_{ij} - N_{sd}\right)^2 + \frac{\lambda_3}{2}\sum_{i=1}^{N_{sd}}\left(\sum_{j=1}^{N_p(i)} V_{ij} - 1\right)^2. \quad (***)$$

To determine the connection weights, we compare Eq. (***) with the generic form given in Eq. (*). 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. (***):

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

where δ_{ik} 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}.

We have observed in our simulation studies that an insufficient number of neurons are typically activated, a problem that can be mitigated by increasing the bias currents. Hopfield and Tank [11] observed the same behavior in their studies of the TSP, as is discussed in Appendix A.

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 may now be expressed as follows:

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

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 [11] in their solution of the TSP. Incorporation of the additional bias currents in this manner permits us to set $I = 0$ in Eq. (**). The resultant expression for bias currents is:

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

3.6 Equations of Motion

The evolution of the input voltage at each neuron is characterized by an equation of motion that is obtained by differentiating the energy function with respect to the output voltage at that neuron. Thus

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

where $\tau = RC$ is the time constant of the RC circuit connected to the neuron. As the system evolves from an initial state, 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 equations of motion may be expressed in iterative form as follows:

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

$$- \Delta t \lambda_2 \left(\sum_{k=1}^{N_{col}} \sum_{l=1}^{N_p(i)} V_{kl} - \alpha N_{sd} \right) - \Delta t \lambda_3 \left(\sum_{l=1}^{N_p(i)} V_{il} - 1 \right) + \Delta t I .$$

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_i's. Also important are the bias current parameters α and I. 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 a legal solution, depends strongly on these parameters. A complete discussion of these and other issues that have arisen in the simulation process is presented in Section 4. At this point, we remark that it is difficult to develop "cookbook" procedures for the choice of specific fixed values of these parameters that will produce high-quality performance for a wide variety of networks. However, we note that we have obtained excellent and highly robust results by using the method of Lagrange multipliers, a technique that permits the connection weights to vary dynamically along with the evolution of the system state.

4.0 PERFORMANCE EVALUATION USING THE PATH-NEURON MODEL

Before describing our simulation results, we remark that additional background and explanations are required to explain our approach fully, and will be provided as needed. Unlike typical well-defined algorithms that are evaluated by simulation, NN-based algorithms are, to a large measure, still in a stage of development. For example, we noted at the end of the previous section that the choice of system parameters is critical to the ability of the NN simulation to converge to good, or even valid, solutions, and that it is difficult to develop cookbook procedures that guarantee convergence to good solutions.

In this section we take somewhat of an "evolutionary" approach that outlines the major steps in our development of what has turned out to be a rather effective and robust method for obtaining solutions to our problem. We start with a discussion of the often-used trial-and-error method of choosing system parameters, and illustrate how this approach can be improved through the use of mean-field-annealing techniques. This sets the stage for our discussion of the method of Lagrange multipliers, which permits the coefficients in the connection weight matrix to be controlled dynamically as the system state evolves. In the course of this development, issues related to convergence and performance evaluation are addressed, and we attempt to share the insight we have gained into the use of Hopfield NNs for optimization problems.

We discuss the application of our method to several networks. First, a rather modest-sized network with 24 nodes (including 10 SD pairs and a total of 52 paths) is considered. After demonstrating the effectiveness of our approach on this network, we discuss a 100-node network

with 40 SD pairs and a total of 327 paths. This example was augmented further by imposing a traffic requirement of three units between each SD pair, which necessitated the definition of three neurons per path, resulting in a total of 981 in the network. In all of these examples, convergence to good solutions is obtained 100% of the time. Therefore, we are confident that this method can be extended to even larger and more complex problems.

4.1 Basic Simulation Issues

The NN model described in Section 3 was simulated 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 path-neuron NN model, i.e., the neurons are defined and indexed such that neuron ij represents the j^{th} path between the i^{th} SD pair, the weight matrix **T** is created, and the bias currents are defined. 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 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 a totally symmetric initial state, which is an undesirable condition, as is discussed in Appendix A. The perturbation 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.[1] 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

[1] Later we discuss the use of simulated annealing in conjunction with our NN model, 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.

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 valid (a legal solution) if the constraints are satisfied, i.e., if exactly one path is activated for every SD pair. Examples of invalid solutions are (1) the activation of two or more paths between a SD pair, or (2) the failure to activate any paths between a SD pair.

Monte-Carlo type simulations were performed. For a particular set of parameter values, the NN was run from 100 different initial states (random number generator seeds), and the results were evaluated in terms of valid convergence percentage (number of valid convergences / number of attempts) and the quality of convergence based on congestion energy E_b or some other related metric of the valid solutions.

Throughout this report, each set of paths between every SD pair has been chosen to be maximally node-disjoint. A set of n maximally node-disjoint paths between a SD pair consists of the n paths between the SD pair that have the fewest nodes in common. These paths have been obtained from the network connectivity matrix by means of Dijkstra's algorithm [13], which is described as follows.

Given an N-node network connectivity matrix and a SD pair, Dijkstra's algorithm attempts to find a set of n maximally node-disjoint paths by iteratively finding n lowest-cost paths. The cost of a path is defined as the sum of the weights of the nodes traversed in the path. Initially, all nodes are assigned a weight of 1, with the exceptions of the source and the destination nodes which are assigned a weight of zero. Then a breadth-first search (see [14]) of the network is used to discover and list a lowest-cost path between the SD pair. As a result of the equal initial weights of all the intermediate nodes (nodes that are neither the source nor the destination), the first listed path is also a shortest (hop) path. The weights of all the intermediate nodes in the listed path are then increased by N, thus completing the first iteration of the algorithm. The iteration (which consists of the breadth-first search, listing of the lowest-cost path, and augmentation of the weights of the intermediate nodes of that path) is repeated n times. At each iteration, the weighting scheme causes the lowest-cost path listed to share the minimum possible number of nodes with the previously listed paths. However, it does not preclude duplicate listings. Any duplicate paths and paths that contain previously listed paths are discarded.[2] This is the reason for the nonuniform number of paths between different SD pairs in the examples presented in this report.

[2] Table B.1 in Appendix B does include one example of a path, path 7, that contains a previously listed path, path 6. The redundant path 7 has never been selected in any of the simulations with this network.

Before we present our numerical results, it is useful to describe and comment on the various initiatives that we took to correct some of the shortcomings of the NN behavior as it was observed in the initial stages of the performance evaluation.

4.1.1 Obtaining Valid Convergences

Preliminary experimentation with the path-neuron model using constant coefficients for the connection weights and small communication networks revealed that obtaining valid convergence is a nontrivial task. There was actually little problem in obtaining convergence. The difficulty was in obtaining convergence to valid states. Examples of convergence to nonvalid states included both those with no path selected for a SD pair and those with two or more paths selected for a SD pair. A great deal of parameter adjustments led finally to valid convergence approximately 50 to 70% of the time. The resultant parameter sets were problem specific; changing the topology required finding a new set of parameters. Although the early results were not particularly good, a great deal of insight into the NN behavior was gained. It was noted that very rapid convergence (< 10 iterations) indicated that the time constant Δt was too large. This resulted in overly large changes in system state from one iteration to the next, which permitted minima to be missed. Time constants on the order of 10^{-4} worked well for a broad range of networks. A bias-current parameter of α = 1.5 (with $I = 0$) generally provided acceptable performance.

4.1.2 Mean Field Annealing

In an effort to increase the percentage of runs that converge to valid solutions, a form of mean field annealing (MFA), similar to the method described in [15], was examined. It involved gradually steepening the slope of the nonlinear neural input-output relationship by gradually decreasing u_o. Recall that the input-output relationship is $V_{ij} = 0.5(1+\tanh(u_{ij}/u_o))$. The nonlinearity parameter u_o is changed as a function of time: $u_o = 10/(c+t/\tau_u)$ while $u_o > u_o'$. The parameter c sets the value of u_o at time $t = 0$, τ_u controls the rate at which the nonlinearity is steepened, and u_o' is the minimum allowed value of u_o (which results in the maximum slope of the nonlinearity). Once the value of u_o reaches u_o', it remains fixed at that value for the remainder of the iteration. Typical values we have used for c, τ_u, and u_o' are 10, 1, and 0.1 respectively. The initial use of a smaller slope in effect permits one to make preliminary vague decisions on the states of the neurons, thus postponing the final decision until a more thorough search of the interior of the hypercube 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 a valid low-energy (although not generally globally optimal) solution.

We remark that our method of MFA differs slightly from that of Van den Bout and Miller [15]. In their studies, u_o was changed in larger steps and held constant at each value until equilibrium was reached.

The introduction of MFA improved the percentage of convergence to valid solutions. Although the parameter sets still had to be found by trial and error, a good set of parameters in conjunction with MFA typically gave valid convergence percentages of 80 to 100%. The successful use of MFA motivated examination of larger, more interesting communication networks.

4.1.3 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.[3] Thus, at any time in the NN evolution, a valid 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 a continuum within the hypercube, however. This mapping from an analog state to a binary one at each step of the iteration is simply for the purpose of assigning a binary interpretation to the state before convergence has been reached.) Ties (i.e., equal output voltages) can be broken arbitrarily, e.g., by choosing the lowest numbered neuron in such a set.

On Tracking System Performance

We have found it advantageous to use the above binary interpretation of the analog state to define a "hypothetical 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 hypothetical values of E_b are necessarily monotonically decreasing.

---

[3] This criterion can be applied only 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.

An Early Termination Criterion

Observation of the evolution of these instantaneously valid solutions revealed 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 4.5) 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 approaches 1 very closely. This early termination criterion is used in all except our early simulation examples.

4.2 Alternative Metrics

A metric is needed to measure the quality of solutions, and thus to allow comparison of different simulations. Given valid solutions, the best solution is the one that provides the lowest level of congestion. As mentioned earlier, the NN formulation is based on a congestion measure, E_b, which measures the NN's performance based on the sum of the pairwise path congestion. Since the connection weights and bias currents are determined by the expression for E_b (and the constraint terms), the quantity E_b is precisely what the NN is programmed to minimize. As just discussed, the hypothetical congestion energy can be determined at each step of the iteration by making a binary decision on which neuron is chosen for each SD path.

Alternative metrics may also be considered. In fact, in Section 4.9 we introduce E_b', an alternative measure of congestion that permits the direct comparison of results obtained using the path-neuron model with those obtained using the link-neuron model of Section 5.

Unfortunately, certain types of desirable metrics cannot be implemented directly using Hopfield NNs. For example, we noted in the previous section that under certain conditions, under 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-based performance measure cannot be directly mapped to the individual neurons in the network because each neuron corresponds to a complete path. However, we do evaluate this quantity in some of our simulation results.

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

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

where

N = the number of nodes

$\Omega(i)$ = max {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 minimization of one does not necessarily minimize the other. However, there appears to be a nearly monotonic relationship between them.

4.3 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 4.4), which has approximately four million different valid solutions, but clearly not practical for the 100-node network (discussed in Section 4.7), which has approximately 5×10^{35} different solutions.

4.3.1 The 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 report, 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.

4.4 Initial Simulation Results for a 24-Node Network

The first network of significant size that we studied in detail was the 24-node network shown in Figure 4.1. A set of ten SD pairs that required the use of multihop paths was selected from this network. The SD pairs are enumerated in Table 1. The paths for each SD pair were generated via Dijkstra's algorithm for finding maximally node-disjoint paths; the paths connecting SD pair 7 are shown in Figure 4.1, and a complete listing of the 52 paths between the SD pairs is given in Appendix B. Approximately four million different valid solutions are possible for this example. An exhaustive search of these solutions found that there are 48 different optimum solutions, which all have congestion energy $E_b = 45$. 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 4.6) 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 using the shortest-path heuristic. Our NN simulation studies are discussed in the following subsections.

Simulation 1: Mean Field Annealing and Constant Connection Weights

Our first simulations of the 24-node network included the use of MFA and constant values of the connection weights. After a period of trial-and-error evaluation of different parameter sets, it was found that the following parameter values yield the highest percentage of valid convergence:

λ_1	λ_2	λ_3	b	α	I	c	τ_u	u_o'	Δt	ε
550	500	200	95.5	1.5	0	1	1	0.1	10^{-4}	0.4

The generous value of ε was needed in order to obtain valid convergences. With this value of ε, convergence was usually obtained at about the same time that the nonlinearity reached its maximum steepness, i.e., at 110 iterations. When a more stringent convergence criterion was used ($\varepsilon \leq 0.3$), the NN failed to converge within the span of 15,000 iterations.

In simulations from 100 different initial states, 100 convergences were obtained, 61 of which were valid. The invalid solutions resulted from the failure of the NN to activate a path for one of the SD pairs. Of the valid solutions, the best one had a congestion-energy value of $E_b = 45$;

this value was shown, in fact, to be optimum by means of an exhaustive search of the solution space. Four other solutions had $E_b = 47$, the same congestion measure as the best solution found with the shortest-path heuristic. The remaining 56 solutions had congestion energy ranging from 48 to 59.

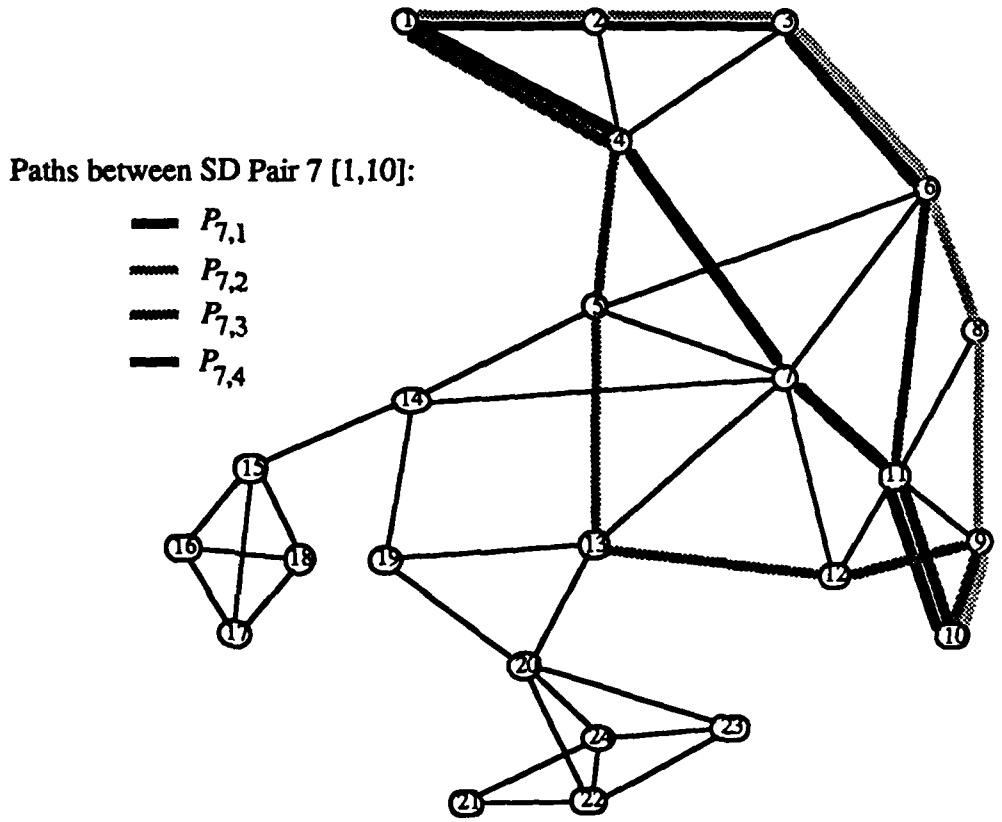


Figure 4.1. A 24-node communication network

Table 1. A listing of SD pairs associated with the network of Figure 4.1

SD pair 1: [4,24]
SD pair 2: [7,17]
SD pair 3: [9,16]
SD pair 4: [1,19]
SD pair 5: [5,11]
SD pair 6: [21,6]
SD pair 7: [1,10]
SD pair 8: [3,18]
SD pair 9: [2,12]
SD pair 10: [14,8]

The results of these simulations are graphically presented in Figure 4.2, where they are compared with the solutions obtained by the shortest-path heuristic. In this figure, the y axis gives

the probability of obtaining a solution with the congestion-energy value given by the x axis. Note that the histogram of the results of Simulation 1 includes the results only of the 61 valid solutions.

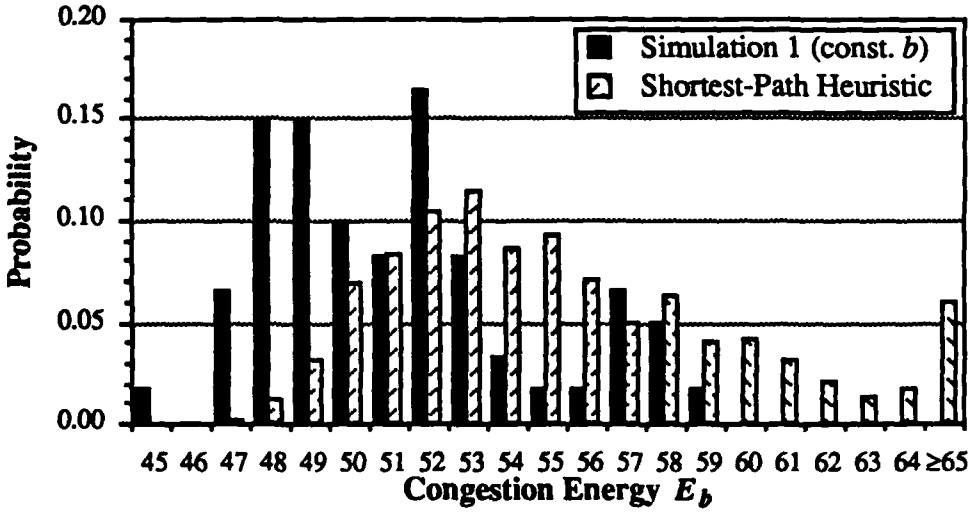


Figure 4.2. Histograms of the valid MFA results and shortest-path heuristic results

Simulation 2: MFA and Linear Ramping of the Congestion-Energy Coefficient

In observing the evolution in time of the neurons' output voltage, it was noticed that there was an initial global trend toward zero. This trend was apparently caused by the initial thrust of the congestion-limiting b term. The initial output voltages of all neurons are approximately equal; thus all neurons are partially active. This gives the appearance of a large amount of congestion, and initially lends overly large emphasis to the b term, which drives the NN to a state from which it is difficult to obtain valid convergence. In order to minimize this undue emphasis, it was found that ramping the congestion-limiting parameter b linearly from 0 to its final value in approximately 100 iterations increased the percentage of runs that converged to valid solutions without significantly sacrificing the congestion-limiting performance of the NN. Congestion can be trivially, albeit illegally, minimized by not activating any paths. Ramping b reduces the tendency toward the trivial minimization and allows the constraint terms to select a preliminary set of paths that form a valid solution.

A Monte-Carlo simulation of the 24-node network, from the same set of 100 different initial states as used in Simulation 1, was run using the parameter values of Simulation 1 in conjunction with MFA and ramping b linearly in 110 iterations to its final value of 95.5. This run yielded 100% convergence and 80% valid convergence. Most convergences occurred in about 110 iterations. Again, invalid solutions resulted from the failure of the NN to activate a path for one

SD pair. Using congestion energy E_b as the metric, one optimum solution was found and a total of three solutions were found that were as good as or better than the best shortest-path heuristic solution.

The valid solutions obtained from Simulations 1 and 2 are compared in Figure 4.3. This figure shows that, on the basis of the congestion energy of valid solutions, better quality results were actually obtained using a constant value of b. However, an examination of the cumulative mass functions of the two simulations, shown in Figure 4.4, permits the evaluation of our results from a different perspective. Although both simulations have about the same likelihood of finding an optimum solution, ramping b linearly gives a much higher frequency of valid solutions. Also shown in Figure 4.4 is the cumulative mass function resulting from exhaustive search. Its inclusion shows that (when valid solutions are found) both simulations produce results much better than those obtained by randomly selecting a valid path set. A total of 61% of the valid solutions obtained through the NN simulations are among the best 1.4% of all legal states, and all of the valid solutions are among the best 28% of all legal states.

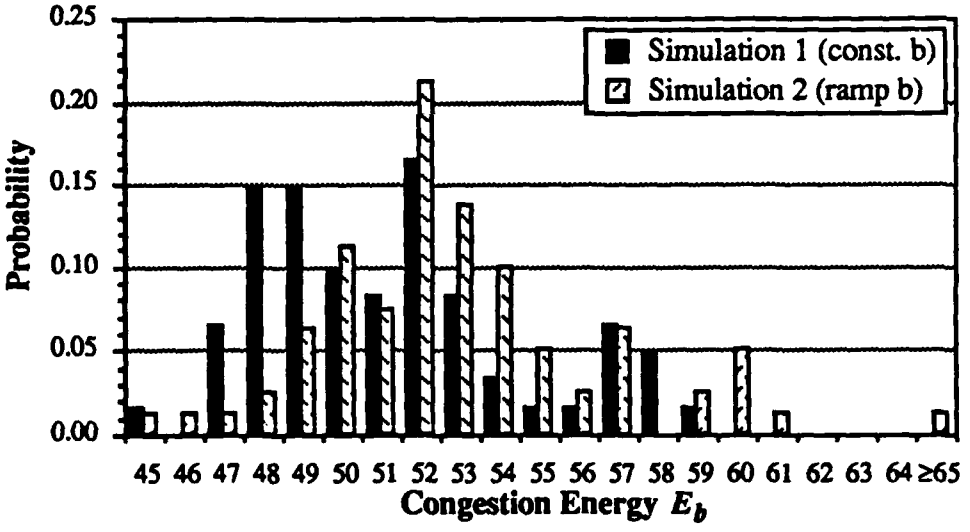


Figure 4.3. Histograms of the valid solutions from Simulations 1 and 2

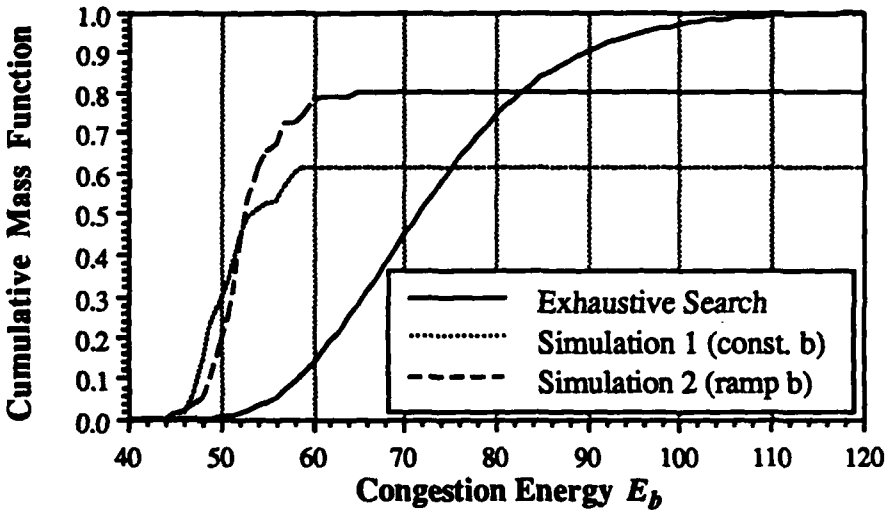


Figure 4.4. Graph of the CMF of Simulations 1 and 2 and the results of exhaustive search

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

All the constraints in this problem are equality constraints. Therefore, as suggested by Wacholder et al. [16], the energy expression corresponding to each equality constraint may be used to update the corresponding connection coefficients dynamically by the method of Lagrange multipliers (LM), as discussed in Appendix A. With this method, each of the constraint connection weights λ_i 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, $\lambda_i(n+1) = \lambda_i(n) + (\Delta t)_\lambda E_i(n)$, where $(\Delta t)_\lambda$ is the LM time constant, which may or may not be different from the global NN time constant Δt. Note that, since $E_i \geq 0$, the quantities λ_i are monotonically increasing. Typically, the Lagrange multipliers are assigned initial values of 1.

Since the use of LM requires equality constraints, it is no longer appropriate to include the additional bias current term in the second equality constraint.[4] Thus we set $\alpha = 1$ and include an additional constant bias current term I, which is positive, in each of the equations of motion.

[4] 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 result in time-varying bias currents.

4.5.1 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 only those SD pairs that were having trouble in turning on exactly one neuron. We call this the method of "multiple Lagrange multipliers" (MLM).

The third constraint formulation

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

may 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 follows

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

where $(\Delta t)_\lambda$ is the Lagrange multiplier time constant. Then the total energy is given by

$$E_{total} = bE_b + \sum_{i=1}^{2}\lambda_i E_i + \sum_{i=1}^{N_{sd}}\lambda_{3i}e_{3i} - I\sum_{i=1}^{N_{sd}}\sum_{j=1}^{N_p(i)} V_{ij},$$

and the iterative equations of motion are

$$u_{ij}(t+\Delta t) = u_{ij}(t) - \Delta t u_{ij}(t) - \Delta t b \sum_{\substack{k=1 \\ k\neq i}}^{N_{nl}} \sum_{l=1}^{N_p(i)} |P_{ij} \cap P_{kl}| V_{kl} - \Delta t \lambda_1 \sum_{\substack{l=1 \\ l\neq j}}^{N_p(i)} V_{il}$$

$$- \Delta t \lambda_2 \left(\sum_{k=1}^{N_{nl}} \sum_{l=1}^{N_p(i)} V_{kl} - N_{sd} \right) - \Delta t \lambda_{3i} \left(\sum_{l=1}^{N_p(i)} V_{il} - 1 \right) + \Delta t I \, .$$

In preliminary work with the LM formulation, it was found that setting $(\Delta t)_\lambda = \Delta t$ did not allow the LMs to grow rapidly enough; the optimization term tends to drive all neurons toward zero output voltage, and the slow growth of the LMs was inadequate to compensate. By establishing a separate and faster time constant for the LMs, the constraint coefficients evolve to meet the optimization term, and the NN relaxes to a state that simultaneously satisfies the constraints and minimizes the congestion energy.

Unless otherwise noted, we have used initial LM values of unity. Limited experimentation with non-unity initial values indicates that the NN is relatively insensitive to the initial value. An abnormally large or small initial LM will be compensated for in the evolution of the entire set of LM values.

The use of Lagrange multipliers allows the connection coefficients to evolve with the NN, and thereby adapt to topological peculiarities and to the remaining constant parameters. It greatly reduces the number of parameter values that must be found by trial and error, thereby making the NN less problem specific. Furthermore, our simulation results demonstrate that the quality of the solutions determined by using this method are typically superior to those obtained where the coefficients are obtained by trial-and-error approaches.

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

Oscillation

In preliminary experimentation with the use of LM, it was found that the speed of convergence is greatly reduced. This was anticipated since the constraint coefficients must be allowed the time to grow, and thereby force valid convergence. Efforts to speed convergence, such as increasing both the NN time constant and the LM time constant, met with some success. However, very large time constants are undesirable because they can sometimes produce a resonance effect in the NN, which results in wild oscillations of neuron output voltages and

consequently of the energy terms. The energy oscillations drive the corresponding LMs to excessively large values, which eventually dampen the oscillation but typically result in poor solution quality. Figure 4.5 shows an example of the oscillation of E_2, the constraint-energy component that corresponds to the activation of exactly N_{sd} paths in the network, when the 24-node network with 10 SD pairs is overdriven.[5] Figure 4.6 shows the corresponding growth of λ_2.

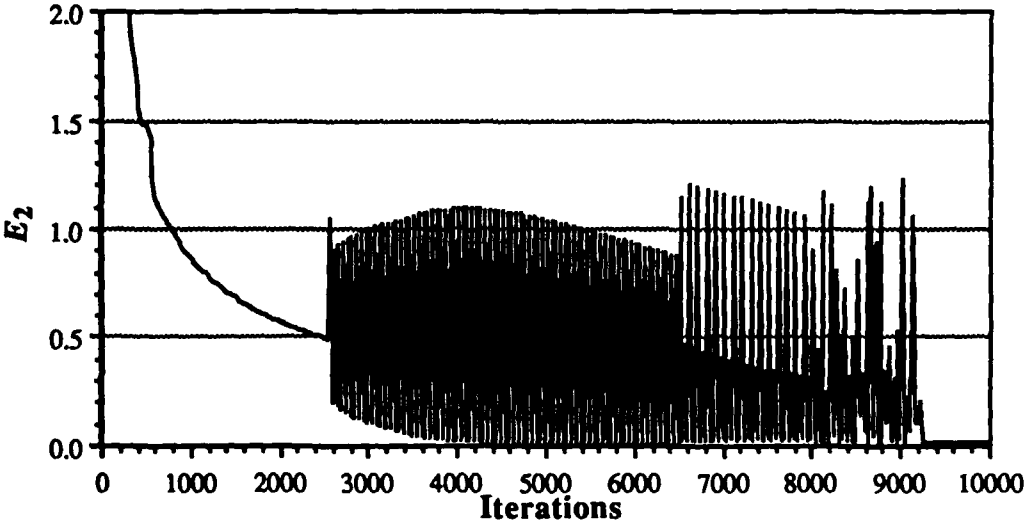


Figure 4.5. An example of constraint energy oscillations

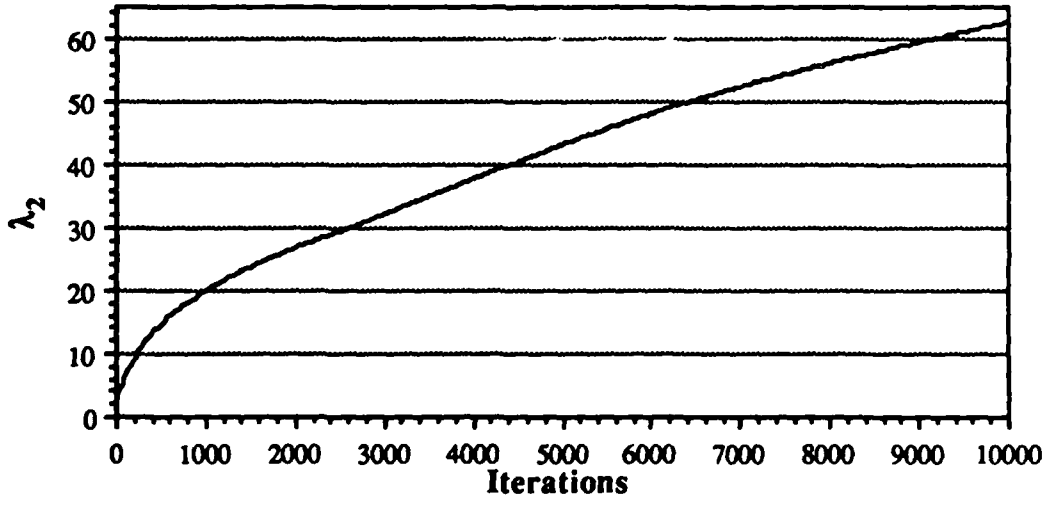


Figure 4.6. Evolution of λ_2 corresponding to the energy of Figure 4.5

[5] The parameter values used in this simulation were: initial $\lambda_i = 1.0$, $b = 5.0$, $I = 21.0$, $\Delta t = 0.005$, $(\Delta t)_\lambda = 0.01$.

To explain this oscillatory behavior, we recall that the connections associated with the E_2 energy term may be either excitatory or inhibitory, depending on the average neuron output voltage in the NN. If the average voltage is too low (corresponding to a condition in which an insufficient number of neurons are active), the connection becomes excitatory. If the average voltage is too high, the connection becomes inhibitory. The change in the input voltage at each iteration is proportional to the time constant. Therefore, if an overly-large time constant is used, large swings in the input voltages may occur when constraint 2 is not satisfied, which result in oscillations of the output voltage.

The best method found to prevent the oscillatory state is to keep the time constants small and allow more iterations to obtain convergence. Alternatively, larger time constants may be used, and any subsequent oscillations may be dampened by incrementally decreasing the time constants. This approach allows rapid convergence (if it is possible for the given system parameters and initial state) and dynamically determines good time-constant values when oscillations are detected. This "oscillation damper" has been used in all the remaining NN simulations, although in most of the simulations its presence was not required.

Simulation 3: LM and the 24-Node Network

The 24-node network (Figure 4.1) with 10 SD pairs (Table 1) was examined using LM and the following parameters ($\lambda_i(0)$ is used to denote the initial Lagrange multiplier value):

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	b	α	I	Δt	$(\Delta t)_\lambda$	u_0	ε
1.0	1.0	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 valid convergence. 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, which was described in Section 4.1.3, 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. 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.

Simulations from 100 different initial states (again, the same initial states as in Simulations 1 and 2) resulted in 100% valid convergence. All of the solutions were better than the best one

found 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 4.4 that there are actually 48 different optimal solutions out of the approximately 4 million total valid solutions for this problem.

The evolution of the constraint energies and the Lagrange multipliers from one of the simulations is shown in Figures 4.7 and 4.8.[6] In Figure 4.7, 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, and the rate of decrease in the values of E_2 and E_3 slows to an asymptotic approach toward zero. Figure 4.8 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.

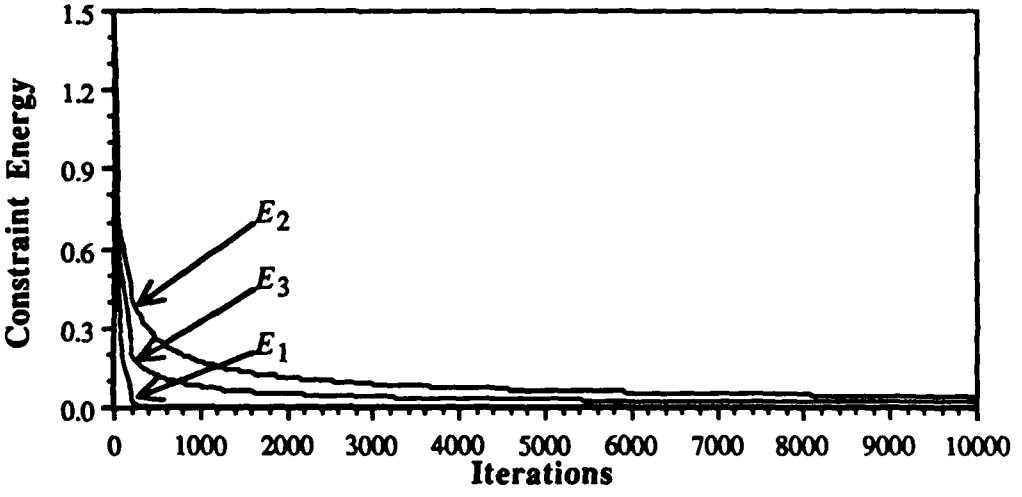


Figure 4.7. Constraint energy evolution

[6] 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.

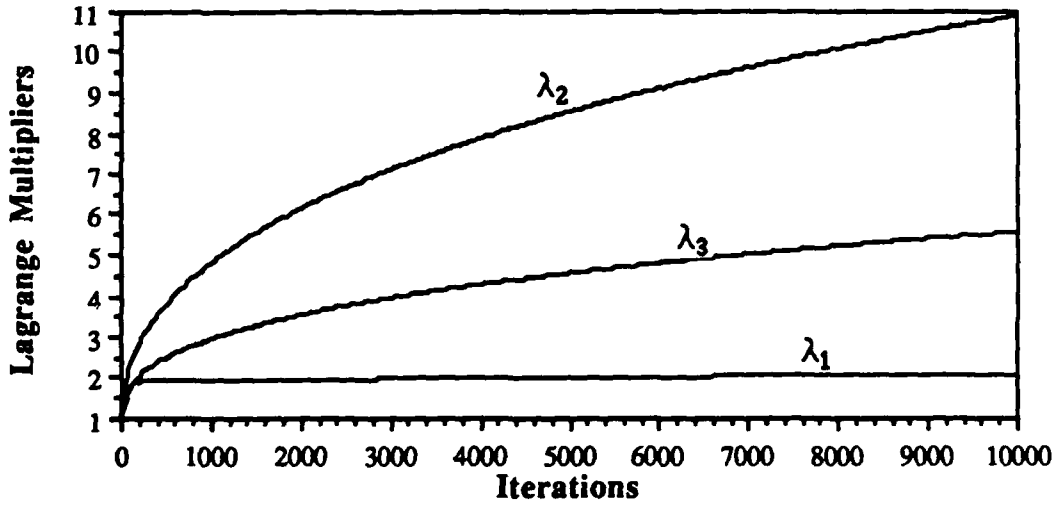
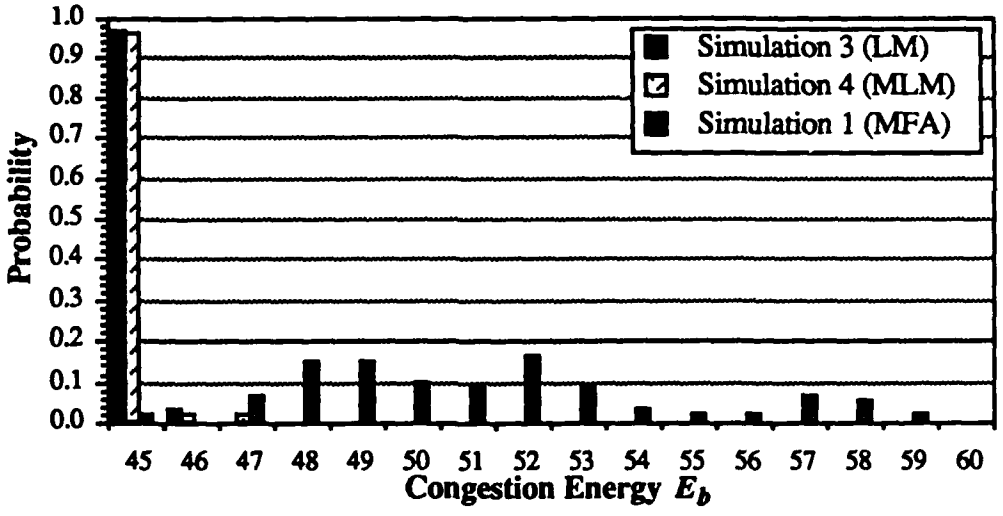


Figure 4.8. Lagrange multiplier evolution

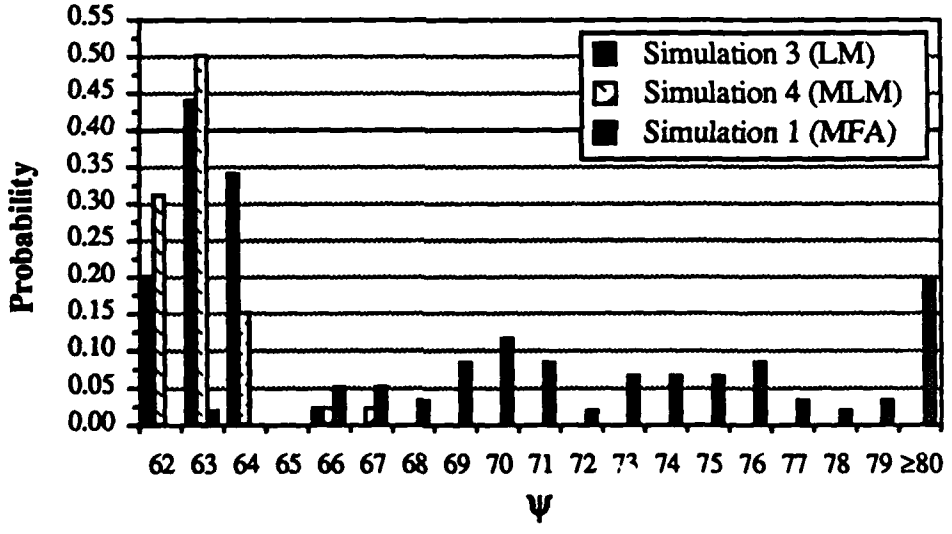
Simulation 4: MLM and the 24-Node Network

Simulation 3 was repeated using MLM with the initial value of each of the λ_{3i} set to 1.0. Figure 4.9 presents the results of this simulation compared with Simulation 3 and the valid solutions of Simulation 1. In Figure 4.9(a) the comparison is on the basis of congestion energy. The figure shows that the use of MLM produces slightly worse results than those obtained in Simulation 3. Simulation 3 found a lowest congestion-energy state 97 times in the set of simulations from 100 random seeds, and the remaining states had congestion energy one unit higher. Simulation 4 with MLM found a lowest energy state ($E_b = 45$) 96 times, two solutions were one unit higher than the optimum value, and the remaining two solutions had $E_b = 47$. The most important conclusion from these results is that the use of LM or MLM, besides giving 100% valid convergences, yields significantly higher quality solutions than those obtained with MFA in conjunction with constant constraint coefficients.

Figure 4.9(b) compares the quality of the same results measured with the ψ metric discussed in Section 4.2. 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 in Simulation 1.



(a)



(b)

Figure 4.9. Histograms of the results of Simulations 3 and 4 and the valid solutions of Simulation 1 in terms of (a) congestion energy, and (b) the ψ metric (optimum ψ = 62)

Simulation 5: LM and a Random Network

In an effort to verify that the NN model using the method of Lagrange multipliers is effective in solving the congestion-minimization problem for a broad class of topologies, a series of random networks were generated and analyzed. Five 24-node random networks and one 100-

node random network were created. The results of NN simulations of the five random networks of the same general scale as that of Figure 4.1, i.e., 24 nodes and mean nodal degree of about 3.8, were compared with the best solutions obtained by means of the shortest-path heuristic. With minor adjustments of the bias currents and the time constants, the NN found better solutions than the shortest-path heuristic for each of the networks. A detailed discussion of the simulation results for one of these 24-node random networks follows. The 100-node network is discussed in Section 4.7.

Thirteen SD pairs were arbitrarily selected in one of the randomly generated 24-node networks. A total of 132 paths were listed when the sets of maximally node disjoint paths connecting the SD pairs were found. The resultant network has 8.8×10^{12} valid solutions, making exhaustive search prohibitive. The best solution found by the shortest-path heuristic, both in terms of congestion energy and ψ, had $E_b = 49$ and $\psi = 67$.

Little parameter optimization was done for this network other than increasing the additional bias current (I) to 10 (to overcome a trend toward under-activation) and decreasing the time constants ($\Delta t = 9 \times 10^{-4}$, $(\Delta t)_\lambda = 10^{-3}$) to allow a more detailed search of the more-complex energy landscape. Otherwise, the parameter values were the same as those used in Simulation 3.

Simulations from 100 different initial states found 100 solutions with $E_b = 47$ and $\psi = 62$. The best E_b and ψ values are not known, but in three of these runs interim NN solutions had congestion values as low as $\psi = 59$ and $E_b = 45$. The interim solutions are the different instantaneous states observed as the NN evolves toward its final solution. We have observed that the occurrence of interim solutions with better quality measures than the final solution is a relatively infrequent phenomenon. However, this event is not unexpected since, although E_{total} is monotonically decreasing, E_b is not necessarily monotonic. In this set of 100 simulations it occurred three times. Each time the high-quality instantaneous state was discovered, it was within the first 25 iterations when the neural state was still highly unstable.

We observe that the ability of our simulations to track the instantaneous state actually permits the selection of the best interim state, rather than the final state, of the system as the solution for that particular simulation run. This property is based on the fact that all instantaneous system states are valid solutions to the problem, as discussed in Section 4.1.3.

We conclude from these studies that our NN model is quite robust. With little or no parameter optimization, it can choose paths with low levels of network congestion in a variety of networks of similar proportions. In the next section, the results of simulations using a much larger

and more complex network further reinforce our confidence that this paradigm is applicable to a broad class of topologies.

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

Simulation 6: LM and a 100-Node Network

The 100-node network shown in Figure 4.10 was used to evaluate the NN's ability to handle larger networks. The network, which was randomly generated, has a mean nodal degree of 4.5 and a diameter of 9. A set of 40 SD pairs was arbitrarily selected, 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 valid solutions (see Table B.2 in Appendix B for a listing of the SD pairs). 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 best solution found by the random search had a congestion energy of $E_b = 567$. The best solution found by the shortest-path heuristic had $E_b = 313$.

Preliminary simulations were run using LM, the parameter values of Simulation 3, and the early termination criterion. The solutions typically had congestion energy of 310. It was found that decreasing Δt to 9×10^{-4} and $(\Delta t)_\lambda$ to 10^{-3} allowed a more thorough search, which consistently resulted in solutions with $E_b = 303$. However, this slight decrease of the time constants caused a significant increase in the number of iterations required, and made it more difficult for the neuron output voltages to reach values sufficiently close to 0 or 1. Increasing the LM time constant back to its original value (0.01), besides helping drive the system state to a corner of the hypercube, also improved the solution quality. The lowest congestion energy found was 291.[7]

[7] Similar efforts with the 24-node network of Simulations 3 and 4 were unsuccessful. Apparently, decreasing the time constants with that network allows the search to find highly-suboptimal inescapable local minima, which are jumped over when faster time constants are used.

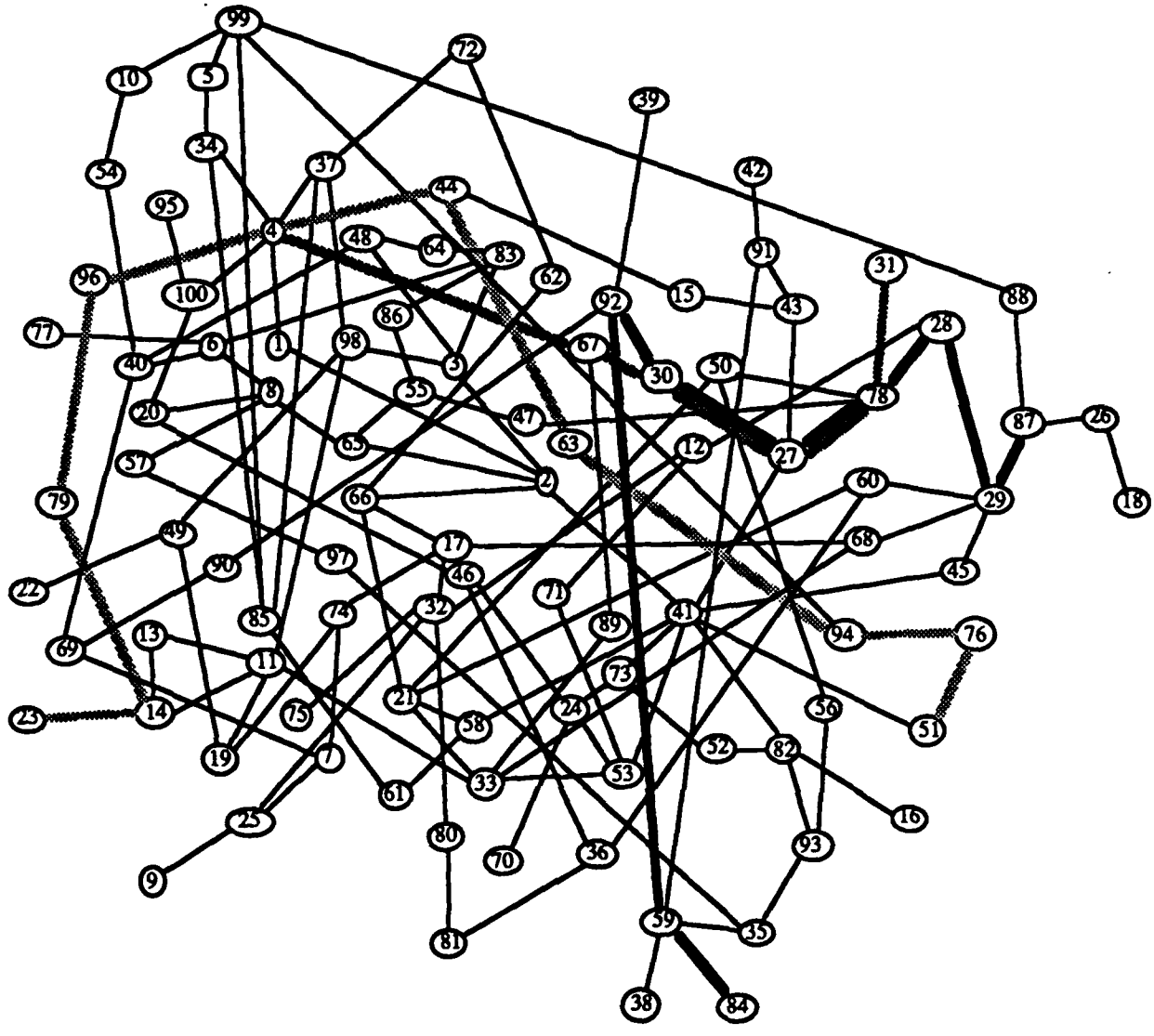


Figure 4.10. A 100-node communication network

Simulations were then performed from 50 different initial states using both the LM NN and the MLM NN with the following parameter values:

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	$\lambda_{3i}(0)$	b	α	I	Δt	$(\Delta t)_\lambda$	u_o	ε
1.0	1.0	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 Figure 4.11. The largest congestion energy value found using either LM or MLM was 303, which is 10 units less than the best solution found by the *shortest-path* heuristic. The best solution found by any method had $E_b = 291$ and was found using the LM NN. The shaded paths in Figure 4.10 represent 3 of the 40 paths selected in this solution. Although the LM and the MLM NNs yield results in the same range, Figure 4.11 shows that the use of a single LM for the third constraint yields slightly better results than the MLM formulation.

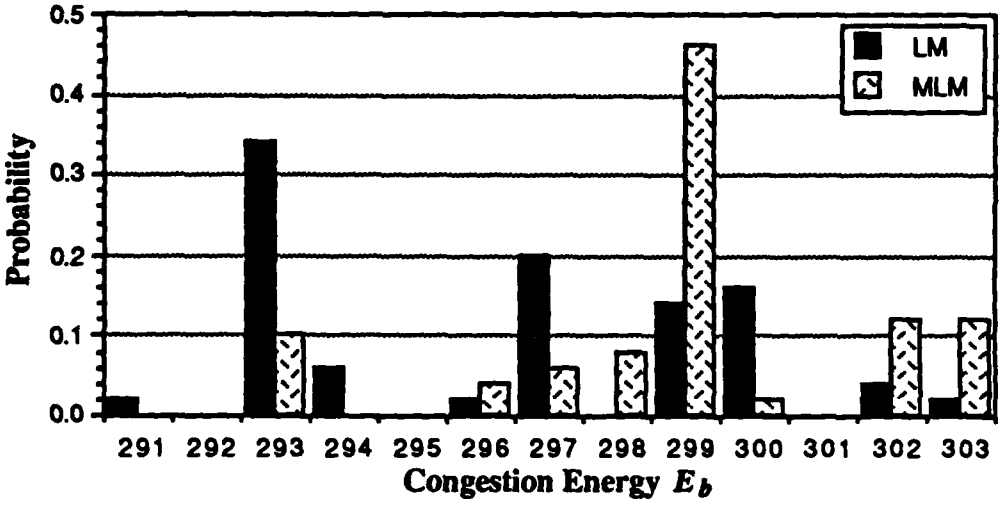


Figure 4.11. Results of Simulation 6 with and without MLM

Simulation 7: LM, MFA and the 100-Node Network

The two primary techniques that we have used to improve NN performance are mean field annealing (MFA) and Lagrange multipliers (LM). The benefits realized from these techniques suggest that it might be advantageous to use them simultaneously. However, there is apparently no synergistic relationship here. Efforts to couple Lagrange multipliers with simulated annealing yielded at best solutions slightly worse than those obtained using either method independently.

The extended search of the interior of the hypercube allowed by MFA causes the neuron output voltages to linger away from the output voltage limits (0 and 1). This delays the approach of the constraint energies toward zero and, in turn, causes the Lagrange multipliers to become excessively large. Slowing the Lagrange multiplier time constant alleviates this effect somewhat, but it also allows the congestion-limiting term to dominate. Such undue initial emphasis on the congestion energy drives the system toward the trivial state of minimum congestion in which no paths are activated, as was noted in the discussion of Simulation 1. When the LMs finally force

convergence to a valid solution, it is highly suboptimal because of the initial domination of the effects of the congestion-energy term. This was verified through simulations with different values of b, which are presented later in this section (Figure 4.13).

The following demonstrates the futility of coupling LM and MFA. Recall that MFA involves steepening the nonlinearity parameter according to $u_o = 10/(c+t/\tau_u)$ while $u_o > u_o'$. A simulation of the 100-node network using both LM and MFA was run repeatedly from the same initial state, varying c so that $u_o(0)$ (the initial value of u_o) took values ranging from 0.2 to 10.0. The duration of the MFA (the number of iterations required for u_o to go from $u_o(0)$ to u_o') was fixed at 500 iterations by changing τ_u as needed. The other parameter values were:

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	b	α	I	Δt	$(\Delta t)_\lambda$ (t<500)	$(\Delta t)_\lambda$ (t≥500)	u_o'	ε
1.0	1.0	1.0	0.85	1.0	5.0	10^{-4}	0.001	0.01	0.1	0.01

The smaller LM time constant during MFA was used to prevent the constraints from overwhelming the optimization term. Note that if $u_o(0) = u_o'$, there is no MFA and the NN uses pure LM. Figure 4.12 shows a plot of the resultant congestion versus $u_o(0)$. Clearly the best results are obtained as the effects of MFA diminish.

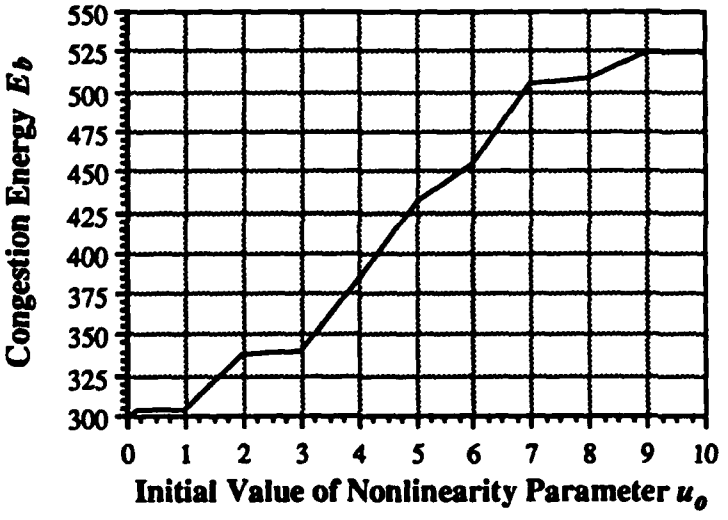


Figure 4.12. Attempted coupling of MFA and LM

Additional efforts to discover lower-energy states through parameter optimization were attempted by varying the congestion-limiting parameter b. A seemingly reasonable way to decrease congestion is to increase b. However, experimentation with this hypothesis revealed that

it is not necessarily true; although increasing the value of b does place increased emphasis on optimization, it does so at the expense of constraint satisfaction. The system is initially driven toward illegal states with minimum congestion energy. The illegal states cause large constraint energies and rapid Lagrange multiplier growth. By the time the LM are large enough to force convergence to a legal state, the optimization term has driven the NN far from a state that is simultaneously legal and optimal. Descent from this state to a corner of the hypercube yields a highly-suboptimal solution. Thus, the magnitude of b must be such that the system state can evolve toward a congestion-energy minimum, while at the same time satisfying the constraints closely.

Conversely, extremely small b allows convergence to a valid solution based almost solely on constraint satisfaction. Once again, highly-suboptimal solutions are generally found. Thus, there is an optimal range for b. This was demonstrated by means of simulations using LM, MFA, and b values ranging from 0.5 to 3. The MFA parameters were $c = 10.0$, $\tau_u = 5.55$, and $u_o' = 0.1$. All other parameters were the same as in the previous simulation. The results are shown in Figure 4.13. For this particular network and these NN parameters, the optimal range of b is [0.79, 0.95].

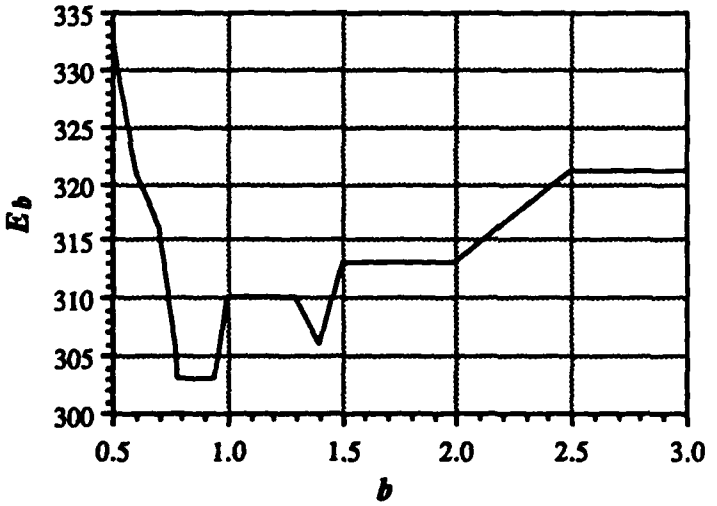


Figure 4.13. Congestion energy vs. b

In conclusion, all efforts to use Lagrange multipliers and MFA simultaneously seem to require the temporary neutralization or dilution of the effects of one or both methods; even when this is done, results of only fair quality are generally obtained. We conclude that the use of Lagrange multipliers alone produces better results than the combined use of these two methods.

4.8 Non-Unit 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 of the traffic over two or more routes, can be advantageous. To permit alternate routing, we may 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 in both the 24-node network (Figure 4.1) and the 100-node network (Figure 4.10) were increased. To accommodate m units of traffic on the i^{th} SD pair, m replicates of each of the neurons $ij, j = 1, ..., N_p(i)$, 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, for which we have already developed a simulation model. The same constraints and optimization connections used for the unit-traffic example are used here as well.

4.8.1 Triplicate 24-Node Network

Three units of traffic were placed on each of the 10 SD pairs in the 24-node network of Figure 4.1. The resultant NN, which consists of 156 neurons, was simulated from 10 different initial states using LM and the following parameter values:

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	b	α	I	Δt	$(\Delta t)_\lambda$	u_o	ε
1.0	1.0	1.0	0.5	1.0	5.0	0.0009	0.01	0.1	0.01

The congestion-energy values of the solutions ranged from $E_b = 547$ to 552. Nine of the NN solutions to the 3-unit traffic problem resulted in traffic splitting at 3 SD pairs. The tenth solution split the traffic at 4 SD pairs. At each instance of splitting, two units were sent on one path and one unit sent on another. Table 2 shows how the traffic was split in representative solutions.

To verify the advantage of alternate routing, these solutions were compared to the one obtained by sending all three units of traffic along a best-solution path set obtained in Simulation 3. For example, if three units of traffic are sent on each path in the path set {0, 6, 11, 12, 20, 23, 36,

42, 45, 50} (see Table B.1 in Appendix B for a listings of the paths that correspond to the numbers in this set), one of the optimum solutions to the unit-traffic problem, the resultant congestion energy is 554.[8] Thus all ten simulations of the triplicate network performed better than the simple triplication of the best solution for a unit-traffic example.

Table 2. Traffic splitting in the 24-node network with 3 units of traffic on each SD pair

Solution	SD pair 3		SD pair 7		SD pair 8		SD pair 10	
with:	$P_{3,3}$	$P_{3,4}$	$P_{7,2}$	$P_{7,4}$	$P_{8,1}$	$P_{8,4}$	$P_{10,1}$	$P_{10,2}$
$E_b = 547$	1	2	1	2	1	2	0	3
$E_b = 550$	1	2	1	2	1	2	1	2
$E_b = 551$	2	1	1	2	2	1	0	3
$E_b = 551$	1	2	0	3	1	2	1	2
$E_b = 552$	2	1	1	2	0	3	1	2

4.8.2 Triplicate 100-Node Network

A similar evaluation was performed for the 100-node network (Figure 4.10). Three units of traffic were placed on each of the 40 SD pairs, resulting in a NN model consisting of 981 neurons. Preliminary simulations were performed with time constants that were too large ($\Delta t = 9 \times 10^{-4}$), and gave solutions with all the neurons output voltages equal to 0. This would not be noteworthy, except in that we were using the early termination criterion, which broke the output voltage ties by selecting the lowest numbered path between each SD pair. As a result of the method used in enumerating the paths between each SD pair, the lowest numbered path between each SD pair is the first path between that SD pair found by Dijkstra's algorithm, and, as such, is a shortest path. Hence, selecting the lowest numbered path between each SD pair effectively routes all three units of traffic on a given SD pair along one of the shortest paths between that SD pair. The congestion energy of this set of paths is $E_b = 4002$.

Again, it would be desirable to determine the best shortest-path 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 best

[8] If path 10 is used instead of path 11 (another optimum solution to the unit traffic problem), the resultant congestion energy resulting from 3 units of traffic is 558.

shortest-path solution to the unit-traffic problem. The congestion energy of this solution is E_b = 3543.

An acceptable time constant was found, and simulations were run from 20 different initial states using LM and the following parameter values:

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	b	α	I	Δt	$(\Delta t)_\lambda$	u_o	ε
1.0	1.0	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 7 SD pairs. The splitting that occurred at SD pair 38 used 3 different paths; the remaining 6 SD pairs sent 2 units of traffic on one path and the third unit on another path.

To determine whether any benefit was actually obtained by the triplication of neurons, these results were compared with those obtained using the unit-traffic formulation. First, we considered the simple triplication of the best unit-traffic solution. Doing so 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 best 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 of dramatic proportions, it is significant in that we are unaware of other methods to find better sets of paths. Perhaps even more noteworthy is the ability of the NN model to handle such a large network and to provide good solutions for a reasonably large percentage of the initial states.

The results of this and the previous section show that the Hopfield network can be used successfully to select good sets of paths and to split traffic in a beneficial manner when uniform traffic (i.e., the same amount of traffic between each SD pair) must be supported by the network. However, in an actual communication network, the case of nonuniform traffic (i.e., non-equal

traffic requirements for the SD pairs) is more likely than these uniform scenarios. This issue is addressed next.

4.8.3 Nonuniform Traffic in the 100-Node Network

In order 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 (Figure 4.10). The NN model was expanded to handle greater than unit traffic as described in Section 4.8.

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 (see Table B.2). Both of the loadings were briefly analyzed using the NN model with LM and the parameter values listed in Section 4.8.2. 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.

4.9 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 the calculation of 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 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 using these two approaches.

Recall that the path-neuron congestion energy formulation used to this point is given by

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

where

P_{ij} = the j^{th} path between SD pair i.
$\left|P_{ij}\cap P_{kl}\right|$ = the number of nodes shared by paths P_{ij} and P_{kl}.

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:

1) 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.

2) A node that is an end-node (a source or destination) for one path and an intermediate node in the other path is labeled a type-2 node, and receives a weight of 0.5.

3) The type-3 node is an end-node for both paths, and is given a weight of 0.25.

Thus E_b' has the same form as E_b, except that $\left|P_{ij}\cap P_{kl}\right|$ is now defined as follows:

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

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 flows support only one or the other. Thus E_b' may, in fact, be a more appropriate performance criterion than E_b. In addition, as noted earlier, the use of E_b' permits the direct comparison of the path-neuron and link-neuron models, as is demonstrated in Section 5.

An exhaustive search of the 24-node network (Figure 4.1) based on the modified congestion energy metric (E_b') found that the best solutions have $E_b' = 33.25$. The best solutions

based on the original metric are characterized by a congestion energy of $E_b = 45$.[9] The best shortest-path heuristic solution, based on the E_b' criterion, resulted in $E_b' = 36.75$. Based on the original metric, the best shortest-path heuristic solution had $E_b = 47$. Note that different states are optimal under the two metrics. A retrospective evaluation of the solutions obtained in Simulation 3 using the modified congestion metric found that 55 of the 100 solutions have $E_b' = 33.75$, 43 have $E_b' = 34.25$, and 2 have $E_b' = 35.25$. Of course, the model used in Simulation 3 was programmed to minimize E_b rather than E_b', so the performance in terms of the modified metric is not expected to be good as that obtained under the programmed metric (97% optimum solutions with $E_b = 45$, and 3% with $E_b = 46$). Note that with this particular 24-node network (i.e., for the particular topology and set of SD pairs) and the modified congestion metric, all the solutions have E_b' values that are odd quarters, i.e., $E_b' = (2i-1)/4$, where i is an integer greater than or equal to 67.

The modified congestion-energy formulation was installed in the path-neuron LM NN model, and simulations of the 24-node network were run. Using the parameter values of Simulation 3, simulations from the same set of 100 different initial states used in Simulation 3 gave solutions of which 86% had $E_b' = 33.75$. The remaining solutions had energy values of 34.25 to 35.25. Parameter values were then varied in an effort to obtain improved results. Our best results to date were obtained using the following parameter values:

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	b	α	I	Δt	$(\Delta t)_\lambda$	u_o	ε
1.0	1.0	1.0	0.5	1.0	5.0	0.0001	0.01	0.1	0.01

In a series of 100 runs, 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.

The 24-Node Random Network Revisited

Simulations of the 24-node random network of Simulation 5 were performed using the E_b' LM NN model (i.e., connection weights based on the E_b' formulation) and the same set of parameters and 100 initial states used in Simulation 5. In 96 of these runs, the congestion energy was $E_b' = 24$, which is 2 units better than the best shortest-path heuristic solution under the new metric ($E_b' = 26$). One NN solution had $E_b' = 26$, and the remaining three solutions had $E_b' \leq 28$. Perhaps of more interest is the fact that 96 of the solutions[10] had $E_b = 45$ and $\psi = 59$. Recall that

[9] Congestion energy is lower under the new metric because type-2 and type-3 nodes are weighted less than type-1 nodes.

[10] The solutions generated using the E_b' model were reevaluated using the E_b cost function.

in Simulation 5 all the solutions had $E_b = 47$ and $\psi = 62$, but interim solutions were found with $E_b = 45$ and $\psi = 59$. Thus, 96% of the E_b and ψ values obtained under the E_b' formulation are actually lower than any of those found using the original E_b congestion-energy formulation despite the fact that the original formulation was designed specifically to minimize E_b.

Thus, the E_b' model performs slightly better for this randomly generated network than the E_b model, based on the E_b congestion-energy formulation. However, in the 24-node network of Figure 4.1, the original model was able to find a global minimum of E_b in a large fraction of the simulations. The new model has yet to find a global minimum of E_b'.

The inability to find a global minimum using the modified congestion energy formulation motivated exploration of the use of simulated annealing. The results of the application of Gaussian simulated annealing to our problem are discussed next.

4.10 The Gaussian Machine

Akiyama et al. [17] have developed a form of simulated annealing (see Appendix A) that uses additive Gaussian noise (AGN) with diminishing variance in conjunction with a mean-field-annealing (MFA) type of nonlinearity steepening.

The use of GSA has no direct effect on the NN energy formulation. Its presence is reflected in the equations of motion only by the additive noise term as follows:

$$u_{ij}' = u_{ij} + \eta,$$

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

$$u_{ij}'(t+\Delta t) = u_{ij}'(t) - \Delta t\left(u_{ij}'(t) - \sum_{k=1}^{N_M}\sum_{l=1}^{N_P(k)} T_{ij,kl}V_{kl} - I_{ij}\right) + \eta,$$

where u_{ij}' 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 which controls the rate of cooling.

Since good results were obtained by using the method of Lagrange multipliers, it was hoped that a modified form of GSA could be developed that included the benefits of the use of LM. As discussed in Section 4.7 (Simulation 7), the simultaneous use of LM and MFA is generally detrimental to NN performance. Therefore, it was decided to evaluate the use of LM in place of MFA in the GSA paradigm. The use of Lagrange multipliers has an effect similar to that of MFA in that the small initial values of the Lagrange constraint coefficients tend to slow the initial movement through the solution space, as does the initially flatter MFA sigmoid function.

4.10.1 Simulation Results for a 24-Node Network

The 24-node network of Figure 4.1 was used in evaluating the use of GSA in conjunction with the use of LM. A constant value of the nonlinearity parameter u_o was used in these runs. After a period of trial-and-error adjustment of the parameters, simulations from 100 different initial states were run with the two different parameter sets that had produced the best performance in preliminary runs. The two parameter sets differed only in the initial noise temperature T_o, as shown in the following table.

$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	b	α	I	T_o	τ_T	Δt	$(\Delta t)_\lambda$	u_o	ε
1.0	1.0	1.0	0.7	1.0	5.0	0.025, 0.00625	200	0.005	0.01	0.1	0.01

The results of these simulations, in terms of congestion energy E_b', are presented in Figure 4.14. For the first time in our studies using the new performance metric E_b', an optimum solution ($E_b' = 33.25$) was found. Figure 4.14 shows that, although the use of the larger initial noise temperature ($T_o = 0.025$) produced one solution with the optimum congestion energy, the use of the smaller initial noise temperature ($T_o = 0.00625$) gave near-optimum solutions more frequently. A total of 84% of the solutions obtained using $T_o = 0.00625$ were better than the best solution found by the shortest-path heuristic ($E_b' = 36.75$), whereas only 40% of the solutions obtained using the larger noise temperature were better than this reference value. Note that as T_o approaches zero, the NN approaches the pure LM model.

Since the simulations using GSA with LM did not produce uniformly good results, simulations using GSA with MFA and constant constraint coefficients were run. The results of simulations from 100 different initial states, which are shown in Figure 4.14, were obtained using the following parameter values:

λ_1	λ_2	λ_3	b	α	I	T_o	τ_T	c	τ_u	u_o'	Δt	ε
550	500	200	95.5	1.0	40	0.02	100	9.09	100	0.1	5×10^{-5}	0.01

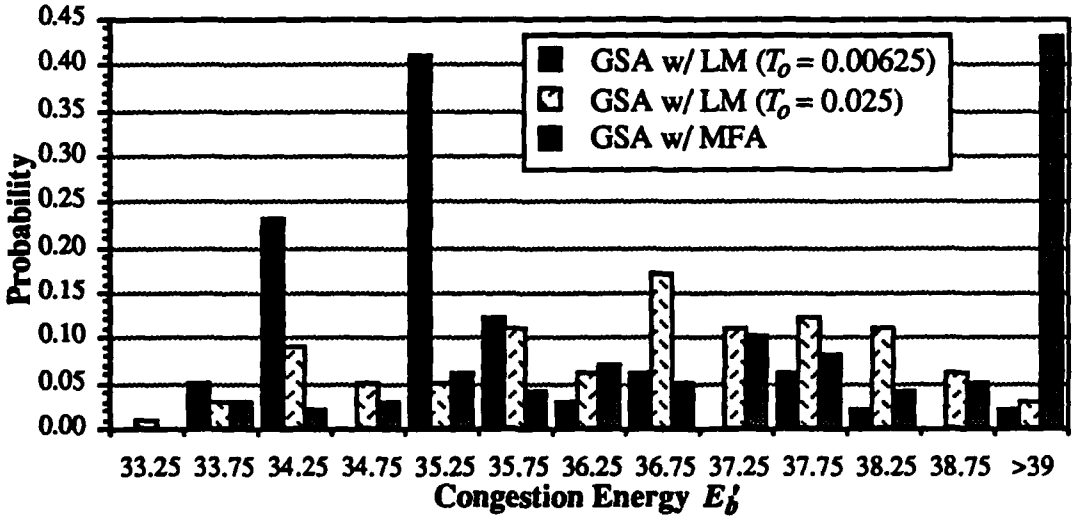


Figure 4.14. Results of Gaussian simulated annealing with (1) LM and no MFA, and (2) MFA and no LM

In Figure 4.14, the results of the use of GSA with MFA (and constant constraint coefficients) are compared with the solutions obtained through the use of GSA with LM (no MFA). The figure clearly indicates that better results were obtained in the simulations that use GSA with LM. Only 25% of the solutions obtained using GSA with MFA were of better quality than the best shortest-path heuristic solution.

Several modifications were evaluated in an attempt to improve performance. One such effort was the limiting of neuron input voltages. The idea was to establish hard limits on the neuron input voltages to keep them near the steep region of the nonlinearity, and thus to prevent extreme input voltage values that were relatively unaffected by the application of noise. Simulations using neuron input-voltage limiting and LM found that the limiting allowed the GSA to be effective for a larger number of iterations, as shown by continuing state changes. However, no appreciable improvement in the solution quality was obtained.

A second modification that was tried was to vary the Lagrange multiplier time constant in time. The schedule used was essentially the inverse of the cooling schedule, i.e., $(\Delta t)_\lambda = (\Delta t)_\lambda'(1+t/\tau)$. By applying the LM time constant schedule, the Lagrange multiplier growth was slowest when the noise magnitude was the largest. As the magnitude of the AGN decreased, the

rate of growth of the LM increased, gradually forcing convergence to valid solutions. Again, no appreciable improvement in the solution quality was obtained.

In conclusion, the application of Gaussian simulated annealing to our problem has provided mixed results. With its use, an optimal solution was indeed found under the E_b' formulation, whereas none were found by any of our other NN models. However, the vast majority of the solutions found using GSA were much worse than those obtained using the method of Lagrange multipliers. The results of these studies suggest that GSA may be a suitable approach for our problem, if it is possible to run a very large number of simulation runs. Although most solutions will generally not be as good as those obtained using the method of LM (without GSA), it is possible to use the best solution that is found. However, generally robust performance is obtained using the LM method alone.

4.11 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. In many of our examples, optimal or nearly-optimal solutions were found. The presentation in this section has been somewhat evolutionary, in that it has discussed a variety of methods we have used to improve system performance, some of which have been extremely successful, while others have not. 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 (a gradual increasing of the slope of the neurons' input-output nonlinearity) provided greatly improved performance over the performance achieved using a nonlinearity of constant slope. However, a solution with a global minimum energy was found only once in a series of 100 runs from different random initial states, and only four other runs performed as well as the best solution 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.

Instead of attempting to determine a good set of constant coefficients, we have used the method of Lagrange multipliers, which permits the coefficients to vary dynamically along with the evolution of the system state. Use of this method has provided optimal or near-optimal solutions

in many of our examples. For example, in our studies of a 24-node network with 10 SD pairs and a total of 52 paths, a global minimum value of the congestion energy was found in 97 out of 100 runs from different random initial states.

The robustness of our NN paradigm was demonstrated by applying it to several randomly-generated networks. For example, in a 24-node network with 13 SD pairs and 132 paths, solutions better than those obtained by using the shortest-path heuristic were found in all 100 runs. A 100-node network, with 40 SD pairs and 327 paths, was then considered. In a series of 50 runs, convergence to a solution better than that produced by the shortest-path heuristic was again obtained in every case. The global-optimum solution for either of these problems is not known, however, because the total number of states makes an exhaustive search prohibitive.

Although some adjustment of system parameters was necessary when studying different networks under the LM method, such adjustments were typically limited to time constants and bias currents, for which acceptable values were determined relatively quickly. We noted above that the choice of the coefficients in the connection weights is the most critical feature of Hopfield NN design. With the use of the LM method, these values are determined dynamically. 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 a valid state with binary neuron values. In particular, the LMs associated with unsatisfied constraints increase at a rate proportional to the current value of the corresponding constraint energy. When, ultimately, a particular constraint is satisfied (in which case the corresponding constraint-energy reaches a value of zero), the corresponding LM remains constant. 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.

The ability of our NN model to handle larger examples was demonstrated by increasing the traffic requirement from one unit to three units of traffic between every SD pair. Studies of the the 100-node network with this triple-traffic requirement represent the largest NN we have simulated to date, i.e., 981 neurons. In a series of 20 runs, eight produced solutions with lower values of E_b than that obtained by simply triplicating the best solution obtained for the unit-traffic case. Thus the benefits of alternate routing were shown, while demonstrating the ability of our NN model to provide good solutions for large, heavily-congested networks. Nonuniform traffic requirements (of one to four units of traffic per SD pair), which resulted in a NN model of comparable size, were also considered. This is perhaps a more useful application for the NN

model than uniform traffic, since the paths chosen by unit-traffic solutions do not necessarily perform well under highly nonuniform traffic requirements.

The LM technique has clearly been the most effective technique we have found to improve system performance. Our attempts to combine this 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.

We also attempted the use of Gaussian simulated annealing, which produced extremely variable results. Although an optimum solution was found once in a series of 100 runs (the only time it was found under the E_b' performance criterion), most results were not as good as those obtained using the LM technique alone.

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.

In conclusion, we have demonstrated the effectiveness of our Hopfield NN model for the minimization of 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 the development of 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 which 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 those among complete paths, must be taken into account. 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 determine the connection weights dynamically. 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. Figure 5.1(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. The corresponding link-neuron NN model is shown in Figure 5.1(b). 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. Several neurons may correspond to the same physical link in the NN model. For example, neurons 221 and 122 in Figure 5.1(b) both represent the physical link connecting nodes 4 and 5. The connection weights shown in the figure are discussed in Section 5.4.

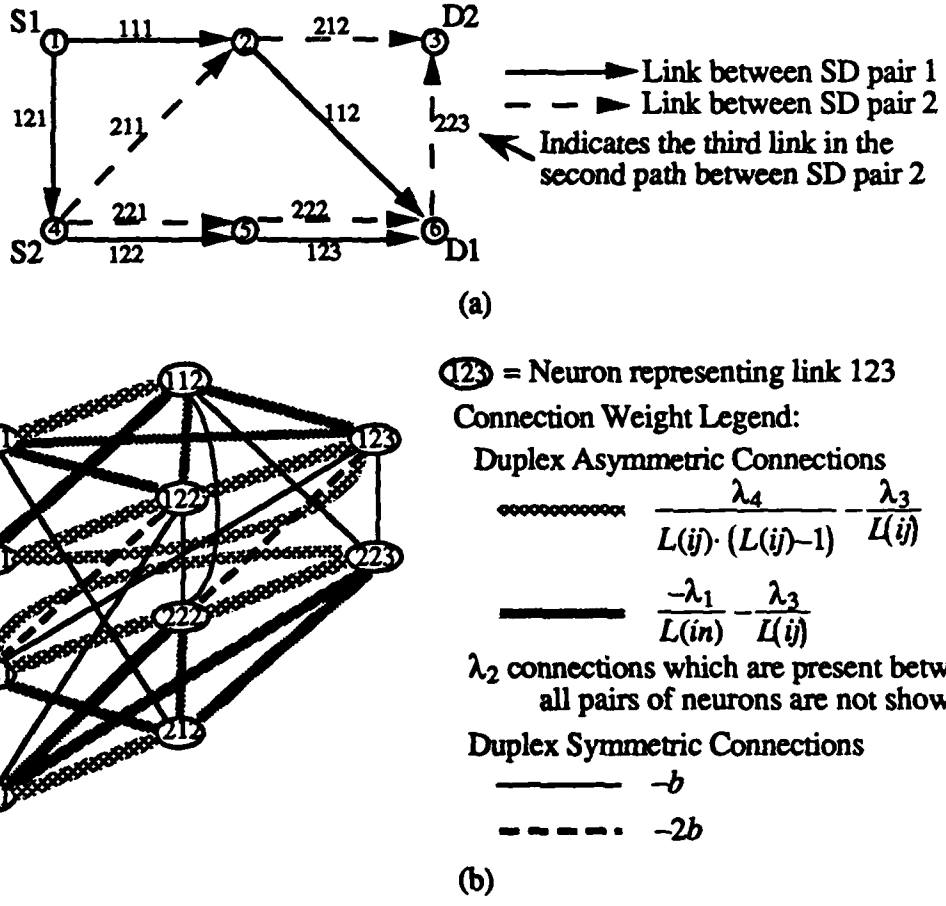


Figure 5.1. An example network, (a) shows a six-node communication network, and (b) is the corresponding link-neuron model

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 may be rewritten as

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

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 taken into account. 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:

$$E_{total} = bE_b + \sum_{i=1}^{4} \lambda_i E_i - I \sum_{i=1}^{N_{nl}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} V_{ijk}. \quad (**)$$

Again, b is a constant coefficient that weights the relative priority given to optimization, as compared to constraint satisfaction. The λ_i are the connection coefficients for the constraint terms. They may be constants or, since all the constraints are equality constraints, they may be variables that are dynamically updated by the method of Lagrange multipliers. The additional bias (I) term, as in the path-neuron formulation, is used to provide 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.

Although some preliminary studies of the link-neuron model used mean-field annealing and constant coefficients to weight the neuron connections, it was found that much better results were again obtained using the method of Lagrange multipliers. Therefore, the development presented here focuses on the LM implementation of the link-neuron model. As before, we first discuss the congestion energy, and then the constraints that must be satisfied.

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.,

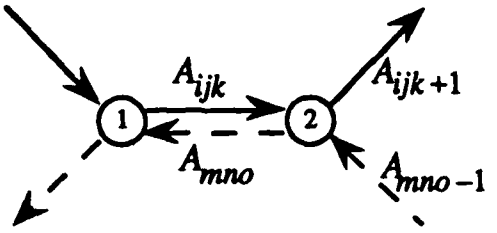
$$E_b^{\ell} = \frac{1}{2}\sum_{i=1}^{N_{sd}} \sum_{\substack{m=1 \\ m\neq i}}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{n=1}^{N_p(m)} \sum_{k=1}^{L(ij)} \sum_{o=1}^{L(mn)} \left| A_{ijk} \cap A_{mno} \right| V_{ijk} V_{mno}$$

where

A_{ijk} = the k^{th} link in the j^{th} path between SD pair i.

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

Note that $\left| A_{ijk} \cap A_{mno} \right|$ 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. These two situations are illustrated in Figure 5.2. When the two links have no common nodes, $\left| A_{ijk} \cap A_{mno} \right| = 0$. 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 will be 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.



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

Figure 5.2. Illustration of adjacent links and links that share the same physical link

The basic difference between the expression for E_b^{ℓ} 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^ℓ with respect to V_{ijk}.

$$\frac{-\partial E_b^\ell}{\partial V_{ijk}} = -\sum_{\substack{m=1 \\ m \neq i}}^{N_{ul}} \sum_{n=1}^{N_{\mu}(m)} \sum_{o=1}^{L(mn)} |A_{ijk} \cap A_{mno}| V_{mno} .$$

5.2.1 Path-Neuron Congestion Metric vs. Link-Neuron Congestion Metric

Recall that the path-neuron congestion formulation was modified to make it comparable to the link-neuron formulation, as discussed in Section 4.9. Now that the congestion measure has been defined for the link-neuron model, we can demonstrate the relationship between the two. The three types of shared nodes that were defined in the path-neuron modification are shown in Figure 5.3.

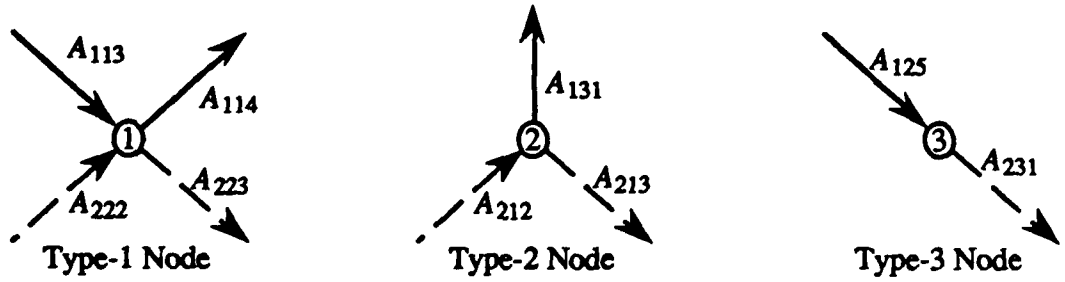


Figure 5.3. Examples of the three types of shared nodes

A type-1 node is an intermediate node in both paths and receives a weight of $1b$ in the path-neuron model. In the link-neuron model, each of the four links in the two paths sharing this type-1 node is inhibited by two adjacent links. For example, link A_{113} in Figure 5.3 is inhibited by links A_{222} and A_{223}. Thus, the sum of the congestion energy resulting from these adjacencies on this type-1 node is $E_b^\ell = 8b/2 = 4b$. A path-neuron type-2 shared node is an end-node for one path and an intermediate node in the other, and is given a weight of $b/2$. In the link-neuron model one of the three links incident on the type-2 node is the first or last link in a path and is inhibited by two adjacencies, e.g., link A_{131} in Figure 5.3 is inhibited by links A_{212} and A_{213}. The remaining two links are each inhibited by only one adjacency (A_{212} and A_{213} are each inhibited by A_{131}), giving the node a link-neuron congestion energy contribution of $E_b^\ell = 4b/2 = 2b$. A type-3 shared node is an end-node in both paths and is given a weight of $b/4$ in the path-neuron model. A similar arrangement in the link-neuron model finds the two links incident on the end-node each inhibited by one adjacent link, yielding a congestion energy of $E_b^\ell = 2b/2 = b$.

Thus the contribution made by each of the three types of shared nodes to the total congestion energy is exactly four times greater under the link-neuron model than under the path-neuron model. Since the total congestion energy under either model is the sum of the contribution associated with each node in the network, the ratio of the total congestion energy under the two models is also four. Therefore, we have

$$E_b^{\ell} = 4E_b'.$$

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 $E_b^{\ell} = 144$ 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 discuss next 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*:

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

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

This constraint supplies an inhibitory connection between all pairs of neurons that represent links that are 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. The need for this normalization can be visualized best by examining the corresponding term in the equation of motion. This term, when weighted by the connection coefficients, corresponds to summing across a row of the weight matrix T. Without the path-length normalization, longer paths (corresponding to states with the same average neuron output voltage) would apply greater inhibition than shorter paths (because more neurons are supplying the

inhibition). This effect is most evident in the first few iterations, since initially all neurons associated with a SD pair are approximately equally activated. For example, consider SD pair i for which two paths are defined of lengths $L(i1) = 2$, $L(i2) = 4$. The initial output voltage of all 6 neurons in this SD pair will be approximately 1/2.[1] If there were no normalization of E_1, at the first iteration both neurons in path $i1$ would receive a total inhibition of $4\lambda_1/2$ from the neurons in path $i2$ while the neurons of path $i2$ would receive an inhibition of only $2\lambda_1/2$ from path $i1$. Thus the longer path $i2$ would be inherently favored. With the normalization, at the first iteration all 6 neurons receive equal total inhibition of $\lambda_1/2$. This allows the NN to select paths on the basis of the constraints and the congestion metric, independently of path length.

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 [11]), 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:*

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

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

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 one 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 one. 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

[1] In general, if there are J paths between a given SD pair, the initial output voltage of each neuron of all J paths will be given an initial output voltage of approximately $1/J$, independent of the path length.

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*:

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

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

This constraint is essentially a reformulation of the second. 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 may 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*:

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

$$\Rightarrow \frac{-\partial E_4}{\partial V_{ijk}} = \sum_{\substack{m=1\\ m\neq k}}^{L(ij)}\frac{V_{ijm}}{L(ij)\cdot(L(ij)-1)}.$$

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.

In contrast to all other constraint-energy formulations examined, E_4 may assume negative values if excess activation occurs. This means that the corresponding Lagrange multiplier is no

longer necessarily monotonically increasing. It appears that in extreme cases of excessive activation, the LM λ_4 may become negative, causing the E_4 connections to become inhibitory so that the level of activation is reduced. However, in all simulations run, the value of E_4 was always greater than or equal to zero. Later we consider a modified version of this constraint, under which the square of the expression for E_4 is used, thus eliminating the need for these concerns.

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. (**). The connection weights and bias currents are then determined by transforming the resulting expression into the form of Eq. (*). Doing so yields

$$T_{ijk,mno} = \frac{-\lambda_1}{L(mn)}\delta_{im}(1-\delta_{jn}) - \frac{\lambda_2 + \lambda_3\delta_{im}}{L(ij)} + \frac{\lambda_4}{L(ij)\cdot(L(ij)-1)}\delta_{im}\delta_{jn} - b\left|A_{ijk}\cap A_{mno}\right|(1-\delta_{im}),$$

where δ_{ik} is the Kronecker delta. The external input bias currents are

$$I_{ijk} = \lambda_2 N_{sd} + \lambda_3 + I\,.$$

The equation of motion for neuron *ijk* is obtained from the energy equation by taking the negative partial derivative with respect to the output voltage V_{ijk}. After discretization in the time domain, the equation assumes the following computable form:

$$u_{ijk}(t+\Delta t) = u_{ijk}(t) - \Delta t u_{ijk}(t) - \Delta t b \sum_{\substack{m=1\\ m\neq i}}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} \left|A_{ijk}\cap A_{mno}\right| V_{mno}$$

$$- \Delta t\lambda_1 \sum_{\substack{m=1\\ m\neq j}}^{N_p(i)} \sum_{n=1}^{L(im)} \frac{V_{imn}}{L(im)} - \frac{\Delta t\lambda_2}{L(ij)}\left(\sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} \frac{V_{mno}}{L(mn)} - N_{sd}\right)$$

$$- \frac{\Delta t\lambda_3}{L(ij)}\left(\sum_{m=1}^{N_p(i)} \sum_{n=1}^{L(im)} \frac{V_{imn}}{L(im)} - 1\right) + \Delta t\lambda_4 \sum_{\substack{m=1\\ m\neq k}}^{L(ij)} \frac{V_{ijm}}{L(ij)\cdot(L(ij)-1)} + \Delta t I\,.$$

As mentioned earlier, the results presented here focus on simulations where the Lagrange multipliers (the λ_i's) are updated dynamically along with the system state. Typically starting with an initial value of 1, the Lagrange multipliers are modified at each iteration by an amount

proportional to their corresponding constraint energy. At the $(n+1)^{st}$ iteration, the new value of λ_i is given by

$$\lambda_i(n+1) = \lambda_i(n) + (\Delta t)_\lambda E_i(n)$$

where $(\Delta t)_\lambda$ is the LM time constant, which may or may not be different from the global NN time constant Δt.

5.5 Simulation Procedure

Simulation of the link-neuron model was performed by means of 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, 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 δ (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.

Termination Criteria

Stable convergence is declared when all the neuron output voltages are within some specified ε value of the output voltage limits. In all of the link-neuron model simulations presented here, a value of $\varepsilon = 0.01$ was used. With this value, stable convergence occurs when all neuron output voltages are greater than 0.99 or less than 0.01.

Before explaining the termination criteria further, let us define valid convergence specifically for the link-neuron model. A valid convergence is a stable convergence that satisfies the problem constraints. As in the path-neuron formulation, all neuron output voltages must be within ε of the output voltage limits. In addition, complete paths must be formed, i.e., the set of neurons that are within ε of 1 must be the set that corresponds to all the links in exactly N_{sd} paths, one of which connects each SD pair. Note that stable convergence does not necessarily imply valid convergence. A stable convergence may occur when all neuron output voltages are zero, clearly violating the constraints.

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.[2] 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. In order 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 may be restated as: Terminate the iteration if the instantaneous state remains unchanged for a specified number (typically several hundred) of iterations.

The use of the second termination criterion (termed the early termination criterion in the path-neuron model development) implies acceptance of the instantaneous state as a valid solution, even though some link-neuron output voltages in a selected path may be very close to 0. It has been found that continued iteration usually leads to valid convergence to the declared state. Hence, it is reasonable to accept the instantaneous state as the solution if it remains unchanged for a sufficiently large number of iterations.

The termination criterion that actually causes termination of a simulation is largely a function of the three parameters that characterize each of the three termination criteria: ε, N_C, and N_{max}, where N_C is the number of iterations specified for the second criterion, and N_{max} is the maximum number of iterations allowed, i.e., the time-out value of the third termination criterion. In preliminary exploratory simulations, both N_C and N_{max} were typically set equal to 15000, and termination usually occurred as a result of satisfaction of the first criterion. Once a suitable set of parameters had been determined, Monte-Carlo type simulations from multiple initial states were run with $N_C \approx 500$ iterations, and terminations usually occurred as a result of satisfaction of the second criterion. Termination based on the third criterion typically occurred only when the value of N_{max} was small relative to the other termination parameters.

[2] The instantaneous state of the system was defined in Section 4.1.3.

Network Description

All of the simulation results presented here are based on the 24-node network shown in Figure 4.1, which we studied earlier using the path-neuron NN model. Table B.1 in Appendix B gives a list of the 10 SD pairs and 52 paths in the network. An exhaustive search of the approximately 4 million valid solutions has shown that the lowest possible value of E_b^ℓ 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 application of the shortest-path heuristic to this network yielded $E_b^\ell = 147$.

5.6 Simulation Results: The Need for Additional Bias Currents

It was originally hypothesized that the inclusion of 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[3] 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_o	ε
1.0	1.0	1.0	1.0	0.5	1.0	0.001	0.01	0.1	0.01

The results shown in Figure 5.4 indicate that the original hypothesis was erroneous. The best solutions found in simulations without additional bias had $E_b^\ell = 151$. With additional bias of 3.0 or 5.0, solutions were found with $E_b^\ell = 135$.[4] Analysis of the evolution of the excitatory fourth Lagrange multiplier indicates that elimination of the additional bias places unnecessary stress on λ_4 by requiring it to provide sufficient stimulation to activate the appropriate number of neurons. With the additional bias term included, the fourth Lagrange multiplier is free to evolve as needed to enforce its corresponding constraint. Figure 5.5 shows the evolution of λ_4 for the cases of $I = 0$ and 5. For approximately the first 400 iterations, λ_4 is essentially the same for these two cases. At that time, the additional bias has allowed the fourth constraint to be very nearly satisfied, resulting in small E_4 values and very little continued growth of λ_4. Without the additional bias, however, λ_4 must continue to grow in order to provide sufficient stimulation to drive the selected neurons toward 1.0, and thereby to satisfy the fourth constraint.

---

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

[4] As a result of the topology of this network and the method of calculating congestion energy, all E_b^ℓ values will be odd integers.

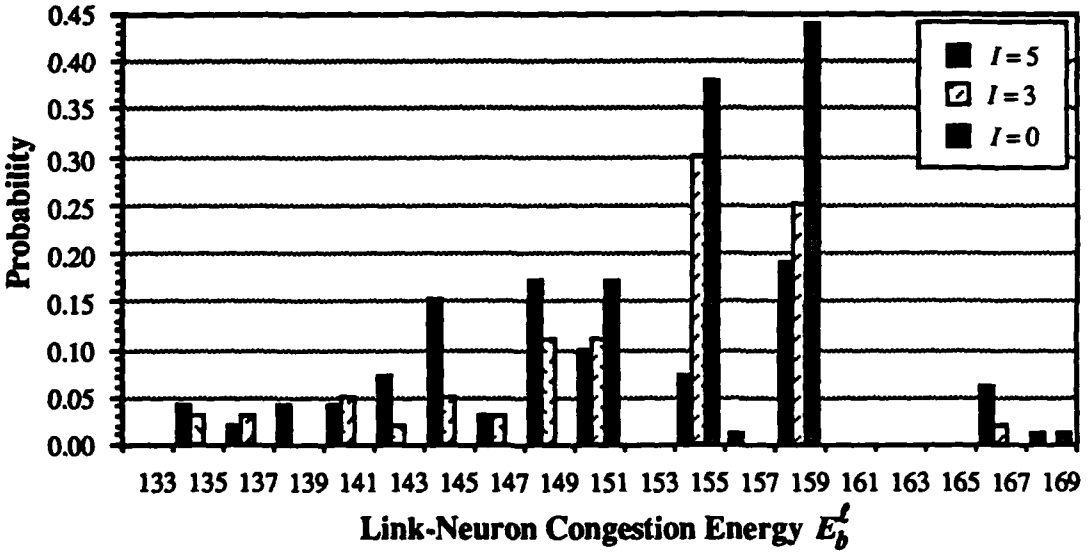


Figure 5.4. Results of simulations with and without additional bias

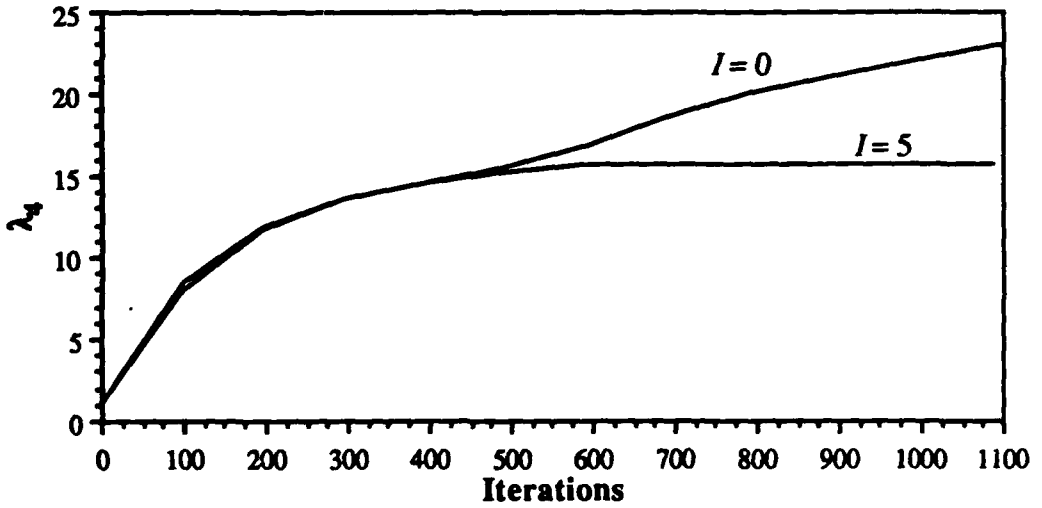


Figure 5.5. Comparison of the growth of the fourth Lagrange multiplier with and without additional bias (I)

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 the use of the additional bias, the results remain inferior to those obtained using the path-neuron model. With the path-neuron model, we were able to find solutions with $E_b' = 33.75$ (which corresponds to $E_b^\ell = 135$) 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.

In an effort to improve the NN performance the use of multiple Lagrange multipliers (MLM) for the excitatory connections was examined next.

5.7 Simulation Results: 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 to implement the fourth constraint, which encourages the activation of complete paths. This constraint may be easily reformulated in the MLM format by writing

$$E_4 = \sum_{i=1}^{N_{sd}} e_{4i} = 0,$$

where

$$e_{4i} = \left(1 - \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \sum_{\substack{m=1 \\ m \neq k}}^{L(ij)} \frac{V_{ijk} V_{ijm}}{L(ij)\,(L(ij)-1)} \right) = 0.$$

Then, as in the path-neuron model, Lagrange multipliers are defined for each SD pair as follows:

$$\lambda_{4i}(n+1) = \lambda_{4i}(n) + (\Delta t)_{\lambda} e_{4i}(n) .$$

The fourth-constraint MLM may be used in place of, or in conjunction with, the E_4 term from which they were derived. In the simulations discussed here, we replaced the E_4 term with this formulation. The total energy is then given by

$$E_{total} = bE_b + \sum_{i=1}^{3} \lambda_i E_i + \sum_{i=1}^{N_{sd}} \lambda_{4i} e_{4i} - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} V_{ijk} ,$$

and the equation of motion is modified by replacing λ_4 with λ_{4i}.

The results of simulations using MLM, the parameter values of Section 5.6, additional bias $I = 5.0$, and the set of 100 different initial states used in Section 5.6, are shown in Figure 5.6.

Also shown for comparison purposes are the best results from section 5.6. The results indicate that the use of MLM in this setting is actually detrimental to the NN performance. The NN with the original E_4 formulation found 39 out of 100 solutions with congestion energy as low as or lower than the best solution found by the shortest-path heuristic ($E_b^\ell = 147$). Using MLM, the NN found only 36 out of 100 solutions with $E_b^\ell \leq 147$.

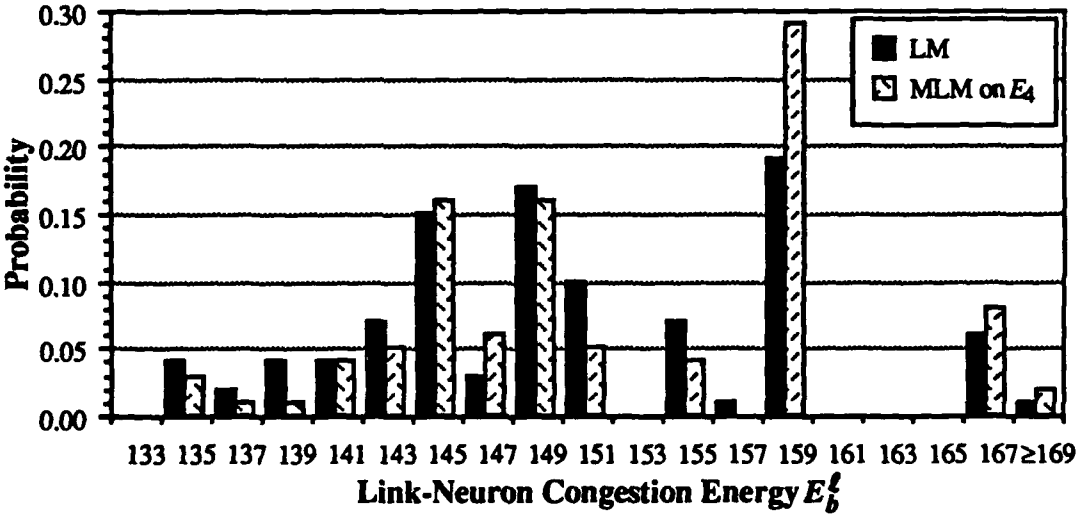


Figure 5.6. Results of simulations using MLM for the excitatory connections

5.8 Simulation Results: A Modified Formulation of the Complete-Path Constraint

The next effort involved a modification of the fourth constraint. As mentioned in the discussion of this constraint, the original implementation of the excitatory connections admits E_4 values less than zero. However, in all of the simulations run to date, the value of E_4 has always remained greater than or equal to 0. As discussed in 5.3, the occurrence of negative values of E_4 is not expected to present additional problems, but the effect of such an event is uncertain. An alternate energy formulation of the fourth constraint, which precludes negative values, is obtained by squaring E_4, i.e.,

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

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 did 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 a valid solution.

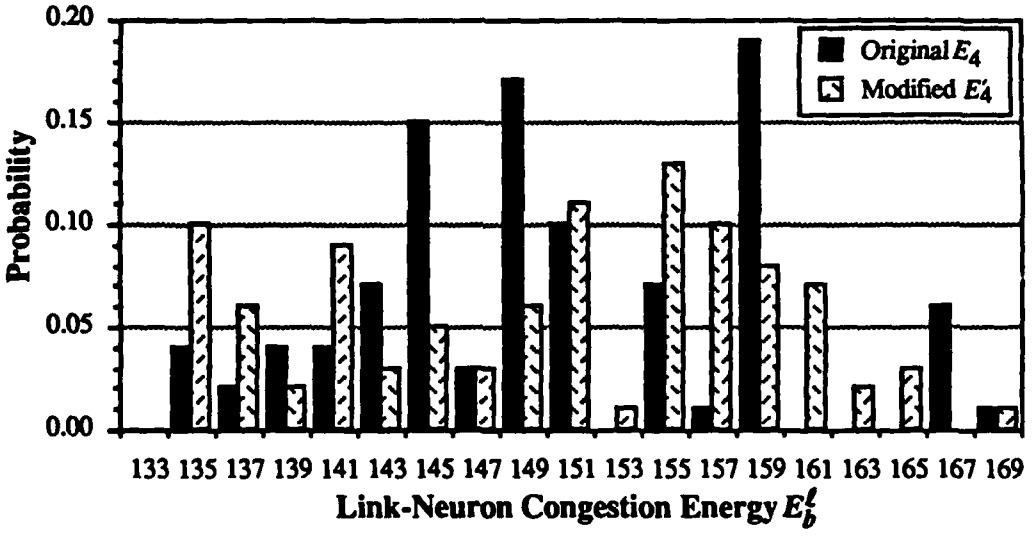


Figure 5.7. A histogram of the results of simulations using E_4' and E_4

In Figure 5.7, the results of simulations using the squared fourth-constraint formulation are compared with the results obtained using the original formulation. The parameter values and the set of 100 different initial states of the previous simulations were again used. With the modifications, 10 solutions were found for which $E_b^\ell = 135$. The original NN found only 4 solutions with this measure of congestion. Another way to compare the performance of these two models is to compare the number of solutions that do better than the best solution found using the shortest-path heuristic. Both models performed approximately equally using this criterion. Comparing the solutions of the E_4' and E_4 models to the best solution found with the shortest-path

heuristic, 38 of the E_4' solutions and 39 of the E_4 solutions had $E_b^\ell \leq 147$. Thus, the E_4' model does not produce significantly better results, nor does it degrade the NN performance. However, since a significantly larger number of best NN solutions were obtained, and since monotonically-increasing Lagrange multipliers are guaranteed, the E_4' formulation was used in the remaining simulations.

5.9 Methods to Overcome a Detrimental Preference for Short Paths

By means of the modifications discussed in Sections 5.6 and 5.8, the performance of the link-neuron paradigm was improved. However, the quality of the solutions remained far below that obtained with the path-neuron model. The link-neuron model was re-evaluated in an attempt to find the reasons for this performance. It was discovered that the energy expressions corresponding to the fourth constraint (E_4 and E_4') contain a subtle bias in favor of shorter paths, which made it difficult to find minimum-energy paths.

Recall that, in order to maintain E_4 as an equality constraint, the sum of the pairwise products of the output voltages of all the neurons that form a path was normalized by the number of pairs, $\binom{L(ij)}{2}$. The normalization is carried through the partial derivative to the equation of motion term, which is rewritten here for convenience.

$$\frac{-\partial E_4}{\partial V_{ijk}} = \sum_{\substack{m=1 \\ m \neq k}}^{L(ij)} \frac{V_{ijm}}{L(ij) \cdot (L(ij) - 1)}.$$

Thus, at each iteration, the excitation applied to neuron ijk as a result of the connections associated with the fourth constraint is the sum of the output voltages of all the other neurons that form path ij, divided by $\{L(ij)^2 - L(ij)\}$. It is this quadratic denominator expression that favors shorter paths. For example, consider two paths, $i1$ and $i2$, between SD pair i with lengths $L(i1) = 3$ and $L(i2) = 5$. If the output voltage of all the neurons that form these two paths is v, the total E_4 excitation applied to each neuron in path $i1$ is $2v/6 = v/3$. The total E_4 excitation applied to each neuron in path $i2$ is $4v/20 = v/5$. Clearly, since $v/3 > v/5$, the shorter path $i1$ receives more excitation than the longer path. The same phenomenon is evident in the E_4' formulation.

Although the best solutions produced by the shortest-path heuristic are usually of reasonably good quality, as discussed in Section 4, 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, and simulation results are presented, in the following subsections.

5.9.1 The Use of Dummy Neurons to Eliminate Short-Path Preference

We first consider a method that equalizes the lengths of all paths between a SD pair by adding dummy neurons as needed. Let $M(i)$ denote the length of the longest path between SD pair i. For each path ij, with length $L(ij) < M(i)$, we create $M(i) - L(ij)$ dummy neurons whose output voltage is given by

$$V_{ij(L(ij)+m)} = \sum_{k=1}^{L(ij)} \frac{V_{ijk}}{L(ij)}, \quad m = 1, \cdots, M(i) - L(ij).$$

That is, the output voltage of dummy neurons ijn ($n = L(ij)+1, \cdots, M(i)$) is equal to the mean output voltage of real neurons ijk, $k = 1, \cdots, L(ij)$. These dummy neurons are used only in the E_4' expressions and their corresponding equations of motion terms;[5] they are, in effect, invisible to all other terms of both the energy equation and the equations of motion. The fourth-constraint expression, with dummy neurons, becomes

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

where the sums are extended to include the dummy neurons. The sums are similarly extended in the corresponding equation of motion term:

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

The results of simulations using dummy neurons are presented next.

---

[5] The dummy neuron approach is equally applicable to the original E_4 formulation. However, as mentioned previously, we have decided to focus on the use of the E_4' model.

Dummy-Neuron Simulation Results

The use of the dummy-neuron formulation results in a marked improvement, as shown in Figure 5.8. As usual, simulations were run from the 100 different initial states used in Section 5.8. Seventeen solutions were found that had congestion energy of $E_b^\ell = 135$. The quality of the solutions found by the dummy-neuron model was better than that of the best solution found by the shortest-path heuristic in 49 of the runs; 12 solutions were found that had congestion-energy values equal to that of the best shortest-path heuristic solution (147).

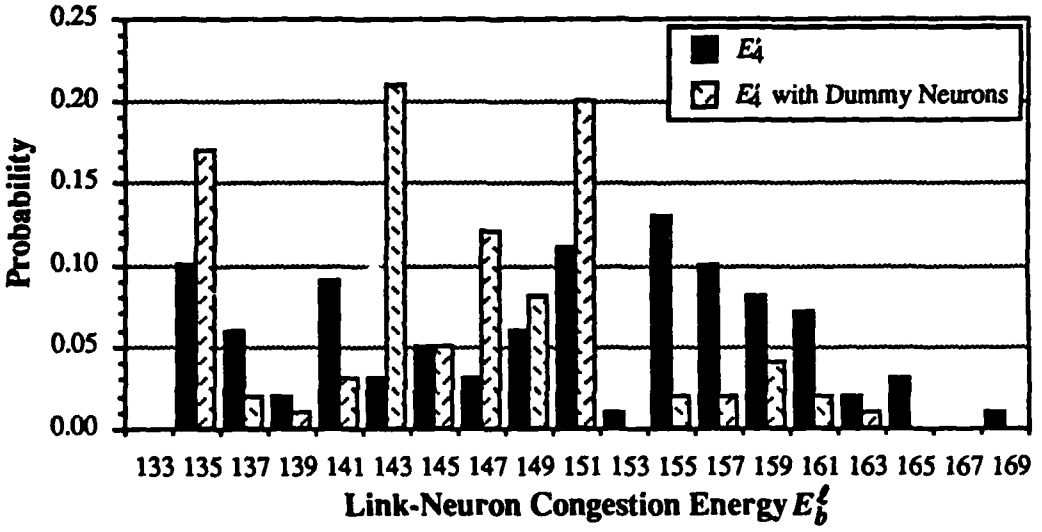


Figure 5.8. Results of simulations using dummy neurons compared with best previous results

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

5.9.2 The Use of Compensatory Bias to Minimize Short-Path Preference

An alternative method to reduce the short-path preference in the NN is to give longer paths an additional "compensatory" bias. Since the short-path preference is most evident when the neuron output voltages are equal, the amount of compensatory bias has been set to a value that exactly neutralizes the short-path preference at iteration zero (assuming no initial perturbation). To formalize this concept, we introduce the following notation: Let $m(i)$ denote the length of the shortest path between SD pair i. Let $n(ij)$ denote the number of hops by which path ij is longer

than the shortest path between SD pair i, i.e., $n(ij) = L(ij) - m(i)$. Denote the amount of compensatory bias added to each neuron on path ij by I^c_{ij}, where

$$I^c_{ij} = \frac{n(ij)}{N_p(i) \cdot m(i) \cdot L(ij)}.$$

The compensatory bias is included in the equations of motion as an additive term and is combined with the additive bias term of the energy equation.

Simulation Results with Compensatory Bias

Using the compensatory bias described above in conjunction with the E_4' formulation, simulations 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 Figure 5.9. 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 $E_b^\ell \leq 147$, the congestion energy of the best solution obtained using the shortest-path heuristic.

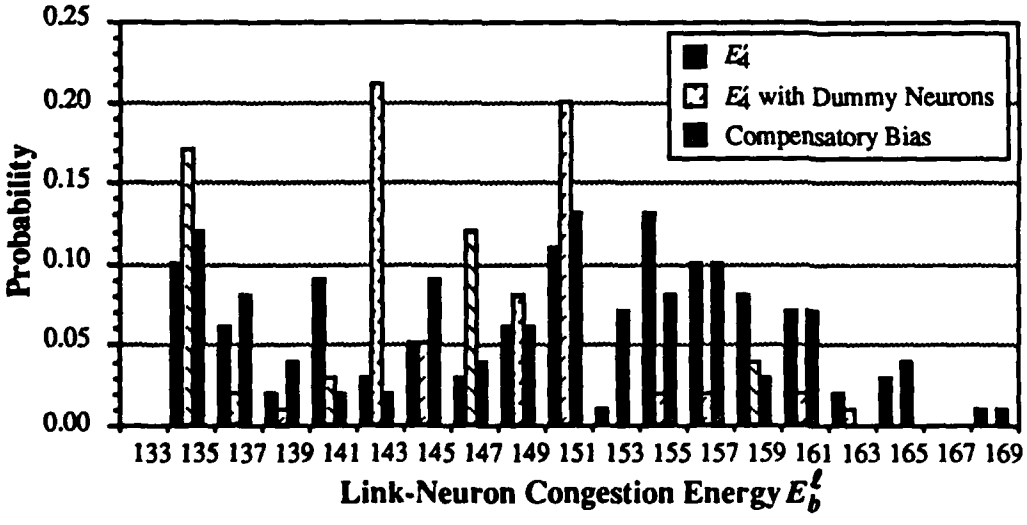


Figure 5.9. A histogram of results using compensatory bias

5.10 Use of Simulated Annealing with the Link-Neuron Model

In an attempt to improve performance further, Gaussian simulated annealing (GSA) was also attempted. The implementation details are exactly the same as presented with the path-neuron model in Section 4.10; a decaying additive Gaussian noise term is included in the equations of motion. As in efforts with the path-neuron model, the time-dependent variation in the slope of the nonlinearity was not implemented. Instead, a constant slope was used in conjunction with dynamically-varying Lagrange multipliers.

In limited experimentation with different cooling schedules, solutions were found with congestion energy values ranging from $E_b^{\ell} = 155$ 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.

5.11 A Reformulation of the Complete-Path Constraint

In the formulation of the complete-path constraint discussed thus far, excitatory connections are placed between every pair of neurons that represent two links in the same path. It appears that this formulation may excessively restrict the NN's search of the state space by coupling the states of all neurons corresponding to any given path too early in the search, thereby limiting the region of the search space that can actually be explored. This observation has led to the hypothesis that a more thorough search *may* be obtained by establishing excitatory fourth-constraint connections only between neurons that represent adjacent links in a given path, rather than between all the neurons that represent links in that path. Regardless of whether or not this hypothesis proves to be valid, this approach would significantly reduce the number of connections required in longer paths, thereby reducing the complexity of the calculation in software simulations as well as the connectivity that must be incorporated into hardware implementations.

The adjacent-link formulation of the fourth constraint is given by:

$$E_4'' = \frac{1}{2}\left(N_{sd} - \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \frac{V_{ijk}(V_{ijk-1} + V_{ijk+1})}{L(ij) - 1} \right)^2 = 0,$$

where we define

$$V_{ijk-1} = \begin{cases} 0, & k = 1 \\ V_{ijk-1}, & \text{otherwise} \end{cases}, \quad V_{ijk+1} = \begin{cases} 0, & k = L(ij) \\ V_{ijk+1}, & \text{otherwise} \end{cases}.$$

As the equation for E_4'' indicates, the use of excitatory connections between only those neurons that correspond to adjacent links on a path results in a normalization constant of $2(L(ij) - 1)$. The corresponding equation of motion term is given by

$$\frac{-\partial E_4''}{\partial V_{ijk}} = \left(N_{sd} - \frac{1}{2} \sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} \frac{V_{mno}(V_{mno-1} + V_{mno+1})}{L(mn) - 1} \right) \left(\frac{V_{ijk-1} + V_{ijk+1}}{2(L(ij) - 1)} \right).$$

Examination of this expression indicates that the problem of short-path preference remains in this formulation. In fact, the short-path preference is more strongly manifested here than in the previous model. To see this, consider two paths, $i1$ and $i2$, between SD pair i with lengths $L(i1)$ and $L(i2)$, $L(i1) > L(i2)$, and assume that all neurons in both paths have identical output voltage levels. Under the E_4' model, the excitation applied to a neuron in the shorter path $i2$ will be $L(i1)/L(i2)$ times the excitation applied to a neuron in path $i1$. Under the E_4'' model, the excitation applied to neurons in the shorter path will be $[L(i1)-1]/[L(i2)-1]$ times the excitation applied to neurons in the longer path. Since $L(i1) > L(i2)$, it follows that $[L(i1)-1]/[L(i2)-1] > L(i1)/L(i2)$; therefore, the short-path preference is stronger in the E_4'' model.

As discussed in Section 5.9.1, the use of dummy neurons (DN) in the E_4' model has been effective in mitigating the effects of the innate short-path preference. Hence, their use in conjunction with the E_4'' model has been also studied. The implementation details are essentially the same as described in Section 5.9.1. One of the reasons for developing the E_4'' model was to allow a broader search by initially focusing the decision process on successive link interaction rather than on entire path interaction. The use of DN in conjunction with the E_4'' model tends to defeat the purpose of the E_4'' model by shifting the emphasis back to path interaction through the averaging effects that the use of DN entails. Therefore, simulations were performed both with and without DN to determine the impact of DN on the E_4'' model. In Figure 5.10, the results of these simulations are compared with the results of Section 5.9.1 (the E_4' model with DN). Each of the simulations was run from the same set of 100 different initial states using the following parameter values:

	$\lambda_1(0)$	$\lambda_2(0)$	$\lambda_3(0)$	$\lambda_4(0)$	b	I	Δt	$(\Delta t)_\lambda$
Dummy Neurons	1.0	1.0	1.0	1.0	0.5	5.0	0.001	0.01
No Dummy Neurons	1.0	1.0	1.0	1.0	0.5	1.0	0.001	0.01

The results show that the use of excitatory connections between only adjacent links yields slightly degraded NN performance relative to the fully-connected E_4' model. However, some compensation for this performance sacrifice is obtained by the reduction in the number of

excitatory connections; it has thus been shown that the reasonably good results can be achieved, even though less information is incorporated into the model. In the simulations using the E_4" model without DN, 10 solutions were found with $E_b^\ell = 135$, while the model with DN found only 6 solutions with this value. However, the DN model found 55 solutions with $E_b^\ell \leq 147$, the best shortest-path heuristic solution congestion-energy value. Without DN, only 44% of the solutions had congestion energy less than or equal to 147. No significant short-path preference was evident in either model; both models activated approximately the same number of non-shortest paths as the E_4' model with dummy neurons.

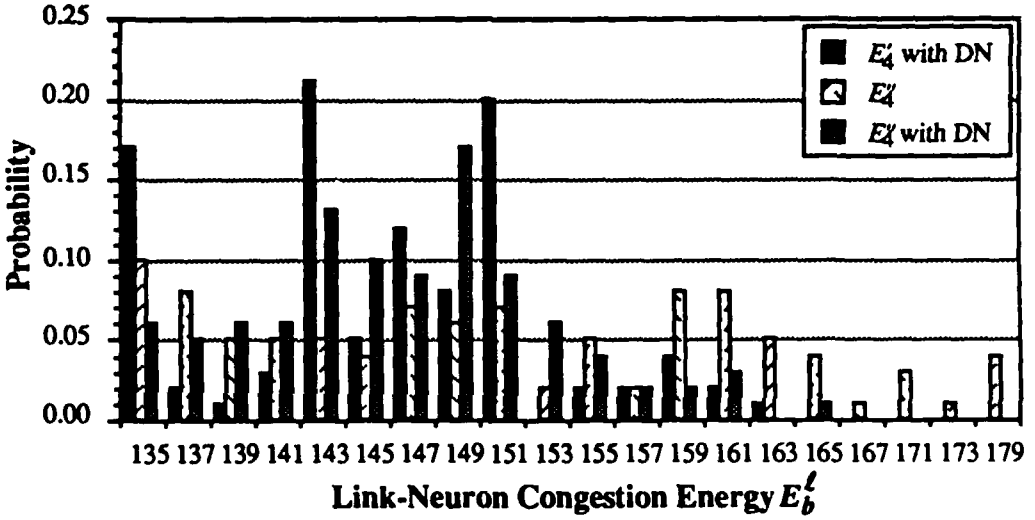


Figure 5.10. Results of simulations of the E_4" model both with and without dummy neurons and the E_4' model with dummy neurons

5.12 A Distributed Implementation of the Link-Neuron Model

The performance of the adjacent-link fourth-constraint formulation described in Section 5.11 suggests that further reductions in the NN complexity may be possible without excessive performance degradation. In this section, 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 addressed the question of distributed operation further for the purpose of decreasing the network complexity through a reduction in the number of neuron interconnections, while maintaining reasonably good results. In Section 5.12.1, the constraints are restated in a distributed form. In Section 5.12.2, the results of simulations of the distributed model are presented. Throughout both sections, deviations from a "purely" distributed model are noted.

5.12.1 Distributed-Model Constraints

In this subsection, we review each of the four centralized link-neuron model constraints and discuss their reformulation as part of a distributed model. A fifth constraint is also introduced to aid in the formation of complete paths.

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

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

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

This constraint requires inhibitory connections between all links in different paths between a SD pair. This constraint may be partially enforced in a distributed manner by establishing inhibitory connections between neurons corresponding to all the "first" links of the paths between the same SD pair; similarly, inhibitory connections can be established between neurons corresponding to all the "last" links of the paths between the same SD pair. These connections

discourage the activation of more than one source link and one destination link at each SD pair. A problem with this approach is that the active source link may be on a different path than the active destination link. Also, the behavior of intermediate links is not reflected at all in this constraint. However, the fourth constraint (activate complete paths) should ensure that the activated source and destination link as well as the intermediate links are all elements of the same path.

The normalization that was required in the centralized formulation to eliminate discrimination on the basis of path length is not required in the distributed formulation because each neuron is connected only to adjacent neurons rather than to all neurons along the path. Since each neuron is allowed only local information, multiple Lagrange multipliers (MLM) are required. Therefore, a Lagrange multiplier is defined for each source and for each destination. These considerations lead to the following reformulation of the first constraint:

$$E_1^d = \sum_{i=1}^{N_{sd}} \left(e_{1i}^f + e_{1i}^l\right) = 0.$$

where

$$e_{1i}^f = \frac{1}{2}\sum_{j=1}^{N_p(i)} \sum_{\substack{m=1 \\ m\neq j}}^{N_p(i)} V_{ij1}V_{im1} = 0, \text{ and } e_{1i}^l = \frac{1}{2}\sum_{j=1}^{N_p(i)} \sum_{\substack{m=1 \\ m\neq j}}^{N_p(i)} V_{ijL(ij)}V_{imL(im)} = 0,$$

and the superscripts f and l refer to first and last, respectively. The corresponding equation of motion terms are

$$\frac{-\partial e_{1i}^f}{\partial V_{ijk}} = \begin{cases} -\sum_{\substack{m=1 \\ m\neq j}}^{N_p(i)} V_{im1}, & k=1 \\ 0, & \text{otherwise} \end{cases}, \text{ and } \frac{-\partial e_{1i}^l}{\partial V_{ijk}} = \begin{cases} -\sum_{\substack{m=1 \\ m\neq j}}^{N_p(i)} V_{imL(im)}, & k=L(ij) \\ 0, & \text{otherwise} \end{cases}.$$

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

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

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

This constraint requires connections between all links. We did not see a way to reformulate this constraint that would result in a reduction in network complexity or would allow it to fit into a nearly-distributed model. Therefore this constraint is not used in the distributed model.

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

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

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

This constraint may be distributedly implemented in a manner similar to that used for the first constraint, i.e., use MLM to define $2N_{sd}$ Lagrange multipliers, one for each set of neurons that represents a set of links incident on a given source node, and one for each set of neurons that represents a set of links incident on a given destination node. Thus, the connections established by the first constraint will be additionally weighted by the third constraint MLM. Again, normalization by the path length is unnecessary. Therefore, the distributed energy formulation is given by:

$$E_3^d = \sum_{i=1}^{N_{sd}}\left(e_{3i}^f + e_{3i}^l\right) = 0,$$

where

$$e_{3i}^f = \frac{1}{2}\left(\sum_{j=1}^{N_p(i)} V_{ij1} - 1\right)^2 = 0, \text{ and } e_{3i}^l = \frac{1}{2}\left(\sum_{j=1}^{N_p(i)} V_{ijL(ij)} - 1\right)^2 = 0.$$

The corresponding equation of motion terms are:

$$\frac{-\partial e_{3i}^f}{\partial V_{ijk}} = \begin{cases} 1 - \sum_{m=1}^{N_p(i)} V_{im1}, & k = 1 \\ 0, & \text{otherwise} \end{cases}, \text{ and } \frac{-\partial e_{3i}^l}{\partial V_{ijk}} = \begin{cases} 1 - \sum_{m=1}^{N_p(i)} V_{imL(im)}, & k = L(ij) \\ 0, & \text{otherwise} \end{cases}.$$

This formulation, like the distributed formulation of the first constraint, recognizes only the constraint violations that involve the activation of two or more first links, or two or more last links, in paths between a given SD pair.

4. Activate complete paths:

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

$$\Rightarrow \frac{-\partial E_4}{\partial V_{ijk}} = \sum_{\substack{m=1 \\ m \neq k}}^{L(ij)} \frac{V_{ijm}}{L(ij) \cdot (L(ij) - 1)}.$$

As discussed in Section 5.11, this constraint may be implemented in a less-centralized fashion by removing all the neuron interconnections established by the centralized fourth-constraint except the ones between adjacent links. However, since the reformulation of Section 5.11 was applied to the E_4' expression, it retained the summation of the product of the output voltages of all adjacent links in each of the paths between a given SD pair, both in the energy term and in the equation of motion. To obtain an expression of this constraint that allows a purely-distributed implementation, we apply the reformulation of Section 5.11 to the original centralized fourth-constraint energy term E_4, rather than applying the reformulation to the energy term E_4'. The constraint energy E_4 was introduced in Section 5.3 and is shown again above. This reformulation yields

$$E_4^d = \sum_{i=1}^{N_{sd}} e_{4i} = 0,$$

where

$$e_{4i} = 1 - \frac{1}{2} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \frac{V_{ijk}(V_{ijk-1} + V_{ijk+1})}{L(ij) - 1} = 0,$$

and we define

$$V_{ijk-1} = \begin{cases} 0, & k = 1 \\ V_{ijk-1}, & \text{otherwise} \end{cases}, \quad V_{ijk+1} = \begin{cases} 0, & k = L(ij) \\ V_{ijk+1}, & \text{otherwise} \end{cases}.$$

The corresponding equation of motion term is

$$\frac{-\partial e_{4i}}{\partial V_{ijk}} = \frac{V_{ijk-1} + V_{ijk+1}}{2(L(ij) - 1)}.$$

To maintain a strictly-distributed model, the quantity e_{4i} may not be used to define fourth-constraint Lagrange multipliers because calculation of the energy e_{4i}, which would be needed to update the Lagrange multipliers, requires knowledge of the output voltages of all the neurons that represent links in paths between SD pair i. A constant coefficient for the fourth constraint retains the purely-distributed implementation. It also removes the equality constraint requirement, and, therefore, allows omission of the normalization which was required to obtain an equality constraint. Our experience has shown that this constraint, by itself, does not guarantee the formation of complete paths. Therefore, we have supplemented it with the following additional constraint.

5. *Adjacent links in a path shall all be activated or shall all be off:*

$$E_5^d = \sum_{k=1}^{N} e_{5k} = 0,$$

where N is the total number of neurons, and

$$e_{5k} = \begin{cases} \frac{1}{2}\left(\sum_{n=k-1}^{k+1} V_{ijn}\right)\cdot\left(3-\sum_{l=k-1}^{k+1} V_{ijl}\right) = 0, \; k \neq 1, L(ij) \\ \frac{1}{2}\left(\sum_{n=k}^{k\pm1} V_{ijn}\right)\cdot\left(2-\sum_{l=k}^{k\pm1} V_{ijl}\right) = 0, \; k = 1, L(ij) \end{cases}$$

If $k = 1$, the upper limit of the summations is $k + 1$; if $k = L(ij)$, the upper limit of the summations is $k - 1$. Thus, a Lagrange multiplier is defined for each neuron and, at each iteration, is increased by an amount proportional to the corresponding e_{5k}.

The corresponding equation of motion term is

$$\frac{-\partial e_{5k}}{\partial V_{ijk}} = \begin{cases} \sum_{n=k-1}^{k+1} V_{ijn} - \frac{3}{2}, \; k \neq 1, L(ij) \\ \sum_{n=k}^{k\pm1} V_{ijn} - 1, \; k = 1, L(ij) \end{cases}$$

This constraint has been added to the distributed model to encourage the formation of complete paths and discourage the activation of partial path segments. Variations of this constraint

form have been examined in conjunction with the centralized link-neuron model with little success. In that setting this "binary-forcing" constraint tends to lock the NN state into the legal state nearest to the initial random state, thus severely limiting the quality of the search of the state space. However, it has been necessary to incorporate this constraint into the distributed model because of the difficulty of ensuring the activation of complete paths.

Congestion-limiting (b-term) considerations

It has been necessary to change the congestion-energy term somewhat from that used in the centralized link-neuron model because the constraints in the distributed model do not capture as much of the desired behavior as those of the centralized model. In particular, in the centralized link-neuron model, the first constraint established inhibitory connections between all link (neuron) pairs that included links from different paths between the same SD pair; i.e., inhibitory connections were placed between all neuron pairs [*ijk,imn*], where $i = 1, \cdots, N_{sd}$; $j, m = 1 \cdots, N_p(i)$; $j \neq m$; $k = 1, \cdots, L(ij)$; $n = 1, \cdots, L(im)$. Therefore, inhibitory *b*-type connections were not installed between adjacent links in different paths between the same SD pair in the centralized model. However, in the distributed model it is appropriate to establish inhibitory *b*-type connections between these adjacent links because the distributed formulation of the first constraint applies inhibitory connections only between first links and between last links. Also in the distributed model, *b*-type connections are established between adjacent links in paths between different SD pairs in a manner similar to that used in the centralized model.

5.12.2 Simulation Results of the Distributed Link-Neuron Model

Simulations of the purely-distributed model have yielded extremely poor results. However, a significant performance improvement was obtained when the complete-path constraint formulation E_4'' of Section 5.11 was used instead of the distributed fourth constraint formulation E_4^d (Section 5.12.1). In fact, 3% of the solutions had the optimum congestion-energy value in simulations of this model. Therefore, the following discussions focus on the results of simulations of an almost-distributed NN model, which we call the QDI (Quasi-Distributed Implementation) model, that uses the distributed formulations of constraints 1, 3, and 5 as described in Section 5.12.1, and the "less-centralized" fourth constraint formulation E_4'' described in Section 5.11.

In a strictly-distributed setting, the use of the early termination criterion and the instantaneous state interpretation is precluded by the lack of global knowledge of the NN state. However, since a strictly-distributed implementation of the NN algorithm is impractical, and since

the QDI model is not a strictly-distributed model, we have used the early termination criterion in simulations of the QDI model and taken the instantaneous state interpretation of the results.

Results of Simulations of the QDI Model

Preliminary simulations of the QDI link-neuron model were performed to obtain a set of parameter values that performed reasonably well. Then simulations from 100 different initial states were run using the 24-node network of Figure 4.1 with the set of paths listed in Table B.1. Again, the initial states were the same as those used in the simulations of Sections 5.6 - 5.9. The parameter values that were used are listed in the following table.

$\lambda_0(0)$ $\lambda_2(0)$	$\lambda_1(0)$ $\lambda_3(0)$ $\lambda_5(0)$	$\lambda_4(0)$	$(\Delta t)_{\lambda 0}$ $(\Delta t)_{\lambda 2}$	$(\Delta t)_{\lambda 1}$ $(\Delta t)_{\lambda 3}$ $(\Delta t)_{\lambda 5}$	$(\Delta t)_{\lambda 4}$	Δt	b	I	N_c	ε
1.0	0.0	40.0	0.01	10^{-4}	1.0	10^{-4}	1.2	1.0	10^4	0.01

Here, $\lambda_0(0)$ denotes the initial value of all the first-constraint first-link Lagrange multipliers that correspond to the energies e^f_{1i}, $\lambda_1(0)$ is the initial value of the first-constraint last-link Lagrange multipliers that correspond to the energies e^l_{1i}, and $\lambda_2(0)$ and $\lambda_3(0)$ are the initial values of the third-constraint first- and last-link LM that correspond to the energies e^f_{3i} and e^l_{3i}, respectively. Again, N_c is the termination parameter that was defined in Section 5.5.

The results of the simulations are shown in Figure 5.11. The results obtained using the centralized model described in Section 5.9.1, which used the E_4' formulation of the complete-path constraint in conjunction with dummy neurons, are also shown. We refer to this centralized model as the CI-DN (Centralized Implementation-Dummy Neurons) model. The CI-DN model clearly provides the best results. Although the distributed model does not perform as well as the centralized one, it does find solutions with congestion energy $E_b^\ell \leq 147$, the energy of the best shortest-path heuristic solution, in more than 40% of the simulations. Given the significant information limitations of the distributed implementation, the performance degradation found in the distributed model is not unexpected. Furthermore, the achieved significant reduction in the NN's complexity, which resulted from the reduced number of neural connections, may make the performance degradation acceptable.

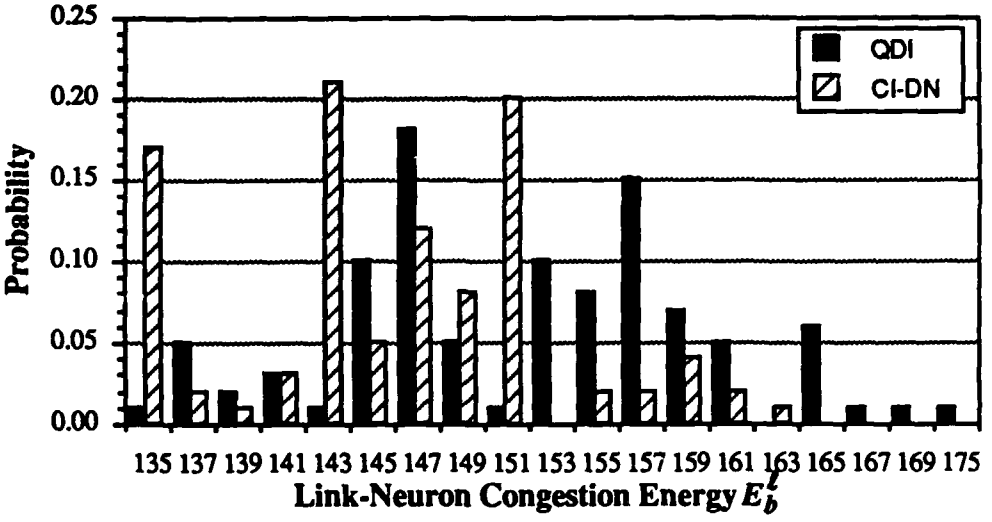


Figure 5.11. A comparison of the results of the QDI model to the results of the centralized link-neuron model CI-DN

Results of Simulations using a Larger Initial Value of λ_4

A second series of simulations of the QDI model were run using a larger initial value of the *complete-path constraint* Lagrange multiplier ($\lambda_4(0) = 200.0$) in conjunction with a smaller fourth Lagrange multiplier time constant ($(\Delta t)_{\lambda 4} = 0.01$). The simulations were run from the usual 100 different initial states using the following parameter values:

$\lambda_0(0)$ $\lambda_2(0)$	$\lambda_1(0)$ $\lambda_3(0)$ $\lambda_5(0)$	$\lambda_4(0)$	$(\Delta t)_{\lambda 0}$ $(\Delta t)_{\lambda 2}$ $(\Delta t)_{\lambda 4}$	$(\Delta t)_{\lambda 1}$ $(\Delta t)_{\lambda 3}$ $(\Delta t)_{\lambda 5}$	Δt	b	I	N_c	ε
1.0	0.0	200.0	0.01	10^{-4}	10^{-4}	1.2	12.5	5×10^3	0.01

The results of these simulations are compared with the CI-DN model simulation results in Figure 5.12. Although it is based on the availability of only limited information, the QDI model (using these parameters) has provided results that rival the best obtained from any of the link-neuron models. In fact, three of the solutions from this set of runs had the optimum congestion-energy value of 133, a result that actually surpasses the performance of the path-neuron model.[6] In Figure 5.13, the cumulative mass functions of the results of these simulations are compared with the CI-DN results from Section 5.9.1 as well as the solutions resulting from the use of a smaller initial

[6] Although optimal solutions were routinely found using the path-neuron model under the original (E_b) criterion, they were not found under the alternate (E_b') criterion, except in one run in which Gaussian simulated annealing was used, as discussed in Section 4.10.

value of λ_4 ($\lambda_4(0) = 40$) in the QDI model. The figure shows clearly that, when using the QDI model, better results are obtained through the use of the larger initial value of λ_4. With $\lambda_4(0) = 200$, 90% of the QDI model solutions had lower congestion energy values than the best shortest-path heuristic solution. The CI-DN model found solutions with congestion energy $E_b^\ell < 147$ in only 49% of the simulations.

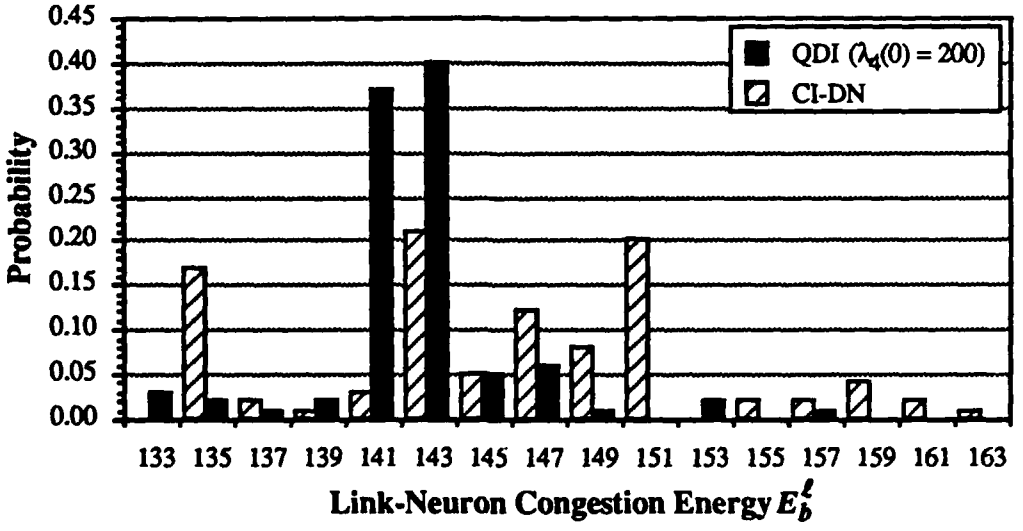


Figure 5.12. Histograms of the results of simulations of the QDI model

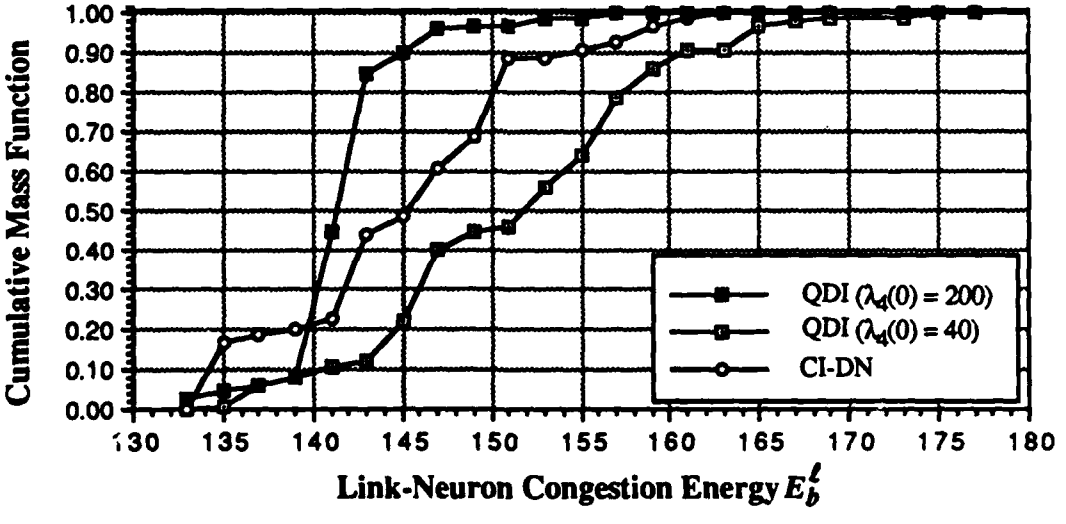


Figure 5.13. Cumulative mass functions of the results from the QDI model with two different values of $\lambda_4(0)$ and the CI-DN model

These simulations of the QDI model indicate that a nearly-distributed link-neuron NN model achieves a reduction in the number neuron interconnections, and hence a reduction in the number of required calculations at each iteration of the equations of motion, with little performance degradation. In fact, the additional degree of freedom obtained by emphasizing link interactions rather than path interactions has allowed the first discoveries (without simulated annealing) of optimal solutions for the 24-node network. However, it was found that the emphasis on link interactions must be somewhat tempered by a large initial value for $\lambda_4(0)$, which enforces the complete-path constraint. The large value of $\lambda_4(0)$, besides preventing the activation of largely-disjoint sets of links in different paths, provides a cohesion among the neurons (links) that form a path. This cohesion allows the congestion and constraint information to be propagated to all neurons corresponding to links on the same path by means of the link interactions on the path. Without such cohesion, both the increased congestion resulting from the activation of conflicting links and the constraint violations caused by the activation of intermediate links in different paths between a SD pair may go virtually unnoticed.

5.13 Conclusions on the Link-Neuron Model

In this section, we have presented a link-neuron NN model for the solution of 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 determine the coefficients in the connection weights dynamically. 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.

After considering a centralized link-neuron model, we investigated a distributed implementation. The practicality of a purely-distributed implementation is questionable for several reasons, which include the large quantity of information that must be exchanged among neighboring nodes, the inability to use the early termination criterion, and the inability to assess the quality of a solution without global knowledge of the states of the other neurons. However, our studies of a quasi-distributed implementation (QDI) have demonstrated that good performance can be achieved even when only incomplete information is made available to the neurons. The advantage of such a system is that fewer connections are needed, thus resulting in greater efficiency in software simulations.

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 may 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 the solution of 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 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

Although the issues of routing and scheduling in packet radio networks are highly interdependent, few studies have addressed them jointly. In this report, we briefly reviewed the major issues associated with the joint study of these problems, and we addressed the problem of routing for the minimization of congestion as a first step toward the solution of the joint routing-scheduling problem. We have posed this as a combinatorial-optimization problem, and we have demonstrated the ability of Hopfield neural network (NN) models to provide good solutions for it.

Much of this report has focused on the path-neuron NN model, in which a neuron is defined for each path between every SD pair in the network. The assumption that paths are predefined is often reasonable, and corresponds to the prespecification of virtual circuits that may be activated as needed. We have also developed and evaluated a link-neuron model, in which individual links, rather than complete paths, are the basic units for system modeling.

The Hopfield NN methodology is rather different from more-traditional algorithmic or simulation approaches to combinatorial-optimization problems. For example, this approach involves the embedding of 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 (which in our problem is directly related to a measure of congestion) and energy components that are related to the constraints that must be satisfied by valid solutions; these constraints enforce the condition that exactly one path is chosen between each SD pair.

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. To determine good values for these coefficients, we have used the method of Lagrange multipliers (LM), which permits the coefficients to vary dynamically along with the evolution of the system state. Most studies of Hopfield NNs have used trial-and-error methods to determine acceptable values for the connection weights, a process that is tedious at best, and often ineffective. We have demonstrated that, especially for the path-neuron model, the LM approach consistently produces high-quality solutions to fairly large problems. In fact, the use of the LM method provides better performance than that which is possible using a set of optimal constant connection weights. This is because the congestion-limiting component dominates the connection weights in the early stages of a simulation run (when the LMs are small), guiding the search toward a region of low congestion. As the run progresses, the LMs increase as necessary, and the impact of the constraints guides the solution toward a valid state with binary neuron values. Although some experimentation is often 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.

The analog nature of the Hopfield NN model is, in fact, a crucial feature of this approach. By permitting the search for an optimal solution to proceed through the interior of a continuous region, better solutions are usually found than are possible with strictly-digital processing elements. System evolution relaxes from some initial state to a final state that is a local minimum of the energy function. Only a local, rather than a global, minimum can generally be guaranteed because the system follows a trajectory of monotonically-decreasing energy. Since the evolution of the NN is deterministic,[1] the search of different portions of the region requires the use of different initial states. Typically, in our examples, we have simulated 100 runs from different initial conditions in our effort to find globally optimal solutions.

To assess the performance of our NN models, a benchmark is needed against which to compare them. In our smaller 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 solution 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. The largest example we have considered consists of the 100-node network with three units of traffic between each SD pair, which has resulted in a NN that consists of 981 neurons. Of the 20 simulation runs for that example, eight performed better than the best solution obtained by simply triplicating the best solution found for the unit-traffic example, thus demonstrating the benefits of alternate routing as well as the ability of our model to handle large, heavily-congested networks.

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

[1] Although we have studied the use of Gaussian simulated annealing (GSA) in conjunction with our NN model, the results obtained using that method have been rather inconsistent. Thus we limit our discussion here to the use of system models without GSA.

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 results demonstrate the effectiveness of our Hopfield path-neuron NN model for the minimization of 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 the development of NN models for the joint routing-scheduling problem. As a step toward this goal, we developed the link-neuron model, which also makes use of the LM technique.

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 applications to network problems. In particular, the ability of this model to generate complete paths by means of excitatory connections between neurons belonging to the same path may 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. It may also be viewed as a first step toward the solution of the joint routing-scheduling problem, in which the time slot for the activation of each individual link along every path must be determined. These problems are the subject of future research. We now comment briefly on the application of Hopfield NN models to scheduling problems.

6.1 Future Studies of Link Activation (Scheduling) and the Joint Routing-Scheduling Problem

In the basic link-scheduling problem, one-hop communication requirements between neighboring pairs of nodes are specified, and the goal is to determine a link-activation schedule that will satisfy this requirement in a minimum amount of time. In [2] this problem is posed as a decision problem; i.e., for a given number of slots k, determine whether a schedule can be found that satisfies the communication requirements, and if so, determine that schedule. In the NN formulation, several neurons are defined to correspond to each link as follows. First, one neuron is defined corresponding to every packet that must be transmitted over that link; each such neuron is then mapped into one neuron for every time slot over which the schedule is to be defined. For example, if a link must deliver three packets and a schedule of length k is being sought, $3k$ neurons are needed to specify the operation of that link. We are looking for stable states of the NN in which exactly one neuron corresponding to each packet is activated at some time during the

schedule. To achieve such minimum-energy states, inhibitory connections are placed between neurons that correspond to interfering links, between neurons that correspond to the same node transmitting two packet simultaneously, and between neurons that correspond to the same packet being transmitted in two different time slots. A reward is incorporated into the energy function for each link that is activated.

The link-activation problem formulation discussed in [2] is based on single-hop communication requirements. We are presently developing methods to extend this model to one in which communication requirements are specified between multihop SD pairs, rather than between neighboring nodes. When the routes are specified a priori, and when the order in which the links of a path are activated is deemed not important, this problem is equivalent to that of [2]. However, in some applications it is desirable to ensure that the links along a path are activated in a sequential manner, i.e., so that a link is not activated before it has received a packet from its upstream neighbor. Such a mechanism can be implemented in a straightforward manner by means of the incorporation of additional inhibitory connections that prevent such behavior. Ultimately, we would like to extend our model to the complete joint-scheduling problem, in which both the paths and the schedules are determined by the NN.

The main difficulty in implementing NN models for the joint routing-scheduling problem results from the number of neurons that are needed. Corresponding to each time slot, one neuron must be defined for each link of every path between every SD pair. Thus it will be difficult to solve this problem for very large networks. However, our high degree of success with NN models consisting of nearly 1000 neurons suggests that the simulation of considerably larger NNs may be feasible. We note that there is no objective function to be minimized in the scheduling and joint routing-scheduling problems. These problems consist solely of finding a state that satisfies the system constraints, i.e., a state in which a complete set of links is activated in a non-interfering manner. We expect that this aspect of the problem may permit the solution of larger problems than we have studied thus far because of the demonstrated ability of the LM method to produce solutions that satisfy system constraints.

We acknowledge that this discussion is far from complete. Its purpose is simply to indicate the flavor of our future research in Hopfield NN methods for routing and scheduling problems. Complete details of these NN models will be presented in a future report.

ACKNOWLEDGMENT

The authors acknowledge many productive discussions with Mr. W. Thoet of Booz-Allen & Hamilton. Also, Mr. Thoet wrote the program that finds sets of maximally node-disjoint paths, which was discussed in Section 4.1.

REFERENCES

1. Wieselthier, J. E., "Code-Division Multiple-Access Techniques and Their Application to the High-Frequency (HF) Intratask Force (ITF) Communication Network," NRL Report 9094, Naval Research Laboratory, Washington, D. C., September 1988.

2. Tassiulas, L., 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.

3. Special Issue on "Neural Networks in Communication," *IEEE Communications Magazine*, 27, November 1989.

4. Special Issue on "Neural Networks in Control Systems," *IEEE Control Systems Magazine*, 10, April 1990.

5. Rauch, H. E. and T. Winarske, "Neural Networks for Routing Communication Traffic," *IEEE Control Systems Magazine*, 8 pp. 26-31, 1988.

6. Tanenbaum, A. S., *Computer Networks, Second Edition*, Englewood Cliffs, NJ: Prentice Hall, 1988.

7. Arikan, E., "Some Complexity Results About Packet Radio Networks," *IEEE Transactions on Information Theory*, IT-30 pp. 681-685, July 1984.

8. Hajek, B. and G. Sasaki, "Link Scheduling in Polynomial Time," *IEEE Transactions on Information Theory*, 34 pp. 910-917, September 1988.

9. Tassiulas, L., "Scheduling Problems in Multihop-Packet Radio Networks," Master's Thesis, University of Maryland, 1989.

10. Tassiulas, L. and A. Ephremides, "An Algorithm for Joint Routing and Scheduling in Radio Networks," *Proceedings of the American Control Conference*, pp. 697-702, June 1989.

11. Hopfield, J. J. and D. W. Tank, "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics*, 52 pp. 141-152, 1985.

12. Ephremides, A. and T. V. Truong, "Scheduling Broadcasts in Multihop Radio Networks," *IEEE Transactions on Communications*, 38 pp. 456-460, April 1990.

13. Bertsekas, D. and R. Gallager, *Data Networks*, Englewood Cliffs: Prentice Hall, pp. 322-324, 1987.

14. Sedgewick, R., *Algorithms*, Addison-Wesley Publishing Co., pp. 461-465, 1988.

15. Van den Bout, D. E. and T. K. Miller III, "Graph Partitioning Using Annealed Neural Networks," *IEEE Transactions on Neural Networks*, **1** pp. 192-203, June 1990.

16. Wacholder, E., J. Han, and R. C. Mann, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem," *Biological Cybernetics*, **61** pp. 11-19, 1989.

17. Akiyama, Y., A. Yamashita, M. Kajiura, and H. Aiso, "Combinatorial Optimization with Gaussian Machines," *Proceedings of the International Joint Conference on Neural Networks*, pp. I-533 - I-540, 1989.

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 which 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.[1] The generic combinatorial-optimization problem that is solved by a Hopfield NN consists of determining which of the neurons should be "on" (i.e., have a value of 1) and which should be "off" (i.e., 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

[1] Other types of NNs are well-suited for learning applications, including a wide range of pattern-recognition tasks. General references on NNs include [A2] and [A3].

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

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

as shown in Fig. A-1. 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_o 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.

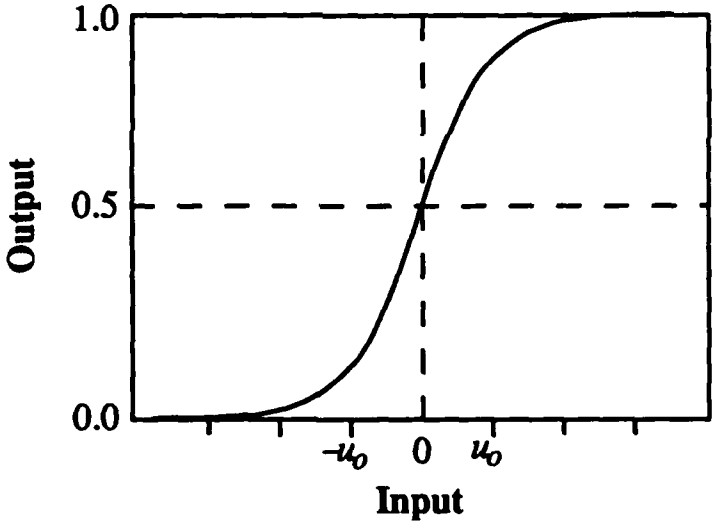


Figure A-1. Input-output nonlinearity

A portion of a Hopfield NN is shown in Fig. A-2. 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 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 may 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 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.

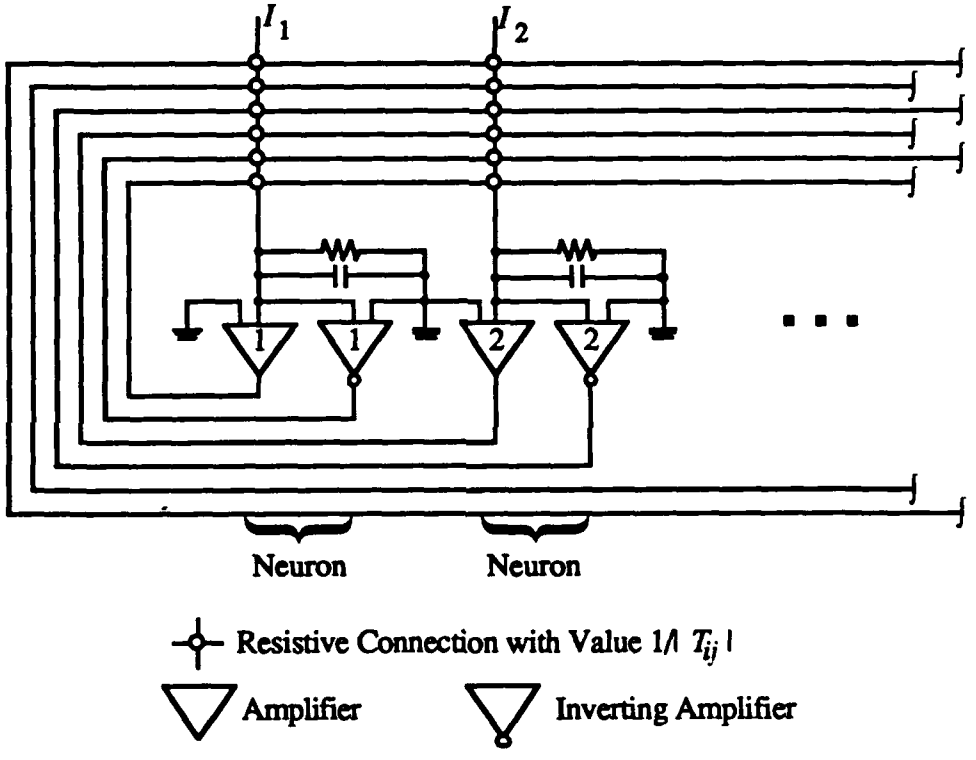


Figure A-2. Portion of a Hopfield NN

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., $T_{ij} = T_{ji}$), and provided that $T_{ii} = 0$ [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}), while 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.

Clearly, the form of the energy function presented above is not completely general, and it is not possible to define a Hopfield net that corresponds to all possible 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 the minimization of congestion in networks.

The evolution of the input voltage at each neuron is characterized by an equation of motion that is obtained by differentiating the energy function with respect to the output voltage at that neuron. Thus

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

where $\tau = RC$ 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 ($1 \le i,j < n$; $i \ne j$). 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 (since the same tour can begin in any city and because reversal of 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, a valid 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} is used to represent 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 [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 Xi is:

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

Recall that $V_{Xi} = 1$ means that city X is visited in the i^{th} position of the tour. Thus, a negative value of du_{Xi}/dt tends to turn neuron Xi 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:

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

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 [A1], but 10^{-5} appears to be a reasonable value for use with these system parameters (see e.g., [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_{Xj}'s are nonzero (which correspond to visiting city *X* in position *j*), the input voltage to neuron *Xi* 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_{Yi}'s are nonzero (which correspond to visiting city *Y* in position *i*), the input voltage to neuron *Xi* 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 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 *Xi*, 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 valid 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 ten-city problem.

An initial state must be chosen as the starting point for the iteration. Although we are searching for a valid 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_{oo}, so that no tour would be preferred above any other a priori. For the ten-city problem u_{oo} 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 $-0.1u_o$ and $+0.1u_o$, 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 valid), 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.[2]

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 valid 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 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_o) confines the search to a region near the edges of the hypercube, and may thus prevent the finding of the best 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

[2] 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.

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 valid 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; invalid 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 and 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 $u_o = 0.02$) and the time step size Δt (the value of which was not specified by Hopfield and Tank).

The choice of system parameters by trial and error is a tedious process. Moreover, convergence to valid 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 valid 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. For example, Aiyer, Niranjan, and Fallside [A12] 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 valid 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 this paper seem to 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 along 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 a valid 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 follows:

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

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 ($\pm$ 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 penalize more-heavily 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 a valid 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 valid 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 the implementation of 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. In Sections 4 and 5 we discuss a number of issues that we have encountered in the application of this method to our problems. For example, the time constant $(\Delta t)_\lambda$ used for the iteration of the Lagrange multiplier values should typically be greater than the value of Δt used for the evolution of system state. Also, performance is somewhat sensitive to the coefficient in the energy equation that represents system performance (the d in the discussion of this appendix) and to the additional bias currents that are added (corresponding to Hopfield and Tank's use of $n = 15$ for a problem with 10 cities). However, 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.[3] Simulated annealing (SA) [A15, A16, 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.

[3] 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.

The terminology of simulated annealing is based on an analogy with the physical annealing process, under 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 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. A-1, 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_o) 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. Detailed discussions of the specifics of our model and our results are presented in Sections 3, 4, and 5.

REFERENCES

A1. Hopfield, J. J. and D. W. Tank, "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics*, **52** pp. 141-152, 1985.

A2. Wasserman, P. D., *Neural Computing: Theory and Practice*, New York: Van Nostrand Reinhold, 1989.

A3. Simpson, P. K., *Artificial Neural Systems*, New York: Pergamon Press, 1990.

A4. Ramanujam, J. and P. Sadayappan, "Optimization by Neural Networks," *Proceedings of the IEEE International Conference on Neural Networks*, pp. II-325 - II-332, July 1988.

A5. Brandt, R. D., 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. Foo, Y. S. and Y. Takefuji, "Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 1. Problem Representation," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, pp. II-275 - II-282, July 24-27, 1988.

A7. Foo, Y. S. 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*, San Diego, CA, pp. II-283 - II-290, July 24-27, 1988.

A8. Lawler, E. L., J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, ed., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Chichester: John Wiley & Sons, 1985.

A9. Wilson, G. W. and G. S. Pawley, "On the Stability of the Traveling Salesman Problem Algorithm of Hopfield and Tank," *Biological Cybernetics*, **58** pp. 63-70, 1988.

A10. Hegde, S. U., 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. Kahng, A. B., "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. Aiyer, S. V. B., M. Niranjan, and F. Fallside, "A Theoretical Investigation into the Performance of the Hopfield Model," *IEEE Transactions on Neural Networks*, **1** pp. 204-215, June 1990.

A13. Wacholder, E., J. Han, and R. C. Mann, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem," *Biological Cybernetics*, **61** pp. 11-19, 1989.

A14. Platt, J. C. and A. H. Barr, "Constrained Differential Optimization," *Proceedings of the IEEE 1987 Neural Information Processing Systems Conference*, 1987.

A15. van Laarhoven, P. J. M. and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, Dordrecht: D. Reidel Publishing Company, 1987.

A16. Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, **220** pp. 671-680, May 13, 1983.

A17. Hajek, B., "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. Hinton, G. E. and T. J. Sejnowski, "Learning and Relearning in Boltzmann Machines," in *Parallel Distributed Processing*, ed. D. E. Rumelhart and J. L. McClelland, pp. 282-317, Cambridge, MA: MIT Press, 1986.

A19. Szu, H. H. and R. L. Hartley, "Nonconvex Optimization by Fast Simulated Annealing," *Proceedings of the IEEE*, **75** pp. 1538-1540, November 1987.

A20. Levy, B. C. and M. B. Adams, "Global Optimization with Stochastic Neural Networks," *IEEE International Conference on Neural Networks*, pp. III-681 - III-689, 1987.

A21. Akiyama, Y., A. Yamashita, M. Kajiura, and H. Aiso, "Combinatorial Optimization with Gaussian Machines," *Proceedings of the International Joint Conference on Neural Networks*, pp. I-533 - I-540, 1989.

A22. Van den Bout, D. E. and T. K. Miller III, "Graph Partitioning Using Annealed Neural Networks," *IEEE Transactions on Neural Networks*, **1** pp. 192-203, June 1990.

APPENDIX B

TABLES OF THE PATHS AND SD PAIRS ASSOCIATED WITH THE NETWORKS OF FIGURES 4.1 AND 4.10

Table B.1. Listing of paths for the 24-node network of Fig. 4.1

SD Pair 1 [4,24]									
Path	Nodes Traversed								
0, $P_{1,1}$	4	5	13	20	24				
1, $P_{1,2}$	4	7	14	19	20	24			
2, $P_{1,3}$	4	3	6	11	12	13	20	24	
3, $P_{1,4}$	4	7	13	20	24				
4, $P_{1,5}$	4	5	14	19	20	24			
5, $P_{1,6}$	4	3	6	8	9	12	13	20	24
SD Pair 2 [7,17]									
Path	Nodes Traversed								
6, $P_{2,1}$	7	14	15	17					
7, $P_{2,2}$	7	11	12	13	19	14	15	17	
SD Pair 3 [9,16]									
Path	Nodes Traversed								
8, $P_{3,1}$	9	11	7	14	15	16			
9, $P_{3,2}$	9	8	6	5	14	15	16		
10, $P_{3,3}$	9	12	13	19	14	15	16		
11, $P_{3,4}$	9	12	7	14	15	16			
SD Pair 4 [1,19]									
Path	Nodes Traversed								
12, $P_{4,1}$	1	4	5	13	19				
13, $P_{4,2}$	1	2	3	6	7	14	19		
14, $P_{4,3}$	1	4	7	14	19				
15, $P_{4,4}$	1	2	3	6	11	12	13	19	
16, $P_{4,5}$	1	4	5	14	19				
17, $P_{4,6}$	1	4	7	13	19				
18, $P_{4,7}$	1	2	3	6	8	9	12	13	19
19, $P_{4,8}$	1	2	3	6	5	14	19		
SD Pair 5 [5,11]									
Path	Nodes Traversed								
20, $P_{5,1}$	5	6	11						
21, $P_{5,2}$	5	7	11						
22, $P_{5,3}$	5	13	12	11					

Table B.1. Listing of paths for the 24-node network of Fig. 4.1 (continued)

SD Pair 6 [21,6]										
Path	Nodes Traversed									
23, $P_{6,1}$	21	22	20	13	5	6				
24, $P_{6,2}$	21	24	20	19	14	7	6			
25, $P_{6,3}$	21	22	20	13	12	11	6			
26, $P_{6,4}$	21	24	20	13	5	6				
27, $P_{6,5}$	21	22	20	19	14	7	6			
28, $P_{6,6}$	21	24	20	13	12	9	8	6		
29, $P_{6,7}$	21	24	20	13	7	6				
30, $P_{6,8}$	21	22	20	19	14	5	6			
31, $P_{6,9}$	21	24	20	13	12	11	6			
32, $P_{6,10}$	21	22	20	13	7	6				
33, $P_{6,11}$	21	24	20	19	14	5	6			
34, $P_{6,12}$	21	22	20	13	12	9	8	6		

SD Pair 7 [1,10]										
Path	Nodes Traversed									
35, $P_{7,1}$	1	4	7	11	10					
36, $P_{7,2}$	1	2	3	6	8	9	10			
37, $P_{7,3}$	1	4	5	13	12	9	10			
38, $P_{7,4}$	1	2	3	6	11	10				

SD Pair 8 [3,18]										
Path	Nodes Traversed									
39, $P_{8,1}$	3	4	5	14	15	18				
40, $P_{8,2}$	3	6	7	14	15	18				
41, $P_{8,3}$	3	6	11	12	13	19	14	15	18	
42, $P_{8,4}$	3	4	7	14	15	18				
43, $P_{8,5}$	3	6	5	14	15	18				
44, $P_{8,6}$	3	6	8	9	12	13	19	14	15	18

SD Pair 9 [2,12]						
Path	Nodes Traversed					
45, $P_{9,1}$	2	4	7	12		
46, $P_{9,2}$	2	3	6	11	12	
47, $P_{9,3}$	2	4	5	13	12	
48, $P_{9,4}$	2	3	6	8	9	12

SD Pair 10 [14,8]						
Path	Nodes Traversed					
49, $P_{10,1}$	14	5	6	8		
50, $P_{10,2}$	14	7	11	8		
51, $P_{10,3}$	14	19	13	12	9	8

Table B.2. A listing of SD pairs, number of paths, and traffic associated with the 100-node network of Fig. 4.10.

	$N_p(i)$	Nonuniform Traffic	
		Loading 1	Loading 2
SD pair 1: [7,48]	6	3	1
SD pair 2: [45,42]	8	4	4
SD pair 3: [95,24]	7	1	3
SD pair 4: [45,58]	5	2	1
SD pair 5: [27,60]	8	3	1
SD pair 6: [61,18]	6	4	3
SD pair 7: [87,8]	10	1	2
SD pair 8: [97,2]	8	2	3
SD pair 9: [19,80]	7	3	4
SD pair 10: [61,6]	9	4	3
SD pair 11: [87,84]	10	1	1
SD pair 12: [33,38]	9	2	4
SD pair 13: [35,96]	9	3	1
SD pair 14: [41,74]	10	4	3
SD pair 15: [67,72]	7	1	4
SD pair 16: [1,70]	10	2	1
SD pair 17: [31,4]	9	3	1
SD pair 18: [57,78]	9	4	4
SD pair 19: [87,40]	9	1	3
SD pair 20: [49,58]	9	2	1
SD pair 21: [3,100]	10	3	4
SD pair 22: [5,75]	9	4	4
SD pair 23: [9,73]	9	1	4
SD pair 24: [10,71]	10	2	3
SD pair 25: [11,69]	7	3	1
SD pair 26: [12,13]	9	4	4
SD pair 27: [14,63]	7	1	3
SD pair 28: [62,37]	5	2	4
SD pair 29: [23,51]	9	3	4
SD pair 30: [36,55]	9	4	3
SD pair 31: [15,60]	7	1	1
SD pair 32: [20,28]	10	2	1
SD pair 33: [30,17]	8	3	4
SD pair 34: [16,54]	9	4	4
SD pair 35: [21,39]	10	1	3
SD pair 36: [53,32]	7	2	3
SD pair 37: [34,50]	9	3	1
SD pair 38: [22,24]	8	4	1
SD pair 39: [25,52]	9	1	4
SD pair 40: [26,29]	1	2	1