



NRL/FR/5520--00-9922

# Throughput Maximization Under Quality of Service Constraints:

## Determination of Optimal Offered Load in Circuit-Switched (Wireless or Nonwireless) Communication Networks

JEFFREY E. WIESELTHIER  
GAM D. NGUYEN

*Communication Systems Branch  
Information Technology Division*

ANTHONY EPHREMIDES

*Pontos, Inc.  
North Bethesda, Maryland  
and  
University of Maryland  
College Park, Maryland*

June 26, 2000

The performance results presented in this report are best understood when viewed in color. A pdf file that includes color versions of all figures is included with this report on a CD-ROM.

Approved for public release; distribution is unlimited.

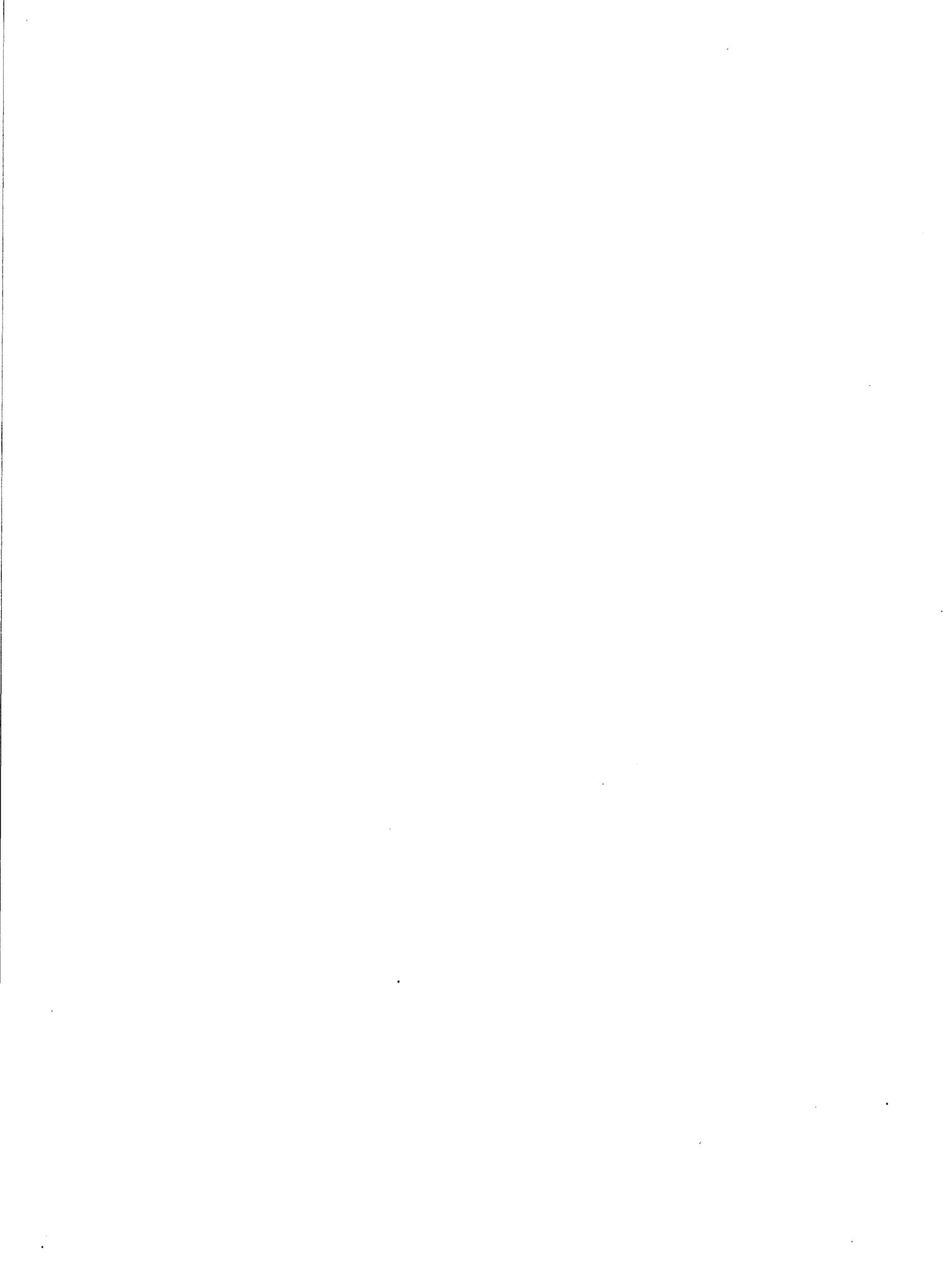


# REPORT DOCUMENTATION PAGE

*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  June 26, 2000	3. REPORT TYPE AND DATES COVERED  Final Report	
4. TITLE AND SUBTITLE  Throughput Maximization Under Quality of Service Constraints: Determination of Optimal Offered Load in Circuit-Switched (Wireless or Nonwireless) Communication Networks		5. FUNDING NUMBERS  PE - 61153N PR - RR015-09-41 WU-DN159-036	
6. AUTHOR(S)  Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides*		8. PERFORMING ORGANIZATION REPORT NUMBER  NRL/FR/5520--00-9922	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Naval Research Laboratory Washington, DC 20375-5320			
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Office of Naval Research 800 North Quincy Street Arlington, VA 22217-5660		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES  *Pontos, Inc., North Bethesda, Maryland, and University of Maryland, College Park, Maryland			
12a. DISTRIBUTION/AVAILABILITY STATEMENT  Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)  We consider the determination of the offered load that maximizes throughput in circuit-switched multihop networks, subject to quality of service (QoS) constraints on circuit blocking probability. This problem is of interest in network design for "sizing" the service capabilities that can be provided, and thereby providing a measure of "network capacity." Lagrangian techniques are used to formulate this nonlinear optimization problem, which incorporates nonlinear inequality constraints. We then describe a heuristic technique that guides the search more directly toward the optimal solution, thereby resulting in faster and more-reliable convergence.  We show that the degree of increase in throughput obtainable when the optimal values of offered load are used varies greatly among the examples we have studied, and can be dramatic in some cases. We demonstrate the effectiveness and robustness of our approach by comparing 18 versions of our algorithm for several network examples. Finally, the relationships between admission control and optimization of offered load are discussed.  <b>The performance results presented in this report are best understood when viewed in color. A pdf file that includes color versions of all figures is included with this report on a CD-ROM.</b>			
14. SUBJECT TERMS  Communications network    Lagrangian techniques Quality of service         Optimization		15. NUMBER OF PAGES  106	16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT  UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE  UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT  UNCLASSIFIED	20. LIMITATION OF ABSTRACT  UL



## CONTENTS

1.	INTRODUCTION .....	1
1.1	Outline .....	2
2.	CIRCUIT-SWITCHED NETWORK MODEL .....	3
2.1	Control Policy .....	5
2.2	Solution and Performance Measures .....	7
2.3	Computational Issues in Evaluating the Solution .....	8
3.	ERLANG CAPACITIES .....	9
4.	THE OPTIMIZATION PROBLEM .....	9
4.1	The Constrained Optimization Problem and the Basic Lagrangian Optimization Method .....	10
4.2	Stepsize Considerations .....	12
5.	EXAMPLE OF RESULTS BASED ON THE BASIC SEARCH PROCEDURE .....	12
6.	GUIDED SEARCH TECHNIQUES .....	16
6.1	Preliminary Approach .....	17
6.2	Generalized Approach .....	19
6.3	On the Development of Improved Iterative Search Algorithms .....	21
7.	ALTERNATIVE VERSIONS OF THE ALGORITHM .....	22
7.1	Projection Rules .....	23
7.2	Stepsize Rules .....	25
7.3	The Update Equation .....	27
7.4	Summary of Algorithm Versions Used in the Core Runs .....	27
8.	TESTING THE ALGORITHM: THE CORE RUNS .....	28
8.1	Description of the Core Runs .....	28
8.2	Milestone Tables .....	30
8.3	Stopping-Rule Tables .....	34

8.4	Comparison of the Solutions Produced by the Different Versions of the Algorithm .....	37
8.5	Search Trajectory .....	40
8.6	Observations Based on Core Runs .....	53
9.	FURTHER TESTING OF THE ALGORITHM .....	54
9.1	Nonzero Values of $\lambda_{\min}$ .....	55
9.2	Nonuniform Admission-Control Thresholds or Distribution of Transceivers .....	56
9.3	Nonuniform QoS Requirements .....	58
10.	AN ALTERNATIVE QOS CONSTRAINT: AVERAGE BLOCKING PROBABILITY .....	62
10.1	Mathematical Model .....	62
10.2	Performance Results .....	63
10.3	Combined Use of Average and Individual QoS Constraints .....	67
11.	THROUGHPUT GAIN FROM NETWORK OPTIMIZATION .....	73
11.1	Core Runs .....	73
11.2	Networks 1, 2, and 3 with Different QoS Values .....	73
11.3	Unequal Number of Transceivers .....	75
11.4	The Multicross Network .....	75
11.5	Discussion .....	76
12.	MISCELLANEOUS PERFORMANCE CONSIDERATIONS .....	76
12.1	The Impact of "Underutilized Circuits" on Performance .....	77
12.2	Fairness Considerations: A Guaranteed Offered Load on Each Circuit .....	77
12.3	An Upper-Bounded Offered Load on Each Circuit .....	79
13.	RELATIONSHIPS BETWEEN ADMISSION CONTROL AND OPTIMIZATION OF OFFERED LOAD .....	80
13.1	Discussion .....	84
14.	SUMMARY AND CONCLUSIONS .....	85
	ACKNOWLEDGMENTS .....	86
	REFERENCES .....	86
	APPENDIX A — Tables for the Core Runs .....	89
	APPENDIX B — Tables for Runs with $\lambda_{\min} > 0$ .....	99

## **THROUGHPUT MAXIMIZATION UNDER QUALITY OF SERVICE CONSTRAINTS:**

### **Determination of Optimal Offered Load in Circuit-Switched (Wireless or Nonwireless) Communication Networks**

#### **1. INTRODUCTION**

The throughput that can be supported by a communication network depends on many factors. These include switching mode (e.g., circuit- vs packet-switched), topology, channel access, routing schemes, and offered loads. Typically, in network optimization problems, several of these factors are maintained fixed, while others (either values of network parameters or the choice of network control policies) are varied to achieve optimal performance. These problems are considerably more difficult when quality-of-service (QoS) constraints must be satisfied, as they must be in many practical military and commercial applications.

In this report, we focus on determination of the offered load that provides optimal performance, i.e., maximum throughput, for the case in which both the routing and admission-control policy are fixed and the channel-access mechanism is frequency-division multiple access (FDMA). This problem entails determining the load that each circuit should offer so that overall network throughput is maximized, subject to a constraint on blocking probability. This problem is of interest for “sizing” the network capability (and thereby providing a measure of “network capacity”) because it is generally difficult to estimate the traffic loads that a network can support or the resulting throughput. The mathematical formulation is that of a constrained optimization problem with nonlinear objective and constraint functions. This is a difficult problem for which the available mathematical theory provides the basic principles for solution but no guarantee of convergence to the optimal point.

Voice and other connection-oriented services in communication networks are characterized by the requirement that delay be not only short, but also of low variance. This requirement motivates the customary approach of establishing circuit-switched paths (i.e., either actual or virtual circuits) between communicating nodes for the duration of each voice call or session. In the present study, we are interested in the case of networks in which the source and destination nodes are not generally within direct communication range; thus, relaying over multihop paths may be required. Although all of our examples represent the case of wireless networks, our formulation is also applicable to wired networks. Furthermore, unlike the case of commercial cellular communication systems or similar cellular architectures, fixed relays are not available in the networks studied here. Therefore, we assume that each of the nodes can assume the role of a relay as well as that of a source or destination. This type of network is sometimes referred to as “ad hoc” or “peer-to-peer” to distinguish it from the more-common structured fixed-relay systems, such as commercial telephone networks that are based on cellular architectures.

A principal aspect of network control concerns the admission and establishment of new sessions so that the network throughput is maximized, subject to satisfaction of various QoS constraints. A comprehensive approach to network optimization and control would address jointly the highly interdependent issues of admission control, routing, offered load, and channel access. However, each of these individual problems is extremely complex, making a complete solution of the joint problem unattainable. Therefore, it is necessary

to simplify the problem by addressing these issues separately in a manner that will, hopefully, lead to useful insights for their joint solution. For example, Refs. 1 and 2 address the use of admission-control policies for circuit-switched networks with fixed routing schemes and fixed offered loads. The need to address the link-activation scheduling problem was eliminated by assuming the use of channelization in the frequency domain by means of FDMA. Alternatively, one could assume channelization in the time domain (by means of time-division multiple access—TDMA) or code domain (by means of code-division multiple access—CDMA) with ensuing differences in analysis and performance.

We do not focus on a detailed description of network traffic [such as ATM (asynchronous transfer mode)-based QoS characterizations]. Instead, we present a more generic formulation that is amenable to analysis and optimization. Specifically, we assume a circuit-switched, fixed-route network with independent Poisson arrivals of session-establishment requests over each of the fixed routes, and with a fixed admission-control policy. The objective is to determine the input traffic levels that maximize throughput, subject to the constraint that blocking probability stays below a given level. We present several versions of an iterative algorithm for the optimization of offered load, subject to QoS constraints. Our approach is based on the use of Lagrangian optimization techniques that have been enhanced by means of heuristics that improve the reliability and quality of convergence. References 3-6 provide preliminary versions of this algorithm, including several heuristics to improve performance.

Extensive performance results are provided, demonstrating the capability of the various versions of the algorithm to provide convergence to nearly optimal solutions in almost all cases. Robust performance has been observed in which the same parameter values work well for a wide variety of network examples. For improved performance, the algorithm has built-in adaptivity features that permit some parameters to vary.

Two versions of the QoS constraint are considered. In the first, we require that blocking probability of each circuit does not exceed a specified QoS value. In the second, we require only that the average blocking probability in the network satisfies such a constraint. We have observed that use of the average form of the QoS constraint results in dramatically faster convergence than the case in which each individual circuit must satisfy the constraint. The faster convergence apparently results because (as a function of the individual input rates) the average blocking probability is much smoother than the maximum blocking probability. We have also observed that the use of the average QoS constraint seems to provide a good initial condition for a search based on the satisfaction of hard QoS constraints on all circuits, thus potentially serving as an alternative (and possibly faster) heuristic for this problem. We hypothesize the conditions under which such an approach may be appropriate. Of course, the network designer/manager must decide whether the satisfaction of QoS on an average basis is sufficient, or whether each circuit must be guaranteed the specified level of QoS performance. The interplay between these QoS philosophies is related to issues of pricing of services in commercial networks, which have been receiving increased attention lately and that may have a counterpart in military networks.

## 1.1 Outline

In Section 2 we discuss the circuit-switched, multihop, wireless network model, including the use of coordinate-convex admission-control policies and the resulting product-form solution. In Section 3 we discuss the notion of Erlang capacity, which is the region of offered load that a network can accommodate, subject to the satisfaction of specific QoS constraints. This discussion leads to the formulation of the constrained optimization problem and the basic iterative Lagrangian optimization method in Section 4. In this formulation, the blocking probability on each circuit is required to satisfy the QoS constraint.

Preliminary studies of the iterative optimization procedure, provided in Section 5, serve as the basis for the development of heuristic improvements, which are discussed in Section 6. In Section 7, several versions of stepsize and projection rules are discussed, and 18 candidate versions of the search algorithm are defined. The primary differences among the versions relate to the choice of stepsize rule and to the use of various forms of a “projection” technique that guides the search along a favorable trajectory.

The remainder of the report discusses the testing of the different versions of the algorithm, as well as the performance improvement that is achieved through their use. In Section 8, we discuss the results of a series of “core runs,” in which the 18 versions of the algorithm were tested on three different networks with uniform parameter sets. We compare the performance achieved under the various versions of the algorithm, demonstrating that all versions worked well, but not equally well, i.e., some appeared to be faster and/or more reliable than others. Further testing of the algorithm is discussed in Section 9, where we discuss the impact of nonuniform parameter values on algorithm performance.

In Section 10, we discuss an alternative form of the QoS constraint, in which only the average blocking probability (rather than each individual circuit blocking probability) is constrained. Convergence to the optimal point is shown to be much faster in this case.

In Section 11, we return to the first form of the QoS constraint (in which the QoS constraint must be satisfied on each circuit), and address the degree of improvement that can be achieved by means of the optimization of offered load. We demonstrate that dramatic improvement can, in fact, be achieved, primarily in asymmetrical networks whose topological structure or resource parameters are highly nonuniform. In Section 12, we discuss miscellaneous performance issues such as fairness, and in Section 13, we discuss relationships between admission control and the optimization of offered load.

Finally, in Section 14, we present our summary and conclusions from this study.

## 2. CIRCUIT-SWITCHED NETWORK MODEL

We consider a wireless network in which FDMA is used to provide contention-free channel access to a multihop circuit over a predetermined path between the source and destination nodes throughout the duration of each accepted voice call. Network topology can be described in terms of the communication resources available at each node and the connectivities between nodes. We assume that unless voice calls are accepted for immediate transmission (in practical terms, this may mean within several hundred milliseconds of their arrival), they are “blocked” and lost from the system; this mode of operation is generally referred to as “blocked calls cleared.” Appropriate performance measures for this mode of operation include blocking probability and throughput.

The number of nodes in the network is denoted by  $N$ , and the number of transceivers at node  $i$  is denoted by  $T_i$ . We consider  $J$  source-destination pairs, each of which is assigned a fixed multihop path (circuit) through the graph of the network that interconnects them. We refer to  $J$  as the “number of circuits” in the network. The establishment and maintenance of each ongoing call requires the use of one transceiver at each node along the path;<sup>1</sup> a call is blocked either if a transceiver is not available at one or more nodes along the path or if an admission-control policy decision is made to block the call. Transceivers are not assigned a priori to particular circuits. We let  $x_j$  denote the number of ongoing sessions (e.g., voice-calls) on the  $j$ th such circuit (i.e., on the path between the  $j$ th source-destination pair). Since it is assumed that each accepted voice call consumes a fixed amount of resource (i.e., one transceiver) at each node in its path, our traffic model is an example of constant bit rate (CBR) traffic sources (using the terminology associated with broadband networks). The state of the system is  $\mathbf{x} = (x_1, x_2, \dots, x_J)$ .

To illustrate our problem formulation with a specific example, we consider the simple seven-node star-network shown in Fig. 1. Nodes 2-7 are each connected only to node 1, thus necessitating the use of node 1 as a relay in all multihop circuits. We consider the three source-destination pairs (2,5), (3,6), and (4,7),

<sup>1</sup>Other models are certainly possible. For example, if traffic is one way rather than interactive, the source node would not have to dedicate a receiver to support the call. Similarly, receive-only nodes would not need transmitters. The approach presented here can be modified straightforwardly to accommodate variations such as these. The implicit assumption that the number of transmitters and receivers at a node are equal, which lets us describe the node in terms of a single parameter  $T_i$ , is used here because of convenience of notation, and can easily be relaxed.

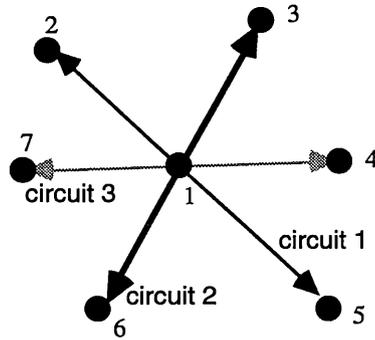
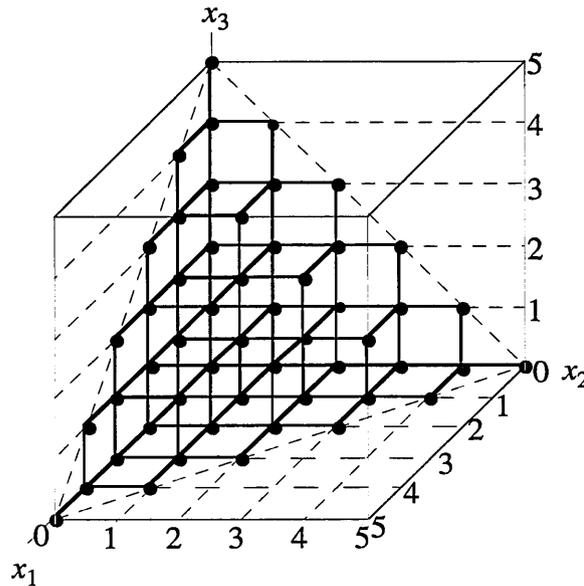


Fig. 1 — A simple three-circuit network

which correspond to circuits 1, 2, and 3, respectively. The state of the system is the three-dimensional vector  $\mathbf{x} = (x_1, x_2, x_3)$ . Although this example is simple in that it is small and has a centrally located node that is connected to all other nodes, this is not a requirement of the model studied in this report. Arbitrary topologies can be modeled, such as those shown in Sections 5 and 8; nodes are not designated a priori as base stations or as relay nodes, and no distinction is made between uplink and downlink channels.

In an uncontrolled system, the system operates in a mode known as “complete sharing” in which a call is accepted as long as there are sufficient resources at all nodes along the multihop path; if node  $i$  has  $T_i$  transceivers, it can support up to  $T_i$  simultaneous calls. We refer to the limits imposed by the values of  $T_i$  as the “capacity constraints.” For example, if each of the nodes in the network of Fig. 1 has five transceivers, the resulting uncontrolled system state space  $\Omega$  is as shown in Fig. 2. Equivalently, the capacity constraints expressed as inequalities describe the system state space:

$$\begin{aligned} x_1 \leq 5; \quad x_2 \leq 5; \quad x_3 \leq 5; \quad & \text{(i.e., no more than five calls may be accepted on any circuit)} \\ x_1 + x_2 + x_3 \leq 5. \quad & \text{(the hub, node 1, can handle no more than five calls).} \end{aligned}$$

Fig. 2 — The admissible state space  $\Omega$  for the network of Fig. 1

In general, a vector description of circuit  $j$  in terms of the nodes it traverses is given by

$$c_j = (c_{j1}, c_{j2}, c_{j3}, \dots, c_{jN}),$$

where

$$c_{ji} = \begin{cases} 1, & \text{if circuit } j \text{ traverses node } i \\ 0, & \text{otherwise} \end{cases},$$

and  $N$  is the number of nodes in the network. For example, circuit 1 in Fig. 1 is represented by

$$c_1 = (1, 1, 0, 0, 1, 0, 0).$$

Now we can express the capacity constraints as

$$\sum_{j=1}^J x_j c_{ji} \leq T_i, \quad i = 1, \dots, N \quad (1)$$

where  $J$  is the number of circuits in the network.

A similar equation can be written for wired networks, in which case  $c_{ji}$  would take on the value of 1 if and only if circuit  $j$  traverses link (rather than node)  $i$ . The capacity constraints would again be written in terms of  $T_i$ , which would now be interpreted as the number of calls that can be supported by link  $i$ . The differences between wireless and wired network operation, and their impact on system operation and performance evaluation, are discussed in Refs. 1 and 2. Although we limit our study to examples that represent wireless networks, the mathematical model and algorithmic approach would apply to wired networks as well.

In this report we do not address the protocol issues that are associated with call setup, such as the control messages that must be exchanged to coordinate the transmission, relay, and reception functions. Our focus is on the development of a mathematical system model that can serve as the basis of performance (in this case, throughput) optimization by proper choice of offered load, subject to blocking probability QoS constraints.

## 2.1 Control Policy

Our ultimate goal is to achieve optimal network performance. In Refs. 1 and 2 we approached this problem by exercising an admission-control policy on calls, under the assumption that routes and offered loads on each of the circuits were fixed. In the present study, we again fix the routes. However, instead of determining the best admission-control policy for a fixed offered load, we determine the offered load that maximizes throughput for a fixed admission-control policy, subject to QoS constraints on blocking probability. In this section, we discuss the nature of the admission-control policies used in our studies, and describe the mathematical model for system performance.

In Ref. 7, control policies are divided into five classes: complete sharing, complete partitioning, trunk reservations, coordinate convex, and dynamic programming. Complete sharing (described near the beginning of Section 2) is the simplest form of control because no control is exercised, i.e., a call is accepted as long as sufficient resources are available at all nodes along the multihop path. Complete partitioning, under which resources are preallocated exclusively to particular call types (e.g., calls between particular source-destination pairs), is the most restrictive form of control; it may provide the least efficient use of network resources, but it easily provides a guaranteed level of service to each call class. Trunk reservations provide a compromise between complete sharing and complete partitioning; all primary-routed calls are accepted (provided that resources are available), but alternate-routed calls are accepted only if some threshold level of resources is available. Both coordinate convex (defined below) and dynamic programming control policies use a modified form of complete sharing within a state space that is a restriction (subspace) of the uncontrolled state space. In our study, coordinate-convex control policies are used because they provide a form of intelligent resource sharing without the complexity of dynamic programming.

A stationary admission-control policy is specified in terms of the set of admissible states  $\Omega$ . A new call is admitted if the state to be entered is in the admissible region; otherwise, it is blocked and lost from the system. The capacity constraints limit the state space  $\Omega$  in which  $\mathbf{x}$  is allowed to take values. We assume that the state space is “coordinate convex” [8], i.e., in a system with  $J$  circuits, if  $\mathbf{x}$  is an admissible state (i.e., if  $\mathbf{x} \in \Omega$ ) and  $x_j \geq 1$ , then  $\mathbf{x}' = (x_1, x_2, \dots, x_{j-1}, \dots, x_j)$  must also be an admissible state (i.e.,  $\mathbf{x}' \in \Omega$ ). This condition implies the very reasonable property that at the completion of a call, the resources it used are immediately available to serve other calls, and that call durations are independent of the system state. It also implies that resources are not reassigned in the middle of a call by rerouting that call through an alternative path. We consider policies that retain the coordinate convexity of the state space. Thus, a coordinate-convex policy is specified in terms of the set of admissible states in a discrete state space; we use the notation  $\Omega$  to refer to either an admission-control policy or the corresponding region in the state space. The control policy is effectively a further restriction of the (already coordinate-convex) admissible state space defined by the capacity constraints (Eq. (1)).

Under our assumption of Poisson arrival statistics at rate  $\lambda_j$  on circuit  $j$ , the use of coordinate-convex policies [8] results in a product-form characterization of the system state [9]. The expected call duration of a session on circuit  $j$  is  $1/\mu_j$ .<sup>2</sup> The product-form solution corresponding to a policy  $\Omega$ , offered load vector  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_J)$  and service rate vector  $\boldsymbol{\mu} = (\mu_1, \dots, \mu_J)$  provides the equilibrium distribution of the state of the voice-call process under that policy.

Although the optimal policy is not necessarily a coordinate-convex one, Jordan and Varaiya have provided examples in which the best coordinate-convex policy performs almost as well as dynamic-programming solutions [7]. The latter (like coordinate-convex policies) are based on a set of admissible states; however (unlike coordinate-convex policies) certain transitions to some states in the admissible region may be prohibited. An advantage of coordinate-convex policies is that they are easy to implement. Furthermore, because their product-form solution is easier to evaluate than a dynamic-programming solution, it is possible to solve much larger problems if we consider only coordinate-convex policies.

For notational purposes, we subdivide the control policy into a set of “threshold controls” and a set of “linear-combination controls.” Threshold controls restrict the number of calls that will be admitted to the individual circuits, and can be expressed as

$$x_j \leq X_j, \quad 1 \leq j \leq J, \quad (2)$$

where  $X_j$  is the threshold on circuit  $j$ .

The linear-combination controls are restrictions on the sums of the number of calls on those circuits that share specific resources (e.g., those that share a common node or a common link), i.e.,

$$\sum_{j \in S_l} x_j \leq Y_l, \quad (3)$$

where  $S_l$  is a subset of the set of all circuits and  $Y_l$  is the threshold on the sum of calls on circuits that are members of  $S_l$ .<sup>3</sup> In this study, transceivers are not assigned a priori to circuits; sessions are accepted as long as the threshold and linear-combination constraints are satisfied. Although not all coordinate-convex policies can be described by Eqs. (2) and (3), this description is sufficiently rich for our purposes since empirical evidence has shown that even threshold policies perform almost as well as the best of the coordinate-convex policies [7]. The optimal policy is determined by evaluating the performance under a large number

<sup>2</sup>It is not necessary to assume that the call duration is exponential. A Poisson arrival process and general service time distribution is sufficient for the product-form solution to apply [10]; knowledge of the means of the service times provides enough information to determine the equilibrium distribution. However, in some of our other studies it was necessary to assume that call durations were exponential to maintain a Markovian state space structure that can be used for dynamic analysis [11-13].

<sup>3</sup>For example, for the network of Fig. 1, all three circuits pass through node 1, which is the only node that supports more than one circuit. The subsets of circuits passing through this node are:  $S_1 = \{1,2\}$ ,  $S_2 = \{1,3\}$ , and  $S_3 = \{2,3\}$ ; it is not necessary to include the complete set of circuits  $\{1, 2, 3\}$ . Additional examples are provided in Ref. 2.

of candidate admission-control policies. In Refs. 1 and 2 we developed recursive and descent-search techniques that reduce the complexity of these calculations.

In the present study, we have limited the class of admission-control policies to coordinate-convex policies with threshold controls only (i.e., we do not consider the use of linear-combination controls).<sup>4</sup> No attempt is made to optimize the admission-control policy; the objective is to determine the offered load that maximizes throughput, subject to QoS constraints on blocking probability, for a specified admission-control policy and routing scheme. It is hoped that the methods developed here can be extended to the solution of the combined problem in which admission control, routing, and offered load are jointly optimized.

## 2.2 Solution and Performance Measures

We assume that calls on circuit  $j$  are characterized by a Poisson arrival process with rate  $\lambda_j$  and a mean duration of  $1/\mu_j$ ; to simplify the problem formulation, we assume that  $\mu_j = 1$ ,  $j = 1, 2, \dots, J$ . Thus the corresponding offered load on circuit  $j$  is  $\rho_j = \lambda_j / \mu_j = \lambda_j$ . Furthermore, control is centralized, and the resources needed to support a circuit are acquired simultaneously when the call arrives and are released simultaneously when the call is completed. Calls are blocked when one or more nodes along the path do not have a transceiver available or when a decision is made not to accept a call, i.e., to accept the call would bring the system state outside the region defined by admission-control policy  $\Omega$ .

Under these conditions, in conjunction with the use of coordinate-convex policies, it has been shown [7, 9] that the system state has the product-form stationary distribution. Assuming use of admission-control policy  $\Omega$ , we have

$$\pi_{\Omega}(\mathbf{x}) = \pi_{\Omega}(0) \prod_{j=1}^J \frac{\rho_j^{x_j}}{x_j!}, \quad (4)$$

where  $\pi_{\Omega}(0)$  is the normalization constant given by

$$\pi_{\Omega}(0) = \left\{ \sum_{\mathbf{x} \in \Omega} \prod_{j=1}^J \frac{\rho_j^{x_j}}{x_j!} \right\}^{-1}. \quad (5)$$

The control policy is defined by the specification of the admissible state space  $\Omega$ . For any such state space, it is straightforward (although time consuming) to evaluate  $\pi_{\Omega}(0)$ , which in turn permits the evaluation of performance measures such as throughput and blocking probability. In state  $\mathbf{x}$ , the total number of active calls is

$$\gamma(\mathbf{x}) = \sum_{j=1}^J x_j. \quad (6)$$

We define throughput  $\Gamma(\Omega)$  to be simply the expected number of active calls averaged over the system state:

$$\Gamma(\Omega) = \sum_{\mathbf{x} \in \Omega} \{\gamma(\mathbf{x})\pi_{\Omega}(\mathbf{x})\}. \quad (7)$$

Our throughput metric is somewhat different from the usual definition, which is the number of *completed* calls per unit time. We have found our metric to be useful because it is closely related to the notion of

<sup>4</sup>The iterative Lagrangian technique developed in this report (beginning in Section 4) could be applied to any admission-control policy for which analytical/numerical evaluation of the system's equilibrium distribution is possible. For example, any admission-control policy for which the product-form solution is satisfied (as discussed in Section 2.2) would be a candidate for our approach. In particular, any coordinate-convex policy would be appropriate, provided that the offered loads arrive according to independent Poisson processes.

“residual capacity” [14], which is a measure of the resources available for data in integrated voice/data networks after voice traffic has claimed its required resources. If calls on all circuits have the same expected length (as they do in our formulation, since  $\mu_j = 1, j = 1, \dots, J$ ), or alternatively, if the throughputs of each circuit are weighted in proportion to their expected length, our definition simply scales (i.e., is proportional to) the usual definition.

The blocking probability associated with circuit  $j$ , designated as  $P_j(\Omega)$ , is the fraction of calls arriving on circuit  $j$  that are blocked. Since we have a stationary system with Poisson arrivals, we can make use of the PASTA (Poisson arrivals see time averages) theorem [15, 16] to recognize that  $P_j(\Omega)$  is simply the probability that the system is in a state in which a call on circuit  $j$  is blocked. Therefore,

$$P_j(\Omega) = \sum_{x \in \Omega} 1((x_j + 1) \notin \Omega) \pi_{\Omega}(x), \quad (8)$$

where  $1(\bullet)$  is the indicator function, which is 1 if the argument is true and 0 otherwise, i.e., the probabilities of the states in which a call on circuit  $j$  is blocked are summed. It is important to note that  $P_j(\Omega)$  can be quite large, even when  $\rho_j = 0$ . This behavior is a consequence of the fact that traffic on other circuits can put the system into states in which a call on circuit  $j$  would be blocked, should such a call arrive.

The overall (averaged over all circuits) blocking probability  $P_{av}(\Omega)$  is the ratio of the expected number of blocked calls per unit time (summed over all circuits) to the expected total number of call arrivals per unit time:

$$P_{av}(\Omega) = \frac{1}{\Lambda} \sum_{j=1}^J \lambda_j P_j(\Omega) \quad (9)$$

where

$$\Lambda = \sum_{j=1}^J \lambda_j$$

is the total offered load. In most of this report, each circuit is required to satisfy the QoS constraint, i.e.,  $P_j(\Omega) \leq Q_j = \text{QoS}$  for all  $j$ . However, we also consider examples in which it is sufficient for the average blocking probability to satisfy the constraint, i.e.,  $P_{av} \leq \text{QoS}$ .

The goal of the problem studied in Refs. 1 and 2 was to restrict the admissible state space to a coordinate-convex region such that the desired performance measure is optimized. The direct approach is to compute  $\pi_{\Omega}(0)$  and the performance index for all possible coordinate-convex regions. We developed a recursive procedure to accelerate the evaluation of a large number of different admission-control policies and a descent-search method to minimize the number of policies that must be evaluated in searching for the optimal one. These techniques are described fully in Refs. 1 and 2.

In the present study, since we fix the admission-control policy and attempt to determine the optimal offered-load vector  $\lambda$ , we can drop the  $\Omega$  in the notation and use the following definitions:

$$S_j = \text{throughput on circuit } j = \lambda_j(1 - P_j), \quad (10)$$

$$S = \text{total throughput} = \sum_{j=1}^J S_j = \Lambda(1 - P_{av}), \quad (11)$$

and

$$P_{av} = \text{overall blocking probability} = \sum_{j=1}^J \frac{\lambda_j}{\Lambda} P_j. \quad (12)$$

### 2.3 Computational Issues in Evaluating the Solution

Since the equilibrium distribution  $\pi_{\Omega}(x)$  has the product-form, the evaluation of system performance becomes greatly simplified. Since the distribution is known to within the normalization constant  $\pi_{\Omega}(0)$ ,

there is no need to solve any balance equations. However, the evaluation of  $\pi_{\Omega}(0)$  is computationally intensive because it requires the evaluation and summation of a large number of terms<sup>5</sup> of the form  $\Pi \rho_j^{x_j} / x_j!$ . (Computational issues associated with loss networks are addressed in Ref. 17.) Considerable effort has been exerted in developing efficient procedures for calculating the normalization constant (see e.g., Ref. 18), but such methods are generally problem specific and of little help to our problem.

In Refs. 1 and 2, where the goal was to choose the best admission-control policy, it was necessary to evaluate the normalization constant for a large number of candidate policies. In the present study, since we assume the use of a fixed admission-control policy, it is not necessary to do so. However, the normalization constant must now be evaluated for a large number of candidate offered-load vectors within a continuous region as we search for the offered load vector that maximizes throughput subject to QoS constraints; hence the new problem is also computationally intensive. This problem is equivalent to the determination of the “Erlang capacity” of the network. Before addressing the optimization problem in detail, we define the notion of Erlang capacity.

### 3. ERLANG CAPACITIES

The “Erlang capacity” [19] of a network with  $J$  circuits is defined as the set of input rates  $\{\lambda_1, \dots, \lambda_J\}$  for which some given QoS criteria can be met. The method of determining optimal admission-control policies that we used in Refs. 1, 2, 14, and 20 relies on fixing the set of “offered” load vector  $\boldsymbol{\lambda}$ , and computing the throughput (or blocking probability) performance for different admission-control policies. Thus the set of  $\lambda_j$ 's was specified a priori, and the goal was to determine the optimal admission-control policy.

To determine the Erlang capacity of a network, we essentially need to reverse our approach by finding (for a fixed set of admission-control policies) the  $\lambda_j$ 's that meet the desired QoS level. Thus it is possible to determine the Erlang capacity associated with a specified admission-control policy and QoS level. The specific QoS constraint that we consider is that the maximum circuit blocking probability  $P_{\max} = \max\{P_1, \dots, P_J\}$  does not exceed a given level.

The determination of the Erlang capacity region is a complicated multidimensional problem that requires a huge search to find the set of offered loads that satisfy the QoS requirements. Rather than attempting to determine this region, we instead focus on the determination of some extremal points in this region that optimize a performance measure.

### 4. THE OPTIMIZATION PROBLEM

Our goal is to find the offered load vector  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_J)$  that maximizes the total network throughput  $S = S(\boldsymbol{\lambda})$  for a fixed coordinate-convex admission-control policy, subject to a circuit blocking probability  $P_j(\boldsymbol{\lambda}) \leq Q_j = \text{QoS constraint for circuit } j$ ,<sup>6</sup> and  $\lambda_j \geq \lambda_{\min}$ .<sup>7</sup> Note that such a vector  $\boldsymbol{\lambda}$  is an extremal point of the Erlang capacity region defined in Section 3. The equilibrium state distribution, and therefore throughput and blocking probability as well, are determined by means of the product-form solution discussed in Section 2. Hence, we have a nonlinear optimization problem that includes nonlinear inequality constraints.

Our optimization problem can be written as

$$\max_{\boldsymbol{\lambda}} \{S(\boldsymbol{\lambda})\}, \quad (13)$$

<sup>5</sup>For our example of Network 1 (see Section 5) with six transceivers at every node and a threshold of six on the number of calls on any circuit, the number of possible states (and hence the number of terms) is 303,248 (reducing the thresholds to four while keeping six transceivers at every node decreases the number of states to 284,115). When the number of transceivers and the threshold values are eight, the number of states is 2,633,094; reducing the thresholds to six decreases the number of states to 2,585,861).

<sup>6</sup>In Section 10 we discuss the use of an alternative QoS constraint, under which the average circuit-blocking probability (weighted by the offered load to each circuit) must not exceed a specified value.

<sup>7</sup>To provide some degree of fairness to all circuits, it may be desirable to specify a nonzero value of  $\lambda_{\min}$ , thereby ensuring that all circuits are permitted at least this value of offered load.

where the total network throughput  $S$  is the sum of the throughputs on each of the  $J$  circuits, i.e.,

$$S = S(\boldsymbol{\lambda}) = \sum_{i=1}^J S_i(\boldsymbol{\lambda}). \quad (14)$$

The throughput  $S(\boldsymbol{\lambda})$  is obtained by the product-form solution associated with the particular network (i.e., the topology, number of transceivers at the nodes, and choice of circuits), the admission-control policy  $\Omega$ , and the offered load vector  $\boldsymbol{\lambda}$ , as described in Section 2.2. Since an analytical solution cannot be obtained (except for trivial problems), numerical evaluation is necessary. In particular, we use iterative search techniques. Such techniques start from an initial value of offered load  $\boldsymbol{\lambda}_0$ , and use gradient information to update the offered load vector  $\boldsymbol{\lambda}$  until convergence to the optimal solution is achieved.

An unconstrained search would proceed along the direction defined by the throughput gradient, namely the  $J$ -dimensional vector

$$\nabla S = \left( \frac{\partial S}{\partial \lambda_1}, \dots, \frac{\partial S}{\partial \lambda_J} \right), \quad (15)$$

whose component in the direction of the  $i$ th circuit is

$$\frac{\partial S}{\partial \lambda_i} = \sum_{j=1}^J \frac{\partial S_j}{\partial \lambda_i}. \quad (16)$$

However, the introduction of constraints complicates the solution process considerably. Since our problem involves a nonlinear objective function and nonlinear constraints, it falls into a class for which there are few available analytical results. Thus, there is no guarantee of convergence to the optimal solution. Nevertheless our experience with Lagrangian techniques, which in some cases are augmented by heuristics, has shown that good (although not necessarily optimal) solutions can be obtained routinely. We now formulate the constrained optimization problem.

#### 4.1 The Constrained Optimization Problem and the Basic Lagrangian Optimization Method

Our goal is to choose the offered load vector  $\boldsymbol{\lambda}$  such that throughput is maximized, subject to constraints on blocking probability. This problem can be written as

**Constrained Optimization Problem:**

$$\max_{\boldsymbol{\lambda}} \{S(\boldsymbol{\lambda})\}, \quad (17)$$

$$\text{subject to: } P_j(\boldsymbol{\lambda}) \leq Q_j,$$

where  $P_j(\boldsymbol{\lambda})$  is the probability that an incoming call to circuit  $j$  is blocked.<sup>8</sup>

**Definitions:**

- We say that an offered-load vector  $\boldsymbol{\lambda}$  is *admissible* if the constraints on blocking probability are satisfied.<sup>9</sup>
- The *admissible region* contains all offered-load vectors that are admissible.
- Corresponding to each admissible vector  $\boldsymbol{\lambda}$  is a value of *admissible throughput*.

<sup>8</sup>In most of our examples, it is assumed that  $Q_1 = \dots = Q_J = QoS$ , i.e., all circuits are subject to the same value of the QoS constraint; however, we do consider several examples in which widely different values are used.

<sup>9</sup>In this report, the term “admissible” is applied to two distinct domains. In Section 2 it is used in the context of the admission-control policy. Here, and in the rest of this report, it relates to the requirement that the offered load be such that the QoS constraints are satisfied (for a specified admission-control policy, set of routes, etc.).

The QoS inequality constraints ( $P_j(\boldsymbol{\lambda}) \leq Q_j$ ) can be written as “equality” constraints as:

$$\min(P_j(\boldsymbol{\lambda}), Q_j) - P_j(\boldsymbol{\lambda}) = \min(0, Q_j - P_j(\boldsymbol{\lambda})) = 0. \quad (18)$$

A similar equality constraint can be derived for the  $\lambda_j \geq \lambda_{\min}$  constraint, but we have found that doing so is unnecessary because this constraint is easily enforced by simply not allowing  $\lambda_j$  to grow smaller than  $\lambda_{\min}$ , as we show below in Eq. (20).

We convert our constrained optimization problem to an unconstrained one by creating a new objective function that increases with  $S(\boldsymbol{\lambda})$  but is decreased (penalized) when the QoS constraint is violated. If the new objective function is properly formed and has appropriate parameters, its maxima should coincide with those of the original problem. For this new objective function, we use the Augmented Lagrangian function [21] given by

$$L(\boldsymbol{\lambda}, \boldsymbol{\gamma}) = S(\boldsymbol{\lambda}) + \sum_{i=1}^J [\gamma_i \min\{0, Q_i - P_i(\boldsymbol{\lambda})\} - \frac{d}{2} (\min\{0, Q_i - P_i(\boldsymbol{\lambda})\})^2]. \quad (19)$$

The coefficients  $\gamma_i$  are Lagrange multipliers. The second term in the summation is a quadratic penalty term with constant coefficient  $d/2$ . This term is the augmentation to the Lagrangian objective function that is helpful in enforcing the QoS constraint. Note that both terms in the summation contribute nothing to the objective function whenever the QoS constraint is satisfied, and they contribute negative values when the constraint is violated.

Our goal is to maximize  $L(\bullet)$  over  $\boldsymbol{\lambda}$ . To do this, we use the iteration

$$\lambda_j(k+1) = \max \left\{ \lambda_{\min}, \lambda_j(k) + \theta \frac{\partial L(\boldsymbol{\lambda}, \boldsymbol{\gamma})}{\partial \lambda_j} \right\};$$

$$j = 1, \dots, J; \quad k = 1, \dots, k_{\max}; \quad \lambda_j(0) = \lambda_{j_0} \geq \lambda_{\min} \quad (20)$$

where  $\theta$  is a stepsize parameter, and

$$\frac{\partial L(\boldsymbol{\lambda}, \boldsymbol{\gamma})}{\partial \lambda_j} = \frac{\partial S}{\partial \lambda_j} + \sum_{i=1}^J 1(P_i(\boldsymbol{\lambda}) > Q_i) \frac{\partial P_i}{\partial \lambda_j} [d(Q_i - P_i(\boldsymbol{\lambda})) - \gamma_i]. \quad (21)$$

Here,  $1(\bullet)$  is the indicator function. This equation involves the gradients of both throughput and blocking probability. We use the product-form solution to calculate the equilibrium state occupancy distribution, from which we can obtain the circuit blocking probabilities  $P_j(\boldsymbol{\lambda})$ , the circuit throughput values  $S_j(\boldsymbol{\lambda})$ , and the partial derivatives  $\partial P_j(\boldsymbol{\lambda})/\partial \lambda_i$  and  $\partial S_j(\boldsymbol{\lambda})/\partial \lambda_i$ . In Ref. 9, Jordan and Varaiya have shown that

$$\frac{\partial P_j(\boldsymbol{\lambda})}{\partial \lambda_i} = \begin{cases} -\frac{\mu_j}{\lambda_i \lambda_j} \text{cov}(x_i, x_j), & i \neq j \\ \frac{\mu_i}{\lambda_i^2} (E\{x_i\} - \text{var}(x_i)), & i = j \end{cases}, \quad (22)$$

and

$$\frac{\partial S_j(\boldsymbol{\lambda})}{\partial \lambda_i} = \frac{\mu_j}{\lambda_i} \text{cov}(x_i, x_j). \quad (23)$$

These quantities are evaluated by using the product-form solution [1, 2].

Thus the search proceeds along the direction of the throughput gradient  $\nabla S$  when the QoS constraints are not violated, and in a direction influenced by both the throughput and blocking probability gradients when the constraints are violated. The max operation in Eq. (20) enforces the  $\lambda_j \geq \lambda_{\min}$  constraint.

The search can be performed either for a fixed number of iterations or, alternatively, until convergence has been obtained. In the latter case, convergence can be defined as a sufficiently small change in  $S(\boldsymbol{\lambda})$  from one iteration to the next. Stopping rules are discussed in Section 8.3.

## 4.2 Step Size Considerations

Stepsize is crucial to the correct and efficient operation of the search. Stepsize must be considered in both updating the Lagrange multipliers and in updating the offered load parameters. The stepsize rules discussed below were used in our preliminary studies, which involved the basic optimization model discussed in this section. Modifications developed in conjunction with our improved search techniques are discussed in Section 7.2.

### Lagrange Multiplier Update

During the iteration, as suggested in Ref. 22, the Lagrange multipliers  $\gamma_j$  are updated according to

$$\gamma_j(k+1) = \gamma_j(k) - 1(P_j(\boldsymbol{\lambda}) > Q_j) \frac{c(Q_j - P_j(\boldsymbol{\lambda}))}{\text{iterations}}, \quad (24)$$

where  $k = 1, \dots, k_{\max}$ ;  $c$  is a positive constant; and  $\gamma_j(0) = \gamma_0, j = 1, \dots, J$ . Note that the  $(Q_j - P_j(\boldsymbol{\lambda}))$  term, which is negative when the indicator function is 1, is subtracted so that  $\gamma_j$  increases to encourage constraint satisfaction when the constraint is violated, but remains unchanged when the constraint is satisfied and the Lagrange multiplier term is inactive. The increment in  $\gamma_j$  is proportional to the amount by which  $P_j(\boldsymbol{\lambda})$  exceeds  $Q_j$  (so that it is increased more on circuits that violate QoS by significant amounts); it is inversely proportional to the number of iterations thus far (to encourage convergence).

### Offered-Load Update

Allowing the stepsize parameter  $\theta$  to vary (in particular, to decrease) is effective in damping oscillatory behavior and in increasing the speed of convergence. The basic  $\theta$ -update rule we used in our preliminary studies is

$$\theta(k+1) = \max \left\{ \theta_{\min}, \theta(k) - 1(\text{crossing?}) \frac{\theta_0 - \theta_{\min}}{r} \right\}, \quad (25)$$

where  $k = 1, \dots, k_{\max}$ ;  $\theta(0) = \theta_0$ ; and the argument of the indicator function is true if the blocking probability value of any circuit has crossed the QoS threshold in either direction in going from iteration  $k-1$  to iteration  $k$ .<sup>10</sup> The parameters  $\theta_0$  and  $\theta_{\min}$  specify the initial and the minimum  $\theta$  values respectively, and the ‘‘damping-rate’’ parameter  $r$  determines the rate at which  $\theta$  is decreased.

## 5. EXAMPLE OF RESULTS BASED ON THE BASIC SEARCH PROCEDURE

In this section, we discuss performance results that are based on the wireless network shown in Fig. 3. This network has  $N = 24$  nodes and  $J = 10$  circuits. We refer to this network as ‘‘Network 1.’’ Additional networks are studied in Section 8. The purpose of this section is to illustrate the nature of the optimization problem under study, as well as to motivate the improvements that are discussed in Sections 6 and 7.

<sup>10</sup>The basis for using the number of threshold crossings in the stepsize update equation is as follows. There may be no need to decrease  $\theta$  before the first threshold crossing because a rapid ascent to the neighborhood of the optimal solution is desired. However, once the search is in the neighborhood of the optimal point, a decrease of  $\theta$  reduces the amount of oscillation, thus permitting convergence to the optimal point. In multidimensional searches such as ours, however, it is less clear that this approach is a good one. The stepsize rules discussed in Section 7 are based solely on the iteration number.

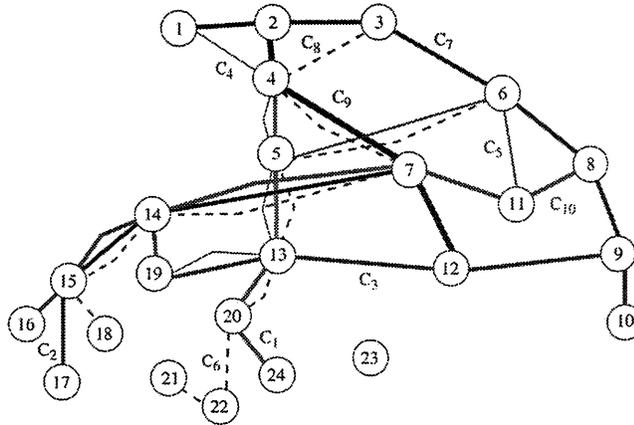


Fig. 3 — Network 1 (24 nodes, 10 circuits)

In the example studied in this section, we assume eight transceivers per node ( $T_1 = \dots = T_{24} = 8$ ), we limit the number of calls that can be accepted on any particular circuit to six (thus we are using a threshold-based coordinate-convex admission-control policy with  $X_1 = \dots = X_{10} = 6$ , as discussed in Section 2),<sup>11</sup> and we define the QoS requirement to be a maximum acceptable circuit-blocking probability common for all circuits (i.e., the blocking probability  $P_j$  on circuit  $j$  must not exceed  $Q_j = \text{QoS} = 0.3; j = 1, \dots, 10$ ).<sup>12</sup> Although our basic search procedure was able to converge to a good solution, convergence was typically slow. In Section 5.1 we discuss the application of the basic search procedure to this network, and in Section 6 we describe an improved search procedure, which makes better use of the available information at each iteration in the search.

Table 1 summarizes the results of five runs that used the same parameters to search for the set of  $\lambda_j$  that maximizes throughput in Network 1. The  $\lambda$  values (which are the Poisson arrival rates  $\lambda_j$  at each circuit) are the values found to yield the highest admissible throughput  $S(\lambda)$  in the respective search (but were not necessarily the values at the last iteration). Each iteration was stopped when convergence was achieved, as determined by sufficiently small changes in the offered-load vector. In most cases, however, convergence was achieved prematurely, apparently as a result of too rapid a decrease of the stepsize  $\theta$ . The number of iterations required to achieve convergence is shown in the last column. The runs differ only in the starting point,  $\lambda_0 = (\lambda_{0,1}, \lambda_{0,2}, \dots, \lambda_{0,10})$ . The  $\lambda_0$  values for Run 1 were obtained by scaling up the best results from searches of the same network with four transceivers per node. Run 2 was started from  $\lambda_0 = (1.75, \dots, 1.75)$ , which is approximately the largest uniform offered load<sup>13</sup> for which the QoS constraint of  $P_{\max} \leq 0.3$  is satisfied. Run 3 was started from  $\lambda_0 = \lambda$  found in Run 2; Run 4 was started from the solution found in Run 3; Run 5 was started from the solution found in Run 4. Thus the concatenation of Runs 2-5 can be viewed as a single composite run of length 6150 iterations in which the stepsize  $\theta$  is varied from its initial (large) value to its final (small) value four times. Note that both Runs 1 and 5 converged to nearly the same point in  $\lambda$  space. Although the choice of  $\lambda_0$  was critical to the quality of the solution in these examples, the improved algorithm discussed in Section 6 and thereafter is relatively insensitive to the initial offered load, provided that a good choice of system parameters is made.

As a result of the stepsize damping, Run 2 converged in 1509 iterations to a throughput of 16.63, a value relatively far from the apparent<sup>14</sup> maximum of  $S(\lambda) \approx 17$ . Figure 4(a) shows the evolution of throughput throughout the concatenation of Runs 2 through 5. Figure 4(b) also shows the throughput, but includes only

<sup>11</sup>We often use the shorthand notation  $T_i = 8$  to indicate that each of the  $N$  nodes has 8 transceivers. Similarly, the notation  $X_j = 6$  means that the threshold on all  $J$  circuits is 6.

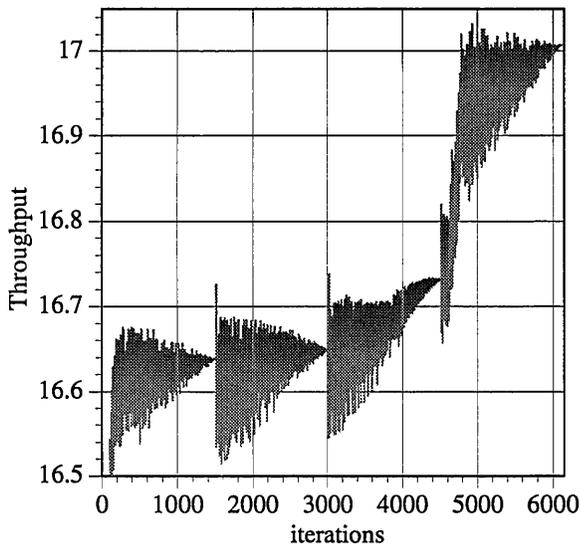
<sup>12</sup>In Section 9.3 we consider networks in which the circuits do not all have the same QoS constraint.

<sup>13</sup>The term “uniform offered load” refers to an offered load vector in which all of the  $\lambda_j$ 's are equal.

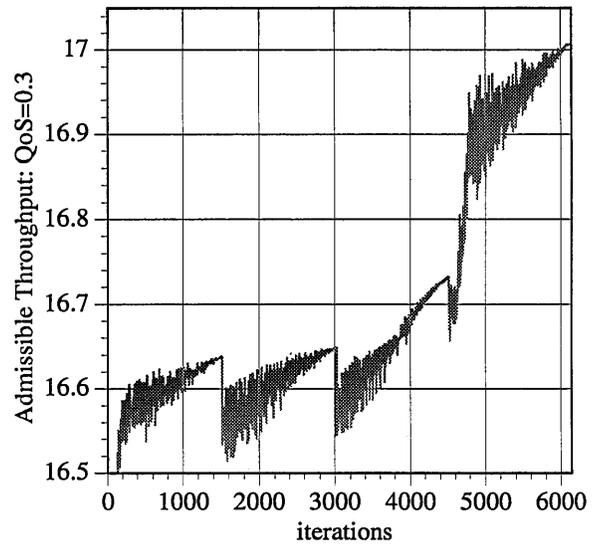
<sup>14</sup>We say the apparent maximum because we have not found a solution that yields a significantly larger throughput, and because there is no known alternative method for determining the true maximum.

Table 1 — Results of Searches of Network 1  
 ( $T_i = 8; X_j = 6; QoS = 0.3; \theta_o = 0.1, \theta_{min} = 2 \times 10^{-6}, r = 1500; \gamma_o = 10, c = 150, d = 100, \lambda_{min} = 0.05$ )

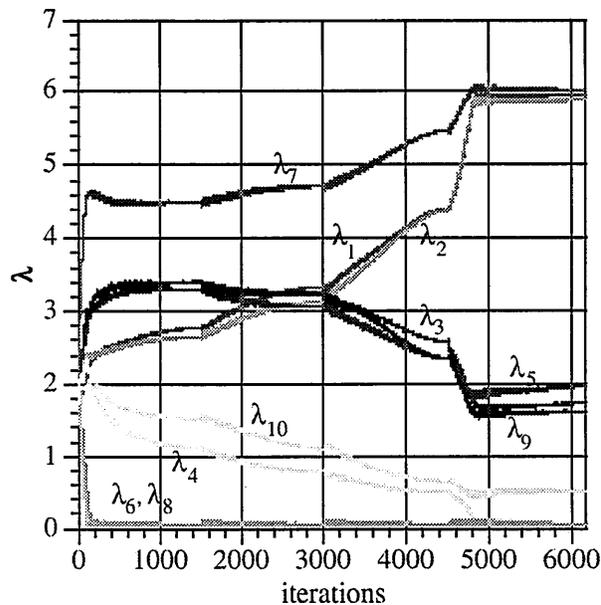
Run	$\lambda_0$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_5$	$\lambda_6$	$\lambda_7$	$\lambda_8$	$\lambda_9$	$\lambda_{10}$	$S(\lambda)$	Iterations
1	scaled	5.9569	5.871	1.6975	0.0632	1.9498	0.05	6.0318	0.0757	1.5814	0.5339	16.992503	1375
2	1.75	2.7769	2.6434	3.4073	1.1265	3.2968	0.05	4.5016	0.05	3.4065	1.5118	16.638538	1509
3	run 2	3.3295	3.1273	3.2642	0.7831	3.0607	0.05	4.7117	0.05	3.2429	1.1041	16.649440	1513
4	run 3	4.4052	4.3835	2.5763	0.4930	2.3540	0.05	5.4694	0.05	2.3507	0.6470	16.732417	1497
5	run 4	5.9616	5.8808	1.7283	0.0515	1.9600	0.05	6.0203	0.0501	1.6222	0.5160	17.006826	1631



(a) Evolution of throughput



(b) Evolution of admissible throughput



(c) Evolution of offered-load  $\lambda$

Fig. 4 — Evolution of throughput and  $\lambda$  as the iteration proceeds through the concatenation of Runs 2-5

those points in the search for which the QoS constraints are satisfied; hence the terminology of “admissible throughput” (throughput values for iterations at which one or more of the QoS constraints are violated are simply ignored). Figure 4(c) shows the evolution of  $\lambda$ , i.e., the values of the offered load values  $\lambda_j$  as the iterative search progressed. It appears that the damping was helpful in each of these runs, in the sense that it permitted some progress toward the optimal solution; without damping the oscillation might have continued indefinitely. Thus, these results seem to suggest that a concatenation of several runs in which  $\theta$  is damped from its maximum to minimum value appears to be more effective than a single run of comparable total length in which  $\theta$  is decreased monotonically. Actually, we demonstrate in Section 8 that such a concatenation is not necessary, provided that the algorithm’s parameters are chosen well or that heuristics are incorporated into the algorithm to direct the search along a better trajectory. We show that reliable convergence can be obtained from arbitrary initial conditions. Furthermore, the speed of convergence of the improved algorithm is much greater than that shown in these examples.

It is interesting to note that the offered load vector  $\lambda$ , after appearing to stabilize, changes rapidly after iteration 3000. It is also interesting that, despite the dramatic variation in the values of  $\lambda$ , the admissible throughput increases by only about 2.4% after iteration 400. Such behavior appears to be typical in this optimization problem and has been observed in numerous examples involving a number of markedly different networks, as is demonstrated in Section 8. We show that throughput is relatively insensitive to changes in the offered load vector, and nearly optimal performance (e.g., 98% of the maximum achievable throughput, while satisfying the QoS constraints) can be obtained over a wide range of offered load values.

Apparently, the starting point for Run 1 is a good one, since the search is able to find an admissible throughput of nearly 17 in 1375 iterations.<sup>15</sup> Figure 5 shows the search trajectory of Run 1 projected onto the  $(\lambda_1, \lambda_3)$  plane. Also shown are two sets of contours. Those depicting throughput  $S(\lambda)$  are self-explanatory. Those depicting  $P_{c-\max}$ , which is identical to  $P_{\max}$ , refer to the largest blocking probability among the circuits; this is an important quantity because the value of  $P_{c-\max}$  determines whether or not an offered load vector is admissible.<sup>16</sup>

The contours are based on the use of the values of  $\lambda_2$  and  $\lambda_4, \dots, \lambda_{10}$  that were determined at the end of Run 1, as shown in the first row of Table 1. All of the  $\lambda_j$ ’s actually vary throughout the search, as does the identity of the dominant circuit (i.e., the circuit with the largest blocking probability); thus, the throughput and blocking probability contours in any plane (such as the  $(\lambda_1, \lambda_3)$  plane shown here) vary throughout the search. Therefore, it is not possible to determine whether or not any particular point on the trajectory is admissible on the basis of this figure alone, and so it is difficult to make definitive conclusions on the basis of figures of this type. Nevertheless, Fig. 5 does provide crucial insight into the nature of the trajectory.

Figure 5 shows that the search moves rapidly to the neighborhood of the maximum and spends most of its time searching there. Examination of this figure has permitted us to make the following observations on the behavior of the search. The throughput gradient  $\nabla S$  is virtually constant (in both magnitude and direction) in the neighborhood of the maximum. Hence, by Eq. (20),  $\lambda$  will change by  $\theta \nabla S$  at any admissible solution in that region, i.e., the step will be approximately  $\theta$  times a constant in the (constant) direction of the throughput gradient. It is very likely that this step will throw the search over the QoS boundary into the inadmissible region. In the inadmissible region, the activation of the constraint-enforcing terms in Eq. (20) causes  $\lambda$  to move in the direction opposite to the  $P_{c-\max}$  gradient, back toward the admissible region, with a small move in the direction of the throughput gradient caused by the always-active  $\nabla S$  term. Thus, a slowly changing oscillation can easily occur while slowly “climbing” the QoS boundary toward the maximum. The

<sup>15</sup>In the early stages of this study, this was considered to be relatively rapid convergence. However, considerably faster convergence has been obtained using our improved algorithms, which provides reliable convergence from virtually any initial condition; e.g.,  $\lambda_0 = 0$  has been used in most of our examples.

<sup>16</sup>In a typical run, the fraction of iterations that produce admissible solutions is difficult to predict; this fraction may be between about 15% and 50%, depending on the network under study and on the parameters of the algorithm. There seems to be no direct correlation between the fraction of admissible solutions and the quality of the solution produced by a particular version of the algorithm.

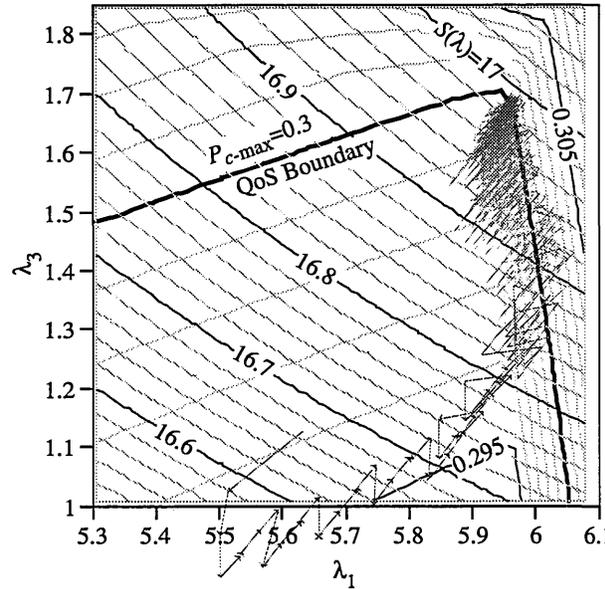


Fig. 5 — Search trajectory in  $\lambda_1$  and  $\lambda_3$  superimposed on throughput and  $P_{c-\max}$  contours;  $\lambda_2$ , and  $\lambda_4, \dots, \lambda_{10}$  are fixed at the values shown in Table 1, Run 1

main feature of the improved search technique discussed in Section 6 is a method to guide the search so that it proceeds essentially parallel to the  $\nabla P_{c-\max}$  contour, rather than at a significant angle to it.

## 6. GUIDED SEARCH TECHNIQUES

The search trajectory shown in Fig. 5 is typical of results obtained using the basic search procedure in that significant (although nonmonotonic) progress is made in the early stage of the iteration, whereas considerably less-productive oscillatory behavior is observed as the search progresses. A common difficulty in constrained optimization problems arises because the optimum lies on the search boundary (i.e., one or more circuit blocking probabilities is at the maximum-permitted QoS value). In unconstrained optimization problems, gradient search procedures are naturally slowed (smaller steps) by the decreasing gradient as they approach the optimum. This slowing allows the search to ascend smoothly to the maximum. However, when the optimum lies on the boundary, as it often does in constrained problems,<sup>17</sup> there is not necessarily a decrease in the gradient in its neighborhood. In this case, typical gradient search techniques rely on damping the stepsize  $\theta$  to slow the search and thereby to facilitate homing in on the optimum. However, an overly rapid decrease in  $\theta$  results in failure to reach the optimal solution; a less rapid decrease in  $\theta$  can result in unacceptably slow convergence.

Constrained optimization problems (especially those such as ours, with many constraints, i.e., one per circuit) often suffer from the impact of irregularly shaped admissible regions. In particular, the need to satisfy blocking probability constraints on each individual circuit creates a situation in which the identity of the dominant circuit can change as a result of small variations in the offered loads. Consequently, it appears that multimodal behavior (i.e., the presence of local optima, which make it difficult to locate the true optimal point), may occur. Such effects appear to be present in our problem, but are not severe in the sense that most versions of the algorithm are usually able to reach an admissible throughput value equal to at least 99% of the optimum. This issue is discussed in Section 13. The theoretical basis for nonlinearly constrained nonlinear optimization problems is not well developed, and so we have not obtained theoretical support for the convergence properties of our algorithm.

<sup>17</sup>We believe that, in our optimization problem, at least one of the circuit blocking probabilities at the optimal point must be at the maximum permitted value. This conjecture is supported by extensive empirical evidence in a variety of network examples. We have observed that, typically, between half and all of the circuit blocking probabilities are at the maximum permitted value.

The degree of progress that can be made at any point in the search depends on which of the following three conditions apply at that time. When none of the QoS constraints are violated, the search tends to proceed along the gradient of increasing throughput, as may be seen in Fig. 5. By contrast, when one or more of the QoS constraints are violated by a significant amount, the constraint terms correct this behavior by directing the trajectory toward the admissible region. The most interesting, and troublesome, behavior occurs when the search trajectory passes near the QoS constraint contour. Figure 5 shows that the violation of a constraint results in oscillatory behavior, with little progress toward the optimal point. The desired behavior would be for the search to proceed along the contour corresponding to the QoS constraint, rather than at a significant angle to it.

We have attempted to mitigate the oscillatory behavior of the “basic algorithm” by using our knowledge of the throughput and blocking probability gradients to guide the search more efficiently. In the following subsections, we introduce the principle of “guided search,” we generalize our approach, and we discuss how the principles of guided search can be used to develop improved search algorithms. These principles are used to develop the versions of the algorithm that are described in Section 7 and tested in subsequent sections.

### 6.1 Preliminary Approach

To illustrate the principle of guided search, we consider an example in which the blocking probability of the “dominant circuit” (i.e., the circuit with the largest blocking probability<sup>18</sup>) is close to (say, to within some  $\epsilon$  of) the specified QoS. We would like to guide the search in a direction of increasing throughput, so that it tends to follow the QoS contour. To simplify the discussion, let us first consider the case in which exactly one of the circuit-blocking probabilities (the dominant circuit) is located within  $\epsilon$  of the QoS constraint, i.e.,  $QoS - \epsilon \leq P_j(\lambda) \leq QoS + \epsilon$ , for exactly one value of  $j \in \{1, 2, \dots, J\}$ . Let us call this circuit  $c$ . In this case, we would like the search to proceed along the component of the throughput gradient  $\nabla S$  that is orthogonal to the circuit-blocking probability gradient  $\nabla P_c$  at our current point in the search. By eliminating the component parallel to  $\nabla P_c$ , we discourage increase in the blocking probability of the dominant circuit. The desired projection can be written as

$$\{\text{Component of } \nabla S \text{ orthogonal to } \nabla P_c\} = \nabla S - \frac{\nabla S \cdot \nabla P_c}{\|\nabla P_c\|^2} \nabla P_c, \quad (26)$$

where

$$\nabla S \cdot \nabla P_c = \sum_{i=1}^J \frac{\partial S}{\partial \lambda_i} \frac{\partial P_c}{\partial \lambda_i} = \sum_{i=1}^J \sum_{j=1}^J \frac{\partial S_j}{\partial \lambda_i} \frac{\partial P_c}{\partial \lambda_i}, \quad (27)$$

and

$$\|X\| = \sqrt{\sum_{j=1}^J X_j^2}$$

is the norm of an arbitrary vector  $X$ . Then we introduce a vector  $D = (D_1, D_2, \dots, D_J)$ , which is equal to this projection when the blocking probability of the dominant circuit is located in a band of width  $2\epsilon$  centered about the QoS contour; otherwise,  $D$  is equal to the throughput gradient  $\nabla S$ :<sup>19</sup>

$$D = \begin{cases} \nabla S - \frac{\nabla S \cdot \nabla P_c}{\|\nabla P_c\|^2} \nabla P_c, & QoS - \epsilon \leq P_c \leq QoS + \epsilon \\ \nabla S, & \text{otherwise} \end{cases}. \quad (28)$$

<sup>18</sup>The notion of dominant circuit does not depend on offered load; in fact, the offered load on the dominant circuit may be zero.

<sup>19</sup>When the blocking probability of the dominant circuit exceeds  $QoS + \epsilon$  the projection is not used, even though the blocking probability of one or more of the other circuits may be located in the band of width  $2\epsilon$ .

We modify the Lagrangian objective function of Eq. (21) by inserting  $D_i$  in place of  $\partial S/\partial \lambda_i$ , as follows:

$$\frac{\partial L(\boldsymbol{\lambda}, \boldsymbol{\gamma})}{\partial \lambda_i} = D_i + \sum_{j=1}^J 1(P_j(\boldsymbol{\lambda}) > Q_j) \frac{\partial P_j}{\partial \lambda_i} [d(Q_j - P_j(\boldsymbol{\lambda})) - \gamma_j]. \quad (29)$$

Thus, when the search is well within the feasible region, it is guided solely by the throughput gradient. When it is sufficiently far into the infeasible region, the penalty term has a significant effect. And when the dominant circuit is within the narrow band centered around the QoS contour, the search is guided by the projection discussed above (and by the constraint terms as well if  $P_c(\boldsymbol{\lambda}) > QoS$ ).

Other approaches to deciding when to use the projection operation are certainly possible. For example, we can consider a “one-sided  $\epsilon$  band” approach in which the projection is applied only when the dominant circuit violates QoS by not more than  $\epsilon$ , but not when  $\boldsymbol{\lambda}$  is in the admissible region close to the QoS boundary. Alternatively, we have considered its use at all times, regardless of whether or not  $P_c$  is very close to QoS. In all of these cases, we discourage the increase of  $P_c$ , while (hopefully) encouraging the throughput to increase.

The use of this projection is the basis of the heuristic techniques we have developed for speeding up the convergence of the algorithm developed in this study. It was certainly not obvious, a priori, that this approach would be successful. The use of  $D_i$ , rather than  $\nabla S$ , in Eq. (29) is equivalent to distorting the shape of the objective function, i.e., we are now maximizing a function whose gradient is  $\mathbf{D}$ , rather than the desired function  $S$  (in either case, subject to the same QoS constraints). Important questions include not only the specification of how the projection is defined (several alternatives are discussed in Sections 6.2 and 7), but also when (i.e., at what, stage of the search) it is to be applied.

The stepsize updating (damping) rule is an important part of algorithm design here, as it is in most iterative algorithms. In the early stages of the iteration,  $\theta$  must be sufficiently large so that adequate progress toward the region in which the desired solution is located is made. Eventually,  $\theta$  must be decreased so that convergence is obtained. If  $\theta$  is decreased too rapidly, convergence may occur before the neighborhood of the desired solution is reached, thus resulting in a solution of poor quality (as in Runs 2, 3, and 4 of Section 5). If  $\theta$  is decreased too slowly, convergence will be delayed (or prevented, if  $\theta$  does not reach a sufficiently small value). For the early stages of the iteration, we have considered approaches in which  $\theta$  is either maintained at a constant value or decreased relatively slowly, e.g., either linearly or exponentially with a factor close to 1. Ultimately,  $\theta$  is decreased proportionally to either  $1/n$  or  $1/\sqrt{n}$ . Here,  $n$  can be either the number of iterations, or a function of the number of iterations that takes into account the number of “special events” that have taken place (e.g., QoS threshold crossings and/or the number of times that the identity of the dominant circuit has changed).<sup>20</sup> In our preliminary studies we used the following  $\theta$ -damping procedure

$$\theta(n+1) = \begin{cases} \theta(n) - \frac{\theta_0 - \theta_{\min}}{r}, & n \leq N \\ \frac{\theta_{\min}}{\sqrt{n-N}}, & n > N \end{cases}. \quad (30)$$

Thus,  $\theta$  is decreased linearly when  $n \leq N$  iterations, and subsequently decreased as  $1/\sqrt{n-N}$ , where  $n$  is the number of crossings of the QoS threshold plus the number of times there has been a change in the identity of the circuit that provides the maximum circuit-blocking probability. In Section 7.2 we discuss the techniques we have used to choose the initial value of  $\theta$ , as well as several damping rules.

<sup>20</sup>The occurrence of such special events suggests that  $\theta$  should be decreased because the search is close to the QoS boundary, and thus homing in toward the desired solution.

## 6.2 Generalized Approach

The use of the projection operation described above removes the component of  $\nabla S$  that is parallel to  $\nabla P_c$ . By doing so,  $\lambda$  is updated in a direction that increases throughput without increasing  $P_c$ . However, the typical consequence of doing so is that one or more of the other circuits will soon violate QoS. At a typical point in the search, it is common for several circuits to violate QoS or to be sufficiently close to the QoS boundary that QoS is in danger of being violated. We have observed behavior in which the chosen circuit for the projection alternates among two or three of the circuits, resulting in oscillatory behavior in which little progress is made toward the optimal solution. To mitigate this behavior, we have considered a generalized form of the projection operation in which several circuits are included in the projection. The inclusion of several circuits takes into consideration the fact that we are dealing with a number of constraints simultaneously. Thus, we would like to update  $\lambda$  in a direction that discourages violation of any of the QoS constraints.

To incorporate the QoS constraints associated with some or all of the circuits into the search-guiding mechanism, we introduce the quantity  $P_\Sigma$ , which is a function of the circuit-blocking probabilities  $P_1, P_2, \dots, P_J$ . In this study we have used the following simple, linear form for  $P_\Sigma$ :

$$P_\Sigma = \sum_{i \in \Sigma} P_i, \quad (31)$$

where  $\Sigma$  is a subset of  $\{1, 2, \dots, J\}$ . The vector  $\mathbf{D}$ , introduced in Eq. (28), is then rewritten as

$$\mathbf{D} = \begin{cases} \nabla S - \frac{\nabla S \cdot \nabla P_\Sigma}{\|\nabla P_\Sigma\|^2} \nabla P_\Sigma, & \text{if } \left\| \nabla S - \frac{\nabla S \cdot \nabla P_\Sigma}{\|\nabla P_\Sigma\|^2} \nabla P_\Sigma \right\| > \tau \|\nabla S\|, \\ \nabla S, & \text{otherwise} \end{cases} \quad (32)$$

This expression is identical to that of Eq. (28), except that  $P_\Sigma$  replaces  $P_c$  in the dot products and the projection operation is used only when the resulting value of  $\|\mathbf{D}\|$  is sufficiently large. The reason for using the projection operation only when it provides a sufficiently large value of  $\|\mathbf{D}\|$  is based on our experimental observation that (in some cases) the trajectory can reach a point at which  $\|\mathbf{D}\|$  is quite small. This behavior results in slow progress toward the optimal point, or even virtually total stopping of the trajectory, resulting in premature convergence. In fact, the trajectory can converge to a point interior to the admissible region (thus none of the circuit-blocking probabilities are at the specified value, a condition not characteristic of the optimal point). This behavior is especially prevalent when the set  $\Sigma$  is large (e.g., we have considered cases in which  $\Sigma$  contains all  $J$  circuits). It occurs when the gradients of  $S$  and  $P_\Sigma$  are nearly parallel to each other. Turning off the projection operation (typically for just a single iteration) permits the trajectory to escape from such undesirable points. We have found that a value of  $\tau = 0.1$  works well.

The projection vector  $\mathbf{D}$  specified by Eq. (32) removes the component of  $\nabla S$  that is in the direction of the gradient of the *average* blocking probability of the circuits included in  $\Sigma$ . Therefore,  $\lambda$  is updated in a direction that tends to increase throughput without increasing the average blocking probability of the circuits in  $\Sigma$ . We have found that the use of average blocking probability in this context can be helpful, even though the problem constraints are expressed in terms of the maximum blocking probability over the set of circuits.

Several approaches are possible for choosing the circuits that are to be included in  $\Sigma$ . For example, we may follow the approach of Eq. (28) and include all circuits for which  $\text{QoS} - \varepsilon \leq P_j(\lambda) \leq \text{QoS} + \varepsilon$ , provided that the dominant circuit is in the band (i.e., don't apply the projection if the blocking probability of the dominant circuit exceeds  $\text{QoS} + \varepsilon$ ). Under this approach, the projection term inhibits movement of  $\lambda$  in a direction that will increase the average blocking probability of the set of circuits that either violate QoS or are close to violating QoS. By avoiding an increase in the average value of these blocking probabilities, it is

hoped that none of the individual blocking probabilities will increase beyond the specified QoS value. However, if they do, the penalty term in Eq. (29) can take care of the situation. Circuits that are not close to violating QoS are not included in  $\Sigma$  because increasing blocking probability on these circuits is not detrimental to system performance.

A straightforward modification of this approach is the use of a single-sided band, i.e., use the same rule, but apply the projection only to circuits that violate QoS. Other approaches are discussed in Section 7.1.

We have found that the use of the generalized search algorithm discussed in this subsection (with several circuits included in the projection) usually provides improved convergence, as compared to the use of the unguided search (no projection term used) or the preliminary version of the guided search algorithm of Section 6.1 (which uses at most the dominant circuit in the projection). For example, use of this search procedure consistently finds the apparent constrained throughput maximum in searches of Network 1, regardless of the search starting point. Table 2 summarizes the results of searches based on the generalized search algorithm that use the same initial offered load vector as Runs 1 and 2 in Table 1, i.e., the  $\lambda_o$  values for Run 1 were obtained by scaling up values obtained from a search of the network with reduced capacity ( $T_i = 4$  and  $X_j = 3$ ), and  $\lambda_o = 1.75$  for Run 2. A two-sided  $\epsilon$  band was used, with  $\epsilon = 0.001$ . With the improved search, Run 1 found a solution that gives a slightly larger throughput value (17.002 vs 16.993) in far fewer iterations (378 vs 1375) than were required in the original run. Recall that with the basic search strategy and a starting point of  $\lambda_o = 1.75$ , we were only able to find the optimum solution after progressively restarting the search from the preceding solution, a procedure that required in excess of 6000 iterations. The improved search strategy found the optimum in only 829 iterations, and with no restarts.

Table 2 — Results of Improved Searches of Network 1

( $T_i = 8$ ;  $X_j = 4$ ; QoS = 0.3;  $\theta_o = 0.4$ ,  $\theta_{\min} = 0.2$ ,  $N = 200$ ;  $\gamma_o = 5$ ,  $c = 10$ ,  $d = 10$ ,  $\epsilon = 0.001$ ,  $\lambda_{\min} = 0.05$ )

Run	$\lambda_o$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_5$	$\lambda_6$	$\lambda_7$	$\lambda_8$	$\lambda_9$	$\lambda_{10}$	$S(\lambda)$	Iterations
1	scaled	5.9754	5.8101	1.6877	0.0549	1.9639	0.05	6.0326	0.05	1.5523	0.6485	17.0023	378
2	1.75	5.9510	5.8674	1.7365	0.0508	1.9616	0.05	6.0172	0.05	1.6259	0.5167	17.0041	829

Figure 6, which should be compared with Fig. 5, shows the trajectory of Table 2, Run 1, projected onto the  $(\lambda_1, \lambda_3)$  plane. Note in Fig. 6(a) that this search more directly ascends the throughput gradient and the unproductive oscillation, which dominated Fig. 5, is not present. We attribute both of these qualities to the use of the projection to guide the search along the QoS boundary. The expanded local view shown in Fig. 6(b) shows that the search does ascend the QoS boundary and finds a solution very near the maximum. As in Fig. 5, because all of the  $\lambda_j$ 's vary throughout the search, the landscape also varies throughout the search. The landscape shown in Fig. 6 is again based on the values of  $\lambda_2$  and  $\lambda_4, \dots, \lambda_{10}$  at the last iteration.

#### The Impact of the Value of $\epsilon$ on Performance

We have experimented with the width of the band in which the projection is used and found that  $\epsilon = 0.001$  strikes a balance between speed of convergence and quality of solution. Typically, a smaller  $\epsilon$ -value delivers a slightly higher throughput value at the expense of added iterations. For example, we repeated Run 2 of Table 2 with  $\epsilon = 0.0005$  and found in 1450 iterations a solution that gives  $S(\lambda) = 17.0056$ . A larger  $\epsilon$  value appears to hinder both the quality of the solution and the speed with which it is obtained. With  $\epsilon = 0.005$ , the Table 2, Run 2 search found in 1367 iterations, a solution that gives  $S(\lambda) = 16.9856$ .

To explain the impact of the value of  $\epsilon$  on both the speed of convergence and quality of the solution, recall our earlier observation that the use of the projection distorts the objective function, thus potentially

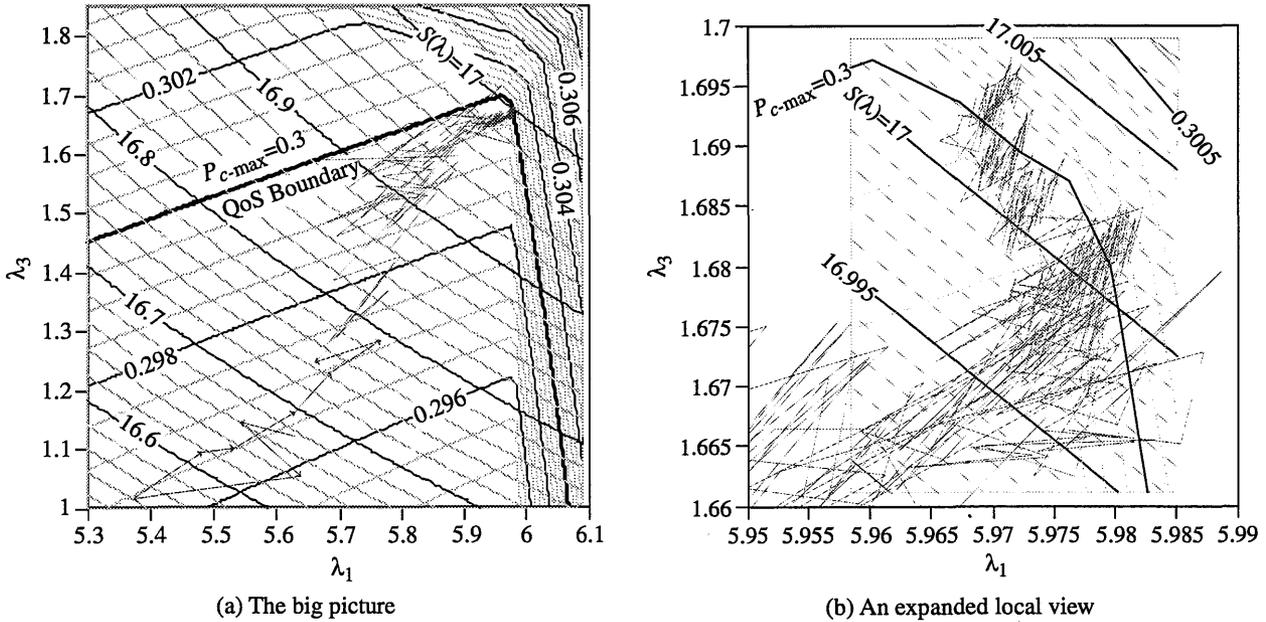


Fig. 6 — Search trajectory in  $\lambda_1$  and  $\lambda_3$  superimposed on throughput and  $P_{c-\max}$  contours;  $\lambda_2$ , and  $\lambda_4, \dots, \lambda_{10}$  are fixed at the values shown in Table 2, Run 1

making it difficult to reach the true (constrained) optimal point. When  $\epsilon$  is large, it is more likely that the blocking probability of the dominant circuit is located within the  $\epsilon$  band, particularly at the final stages of the iteration when  $\theta$  is small. Thus, the distortion is included in the objective function at most, if not all, iterations. By contrast, when  $\epsilon$  is very small, the distortion is not present at every iteration, thus potentially permitting the search to converge to a point that is not directly affected by the projection. However, when  $\epsilon$  is too small the projection is rarely applied, and thus has little effect in guiding the search toward the optimal solution.

### 6.3 On the Development of Improved Iterative Search Algorithms

Based on the results discussed above, as well as on similar results for many other test runs, we have been able to enhance the iterative search algorithm of Section 6.2. A crucial observation has been that it is best to turn off the projection term at some point during the iteration. When this term is kept active throughout the iteration, it is common to see the search approach the neighborhood of the optimal solution, only to proceed past it, eventually converging to a point relatively far from the optimal. This behavior is especially pronounced when  $\epsilon$  is relatively large. However, when the projection is turned off, the final approach to the optimal solution can be made without the presence of distortion.

However, as suggested by the preliminary results presented in this section, the use of the projection can be helpful, at least at some point in the iteration. Although  $\epsilon = 0.001$  was a relatively good choice under the constraint that the projection (with a fixed value of  $\epsilon$ ) is applied throughout the entire search, the option of turning off the projection before the final approach to the optimal point raises other possibilities as well. For example, during the early stages of the iteration it may be appropriate to include a potentially large number of circuits into the projection (either by means of a large value of  $\epsilon$  or an alternative criterion for including circuits in the set  $\Sigma$ ). Consequently, application of the projection, which tends to prevent the increase of the average value of the blocking probability of a large number of circuits, also tends to keep each of the blocking probabilities from straying too far into the inadmissible region. (Recall that the penalty term is present to take care of those circuits that enter the inadmissible region.)

The choice of stepsize ( $\theta$ ) damping rule is also critical to the performance of the algorithm. The parameter  $\theta$  must be large enough at the beginning of the iteration to make significant progress toward the region of the optimal solution, and must decrease in a manner that permits convergence to the optimal solution. An excessively fast decrease can cause premature convergence of the algorithm, and hence failure to reach the optimal solution, whereas an excessively slow decrease can prevent convergence within the desired time constraints. In fact, we demonstrate in Section 8 that the use of a good stepsize rule can provide reliable convergence to the optimal solution, even when the projection is not used; the slow convergence of the runs described in Section 6.1, including the need to restart with a the original stepsize, can be attributed largely to the use of too small a stepsize. However, we also show that in many examples the use of the projection does indeed enable the search to reach the neighborhood of the optimal solution considerably faster than is possible when the projection is not used.

Care must also be taken in the choice of several other parameters used in the algorithm, such as the choice of  $c$  (used in updating the Lagrange multipliers in Eq.(24) and  $d$  (which weights the penalty term in Eq. (21)). The use of  $c = d = 50$  worked well for high values of QoS (e.g.,  $\geq 0.2$ ), but not for more realistic values. The relatively poor performance for low values of QoS was observed because the gradient terms  $\partial P_j / \partial \lambda_i$  were too small to drive the search back into the admissible region at the low offered loads that are characteristic of low values of QoS. This problem has been mitigated by weighting the constraint-violation terms by  $1 / \sqrt{\text{QoS}}$  (while maintaining  $c = d = 50$ ):

$$\frac{\partial L(\boldsymbol{\lambda}, \boldsymbol{\gamma})}{\partial \lambda_i} = D_i + \alpha \sum_{j=1}^J 1(P_j(\boldsymbol{\lambda}) > Q_j) \frac{\partial P_j}{\partial \lambda_i} \frac{[d(Q_j - P_j(\boldsymbol{\lambda})) - \gamma_j]}{\sqrt{Q_j}}, \quad (33)$$

where  $\alpha$  is a “kick-up” factor that can be updated (increased from an initial value of 1) as necessary. For example,  $\alpha$  can be increased if too many consecutive inadmissible solutions are observed, or decreased if too many consecutive admissible solutions are observed (after the inadmissible region has been entered at least once).

In Section 7 we define 18 versions of our iterative algorithm, which differ in their use of projection and stepsize rules. Some of these versions use no projection at all, while others use it only during the relatively early stages of the iteration. We discuss several options for stepsize rules, and indicate how system parameters can be chosen to provide robust performance. The 18 versions of the algorithm have been tested extensively in a series of “core runs” for three different networks and various “uniform” parameter sets; these runs are described in Section 8. These results demonstrate that some versions are capable of providing nearly optimal performance (e.g., admissible solutions that provide 98% of the maximum admissible throughput value) considerably earlier in the iteration process than others. The key to obtaining “good” (although suboptimal) performance rapidly appears to be the use of the projection with an appropriate rule for the determination of the set  $\Sigma$  (of circuits to be included), combined with a good stepsize rule. However, there is relatively little difference in performance in either the quality of the ultimate solution or the speed with which it is reached for cases in which a solution that is extremely close to the optimal is required (e.g., one that provides 99.9% of the optimal throughput value). We have also found a number of examples (typically with relatively “extreme” parameter values) for which some versions of the algorithm provide higher throughput than others; however, this difference is rarely more than 1%.

## 7. ALTERNATIVE VERSIONS OF THE ALGORITHM

The preliminary studies described in Section 6 have served as the basis for the development of an improved algorithm for the determination of optimal offered load. The different versions of the algorithm are all based on the iterative procedure described by Eq. (33); they differ in their use of the dot-product projection and in the stepsize update rule. In this section we discuss the various options that we have considered for the use of the projection and stepsize rules, as well as the interrelationships between these two critical aspects of algorithm design. We also discuss considerations relating to the choice of other parameters in the algorithm.

Specifically, in Section 7.1 we discuss alternative projection rules, i.e., alternative ways to specify the set  $\Sigma$  of circuits that are to be included in the projection. In Section 7.2 we discuss factors relating to the stepsize rule, and present four particular stepsize rules that we have used extensively in our tests of the algorithm. Finally, in Section 7.3 we characterize 18 versions of the algorithm in terms of their stepsize and projection rules.

In our discussion of projection rules, it is implicitly assumed that  $Q_j = \text{QoS}$ ,  $j = 1, \dots, J$ , (i.e., that all circuits are subject to the same constraint on maximum blocking probability), although it is certainly possible to define projection rules that incorporate different QoS values. In Section 9.3 we discuss several examples in which the circuits are divided into two groups of equal size, which are subject to widely differing values of the QoS constraint. In those examples, significantly better performance is obtained when the projection operation is not used. However, the results for many of the examples presented in this report (with equal QoS values) demonstrate that appropriate use of the projection operation can often provide improved performance, in terms of speed of convergence and/or quality of solution.

### 7.1 Projection Rules

The projection rule, as described in Section 6.2, guides the search by removing the component of the throughput gradient that is parallel to  $\nabla P_\Sigma$ , where  $P_\Sigma = \sum_{j \in \Sigma} P_j$ , for some subset  $\Sigma$  of  $\{1, 2, \dots, J\}$ . For convenience, we recall the vector  $\mathbf{D}$  that was defined as

$$\mathbf{D} = \begin{cases} \nabla S - \frac{\nabla S \cdot \nabla P_\Sigma}{\|\nabla P_\Sigma\|^2} \nabla P_\Sigma, & \text{if } \left\| \nabla S - \frac{\nabla S \cdot \nabla P_\Sigma}{\|\nabla P_\Sigma\|^2} \nabla P_\Sigma \right\| > \tau \|\nabla S\|, \\ \nabla S, & \text{otherwise} \end{cases}, \quad (34)$$

where we have used  $\tau = 0.1$ . The effect of the projection is to remove the component of the throughput gradient that is in the direction of the gradient of the sum of the blocking probabilities (or, equivalently, the gradient of the average blocking probability<sup>21</sup>) of the circuits included in  $\Sigma$ . By including several circuits in  $\Sigma$ , it is possible to discourage (although not necessarily prevent) the blocking probabilities of these circuits from exceeding the QoS constraint. In addition, the oscillatory behavior that results from the use of a single circuit (the identity of which typically alternates among a small set of circuits) in the dot product is reduced. However, it must be acknowledged that the use of the projection is a heuristic approach. In particular, efforts to avoid increase in the average blocking probability of the circuits in  $\Sigma$  do not take into consideration the true nature of the capacity constraints, i.e., that the blocking probability on each circuit must not exceed the QoS constraint; we rely on the penalty term in Eq. (33) to enforce these constraints. Thus, it is hard to predict a priori whether the use of the projection based on a direction defined by several circuits will be helpful. The performance results presented in Section 8 demonstrate that, if used judiciously, the projection can, in fact, be very helpful.

The projection rule is defined by specifying the set  $\Sigma$ . Several distinct alternatives for choosing  $\Sigma$  are discussed below.

*No projection:*

$$\Sigma = \emptyset = \text{the empty set}$$

This is simply the ‘‘basic search’’ procedure that was introduced in Section 4, for which  $\mathbf{D} = \nabla S$ . Although the performance results presented in Section 5 demonstrated the deficiencies of this approach and

<sup>21</sup>Here, we do not weight the blocking probabilities by the corresponding offered loads; thus, by average blocking probability we simply mean the numerical average of the blocking probabilities of the individual circuits, rather than the fraction of calls to the overall network that are blocked. By contrast, in Section 10 when we study an alternative version of the QoS constraint that is defined in terms of overall average blocking probability, the individual blocking probabilities are weighted by the corresponding offered loads, thereby producing the fraction of calls to the overall network that are blocked.

motivated the use of the projection operation, we have concluded that  $\Sigma = \emptyset$  is best during the final approach to the optimal solution, because use of this approach does not introduce the distortion that is associated with the dot-product terms. Also, we demonstrate in Section 8 that the use of  $\Sigma = \emptyset$  throughout the search can sometimes result in convergence to the optimal solution, provided that a good stepsize rule is used. However, convergence to a “good” solution (e.g., one that provides an admissible solution with throughput of 98% of the maximum admissible throughput) is typically much slower than the approaches that do make use of the projection.

*Projection based on the dominant circuit only:*

$$\Sigma = \begin{cases} \{j: P_j = P_{\max}\}, & \text{if } \text{QoS} - \varepsilon \leq P_{\max} \leq \text{QoS} + \varepsilon \\ \emptyset, & \text{otherwise} \end{cases},$$

where

$$P_{\max} = \max\{P_1, P_2, \dots, P_J\}.$$

Here,  $\Sigma$  consists of the dominant circuit only,<sup>22</sup> provided that its blocking probability is within  $\varepsilon$  of QoS; otherwise  $\Sigma$  is the empty set. This is the approach described in Section 6.1.

An alternative version of this rule is

$$\Sigma = \{j : P_j = P_{\max}\},$$

i.e.,  $\Sigma$  consists of the dominant circuit only as above, but differs in that the projection is used regardless of the value of  $P_{\max}$ . This case is obtained by letting  $\varepsilon = 1$ . The motivation behind this approach is to turn the trajectory away from the QoS constraint contour even before it comes close to it.

*Projection based on several circuits, with  $\Sigma$  defined in terms of a fixed band:*

$$\Sigma = \begin{cases} \{j: \text{QoS} - \varepsilon \leq P_j \leq \text{QoS} + \varepsilon\}, & \text{if } \text{QoS} - \varepsilon \leq P_{\max} \leq \text{QoS} + \varepsilon \\ \emptyset, & \text{otherwise} \end{cases},$$

i.e.,  $\Sigma$  contains all circuits in a band of width  $2\varepsilon$ , centered about the QoS contour, provided that the dominant circuit is in this band. The inclusion of several circuits in the dot product discourages the increase of their average blocking probability. This, in turn, tends to discourage (although not necessarily prevent) violation of the QoS constraint on these circuits.<sup>23</sup> If a large value of  $\varepsilon$  is used, then most (or perhaps even all) of the circuits are included. In this case, the trajectory tends to stay within the admissible region most of the time. Although this approach has been shown to be useful in finding good solutions rapidly, it is ineffective in finding the optimal solution. In some examples, the offered-load trajectory has approached the vicinity of the optimal point, only to proceed past it to a solution of lower throughput because of the distortion associated with the dot-product term, which prevented the search from proceeding in the direction of increasing throughput gradient.

A slight modification of the above produces:

$$\Sigma = \begin{cases} \{j: \text{QoS} \leq P_j \leq \text{QoS} + \varepsilon\}, & \text{if } \text{QoS} < P_{\max} \leq \text{QoS} + \varepsilon \\ \emptyset, & \text{otherwise} \end{cases}.$$

<sup>22</sup>In the event of ties, an arbitrary rule can be used to choose one circuit as the dominant one.

<sup>23</sup>The constraint-violation terms (i.e., the summation terms in the Augmented Lagrangian function of Eq. (19) and in the partial-derivative expressions of Eqs. (21, 29, and 33)) become active when the QoS constraints are violated.

This is the same as the preceding case, except that  $\Sigma$  contains only the circuits for which the QoS constraint is violated. Thus, the “basic search” is used inside the admissible region, and the projection with several circuits is used in the band of width  $\epsilon$  located in the inadmissible region. This approach appears less likely to result in movement away from the neighborhood of the optimal point, apparently because the distorting effects of the dot product are not present when the search point is located in the admissible region.

*A fixed number of circuits in  $\Sigma$ :*

$$\Sigma = \{\text{the } J' \text{ circuits with the largest blocking probabilities}\},$$

where  $J'$  is a parameter that represents the number of circuits to be included in the projection. Rather than use a fixed band to determine which circuits are to be included, this approach selects the  $J'$  circuits that have the largest blocking probability, independently of the distribution of blocking probability values.

*Projection based on several circuits, with  $\Sigma$  defined in terms of relative sizes of blocking probabilities:*

$$\Sigma = \{j: P_j \geq P_{\min} + v(P_{\max} - P_{\min})\},$$

where

$$P_{\min} = \min\{P_1, P_2, \dots, P_J\}$$

and  $v \in [0, 1]$ .

This approach is similar to the previous one in that it does not depend on a fixed band. However, the number of circuits to be included is not fixed, but instead depends on both the value of the parameter  $v$  and the distribution of the blocking probabilities over the range from  $P_{\min}$  to  $P_{\max}$ . For example, if  $v = 0.5$ , as many as  $J - 1$  circuits (if the next-to-smallest blocking probability is greater than the average of  $P_{\min}$  and  $P_{\max}$ ) or as few as one circuit (if the next-to-largest blocking probability is less than the average of  $P_{\min}$  and  $P_{\max}$ ) could be included. As  $v$  is decreased, the number of circuits contained in  $\Sigma$  tends to increase, e.g., if  $v = 0$ , then all circuits are included.

This approach uses the projection even before the trajectory reaches the vicinity of the QoS constraint contour, and  $\Sigma$  includes those circuits whose blocking probability is sufficiently close to  $P_{\max}$ . Thus the projection begins to guide the search at the earliest stages of the iterative process. Our experience has been that use of this approach with  $v = 0.2$  (which includes a relatively large number of circuits, but not those with blocking probability very close to  $P_{\min}$ ) often guides the search rapidly to a reasonably good solution. Once this point is reached, the projection term can be turned off (i.e., set  $\Sigma = \emptyset$ ) for the final approach to the optimal solution.<sup>24</sup>

## 7.2 Stepsize Rules

The choice of stepsize ( $\theta$ ) damping rule is critical to the performance of the algorithm, as it is in most iterative algorithms. The two basic components of the stepsize rule are the initial stepsize and the decay rule. The parameter  $\theta$  must be large enough at the beginning of the iteration to make significant progress toward the region of the optimal solution. Typically, we have chosen the initial stepsize  $\theta_0$  on the basis of a short pilot run in which the projection is not used. It is chosen so that, starting at  $\lambda_j = 0$ , the trajectory exits the admissible region for the first time after about five to fifteen iterations. The same value of  $\theta_0$  is used whether or not the projection is used in the actual search.

We have found that a first exit point of five iterations works well for large values of the QoS constraint, e.g., 0.3. However, this approach appears to produce an excessively large initial stepsize for small values of the QoS constraint, e.g., 0.001. Thus, in some of our examples for QoS = 0.001 we have used an initial value of  $\theta$  that is half that produced by using the rule based on exiting the admissible region for the first time at the fifth iteration. To explain the difference in behavior, consider the terms  $\partial S_j / \partial \lambda_j$  in Eq. (21), which are usually

<sup>24</sup>Alternatively, an intermediate phase can be introduced in which a larger value of  $v$  is used. Doing so may guide the search to a point closer to the optimum, at which point the projection term would be turned off.

significantly larger than the terms  $\partial S_j / \partial \lambda_p$ , when  $j \neq i$ . These “diagonal” terms have a value close to 1 at the low offered loads that are characteristic of low blocking probability. However, these terms are considerably smaller at offered loads characteristic of significantly higher blocking probability (typical average values are approximately 0.3 in many of our examples for QoS = 0.3). Thus the use of smaller stepsizes at low QoS values compensates for the larger values of throughput gradient at the corresponding offered loads.

Equally important is the rate of decrease of  $\theta$ . Too fast a decrease can cause premature convergence of the algorithm, and hence failure to reach the optimal solution, whereas too slow a decrease can prevent convergence within the desired time constraints.

The following guidelines are often used for choosing stepsize ( $\theta_k$  at iteration  $k$ ) in iterative problems:

$$\lim_{k \rightarrow \infty} \theta_k = 0$$

and

$$\lim_{k \rightarrow \infty} \sum_{j=1}^k \theta_j = \infty.$$

For example,  $\theta_k$  proportional to  $1/k$  satisfies these conditions. Thus, in an infinitely long search, it would be appropriate to decrease the stepsize proportionately to  $1/k$  in the later stages of the search.<sup>25</sup> However, since we typically limit our search to 1000 iterations, it is necessary to make compromises in the design of stepsize rules. Therefore, there is no guarantee that the true optimal solution will be found. Furthermore, since unimodality of the constrained throughput function has not been proven (hence there may be local maxima), there is no guarantee that any stepsize rule would produce the globally optimal solution. Nevertheless, the various versions of our algorithm, which use several different stepsize rules, have performed well, as we demonstrate later in this report.

Figure 7 shows the four stepsize rules used in the set of “core” runs. In each case, the total number of iterations per run is 1000, and the stepsize is reduced to 0.001 of its original value at that point. The stepsize rules divide the 1000 iterations into either two phases (Rule 1) or three phases; each such phase is characterized by a change in the value and/or rate of decrease of the stepsize. In addition, the projection rule may change as the stepsize rule enters a new phase. The segments of the curves are either constant or exponentially decaying (hence linear on the log scale shown here). A particular version of the algorithm is characterized by the joint specification of the stepsize rule and projection rule. In Section 7.4 we summarize the 18 versions of the algorithm that were used in the core runs.

The motivation behind the use of a constant stepsize during the early part of the iteration is that a large stepsize is needed to approach the neighborhood of the optimal solution. Also, the use of a constant stepsize permits us to isolate the effect of the terms  $\partial L / \partial \lambda_i$  on the trajectory of the offered loads, and thus to assess the impact of particular projection rules and parameter values without having to address the impact of the concurrent decrease in stepsize (Eq. (20)). However, the disadvantage of this approach is that large oscillations persist until the stepsize is reduced. A slowly decaying stepsize, which gradually decreases the level of oscillation, was also considered for the first phase (Rule 3); this rule is based on the assumption that the neighborhood of the optimal solution is reached rather quickly, suggesting that we may be able to increase the speed of convergence by trying to home in on the optimal point relatively early in the search.

The motivation behind Rule 2, in which the stepsize is cut by 50% when phase 2 is entered, is based on a comparison of “effective stepsizes” when the projection is used to when it is not. Since  $\|D\| \leq \|\nabla S\|$ , it may be helpful to decrease the stepsize when the projection operation is turned off to compensate for the increase in effective stepsize that would result if this is not done.<sup>26</sup>

<sup>25</sup>It is helpful to keep the stepsize at a relatively large value until the neighborhood of the optimal solution is reached. The rate of decrease of  $1/k$  is often too rapid for small values of  $k$ .

<sup>26</sup>Our studies suggest that  $\|D\|$  tends to decrease as more circuits are included in the set  $\Sigma$ .

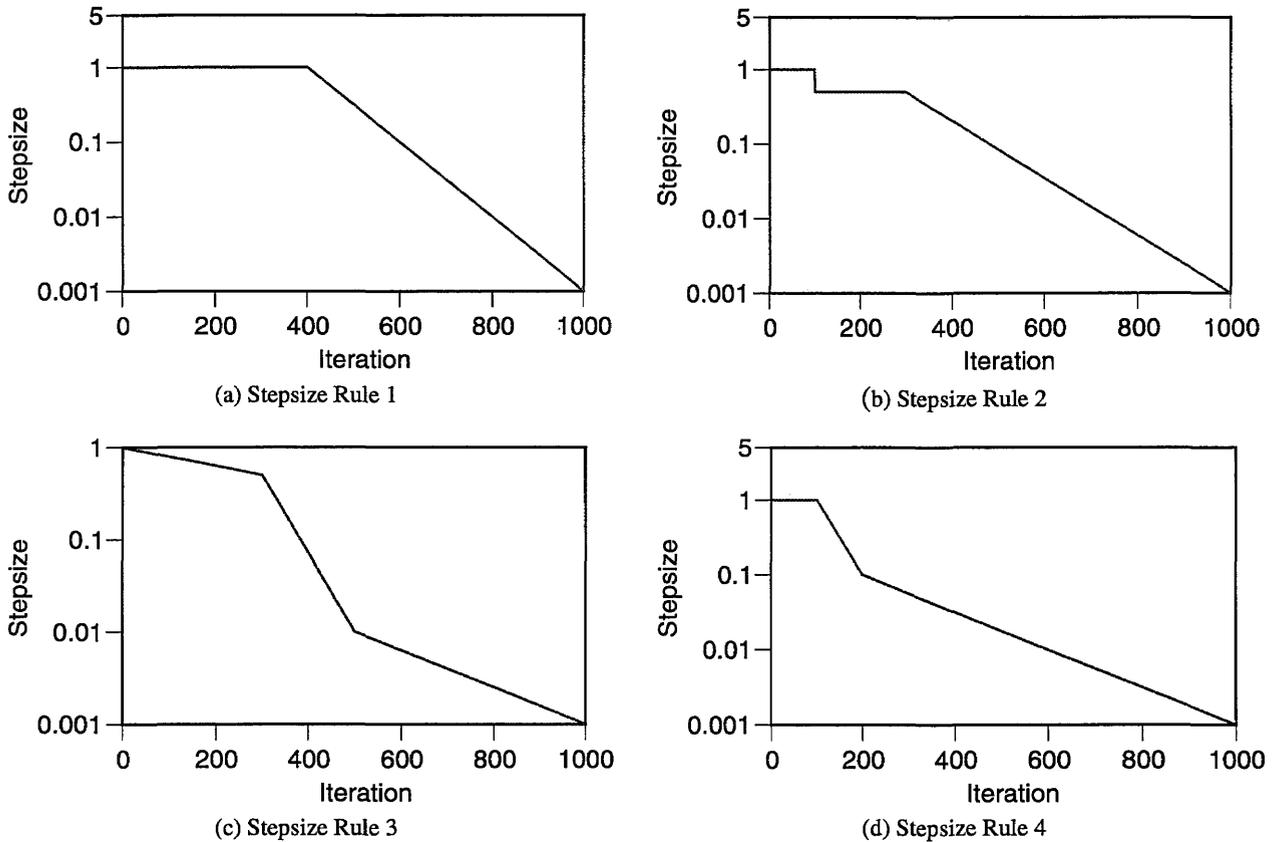


Fig. 7 — Normalized stepsize as a function of iteration number

### 7.3 The Update Equation

Recall from Section 6.3 that the equation used to update the offered-load values is

$$\frac{\partial L(\boldsymbol{\lambda}, \boldsymbol{\gamma})}{\partial \lambda_i} = D_i + \alpha \sum_{j=1}^J 1(P_j(\boldsymbol{\lambda}) > Q_j) \frac{\partial P_j}{\partial \lambda_i} \left[ \frac{d(Q_j - P_j(\boldsymbol{\lambda})) - \gamma_j}{\sqrt{Q_j}} \right] \quad (35)$$

In most of our production runs we have used the following parameter values:

$c = 50$  (used in Lagrange multiplier update, Eq. (24));

$\gamma_0 = 5$  (initial value of Lagrange multipliers used in Eq. (24));

$d = 50$  (coefficient in penalty term of Eq. (35)).

$\alpha =$  “kick-up” factor (initial value = 1; it is doubled whenever 10 consecutive iterations are inadmissible, and halved whenever five consecutive iterations are admissible).

### 7.4 Summary of Algorithm Versions Used in the Core Runs

To test the effectiveness of our algorithm, and to determine which versions perform best, we have extensively tested 18 versions of the algorithm on several network examples. Our basic test consisted of the “core” runs, described in Section 8, in which each of these versions was tested for three distinct networks and a variety of parameter values.

The different versions of the algorithm are based on the projection formulations discussed in Section 7.1 and the stepsize rules discussed in Section 7.2. The versions are identified by the notation X.Y, where

$X \in \{1, 2, 3, 4\}$  specifies the stepsize rule defined in Fig. 7 and  $Y \in \{0, 1, 2, 3, 4\}$  specifies the projection rule. Sometimes in our discussion, we consider several versions together in a group. For example, when we refer to versions of type 1.Y, we mean versions that use Stepsize Rule 1 and any of the projection rules; similarly, X.0 refers to versions that use Projection Rule 0 and any of the stepsize rules.

In particular, the stepsize rules are specified as:

X = 1: Stepsize Rule 1 (Fig. 7(a));

X = 2: Stepsize Rule 2 (Fig. 7(b));

X = 3: Stepsize Rule 3 (Fig. 7(c));

X = 4: Stepsize Rule 4 (Fig. 7(d)).

The projection rules are specified as:

Y = 0: no projection used at any time during the iteration;

Y = 1: the projection based on only the dominant circuit is used throughout the entire iteration;

Y = 2: the projection with  $\nu = 0.2$  is used during phase 1, and no projection is used during phases 2 and 3;

Y = 3: the projection with  $\nu = 0.8$  is used during phase 1, and no projection is used during phases 2 and 3;

Y = 4: the projection with  $\nu = 0.2$  is used during phase 1,  $\nu = 0.8$  is used during phase 2, and no projection is used during phase 3.

Thus the projection rules that have been tested cover a wide range that includes the use of no projection ( $Y = 0$ ), one circuit only ( $Y = 1$ ), many circuits ( $Y = 2$ ), few circuits ( $Y = 3$ ), and switching from many circuits initially to few ( $Y = 4$ ). In all cases except  $Y = 1$ , the projection is not used during the final phase(s). Of the 20 possible X.Y combinations, 18 versions of the algorithm (all except 1.3 and 1.4) were actually tested (the latter is undefined because Stepsize Rule 1 has only two phases).

## 8. TESTING THE ALGORITHM: THE CORE RUNS

We have performed extensive testing of our algorithm on three networks, namely Network 1 of Fig. 3 and Networks 2 and 3 shown in Figs. 8 and 9. Network 2, like Network 1, supports 10 circuits, whereas Network 3 supports eight. Our basic test consisted of the “core” runs, in which the 18 versions of the algorithm described in Section 7.4 were tested for these three networks and a variety of parameter values. The core runs are characterized by “uniform” parameter sets, as defined below. Results are presented for blocking-probability QoS constraints of 0.001 and 0.3, thus illustrating performance evaluation over a wide range of values of this parameter. In the core runs, all 18 versions of the algorithm provided nearly identical throughput, although there were significant differences in the speed of convergence. Additional (noncore) examples are discussed in Section 9, where it is shown that some versions are more reliable than others in terms of convergence to the optimal throughput.

### 8.1 Description of the Core Runs

The “core” runs consist of tests of the 18 versions of the algorithm on Networks 1, 2, and 3 for “uniform” network parameters. By uniform we mean that all nodes have the same number of transceivers ( $T_1 = \dots = T_N$ ), and all circuits have the same threshold ( $X_1 = \dots = X_J$ ) on the permitted number of calls. In our discussion of particular network examples, the shorthand notation “ $T_i = 6$ ” means  $T_i = 6$ ,  $i = 1, 2, \dots, N$  (i.e., there are six transceivers at each node). Similarly, “ $X_j = 4$ ” means  $X_j = 4$ ,  $j = 1, 2, \dots, J$  (i.e., there are at most four calls on any circuit at any time). The subscript  $i$  normally refers to node number, and the subscript  $j$  normally refers to circuit number.

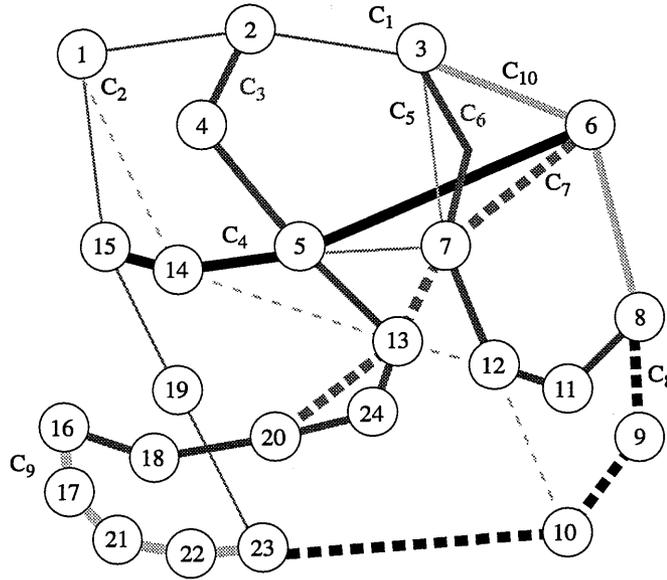


Fig. 8 — Network 2 (24 nodes, 10 circuits)

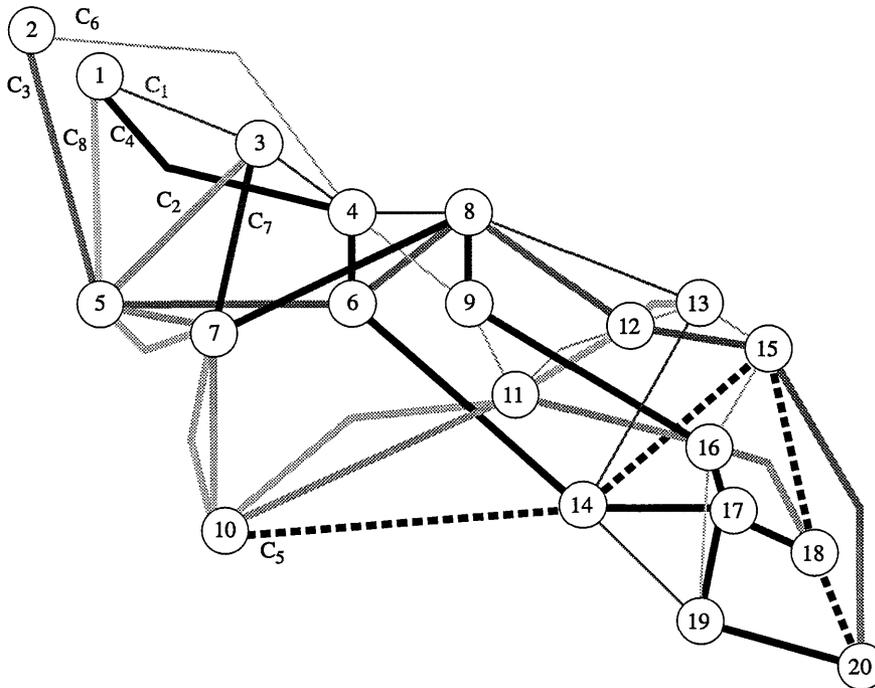


Fig. 9 — Network 3 (20 nodes, 8 circuits)

In addition, in the core runs all circuits must satisfy the same QoS constraint on blocking probability (i.e.,  $Q_1 = \dots = Q_J$ ). Again, we use a shorthand notation; e.g., “ $Q_j = 0.001$ ” indicates that  $Q_j = 0.001, j = 1, 2, \dots, J$ . We have performed core runs for  $Q_j = 0.001$  and 0.3. Furthermore, there are no restrictions on the offered loads; thus,  $\lambda_{\min} = 0$  (which means that the offered load values are not prevented from decreasing to zero), and no bounds are placed on the maximum values of the offered loads. Finally, the same rule is used to pick the initial stepsize value  $\theta_0$  in all of the core runs, namely that  $\theta_0$  is chosen so that the first inadmissible solution is obtained at the fifth iteration for the case in which the projection is not used (see Section 7.2); the value of  $\theta_0$  is the same for all 18 versions of the algorithm for each network example (i.e., for a specified network topology, set of circuits, number of transceivers, admission-control policy, and QoS value), but different for different network examples.

We have also performed many noncore runs in which one or more of the following conditions apply: not all nodes have the same number of transceivers, not all of the circuits have the same threshold, not all  $Q_j$  values are the same,  $\lambda_{\min} > 0$  (which means that each circuit is guaranteed at least the specified offered load value  $\lambda_{\min}$ ), and/or different rules are used to pick the initial stepsize  $\theta_0$ . These cases are discussed in Sections 9, 11, and 12.

In the core runs, for all three networks we considered examples with  $T_i = 6$ , (i.e., six transceivers at each node) and  $X_j = 4$  (i.e., at most four ongoing calls on any circuit).<sup>27</sup> In addition, we also considered Network 1 with  $T_i = 6$  and  $X_j = 3$  and Network 3 with  $T_i = 8$  and  $X_j = 6$ . In all cases, each of the 18 versions of the algorithm was run for 1000 iterations.

The versions have been evaluated based on their ability to find the optimal solution, as well as on their speed. The results of these tests are summarized in three types of tables, which we refer to as “milestone” tables, “stopping-rule” tables, and “offered-load” tables. Several examples of these tables are provided in Sections 8.2 through 8.4; additional tables are provided in Appendix A. Milestone tables show the number of iterations required to reach various levels of throughput performance (for a total of 1000 iterations in each run); stopping-rule tables show the performance that is achieved when the runs are stopped when convergence is obtained (for several values of the convergence parameter), rather than for a fixed number of iterations; “offered-load” tables show the offered loads and normalized blocking probabilities produced by the optimization algorithm. Before discussing the performance of the algorithm over a wide range of scenarios, we discuss the interpretation of these tables using a particular example.

## 8.2 Milestone Tables

Table 3 provides (for each of the 18 versions of the algorithm) a summary of the important “milestones” of each run for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ . The initial values of the offered loads are zero for all circuits. The “benchmark throughput” value (listed in the figure caption) is the highest value of admissible throughput that was obtained for any of the 18 runs. Table 4 provides similar results for  $Q_j = 0.3$ ; milestone tables for additional core runs are provided in Appendix A. The columns in the table are summarized as follows:

- **algo**: the version of the algorithm, in X.Y notation;
- **adm thruput at first exit**: the upper entry indicates the last iteration before the trajectory leaves the admissible region for the first time; the lower entry shows the percentage of the benchmark throughput value obtained at this iteration;
- **best adm thruput**: the iteration at which the highest admissible throughput is found, and the percentage of benchmark throughput obtained at this iteration;
- **last adm thruput**: the last iteration for which the solution is in the admissible region, and the percentage of benchmark throughput obtained at this iteration;
- **95.0%**: the first iteration at which the solution is admissible and equal to at least 95% of the benchmark throughput value;<sup>28</sup>

and similarly for the higher percentage values.<sup>29</sup>

<sup>27</sup>Had we set  $X_j = 6$ , the resulting system would be an “uncontrolled” network, in which all calls would be accepted as long as network resources (i.e., transceivers) are available to service them. For the case of QoS = 0.3, there is little difference in throughput; however, it is more difficult to find the optimal solution. For the case of QoS = 0.001, there is a modest increase in throughput, with somewhat increased difficulty of finding the optimal solution. The impact of the admission-control policy on achievable throughput and robustness of the algorithms is discussed in Section 13.

<sup>28</sup>The throughput value (whether admissible or not) does not increase monotonically; thus the throughput can decrease to values less than the milestone values, as in most iterative algorithms.

<sup>29</sup>Blank entries (if any) indicate that the corresponding milestones were not achieved by that particular version of the algorithm.

Table 3 — Milestone Table for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$   
(benchmark throughput: 2.6645)

algo	adm thruput at first exit	best adm thruput	last adm thruput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4 67.54%	986 99.99%	1000 99.99%	51	517	633	707	775
1.1	7 87.13%	976 99.99%	999 99.97%	470	569	576	675	778
1.2	13 84.77%	993 99.98%	998 99.96%	52	570	640	701	821
2.0	4 67.54%	991 99.99%	999 99.97%	51	376	468	468	786
2.1	7 87.13%	986 99.99%	1000 99.96%	145	390	461	591	750
2.2	13 84.77%	993 99.99%	1000 99.94%	52	439	563	661	828
2.3	9 90.91%	986 99.99%	1000 99.97%	92	468	554	597	735
2.4	13 84.77%	990 100.00%	999 99.98%	52	464	479	602	743
3.0	4 67.39%	856 99.99%	980 99.98%	252	361	397	434	508
3.1	7 86.73%	996 100.00%	1000 99.98%	135	368	378	414	534
3.2	13 83.80%	999 99.97%	999 99.97%	55	397	403	429	536
3.3	9 90.24%	941 99.99%	1000 99.97%	135	389	415	468	566
3.4	13 83.80%	959 99.99%	999 99.98%	55	353	389	411	548
4.0	4 67.54%	985 99.99%	998 99.97%	51	191	198	220	571
4.1	7 87.13%	956 100.00%	1000 99.98%	137	171	233	233	587
4.2	13 84.77%	979 100.00%	998 99.98%	52	193	203	379	626
4.3	9 90.91%	993 100.00%	999 99.99%	92	208	219	391	648
4.4	13 84.77%	892 99.99%	999 99.98%	52	174	194	288	647

first column: version of algorithm in X.Y notation

upper entries (columns 2-9): iteration at which milestone is reached

lower entries (columns 2-4): percentage of benchmark throughput at this iteration

Table 4 — Milestone Table for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$   
(benchmark throughput: 11.5380)

algo	adm thruput at first exit	best adm thruput	last adm thruput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4 84.29%	930 99.92%	997 99.91%	41	467	517	601	833
1.1	6 84.95%	984 99.90%	995 99.90%	49	496	561	626	914
1.2	31 96.05%	973 99.96%	999 99.96%	30	78	521	609	770
2.0	4 84.29%	996 99.91%	1000 99.90%	41	186	495	583	864
2.1	6 84.95%	981 100.00%	1000 99.99%	49	339	428	478	638
2.2	31 96.05%	997 99.93%	1000 99.92%	30	78	419	544	833
2.3	7 87.99%	980 99.28%	1000 99.28%	15	56	540		
2.4	31 96.05%	942 100.00%	999 99.99%	30	78	189	477	657
3.0	4 84.22%	953 99.98%	998 99.98%	45	239	347	386	449
3.1	6 84.81%	969 99.99%	1000 99.99%	36	302	342	386	434
3.2	30 95.94%	970 99.91%	998 99.91%	30	82	345	391	679
3.3	7 87.82%	959 100.00%	999 100.00%	16	187	333	395	470
3.4	30 95.94%	983 99.75%	1000 99.74%	30	82	330	383	
4.0	4 84.29%	994 99.72%	996 99.72%	41	140	203	454	
4.1	6 84.95%	993 99.05%	999 99.05%	49	151	474		
4.2	31 96.05%	929 99.94%	999 99.93%	30	78	175	221	564
4.3	7 87.99%	991 99.01	999 99.01%	15	56	823		
4.4	31 96.05%	951 99.94%	1000 99.93%	30	78	158	237	524

We first consider  $Q_j = 0.001$ , as shown in Table 3. Virtually identical throughput values were obtained for all 18 versions of the algorithm summarized in this table. In all cases, the throughput was at least 99.9% of the benchmark value. For  $Q_j = 0.3$ , shown in Table 4, all 18 versions provided at least 99% of the benchmark throughput value, 15 versions provided at least 99.5%, and 13 versions provided at least 99.9%.

Such convergence to nearly optimal solutions is typical behavior over a wide variety of network scenarios. This is evident from the milestone tables for the nine other core runs, which are provided in Appendix A. In fact, for nine of the 11 network examples in the core runs, all 18 versions of the algorithm were able to provide at least 99.9% of the optimal throughput value. For 10 of the 11 network examples, all 18 versions were able to provide at least 99.5% of the optimal throughput value. In all 11 network examples, all 18 versions were able to provide at least 99% of the optimal throughput value.

Several important observations, which also apply to the other core runs, can now be made. First, all 18 versions of the algorithm do, in fact, converge. This is evident from the fact that the throughput value at the last admissible iteration is virtually identical to that at the best iteration.<sup>30</sup> In some cases (e.g., Table 3) all versions converge to virtually the same point, whereas in others (e.g., Table 4) there is greater variation among the different versions. In Section 8.4 we compare the offered loads (and the resulting blocking probabilities) obtained by the different versions of the algorithm. Although there is no guarantee that the benchmark throughput is the true optimal solution, the fact that several versions of the algorithm converged to nearly the same point (i.e., a virtually identical set of offered loads), by following different trajectories, suggests that it is.

All of the milestone tables are based on runs of 1000 iterations in duration. The stepsize rules were designed to achieve the maximum possible throughput in at most 1000 iterations, rather than the fastest convergence to a “good” solution that provides 98% or 99% of the true optimal performance. Thus reliable convergence to the best solution within 1000 iterations is the primary goal. Had the goal been to determine the best possible throughput in a smaller number of iterations, different stepsize rules would have been used. The four stepsize rules tend to be conservative<sup>31</sup> in the sense that the stepsize is kept at a relatively large value long enough to ensure that the neighborhood of the optimal solution is approached. If the durations of the phases are shortened (either in the same proportion or otherwise), it is expected that better solutions will typically be found faster, although with an increased risk of convergence to a suboptimal solution.

The purpose of the “adm thruput at first exit” column is to indicate the percentage of the benchmark throughput that is achieved prior to exiting the admissible region for the first time. This can be viewed as a measure of how well the algorithm behaves in the early stages in the sense of the directness of the approach to the optimal solution. Generally, versions of the algorithm that do not use the projection (i.e., X.0) exit the admissible region sooner than the versions that do use the projection when the same stepsize is used. One reason for this behavior is that, since  $\|D\| \leq \|VS\|$ , the offered loads (and hence the resulting throughput) typically change by a smaller amount at each iteration than for examples in which the projection is not used.<sup>32</sup> We refer to this effect as that of a smaller “effective stepsize” resulting from the use of the same value of  $\theta$ . Another (related) reason is that, when the projection is not used, the trajectory is not affected by the QoS constraint until the admissible region is exited for the first time. By contrast, Versions X.1, X.2, X.3, and X.4, which do use the projection, are affected by the QoS constraint throughout the iterative search, and therefore may turn away from the QoS contour before crossing it.

Moreover, in most cases, the percentage of the benchmark throughput that is achieved just prior to exiting the admissible region for the first time is generally considerably higher for the versions that use the projection with a large number of circuits. For example, when  $Q_j = 0.3$ , the X.2 and X.4 versions of the algorithm provide about 96% of the benchmark throughput prior to exiting the admissible region for the first time. By contrast, the X.1 and X.3 versions provide only about 85% to 88% of the benchmark throughput. These versions are similar to each other in that they use one circuit (X.1) and a small number of circuits (X.3) in the projection. It thus appears that the versions of the algorithm that use a large number of circuits in the projection provide a more-direct ascent to an early “good” solution (in which the offered loads still may be far from the optimal solution). The results for  $Q_j = 0.001$  shown in Table 3 are not as conclusive, although it is clear that the X.0 versions provide the lowest percentage of benchmark throughput prior to the first exit of the admissible region. Table A3 in Appendix A shows the results for the same case shown in Table 3, except that the initial stepsize has been reduced by 50%. Table A3 demonstrates that use of Versions X.2 and X.4 with the reduced stepsize for this problem consistently provides between 93% and 94% of the benchmark throughput prior to exiting the admissible region for the first time, whereas the Versions X.0,

<sup>30</sup>We noted in Section 7.1 that the use of the projection throughout the entire search often resulted in movement away from the optimal point.

<sup>31</sup>“Conservative” is a relative term. For some problems (perhaps for problems with a larger number of circuits or a severely constrained search space), even the most conservative of these stepsize rules may be overly aggressive.

<sup>32</sup>We have observed that  $\|D\|$  can be considerably smaller than  $\|VS\|$ , especially when many circuits are included in the set  $\Sigma$ .

which do not use the projection, provide only about 78% of the benchmark throughput. The behavior of the various versions of the algorithm prior to the first exit of the admissible region is discussed in greater detail in Section 8.5. Explanations that suggest the reasons for the benefit of the use of a smaller stepsize for low values of  $Q_j$  are discussed in Section 8.5.

We now address the question of speed of convergence. It is of interest to see how fast the different versions of the algorithm can produce “good” solutions. This is the purpose of the columns denoted by 95%, 98%, 99%, 99.5%, and 99.9% in the milestone tables. These columns indicate the iteration at which an admissible solution with throughput equal to the corresponding percentage of the benchmark throughput is obtained for the first time. Of course, it is impossible to know in real time when these milestones have been reached because they are based on knowledge of the benchmark throughput value, which is not known until the completion of the 1000 iterations for all 18 versions of the algorithm.

### 8.3 Stopping-Rule Tables

The milestone tables presented in Section 8.2 are based on running the different versions of the algorithm for a fixed number of iterations, in this case 1000. However, in most applications, iterative algorithms such as those developed in this report would be terminated when a suitable convergence criterion is satisfied. The use of appropriate stopping rules can save considerable computational time if convergence is obtained early; on the other hand, failure to converge indicates that additional iterations are needed.

We now consider the possibility of stopping the iteration when convergence has been achieved, instead of using a fixed number of iterations. A complete specification of the algorithm must now also include the particular stopping rule that is being used. The effectiveness of algorithms with particular stopping rules can be evaluated in terms of speed (number of iterations until convergence is declared) and quality of the resulting solution (in our case, percentage of the benchmark throughput that is obtained when the algorithm is stopped).

To evaluate the stopping rules, we did not perform additional runs; instead we examined data from the runs with duration equal to 1000 iterations and determined the outcome that would have occurred had these stopping rules been used. The convergence criterion used in our studies is

$$\frac{|S_{k+1} - S_k|}{\max\{S_k, S_{k+1}\}} < \delta, \quad k = m, m+1, \dots, m+4 \quad \text{for some } m, \quad (36)$$

i.e., that five consecutive throughput values (whether or not they are admissible) should not differ from the previous throughput value by more than a specified fraction, which we denote by  $\delta$ . Table 5 shows the effect of using different stopping rules for the determination of convergence for the same example for which the milestone table was just discussed, i.e., for  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ ; Table 6 shows similar results for  $Q_j = 0.3$ . For example, three columns are headed by “max it = 300” and show the effect of stopping the run when convergence to the specified tolerance ( $\delta = 0.1, 0.01, \text{ or } 0.001$ ) has been achieved, or at iteration 300 if convergence to the specified tolerance is not achieved. Results are also shown for a maximum of 600 and 1000 iterations.

When the iterative search is stopped (either because the convergence criterion is satisfied, or because the specified number of iterations has been reached), the solution to the problem is declared to be the set of offered loads that has provided the highest admissible throughput thus far during the course of the run. The best solution is not necessarily the offered load at the stopping point because of the nonmonotonic nature of the algorithm. Specifically, consider the case of Version 1.2 in Table 5. The entry of 38 in the column headed by max it = 300 and 0.1 indicates that convergence to within a tolerance of  $\delta = 0.1$  is obtained at iteration number 38. The entry of 94.205 under the 38 indicates that use of this stopping criterion provides 94.205% of the benchmark throughput (2.6645), i.e., the best admissible throughput observed thus far (not necessarily at this iteration—it may have occurred earlier) is 94.205% of the benchmark throughput. Similarly the entries in the next column indicate that convergence to within 0.01 is not obtained during the first

Table 5 — Stopping-Rule Table for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$   
(benchmark throughput: 2.6645)

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.01	0.001	0.0001
1.0	300 96.713	300 96.713	300 96.713	462 96.713	591 98.512	600 98.512	591 98.512	852 99.933	1000 99.989
1.1	300 94.032	300 94.032	300 94.032	470 95.487	600 99.279	600 99.279	634 99.279	833 99.949	953 99.987
1.2	38 94.205	300 95.293	300 95.293	38 94.205	600 98.346	600 98.346	678 99.217	864 99.924	1000 99.981
2.0	148 96.713	300 96.713	300 96.713	148 96.713	600 99.791	600 99.791	605 99.791	803 99.903	1000 99.994
2.1	110 93.636	300 96.739	300 96.739	110 93.636	537 99.366	600 99.551	537 99.366	783 99.973	1000 99.991
2.2	38 94.205	300 97.088	300 97.088	38 94.205	600 99.099	600 99.099	600 99.099	698 99.703	1000 99.988
2.3	108 95.083	300 95.215	300 95.215	108 95.083	586 99.370	600 99.508	586 99.370	796 99.938	1000 99.988
2.4	38 94.205	300 97.409	300 97.409	38 94.205	521 99.062	600 99.334	521 99.062	840 99.971	1000 99.996
3.0	300 96.295	300 96.295	300 96.295	315 96.295	436 99.789	564 99.920	436 99.789	564 99.920	1000 99.987
3.1	191 96.188	300 97.154	300 97.154	191 96.188	390 99.019	576 99.934	390 99.019	576 99.934	1000 100.00
3.2	36 90.756	300 97.543	300 97.543	36 90.756	412 99.088	600 99.942	412 99.088	634 99.942	1000 99.973
3.3	92 92.781	300 97.776	300 97.776	92 92.781	454 99.489	600 99.902	454 99.489	686 99.938	1000 99.985
3.4	36 90.756	300 97.543	300 97.543	36 90.756	402 99.251	445 99.808	402 99.251	445 99.808	475 99.835
4.0	147 96.713	300 99.541	300 99.541	147 96.713	343 99.712	511 99.848	343 99.712	511 99.848	1000 99.992
4.1	132 93.636	292 99.742	300 99.742	132 93.636	292 99.742	600 99.922	292 99.742	690 99.964	1000 99.998
4.2	38 94.205	240 99.200	300 99.358	38 94.205	240 99.200	600 99.884	240 99.200	604 99.884	1000 99.996
4.3	135 95.083	283 99.102	300 99.374	135 95.083	283 99.102	600 99.871	283 99.102	636 99.871	1000 100.00
4.4	38 94.205	260 99.481	300 99.565	38 94.205	260 99.481	600 99.886	260 99.481	719 99.926	1000 99.993

max it = maximum number of iterations (= 300, 600, or 1000)

$\delta$  = tolerance level used in stopping rule (= 0.1, 0.01, 0.001, or 0.0001)

upper entries: point at which iteration is stopped (either because convergence criterion is satisfied, or because maximum number of iterations has been reached)

lower entries: percentage of benchmark throughput at stopping point

Table 6 — Stopping-Rule Table for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$   
(benchmark throughput: 11.5380)

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.01	0.001	0.0001
1.0	14 90.155	300 96.349	300 96.349	14 90.155	551 99.142	600 99.497	551 99.142	737 99.830	937 99.916
1.1	10 92.348	300 96.223	300 96.223	10 92.348	517 98.665	600 99.492	517 98.665	724 99.790	922 99.901
1.2	13 79.572	29 94.112	300 98.088	13 79.572	29 94.112	600 99.363	29 94.112	738 99.868	907 99.943
2.0	14 90.155	300 98.468	300 98.468	14 90.155	462 98.869	600 99.584	462 98.869	699 99.807	940 99.910
2.1	10 92.348	300 97.768	300 97.768	10 92.348	355 98.283	600 99.869	355 98.283	641 99.905	880 99.992
2.2	13 79.572	29 94.112	300 98.088	13 79.572	29 94.112	600 99.685	29 94.112	674 99.822	933 99.932
2.3	10 87.987	132 98.263	300 98.263	10 87.987	132 98.263	559 99.101	132 98.263	559 99.101	744 99.262
2.4	13 79.572	29 94.112	300 99.181	13 79.572	29 94.112	600 99.815	29 94.112	622 99.839	917 99.990
3.0	12 89.292	300 98.350	300 98.350	12 89.292	330 98.402	467 99.916	330 98.402	467 99.916	773 99.980
3.1	10 92.220	300 97.786	300 97.786	10 92.220	338 98.842	458 99.902	338 98.842	458 99.902	767 99.989
3.2	13 79.197	81 97.707	293 98.762	13 79.197	81 97.707	293 98.762	81 97.707	293 98.762	823 99.906
3.3	10 87.817	16 95.586	300 98.509	10 87.817	16 95.586	408 99.768	16 95.586	408 99.768	840 99.993
3.4	13 79.197	81 97.707	293 98.762	13 79.197	81 97.707	293 98.762	81 97.707	293 98.762	499 99.659
4.0	14 90.155	183 98.799	300 99.216	14 90.155	183 98.799	384 99.374	183 98.799	384 99.374	781 99.694
4.1	10 92.348	152 98.003	300 98.851	10 92.348	152 98.003	387 98.953	152 98.003	387 98.953	781 99.048
4.2	13 79.572	29 94.112	300 99.690	13 79.572	29 94.112	402 99.815	29 94.112	402 99.815	793 99.931
4.3	10 87.987	129 98.263	242 98.809	10 87.987	129 98.263	242 98.809	129 98.263	242 98.809	640 98.988
4.4	13 79.572	29 94.112	215 99.380	13 79.572	29 94.112	215 99.380	29 94.112	215 99.380	877 99.932

300 iterations, and that the best solution in the first 300 iterations provides 95.293% of the benchmark throughput. The criterion of  $\delta = 0.01$  is not satisfied until iteration 678, at which point 99.217% of the benchmark throughput is obtained. The criterion of  $\delta = 0.001$  is satisfied at iteration 864, where 99.924% of the benchmark throughput is obtained.

There is no way to know a priori the quality of the solution that will be determined by a particular level of convergence. For the network example of Table 5, when comparing all 18 versions of the algorithm, we see that a convergence criterion of  $\delta = 0.1$  (see the column for 600 iterations) provides a solution with throughput that is somewhere between 90.7% and 96.7% of the benchmark value. A criterion of  $\delta = 0.01$  (see the column for 1000 iterations) provides a solution between 98.5% and 99.8% of the benchmark value. Finally, a convergence criterion of  $\delta = 0.001$  (see the column for 1000 iterations) provides a solution that is

better than 99.7% of the benchmark value in all cases. These numbers vary significantly for the different network examples we have considered.

Let us now consider Table 6, the stopping-rule table for the same example as that of Table 5, but with  $Q_j = 0.3$ . In this example, a convergence criterion of  $\delta = 0.01$  (in the column for 600 iterations) provides only 94.1% of the benchmark throughput for five versions of the algorithm (i.e., 1.2, 2.2, 2.4, 4.2, and 4.4) and 95.6% for one version (3.3); all other versions provide at least 97.7%. A common feature of the six versions that provided less than 96% of the benchmark throughput is that the convergence criterion was satisfied very early in the search. For example, the five versions that reached only 94.1% at convergence, actually converged (at iteration 29) prior to the first exit from the admissible region. Version 3.3 (which had already exited the admissible region) converged at iteration 16. This overly rapid convergence has suggested that we modify the stopping rule so that the test for convergence is not performed before the 50th iteration. This additional requirement, coupled with the convergence criterion of  $\delta = 0.01$ , produces solutions that provide at least 97.3% of the benchmark throughput for this network example.

Several factors affect the rate of convergence. For example, small stepsizes result in small changes in the offered load (and hence throughput). Thus a rule that reduces stepsize rapidly would result in rapid satisfaction of the convergence condition, although not necessarily convergence to the optimal point. Also, we have observed that use of the projection operation with a large number of circuits tends to result in a smoother trajectory (not only a more-direct path, but also a smaller “effective stepsize” for a given  $\theta$  than the use of no projection), and thus the increased possibility of declaring convergence prematurely. In particular, premature convergence was observed for Versions 1.2, 2.2, 2.4, 4.2, and 4.4 (all of which are identical during the first 100 iterations) in the example of Table 6. The additional requirement of at least a minimum specified number of iterations appears to be helpful here, as discussed in the previous paragraph.

#### 8.4 Comparison of the Solutions Produced by the Different Versions of the Algorithm

Our primary conclusion, obtained by examining the data in the tables presented earlier in this section and in Appendix A, is that almost all versions of the algorithm perform well, based on the criterion of providing optimal (or nearly optimal) throughput within 1000 iterations. In this subsection we compare the solutions that we have obtained by using the 18 different versions of the algorithm. Although we limit our discussion to six particular network examples (Networks 1, 2, and 3, with two  $Q_j = 0.001$  and 0.3 for each), the qualitative observations made here apply to the other core runs as well.

One characteristic property of the optimal solution in constrained optimization problems such as ours is that at least one of the circuit blocking probabilities must be at the maximum permissible value of  $Q_j$ . To measure how close the individual circuits approach this value, we introduce the normalized circuit blocking probabilities

$$\hat{P}_j = P_j / Q_j, \quad j = 1, \dots, J.$$

Thus  $\hat{P}_j = 1$  when  $P_j = Q_j$ .<sup>33</sup>

Table 7, which we refer to as an “offered-load” table, summarizes the solutions obtained for Networks 1, 2, and 3 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$  and 0.3.<sup>34</sup> For each of these six cases, two solutions are listed, namely the “best” (highest admissible throughput) and “worst” (lowest admissible throughput) solutions obtained among the 18 versions of the algorithm. (For Network 1 with  $Q_j = 0.3$ , we also include a third solution, for reasons discussed below.) The entries shown are the offered load values ( $\lambda_j$ 's), the corresponding normalized circuit blocking probabilities ( $\hat{P}_j$ 's), and the throughput ( $S$ ) at the best point obtained for each run. The 10 columns, with double entries in each cell, show the offered loads and normalized circuit blocking probabilities. For example, consider the top row, which shows the best solution obtained for

<sup>33</sup>In most examples in this report, we consider the case for which  $Q_j = \text{QoS}$ ,  $j = 1, \dots, J$ , i.e., the QoS constraint is the same for all circuits; however, in Section 9.3 we study examples for which the circuits are divided into two groups with significantly different QoS values.

<sup>34</sup>Offered-load tables for the other core runs are provided in Appendix A.

Table 7 — Offered-Load Table for Networks 1, 2, and 3 with  $T_i = 6$ ,  $X_j = 4$ ; “Best” and “Worst” Results Shown for  $Q_j = 0.001$  and  $0.3$

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	$S$
Network 1; $Q_j = 0.001$											
best	0.2546 1.0000	0.3262 1.0000	0.3095 1.0000	0.2353 0.9990	0.3417 1.0000	0.1512 0.9780	0.3949 1.0000	0.0123 0.9140	0.3399 1.0000	0.3016 1.0000	2.6645
worst	0.2553 0.9980	0.3224 0.9990	0.3094 0.9990	0.2359 0.9980	0.3444 0.9990	0.1444 0.9570	0.3960 1.0000	0.0254 0.9380	0.3359 1.0000	0.2974 0.9990	2.6638 99.97%
Network 1; $Q_j = 0.3$											
best	3.3160 0.9997	3.5230 0.9999	1.9599 0.8975	0.0009 0.8862	2.0194 0.7656	0.0008 0.9999	3.3179 1.0000	0.0005 1.0000	1.9675 0.9011	0.0036 0.7542	11.5380
4th worst	3.1770 0.9648	3.2602 0.9749	1.8929 0.9105	0.1036 0.8509	2.0215 0.7819	0.0001 0.9999	3.3605 0.9999	0.0001 1.0000	1.7802 0.8551	0.3478 0.7919	11.5057 99.72%
worst	2.0459 0.8016	2.0080 0.8136	2.4057 0.9998	0.4387 0.7946	2.5897 0.8666	0.0004 0.9998	2.9367 0.9997	0.0003 0.9999	2.4056 0.9998	0.9052 0.8405	11.4239 99.01%
Network 2; $Q_j = 0.001$											
best	0.2326 1.0000	0.3724 1.0000	0.3129 0.9990	0.3571 0.9990	0.2460 1.0000	0.2819 1.0000	0.2941 0.9990	0.3546 0.9990	0.3856 1.0000	0.2357 1.0000	3.0700
worst	0.2323 0.9970	0.3723 0.9990	0.3129 0.9990	0.3570 0.9980	0.2458 0.9970	0.2818 0.9970	0.2942 0.9990	0.3548 1.0000	0.3856 1.0000	0.2355 0.9970	3.0693 99.98%
Network 2; $Q_j = 0.3$											
best	0.0002 0.9999	3.2507 0.9999	1.1397 0.9999	2.0615 1.0000	2.3653 0.9819	1.3159 1.0000	1.2968 0.9999	2.5453 0.9999	3.4367 0.9999	1.7296 0.9989	13.4129
worst	0.0298 0.9998	3.2340 1.0000	1.1582 0.9998	2.0825 0.9999	2.3093 0.9761	1.3500 1.0000	1.2961 0.9997	2.5239 0.9999	3.4291 0.9999	1.7127 0.9998	13.4050 99.94%
Network 3; $Q_j = 0.001$											
best	0.2481 0.9990	0.2529 1.0000	0.2852 1.0000	0.3242 1.0000	0.2808 1.0000	0.2777 1.0000	0.3122 1.0000	0.2646 1.0000			2.2436
worst	0.2479 0.9980	0.2527 0.9990	0.2851 1.0000	0.3243 1.0000	0.2809 1.0000	0.2778 1.0000	0.3123 1.0000	0.2646 0.9990			2.2434 99.99%
Network 3; $Q_j = 0.3$											
best	1.1174 1.0000	1.2697 0.9998	1.5078 0.9999	2.2527 1.0000	1.6666 0.9999	1.7254 1.0000	2.1911 1.0000	1.7158 0.9999			9.4128
worst	1.1127 0.9998	1.2428 0.9959	1.5010 1.0000	2.2480 0.9999	1.6754 0.9997	1.7293 0.9999	2.2056 1.0000	1.7274 0.9999			9.4116 99.99%

best: offered load that provides highest admissible throughput among 18 versions of algorithm

worst: offered load that provides lowest admissible throughput among 18 versions of algorithm

upper entries:  $\lambda_j$

lower entries:  $\hat{P}_j$  (normalized blocking probability)

Network 1 with  $Q_j = 0.001$ . The cell in the  $\{\lambda_1, \hat{P}_1\}$  column, with entries 0.2546 and 1.0000, indicates that  $\lambda_1 = 0.2546$  and  $\hat{P}_1 = 1.0000$  at the best result for that run. The column at the far right shows the throughput achieved for that run, which is the same as the benchmark throughput in Table 3. The second row contains the offered loads and normalized blocking probabilities at the best point in the worst run for Network 1 with  $Q_j = 0.001$ . The lower entry in the far-right column of 99.97% indicates that the worst version of the algorithm provides 99.97% of the best (benchmark) throughput.

With the exception of Network 1 with  $Q_j = 0.3$ , there is little difference in the offered loads found by the different versions of the algorithm. First, let us consider Network 1 with  $Q_j = 0.001$ . In the best solution, eight of the circuits have normalized blocking probabilities of at least 0.9990; on the two remaining circuits, we have  $\hat{P}_6 = 0.9780$  and  $\hat{P}_8 = 0.9140$ . In the worst solution, eight of the circuits have normalized blocking probabilities of at least 0.9980; on the two remaining circuits, we have  $\hat{P}_6 = 0.9570$  and  $\hat{P}_8 = 0.9380$ .<sup>35</sup> The fact that all of the circuit blocking probabilities are close to the maximum permitted value suggests strongly (although does not prove) that our solution is, indeed, close to the true optimal point. Furthermore, the fact that the solutions produced by the 18 versions of the algorithm are very similar to each other, despite the fact that the 18 versions produced very different trajectories, offers further support for this conclusion.

For  $Q_j = 0.3$  there is a considerably wider variation in the solutions, both in terms of the offered loads (and the resulting normalized blocking probabilities) and in the throughput. For this case we also include results for the solution provided by the fourth worst version of the algorithm (which provides 99.72% of the benchmark throughput); the solutions produced by the 14 best-performing versions for this example are similar to each other. Typically, the normalized blocking probability is 0.999 or greater on at most five of the 10 circuits (although not always on the same set of five circuits); on the other circuits it is significantly lower than the maximum permitted QoS value (as low as 0.7542). This behavior is in marked contrast to that for  $Q_j = 0.001$ , in which all circuits were close to the maximum permitted QoS value.

The fact that not all blocking probabilities are near the specified QoS level when  $Q_j = 0.3$  is not surprising. It is not a failure of the algorithm, but rather reflects the fact that the level of interaction among the circuits increases as offered load increases. Thus, a set of offered-load values does not exist for which all blocking probabilities are at the maximum permitted QoS value when that value is relatively high (e.g., 0.3). Note that the best solution for this example includes two normalized blocking probability values that are lower than any of the values in the worst and fourth worst solutions. Thus, we see that in examples where a solution does not exist for which the QoS value is reached on all circuits, the best solution may include some relatively extreme (i.e., small) values of normalized circuit blocking probability.

Although there is certainly interaction among the circuits at offered loads characteristic of blocking probabilities of the order of 0.001, the level of interaction increases as the offered loads are increased. In particular, the values of the partial derivatives ( $\partial P_j / \partial \lambda_i$ ) used in the update equations increase as offered load increases.

The optimal solutions for Networks 2 and 3 are quite different from that of Network 1 in the sense that all of the blocking probabilities are extremely close to the QoS constraint value, even when  $Q_j = 0.3$ . In all cases, the solutions produced by all versions of the algorithm (i.e., the offered load values) were virtually identical. For Network 2 with  $Q_j = 0.3$ , in the benchmark solution the lowest value of normalized blocking probability is  $\hat{P}_5 = 0.9819$ ; all of the others are 0.9989 or greater. In the worst case, the lowest value of the normalized blocking probability is  $\hat{P}_5 = 0.9761$ ; all of the others are 0.9997 or greater. Similarly, for Network 3 with  $Q_j = 0.3$ , in the benchmark solution the lowest value of normalized blocking probability is  $\hat{P}_2 = 0.9998$ . In the worst case, the lowest value of the normalized blocking probability is  $\hat{P}_2 = 0.9959$ .

On the basis of these observations, it appears that whenever the optimal solution lies very close to the QoS contour in all dimensions, there is very little difference in the quality of the solutions produced by the various versions of the algorithm. Also, it appears that our algorithm is more robust in such cases. Often,

<sup>35</sup>The solutions produced by the other 16 versions of the algorithm were similar to the two shown here.

fewer iterations are needed, and more-aggressive stepsize rules (resulting in faster achievement of milestones) are usually successful. Furthermore, we believe that one can have more confidence in the quality of the solution if the blocking probabilities are all close to the QoS constraint value.<sup>36</sup> However, it is difficult to assess, before running the optimization algorithm, whether or not the optimal solution for a particular problem has this property. For example, in Section 13 we consider an example in which Network 1 operates in an “uncontrolled” manner, i.e., calls are admitted as long as transceivers are available at all nodes along the path (in particular,  $X_j = T_i = 6$ ). In this case, even when  $Q_j = 0.001$ , there is no solution for which all of the circuit blocking probabilities are close to the QoS constraint contour, and there is significant variation among the solutions produced by the various versions of the algorithm.

## 8.5 Search Trajectory

To better understand the effects of stepsize and the projection operation on the optimization algorithm, we have studied the evolution of the offered load vector and admissible throughput under the 18 versions of the algorithm. In this subsection we again look at Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and two values of  $Q_j$ , namely 0.001 and 0.3. We examine several versions of the algorithm in detail and plot the evolution of admissible throughput and offered load (i.e., the values of the  $\lambda_j$ 's) throughout the iterative process. The understanding developed here is expected to facilitate the choice of the appropriate version of the algorithm, as well as specific parameter values for specific network applications.

In all cases, the initial value of the offered load is  $\lambda_j = 0$ ,  $j = 1, \dots, 10$ . In most cases, the initial stepsize  $\theta_0$  is chosen so that the trajectory leaves the admissible region for the first time at iteration 5 when the projection is not used (see Section 7.2); exceptions are noted as appropriate. In particular, for Network 1 with  $T_i = 6$  and  $X_j = 4$ , when  $Q_j = 0.001$  we use  $\theta_0 = 0.06$ , and when  $Q_j = 0.3$  we use  $\theta_0 = 0.5$ . Other parameter values are  $c = d = 50$  and  $\gamma_j(1) = 5$ .

### Version 1.0

Figure 10 shows the evolution of the admissible throughput and offered load for Version 1.0 for  $Q_j = 0.001$ . (Note that although throughput values are plotted only for admissible solutions, the offered loads are plotted at all iterations.) Milestone points (e.g., the first iteration at which an admissible solution that provides 95% of the benchmark throughput is obtained) are indicated. The first 100 iterations are rather chaotic, with large variations in the offered loads (the  $\lambda_j$ 's, which are denoted as “arrival rates” in the graph) and admissible throughput. Although the 95% milestone is reached at iteration 51 (which is relatively early in the iteration), this level of performance is not maintained. Between iteration 100 and 400, the admissible throughput is approximately 75% of the benchmark value. For much of this period, it alternates between two values.<sup>37</sup> Also, during much of this period, each of the offered loads alternates among four distinct, and widely separated, values. No further progress is made toward a better solution until the stepsize is decreased significantly. Ultimately, a point that provides more than 99.9% of the benchmark throughput is, in fact, reached; little variation in the offered load values is seen after this point.

Figure 11 shows the trajectories for Version 1.0 with  $Q_j = 0.3$ . Like the case for  $Q_j = 0.001$ , the 95% milestone is met early (at iteration 41), but (again as for  $Q_j = 0.001$ ) the later milestones are not met until the stepsize is reduced. Although the offered loads and admissible throughput do not alternate among several levels as they did for  $Q_j = 0.001$ , little progress is made until the stepsize is reduced significantly.

On the basis of these observations, it is clear that Version 1.0 is overly conservative for these network examples. Tables 3 and 4 show that most of the milestones are achieved relatively late for Versions 1.Y, i.e., for all versions of the algorithm that use Stepsize Rule 1. Similarly, the milestone tables in Appendix A indicate that this observation applies to our other network examples as well.

<sup>36</sup>In some cases (particularly when several of the blocking probabilities are far from the QoS boundary), the network designer/manager might want to run several versions of the algorithm to be sure that the true optimal solution has been found.

<sup>37</sup>The throughput actually alternates among four distinct levels, two of which are admissible and two are not; the inadmissible throughput values are 2.52 and 2.58.

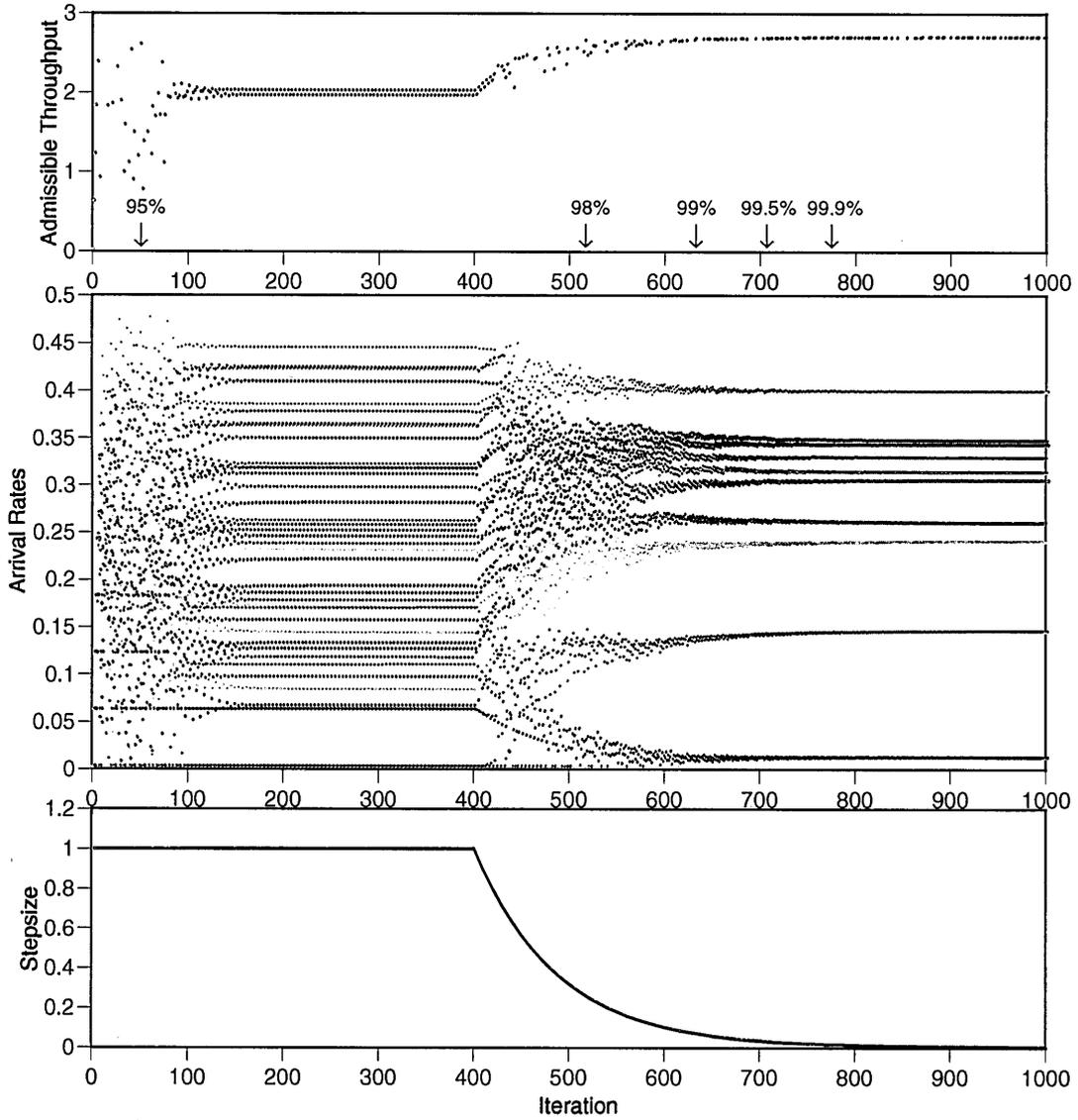


Fig. 10 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ : Version 1.0

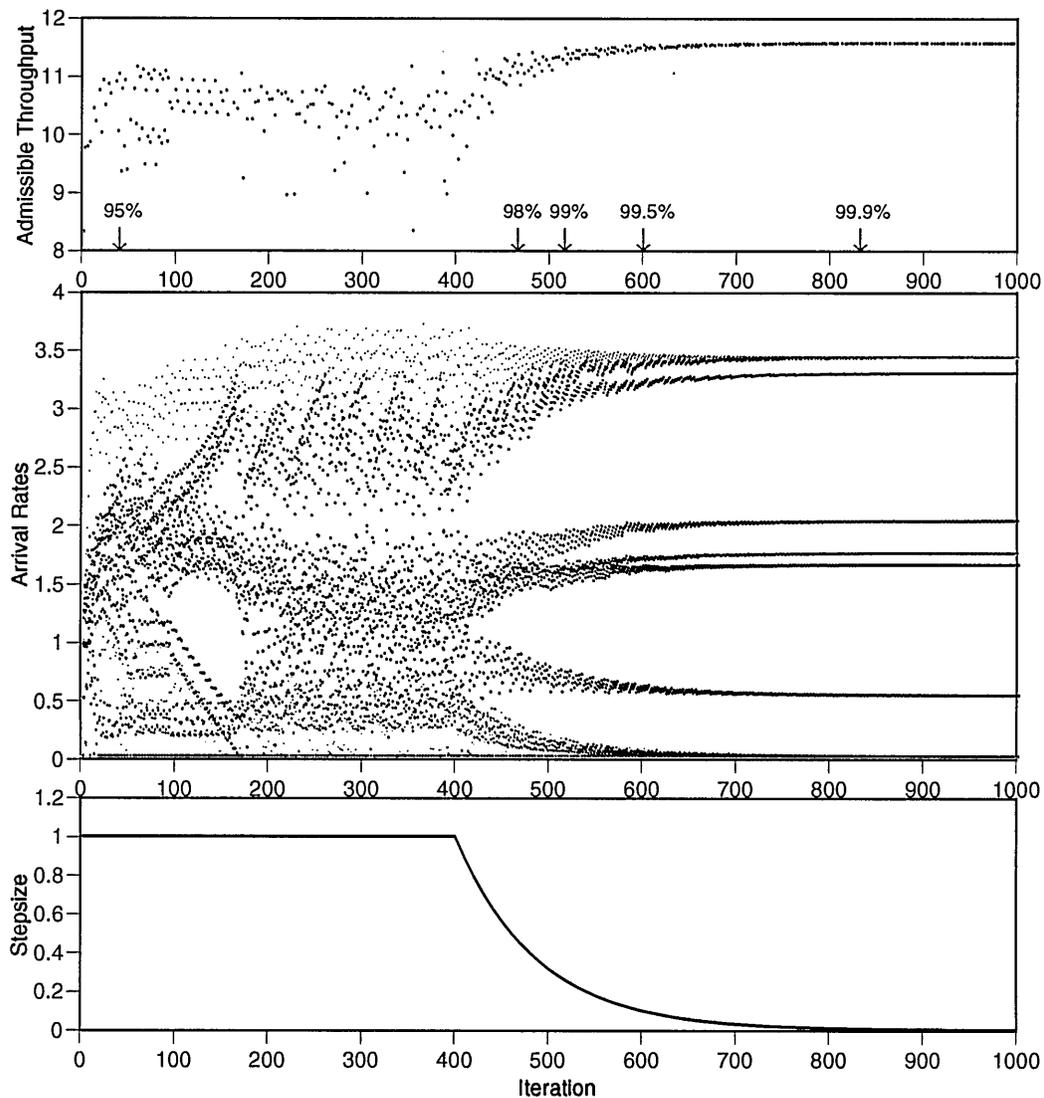


Fig. 11 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$ : Version 1.0

### Version 2.0

Stepsize Rule 2 is less conservative than Rule 1. The stepsize is cut in half at iteration 100, and the exponential descent begins at iteration 300, rather than 400. Figures 12 and 13 show the trajectories for Version 2.0 for  $Q_j$  values of 0.001 and 0.3, respectively.<sup>38</sup> The trajectory during the first 100 iterations is identical to that of Version 1.0. As a result of the use of a smaller stepsize after this point, the amount of oscillation in admissible throughput and offered load values is considerably smaller than that which was observed for Version 1.0. For example, for  $Q_j = 0.001$  the offered loads and throughput no longer alternate periodically among a small set of distinct values. Also, most of the milestones are obtained earlier than in the case of Version 1.0.

<sup>38</sup>In Fig. 12 no milestone is shown for 99% because the 99.5% milestone was reached at the same time. Similar behavior is observed in several of the examples presented in this report.

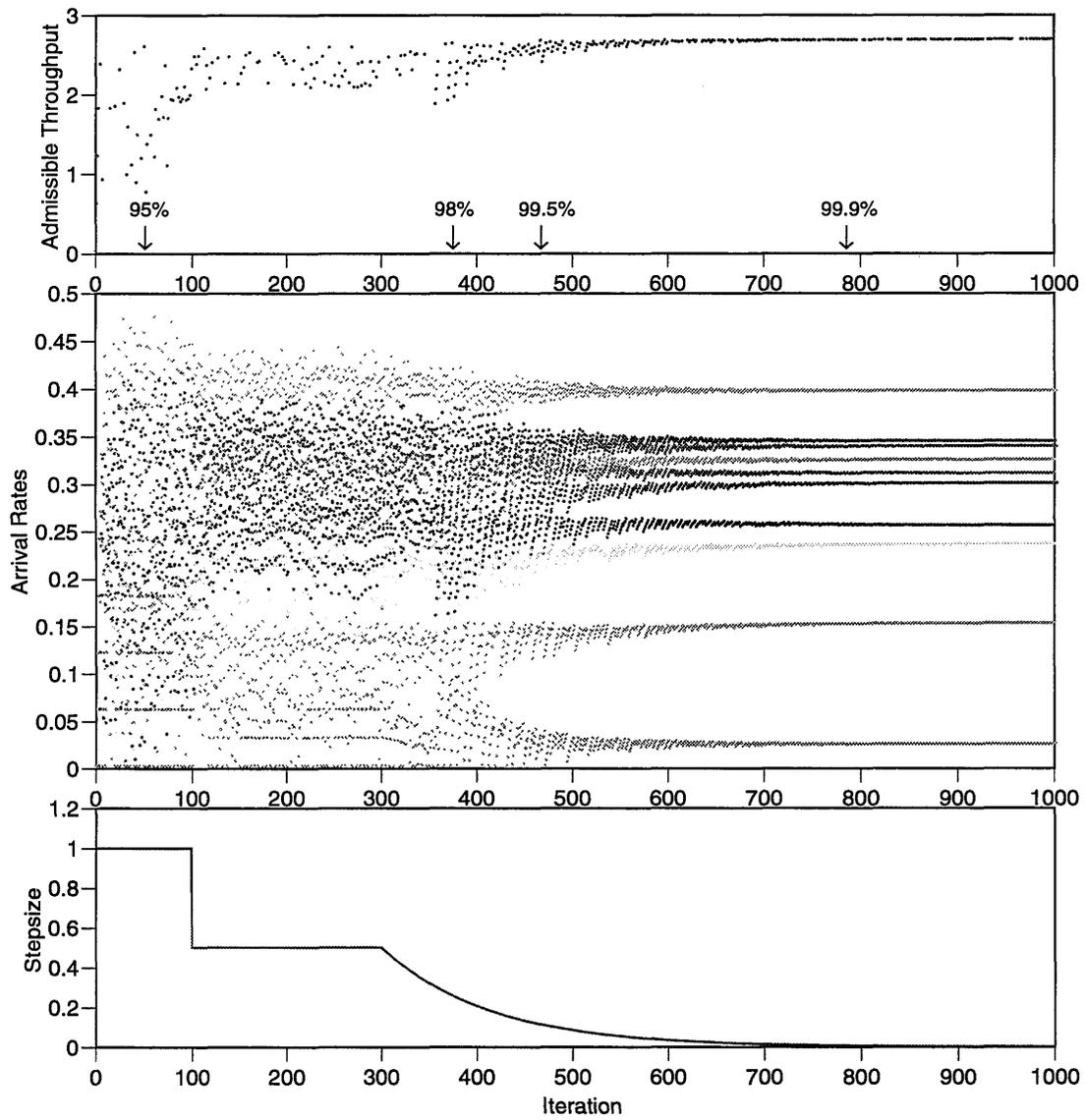


Fig. 12 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ : Version 2.0

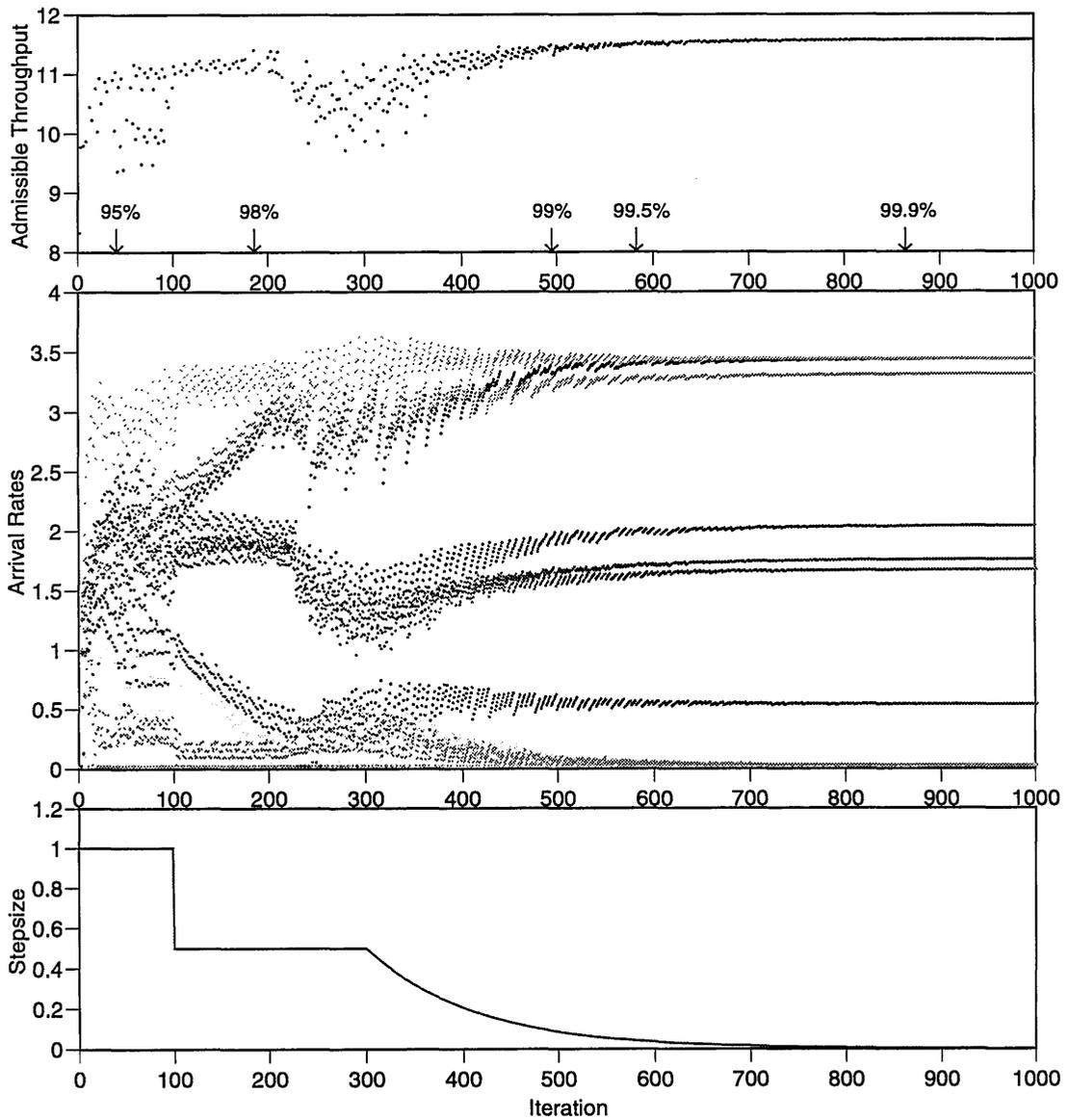


Fig. 13 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$ : Version 2.0

### Version 3.0

The three phases of Step Size Rule 3 are all characterized by an exponential decrease in step size. Although the normalized step size value is 0.5 for both Rules 2 and 3 at iteration 300, the decrease between iterations 300 and 500 is much faster for Rule 3 (see Fig. 7 for details). Figures 14 and 15 show the trajectories for Version 3.0 for  $Q_j = 0.001$  and 0.3, respectively. The rapid decrease during the second phase apparently results in faster attainment of the higher milestone values and faster convergence to the final values of offered load.

### Version 4.0

Step Size Rule 4 is the most aggressive, in the sense that the rapid decrease in step size begins at iteration 100. Figures 16 and 17 show the trajectories for Version 4.0, for  $Q_j = 0.001$  and 0.3, respectively. These figures show that the offered loads reach the neighborhood of their final values quite rapidly. However, failure to reach the 99.9% milestone when  $Q_j = 0.3$  suggests that the rapid decline in step size may have been too extreme in this case.

### Versions that Use the Projection

The versions of the optimization algorithm discussed in the previous subsections did not use the projection formulation. In this subsection we discuss the performance obtained through the use of the four projection rules in conjunction with Step Size Rule 4 (shown in Fig. 17). For these examples, we consider only the admissible throughput, because the behavior of the offered loads does not provide further insight beyond that already discussed for the cases without the projection operation.

Figure 18 shows the admissible throughput for  $Q_j = 0.001$  for Versions 4.1, 4.2, 4.3, and 4.4, and for an initial step size value of  $\theta_0 = 0.06$  (which was determined based on Version 4.0's leaving the admissible region at iteration 5). Figure 19 shows similar results for  $\theta_0 = 0.03$  (half of that value). Use of the smaller initial step size permits a smoother ascent to relatively high throughput values, and results in considerably less oscillation about the optimal point (see Section 7.2). Furthermore, in this example most of the milestones are achieved faster when the smaller value of  $\theta_0$  is used. All four 4.Y versions of the algorithm provide comparable performance in the example of Fig. 18, in terms of both the throughput value obtained and the speed with which the milestones are reached. However, based on the plots in Fig. 19, Versions 4.2 and 4.4 are somewhat better behaved in that the magnitude of the oscillation of admissible throughput is less for these versions than for Versions 4.1 and 4.3.

Figure 20 shows admissible throughput for  $Q_j = 0.3$ . The differences among the 4.Y versions are more apparent for this example. Most evident is the fact that Versions 4.1 and 4.3 do not reach the 99.5% milestone. Some insight into the behavior of these versions can be obtained by examining their basic characteristics.

Version 4.1 uses the projection operation with only the dominant circuit throughout the entire search. During the first 100 iterations, large swings in admissible throughput are observed, and little progress is made. This behavior can be attributed, at least in part, to the fact that the identity of the dominant circuit changes frequently throughout the search. In addition, we have observed that the effective step size for Versions X.1 (i.e., only a single circuit is used in the projection) is comparable to that of Versions X.0 (i.e., versions of the algorithm that do not use the projection).

The behavior of Version 4.3, which includes only a few circuits in the projection operation (because  $v = 0.8$ ) during phase 1, is similar to that of Version 4.1. When the projection operation is turned off at iteration 100, the step size is decreased too rapidly for the trajectory to move to the optimal point in this example.

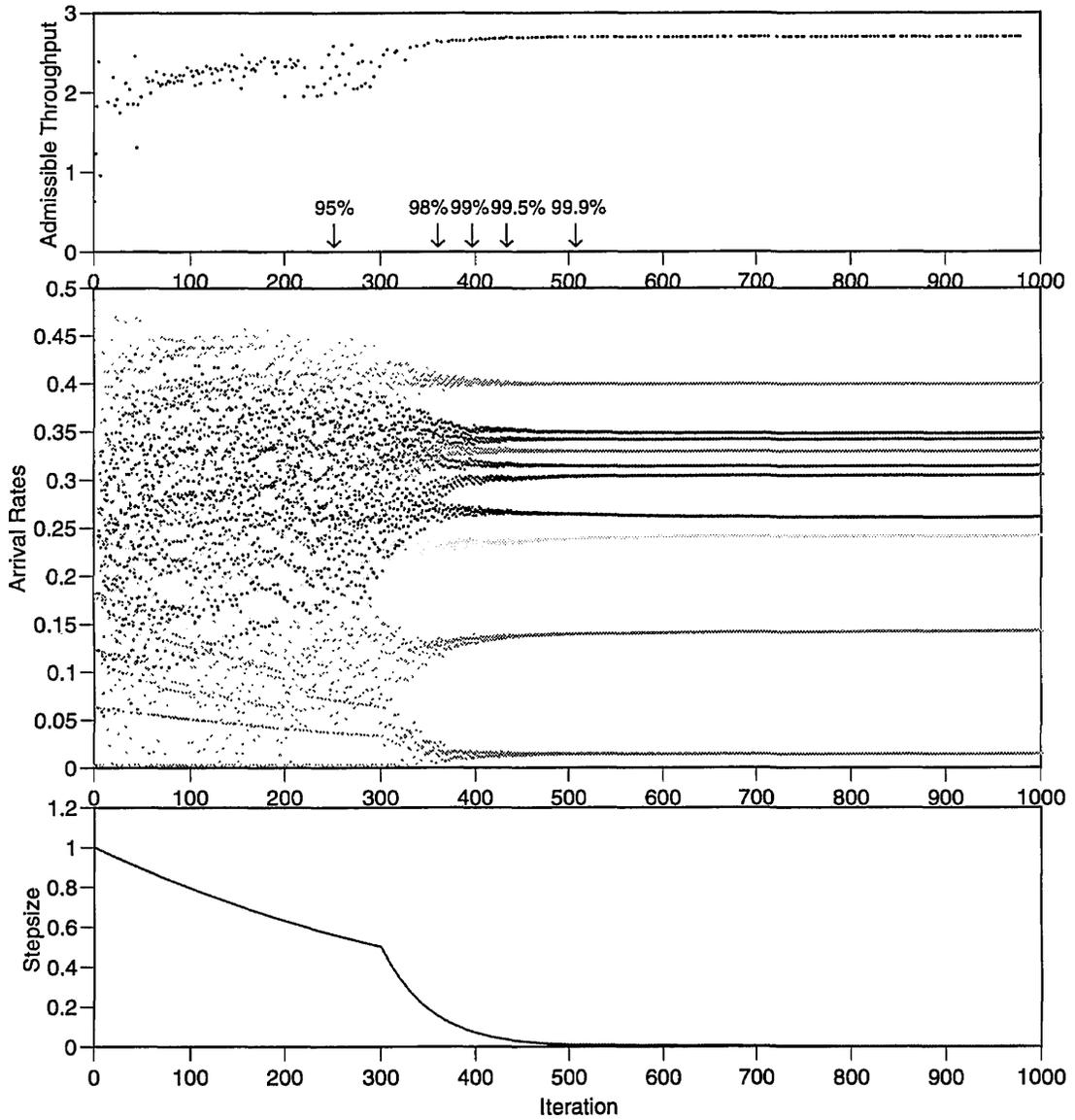


Fig. 14 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ : Version 3.0

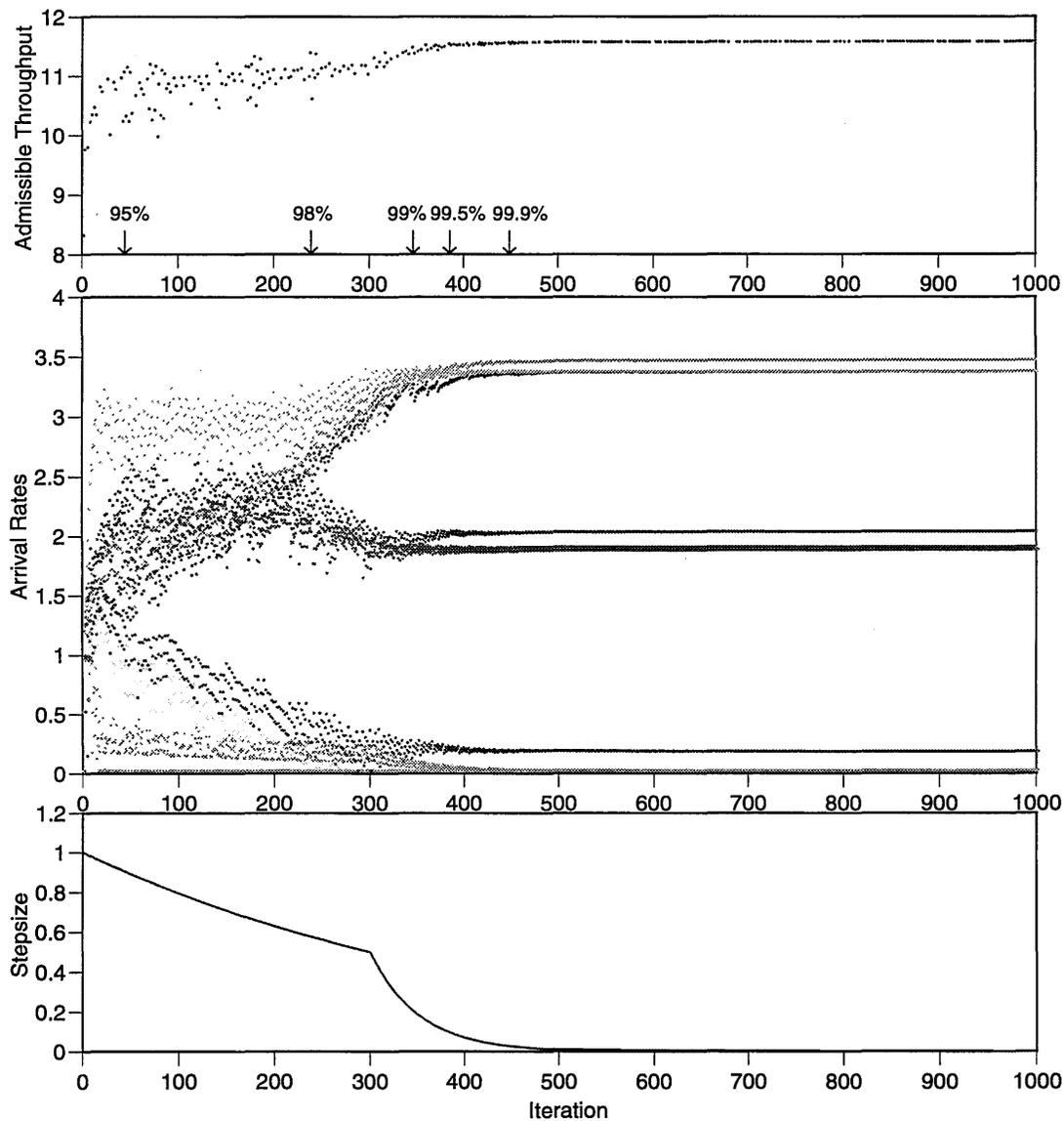


Fig. 15 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$ : Version 3.0

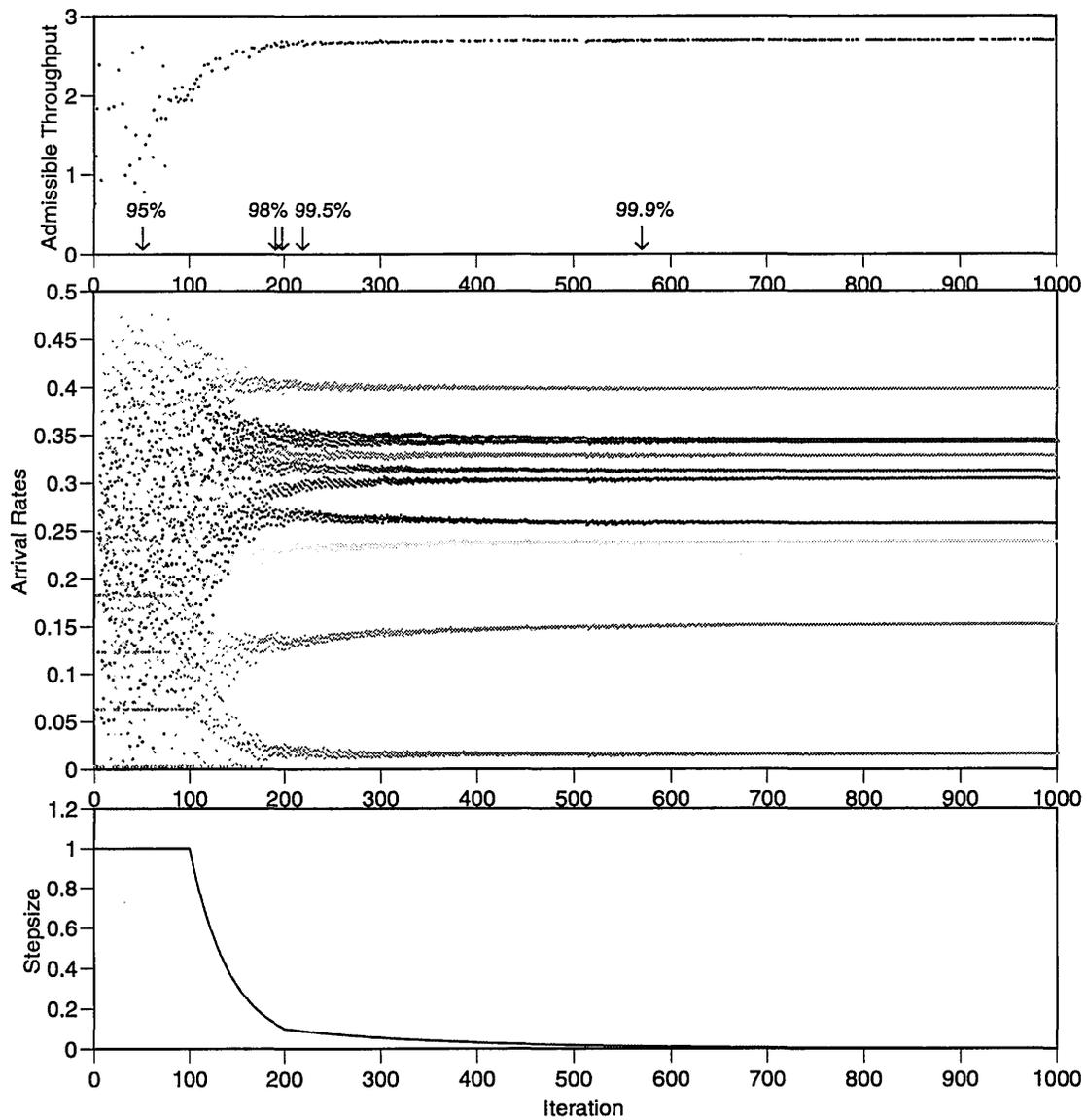


Fig. 16 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ : Version 4.0

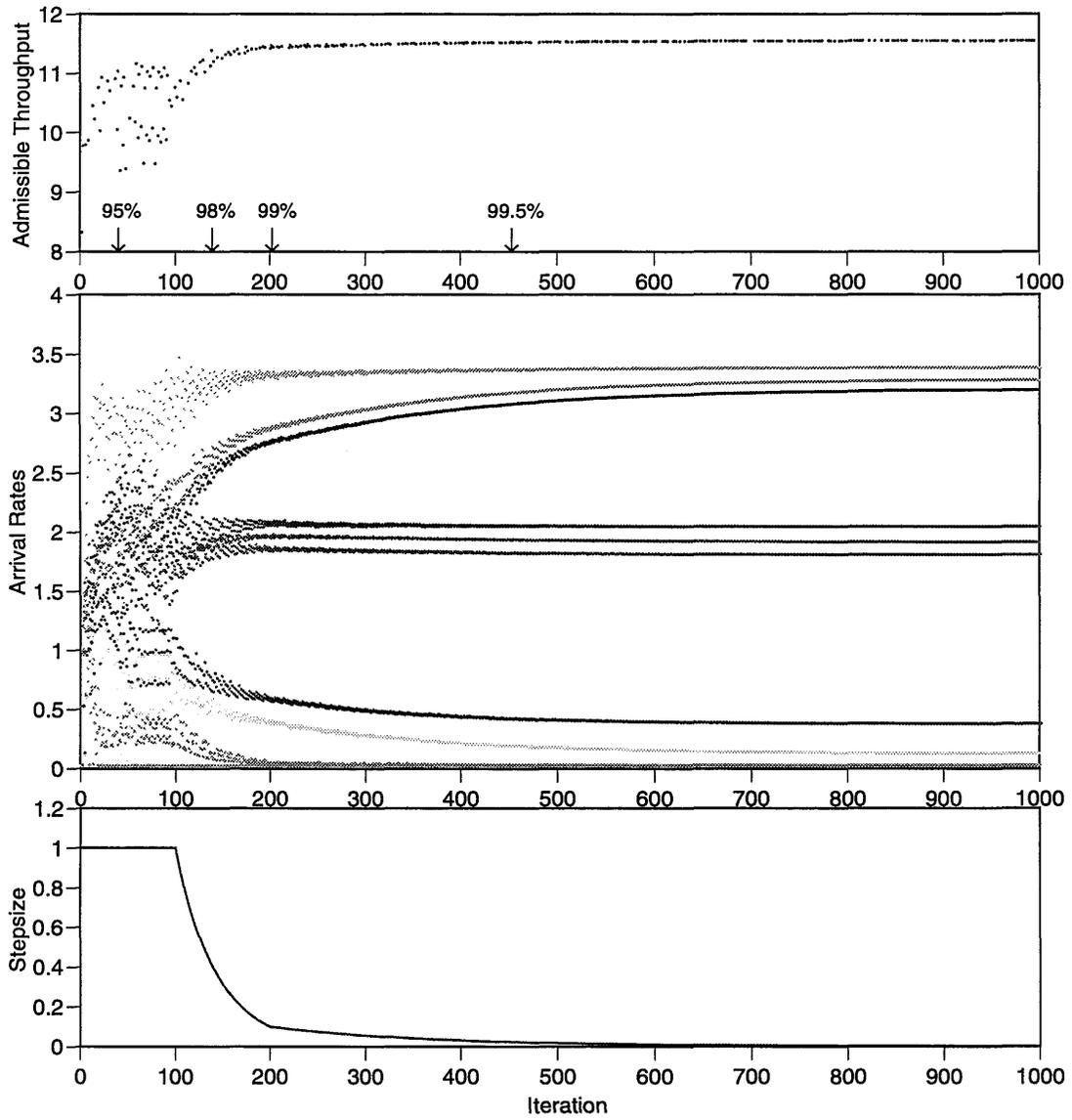
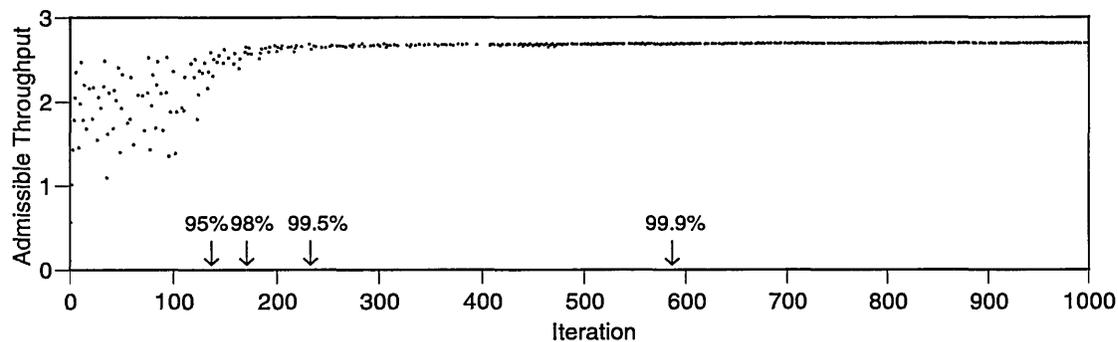
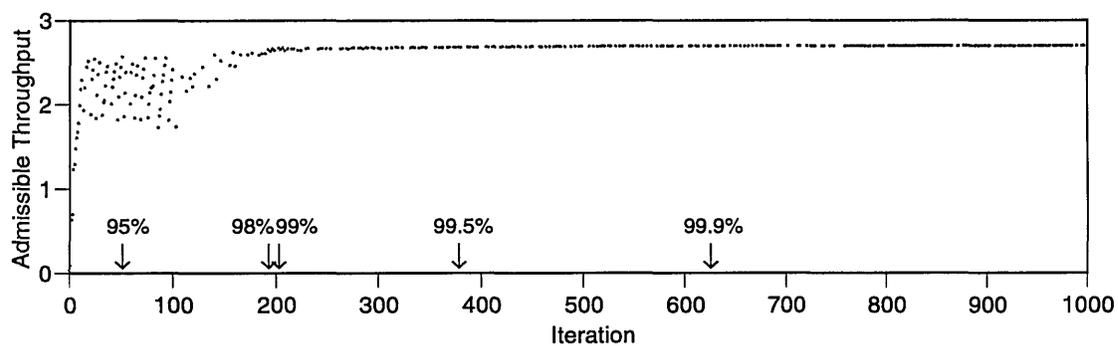


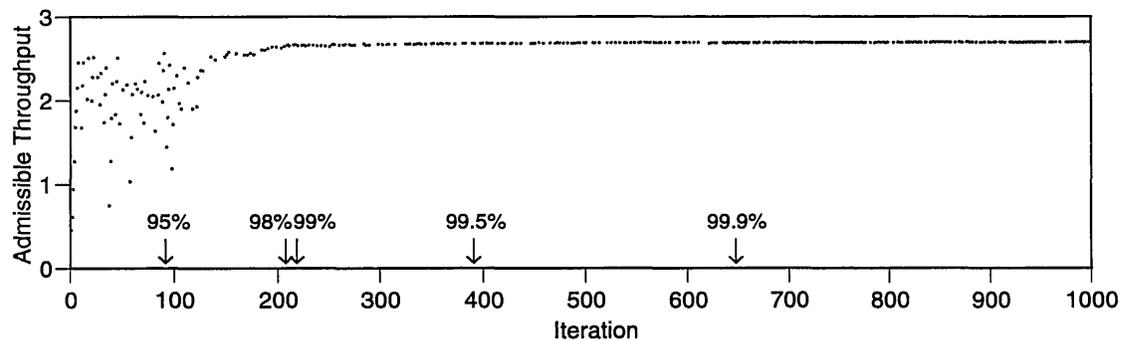
Fig. 17 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$ : Version 4.0



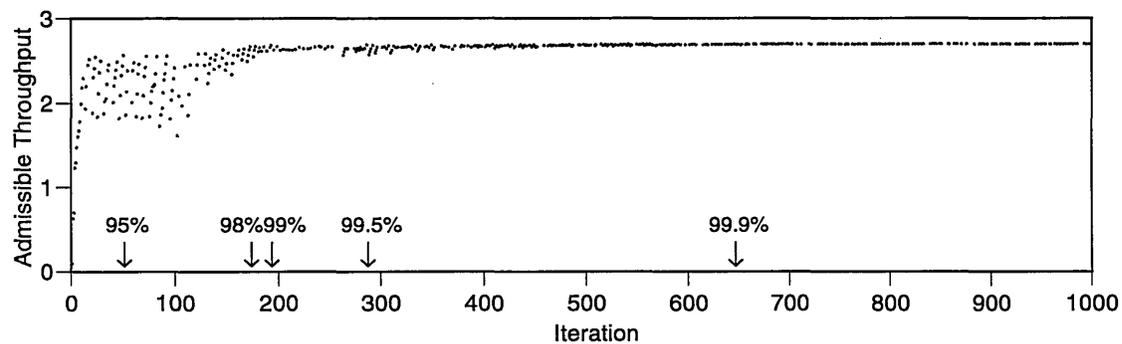
(a) Version 4.1



(b) Version 4.2

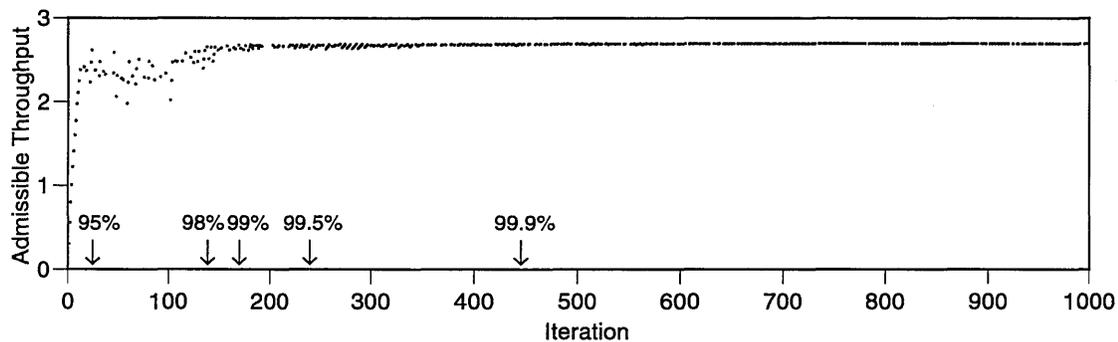


(c) Version 4.3

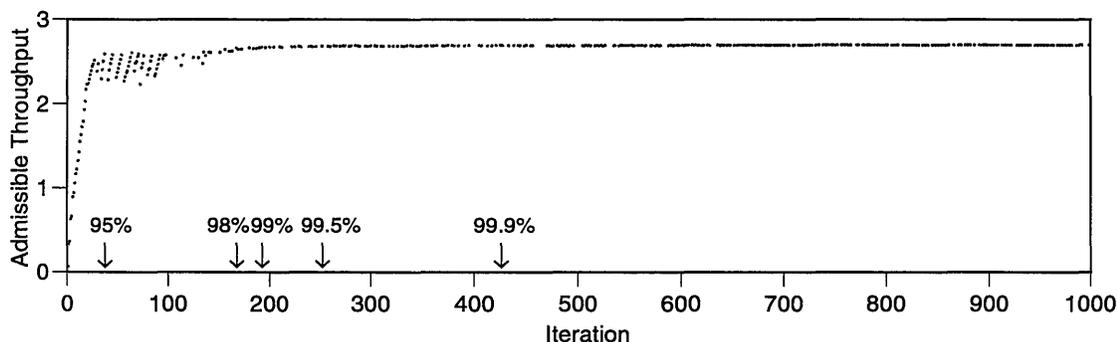


(d) Version 4.4

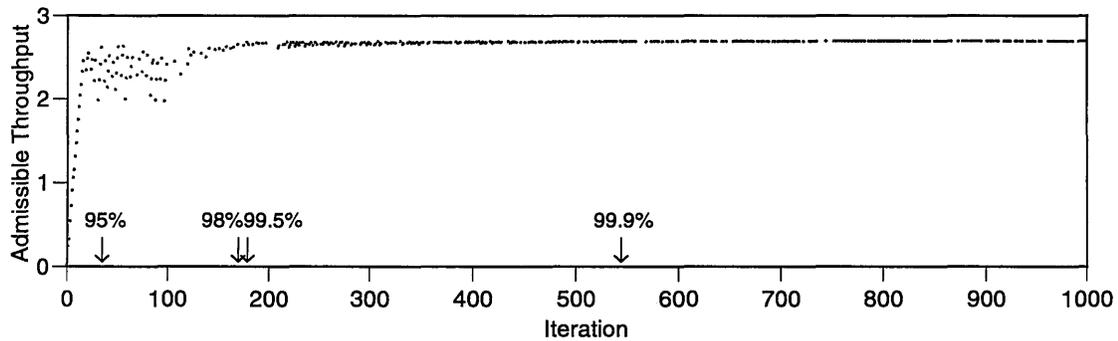
Fig. 18 — Evolution of admissible throughput for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $Q_j = 0.001$ , Stepsize Rule 4, and four variations of the projection rule ( $\theta_0 = 0.06$ )



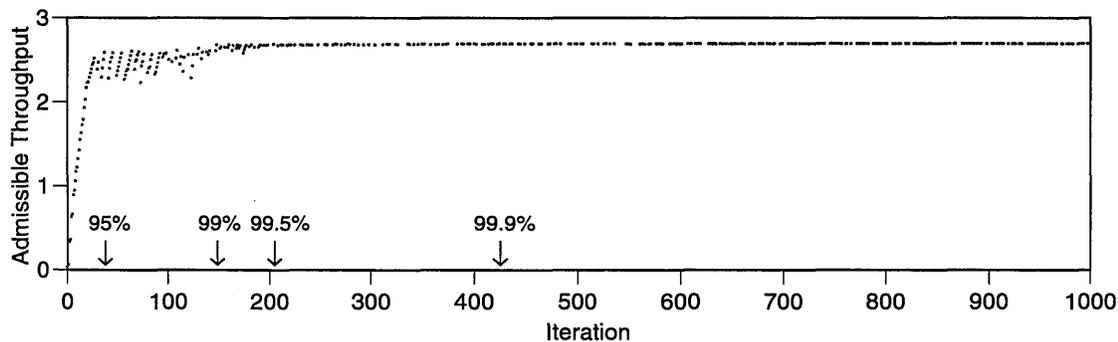
(a) Version 4.1



(b) Version 4.2



(c) Version 4.3



(d) Version 4.4

Fig. 19 — Evolution of admissible throughput for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $Q_j = 0.001$ , Stepsize Rule 4, smaller initial stepsize, and four variations of the projection rule ( $\theta_0 = 0.03$ )

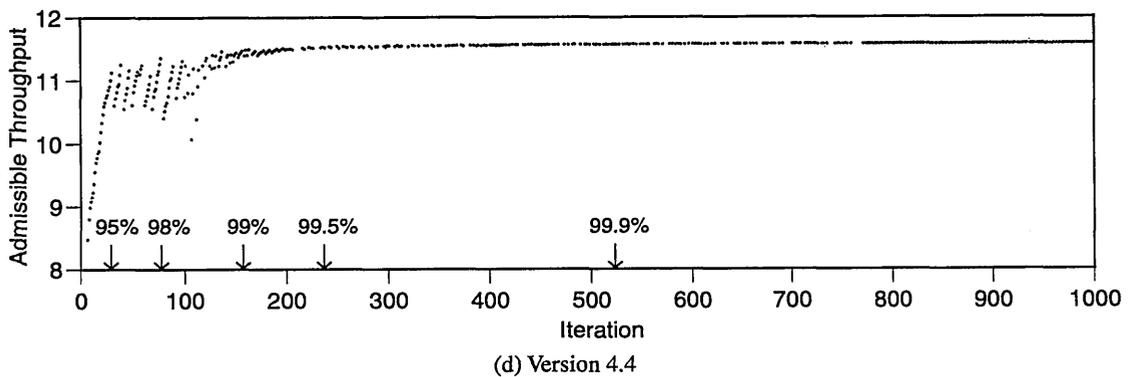
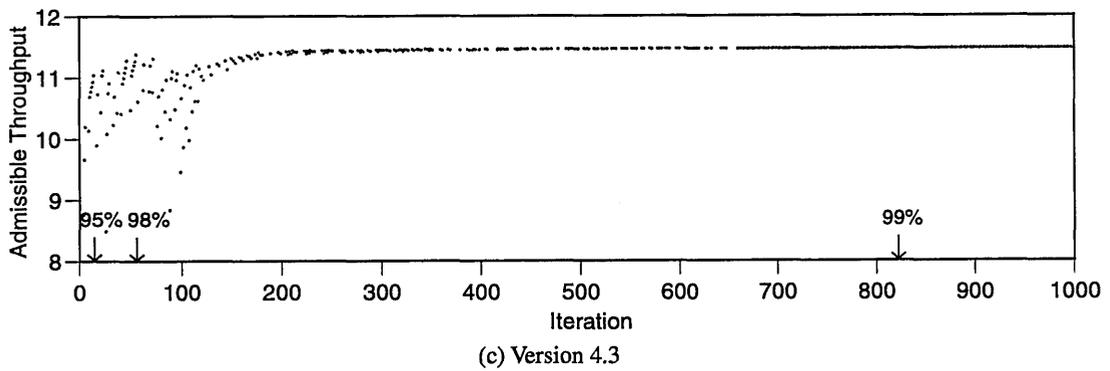
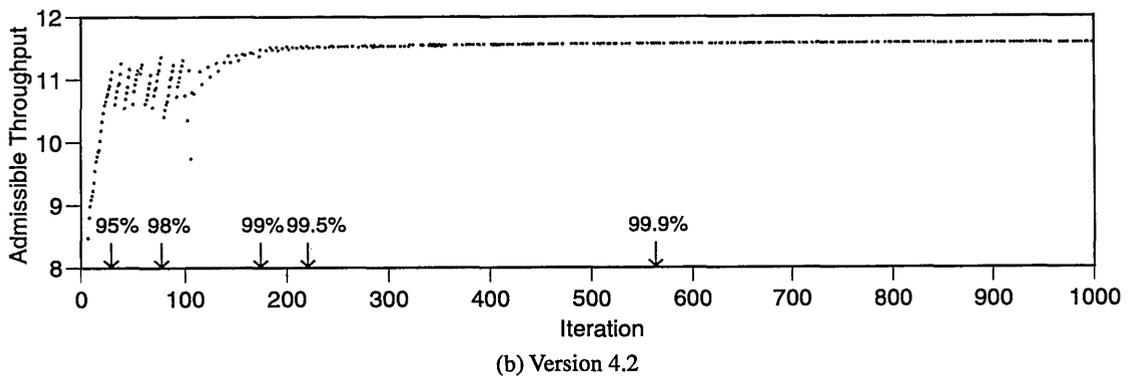
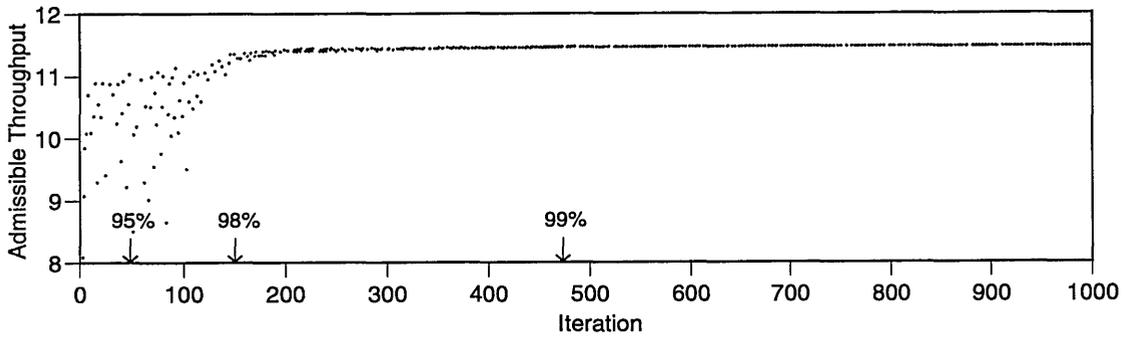


Fig. 20 — Evolution of admissible throughput for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $Q_j = 0.3$ , Stepsize Rule 4, and four variations of the projection rule

Versions 4.2 and 4.4 reach the milestones relatively rapidly. The evolution of admissible throughput shows that these two versions are characterized by nearly monotonically increasing throughput until the admissible region is exited for the first time. In fact, at the last admissible throughput prior to leaving the admissible throughput for the first time, the throughput reaches 96.05% of the benchmark value for these two versions for  $Q_j = 0.3$  (see Table 4). When  $Q_j = 0.001$  and the smaller initial stepsize is used, these two versions provide 93.21% of the benchmark value prior to the first exit (Fig. 19); however, only 84.77% of the benchmark throughput is obtained prior to the first exit when the larger initial stepsize is used (Fig. 18 and Table 3).

## 8.6 Observations Based on Core Runs

Based on the core runs discussed in this section, as well as those summarized in Appendix A, we are able to make a number of observations on both the sensitivity of the throughput to offered load and on the performance of the various versions of the algorithm that are designed to find that load.

### *Properties of the Solution*

We have observed that throughput, when the blocking probability on each circuit is constrained to the QoS value, is a relatively flat function of the offered load in the region of the optimal solution. Thus, rather large deviations in several of the  $\lambda_j$  values may be observed as the throughput increases from the 95% to the 98% and higher milestone values.

We noted in Section 6 that, at the optimal solution to our problem, at least one of the circuit blocking probabilities is at the maximum permitted QoS value. We have examined the best solutions found for several network examples to see how close the circuit blocking probabilities (the  $P_j$ 's) are to the specified QoS value. Although there is not necessarily a correlation between the circuit blocking probabilities and the quality of the solution (the best admissible throughput value), we feel that some insight can be obtained by studying this property.

For Network 1 with  $T_i = 6$  and  $X_j = 4$ , there is significant difference in the behavior of the normalized circuit blocking probabilities between the cases of low and high values of the QoS constraint. For low values of  $Q_j$  (e.g., 0.001), most of the circuit blocking probabilities are very close to the specified QoS value. However, for high values of  $Q_j$  (e.g., 0.3), several (in our example, about half) of the  $P_j$ 's are significantly lower than the QoS value. For this problem a set of offered load values does not exist for which all blocking probabilities are at the maximum permissible value. This is not a failure of the algorithm, but rather is a property of the particular network example. In Section 8.4 we attributed this behavior to the increased interaction among the circuits at higher values of offered load. Nevertheless, we demonstrate in Section 13 that the requirement of a low value of  $Q_j$  does not ensure the existence of a solution in which all blocking probabilities are close to the QoS constraint value.

The behavior of our algorithm was more robust for examples involving Networks 2 and 3 than those for Network 1, based on the criterion of consistency of solutions produced by the various versions of the algorithm (i.e., all versions converge to essentially the same point) and speed with which the milestones were reached. Networks 2 and 3 were different from Network 1 based on another criterion as well, namely that for Networks 2 and 3, all of the  $P_j$ 's were very close to the QoS value for both values of the QoS constraint.

We hypothesize that our algorithm is more robust (i.e., all versions are able to reach the optimal point) when the condition of all blocking probabilities in the optimal solution being close to the specified QoS value is satisfied. Since the blocking probability on all circuits can approach the QoS value simultaneously, there appears to be less conflict among the circuits, and hence an easier ascent to the optimal point. In such cases, aggressive stepsize rules are capable of reaching the optimal solution rapidly. However, we have not discovered a method to determine a priori whether or not this condition will be satisfied.

### *Properties of the Algorithm*

Based on the examples discussed in this section as well as those summarized in Appendix A, one of our principal conclusions is that all versions perform well in the sense of providing nearly optimal throughput. However, there are some differences in performance, both in terms of the speed of achieving “good” (although not necessarily optimal) solutions and in terms of reaching the optimal point.

The difference in the ultimate performance of the different versions of the algorithm (i.e., the best solution they find in a run of 1000 iterations) has been very small in most of the core runs. For example, we remarked in Section 8.2 that for nine of the 11 network examples (which represent three distinct networks, i.e., Networks 1, 2, and 3) in the core runs, all 18 versions of the algorithm were able to provide at least 99.9% of the optimal throughput value. For 10 of the 11 network examples, all 18 versions were able to provide at least 99.5% of the optimal throughput value. In all 11 network examples, all 18 versions were able to provide at least 99% of the optimal throughput value. Although we later discuss some examples in which some versions of the algorithm provided somewhat less than 98% of the benchmark throughput values, our basic conclusion remains valid, namely that all versions perform well when performed for a limited number of iterations (1000 in most of our examples).

We discovered in our early studies that the use of the projection operation often prevents convergence to the optimal solution, especially when a relatively large number of circuits are included in the projection, as is discussed in Section 7.1. The only versions of the algorithm studied in the core runs that use the projection operation throughout the entire iteration are the X.1 versions, which use only the dominant circuit in the projection. In the X.2 and X.3 versions, the projection term is turned off (i.e., set  $\Sigma = \emptyset$ ) at the beginning of phase 2, and in the X.4 versions it is turned off at the beginning of phase 3. We may view the use of the projection term in the first phase(s) as the determination of an “initial condition” for the “undistorted” version of the algorithm (i.e., the version without the projection term). Thus, as long as the trajectory is brought sufficiently close to the neighborhood of the optimal solution in the early phase(s), the undistorted version of the algorithm should bring the solution close to the optimal point before the end of the allotted 1000 iterations.

Even for network examples in which all versions converge to nearly the same point, the use of the projection operation can have a profound impact on the behavior of the algorithm. For example, when a large number of circuits are included in the projection set  $\Sigma$  (e.g., by using a relatively small value of  $v$  such as 0.2, as in Versions X.2), a relatively smooth (although perhaps somewhat slow) trajectory is observed in which the throughput increases monotonically to a large percentage of the benchmark throughput value before exiting the admissible region for the first time. By contrast, when the projection operation is not used, the trajectory is much rougher, with considerably larger deviations in offered load and throughput from one iteration to the next. Although it is indeed possible to achieve some of the high milestone values relatively early in the run when the projection is not used, it may be a matter of “luck” as to whether or not such points are indeed found early. Even if they are found, the trajectory will often move far from these points because of the large stepsize.

Among the stepsize rules we have used, Rule 4 is the most aggressive, thereby providing the possibility of rapid convergence but with greater risk of not reaching the optimal point than the more-conservative stepsize rules. In the core runs discussed in this section and in Appendix A, it appears that the smoothing effect of the projection operation with  $v = 0.2$  permits the effective use of such an aggressive stepsize rule. However, in Section 13 we provide examples in which the use of more-conservative stepsize rules provides higher admissible throughput values.

## **9. FURTHER TESTING OF THE ALGORITHM**

In addition to the core runs of Section 8, we have extensively tested our algorithm to address a number of interesting cases. In particular, we study the impact of nonzero values of  $\lambda_{\min}$ , a nonuniform number of

transceivers at the network's nodes, and nonuniform QoS constraints. We are interested in not only the throughput that is achievable in such cases, but also in the ability of our algorithm to handle such cases.

### 9.1 Nonzero Values of $\lambda_{\min}$

It is evident from the results presented in Section 8 that the values of offered load at the optimal solution are often quite different from one circuit to another. In some cases, some of them are close to zero, while others are significantly higher than the average value. This high variance among the offered loads suggests that it may be advisable, in some cases, to impose a "fairness" mechanism that guarantees that each circuit is permitted at least a nominal offered-load value, even if doing so would decrease the total throughput.

We now address the impact of guaranteed offered-load levels on the performance of our algorithm, and show that when  $\lambda_{\min}$  is relatively large, the iteration is more likely to converge to a suboptimal solution than when  $\lambda_{\min} = 0$ . In Section 12 we address the impact of fairness considerations on the achievable throughput.

Recall that the offered-load values are updated as follows (see Eq. (20)):

$$\lambda_j(k+1) = \max \left\{ \lambda_{\min}, \lambda_j(k) + \theta_k \frac{\partial L}{\partial \lambda_j} \right\}.$$

Thus the offered-load values are constrained at every iteration from decreasing below  $\lambda_{\min}$ .

Based on the core runs discussed in Section 8, we have concluded that an especially good candidate for testing the impact of nonzero values of  $\lambda_{\min}$  is that of Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$ . This example is characterized by a highly asymmetrical solution in which the optimal offered load vector includes circuits with nearly zero offered load. We consider a circuit to be "underloaded" if its offered load is less than 1% of the average offered load among the  $J$  circuits in the network. In particular, the optimal offered load for this example with no constraint on the  $\lambda_j$ 's is characterized by three or four underloaded circuits.<sup>39</sup> Thus, the imposition of the requirement that all circuits have a significant value of offered load is expected to have a greater impact on this example than on examples in which the solution is less asymmetrical. Also, the fact that the normalized blocking probability of only five (out of ten) of the circuits is close to the QoS constraint value is another characteristic of asymmetry. Table B1 in Appendix B provides milestone and stopping-rule tables for this example with  $\lambda_{\min} = 0.5$ . One consequence of imposing this constraint on  $\lambda_{\min}$  is that the benchmark throughput is lowered from 11.5380 to 11.2368, a decrease of 2.6%. In this section, we are primarily interested in the impact of  $\lambda_{\min}$  on the properties of the optimization algorithm. When using nonzero values of  $\lambda_{\min}$ , we have used the same initial stepsize  $\theta_0$  as that used when  $\lambda_{\min} = 0$ .

Only five of the 18 versions of the algorithm provided 99.5% or greater of the benchmark throughput for this example; all of these actually reached the 99.9% milestone (see Table B1). By contrast, when  $\lambda_{\min} = 0$  (see Table 4), 15 of the 18 versions provided at least 99.5% of the benchmark throughput, and 13 reached the 99.9% milestone. Imposing a relatively large value of  $\lambda_{\min}$  makes convergence to the optimal solution more difficult by denying the trajectory the opportunity to pass through the region in which one or more of the  $\lambda_j$ 's are less than  $\lambda_{\min}$ . Of the five versions that did provide at least 99.9% of the benchmark throughput, all used the projection formulation with a large number of circuits in at least phase 1. Thus, it appears that use of the projection facilitated the guiding of the trajectory around the inadmissible region of low values of  $\lambda_j$ . By contrast, when the projection was not used the trajectory seemed to be unable to escape from a region constrained by the requirement that  $\lambda \geq \lambda_{\min}$  for large values of  $\lambda_{\min}$ . Based on the various versions of the algorithm studied, it appears that the performance depends strongly on the characteristics of the algorithm during phase 1. It is especially helpful during this phase to use a large number of circuits in the projection, and to keep the stepsize at a relatively large value.

<sup>39</sup>Of the 13 versions of the algorithm that provided at least 99.9% of the benchmark throughput, one had four underloaded circuits and the others had three.

The case of Network 2 with the same parameters ( $Q_j = 0.3$  and  $\lambda_{\min} = 0.5$ ) is summarized in Table B3. The imposition of the constraint  $\lambda_{\min} = 0.5$  reduces the benchmark throughput from 13.4129 to 13.2950, a decrease of less than 1%. In this case, all 18 versions of the algorithm reached the 99.9% milestone. An examination of the solution for the case of  $\lambda_{\min} = 0$  (see Table 7) may provide some insight into why this network example is “easier” to solve than that of Network 1 with the same values of  $Q_j$  and  $\lambda_{\min}$ . In particular, there was only one underloaded circuit in the optimal solution for Network 2 with  $\lambda_{\min} = 0$ ; all of the other  $\lambda_j$ 's were 1.1 or greater. Thus, the imposition of the constraint  $\lambda_{\min} = 0.5$  has a smaller impact than for the case of Network 1. Additionally, as we commented in Section 8.4, our algorithm is more robust for Network 2 (with  $\lambda_{\min} = 0$ ) than Network 1 because all of the blocking probabilities are close to the QoS value at the optimal point (see Table 7). For the present case of  $\lambda_{\min} = 0.5$ , all except one of the normalized blocking probabilities are greater than 0.99, and the smallest is 0.9144. Thus this property is present for the case of  $\lambda_{\min} = 0.5$  as well.

## 9.2 Nonuniform Admission-Control Thresholds or Distribution of Transceivers

In all examples discussed thus far, the admission-control thresholds have been the same for each circuit, and the number of transceivers has been the same at each node. We now address several examples in which one of these conditions does not apply, and demonstrate how these examples provide added insight to the optimization problem under study.

### *Nonuniform Admission-Control Thresholds*

We examined the effect of nonuniform admission-control thresholds by considering Networks 1 and 2 with  $T_i = 6$  (i.e., six transceivers at each node), but  $X_j = 2$  for half of the circuits ( $j = 1, 3, 5, 7,$  and  $9$ ) and  $X_j = 4$  for the remaining circuits ( $j = 2, 4, 6, 8, 10$ ). Searches were performed for  $Q_j = 0.001$  and  $0.3$ . In all four of these examples, all 18 versions of the algorithm reached at least the 99.9% throughput milestone. It is interesting to compare these results to those obtained for examples in which  $X_j = 4$  for all 10 circuits. Of the four corresponding examples considered (i.e., two networks and two QoS values for each) for  $X_j = 4$  on all circuits, three produced results in which all 18 versions reached at least the 99.9% milestone and one did not (the case for Network 1 with  $Q_j = 0.3$ , which was illustrated in Table 4). Thus, it appears that in the example with nonuniform thresholds (half with thresholds of 4 and half with thresholds of 2) our algorithm is more robust than for the case in which all thresholds are equal to 4 (in the sense that all versions of the algorithm produce virtually the same solution, and that aggressive stepsize rules can reach the high milestones quickly).

Table 8 is an offered-load table for these four examples with nonuniform thresholds. We are most interested in the case of Network 1 with  $Q_j = 0.3$  because this case seemed to offer the most resistance to our algorithm for examples with a uniform threshold of four on each circuit. In the best solution for this case, all of the normalized blocking probabilities except  $\hat{P}_8$  are at least equal to 0.9998; in the worst solution, all except  $\hat{P}_8$  reach the value 0.9989. By contrast, four of the normalized blocking probabilities are typically less than 0.9 when  $X_j = 4$  on all circuits. Similar results are observed for the other three examples as well. Thus, these examples provide further support to our hypothesis that our algorithm is more robust for problems for which all of the circuit blocking probabilities at the optimal solution are close to the QoS value (see Section 8.4).

The above discussion raises the question of why the examples with reduced thresholds on some circuits permit solutions in which the blocking probabilities are close to the specified QoS values on a larger number of circuits. It appears that reducing the threshold on some circuits reduces the competition for transceivers, thus reducing the interaction among circuits and hence permitting more of them to approach the maximum permitted QoS value. Also, the smaller threshold values reduce the size of the search space, thus facilitating the search process.

Table 8 — Offered-Load Table for Networks 1 and 2 with  $T_i = 6$ ,  
 $X_1 = X_3 = X_5 = X_7 = X_9 = 2$ ;  $X_2 = X_4 = X_6 = X_8 = X_{10} = 4$ ;  
 “Best” and “Worst” Results Shown for  $Q_j = 0.001$  and  $0.3$

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	S
Network 1; $Q_j = 0.001$											
best	0.0361 1.0000	0.3462 1.0000	0.0221 0.9990	0.4008 1.0000	0.0400 1.0000	0.4130 1.0000	0.0457 0.9990	0.3198 1.0000	0.0257 1.0000	0.3462 1.0000	1.9937
worst	0.0362 0.9950	0.3483 1.0000	0.0131 0.8300	0.4019 1.0000	0.0399 0.9970	0.4142 1.0000	0.0456 0.9970	0.3221 1.0000	0.0248 0.9990	0.3483 1.0000	1.9922 99.92%
Network 1; $Q_j = 0.3$											
best	0.4803 0.9999	2.5860 0.9999	0.6257 0.9999	2.5336 1.0000	0.9132 0.9999	2.1888 0.9998	1.4205 1.0000	0.0007 0.9574	0.9613 1.0000	2.5860 0.9999	10.0075
worst	0.4645 0.9989	2.5318 0.9999	0.6053 0.9998	2.5164 0.9999	0.9059 1.0000	2.2301 0.9998	1.4190 0.9999	0.1465 0.9722	0.9323 0.9999	2.5318 0.9999	10.0003 99.93%
Network 2; $Q_j = 0.001$											
best	0.0437 1.0000	0.4187 1.0000	0.0456 0.9970	0.4153 1.0000	0.0437 0.9990	0.3383 1.0000	0.0428 0.9950	0.3642 1.0000	0.0457 0.9990	0.3307 1.0000	2.0864
worst	0.0437 0.9990	0.4187 1.0000	0.0454 0.9910	0.4153 1.0000	0.0437 0.9990	0.3383 0.9990	0.0427 0.9910	0.3641 0.9990	0.0455 0.9920	0.3307 0.9990	2.0859 99.98%
Network 2; $Q_j = 0.3$											
best	1.1203 1.0000	2.9031 1.0000	1.1536 1.0000	2.9785 0.9999	1.1174 0.9998	2.1149 1.0000	1.1862 0.9996	3.0568 0.9999	1.2783 0.9997	0.7333 0.9786	12.3549
worst	1.1162 0.9992	2.9185 0.9999	1.1544 0.9999	2.9606 0.9999	1.1201 0.9997	2.0767 0.9996	1.1783 0.9996	3.0482 0.9997	1.2805 0.9998	0.7892 0.9894	12.3537 99.99%

*Nonuniform Number of Transceivers*

We considered examples involving Network 1 with  $X_j = 4$ , in which seven nodes have eight transceivers each (nodes 4, 5, 6, 7, 13, 14, and 15)<sup>40</sup> and the rest have six transceivers each. QoS values of  $Q_j = 0.001$  and  $0.3$  were studied. Thus these examples differ from those of Tables 3 and 4 in that seven of the nodes have two additional transceivers. All 18 versions converged to at least the 99.9% throughput milestone for both QoS values. Table 9 shows that the normalized blocking probabilities are all close to 1.0, once again supporting our hypothesis that our algorithm is highly robust when addressing problems for which a solution exists in which all normalized blocking probabilities are close to 1.

It is intuitively satisfying that our algorithm becomes more robust as the number of transceivers at several nodes increases (as long as the admission-control thresholds are maintained at the same value). Similar to the case of reduced thresholds (Section 9.2.1), the availability of additional transceivers reduces the competition for transceivers among circuits (provided that the additional transceivers are located at heavily congested nodes).

<sup>40</sup>These particular nodes were chosen to receive the additional transceivers because they support a large number of circuits.

Table 9 — Offered-Load Table for Network 1 with  $X_j = 4$ ;  
 $T_i = 6$ , except  $T_4 = T_5 = T_6 = T_7 = T_{13} = T_{14} = T_{15} = 8$ ;  
 “Best” and “Worst” Results Shown for  $Q_j = 0.001$  and 0.3

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	$S$
$Q_j = 0.001$											
best	0.3814 1.0000	0.4074 0.9990	0.3774 0.9980	0.3700 0.9990	0.4098 0.9990	0.3981 1.0000	0.3902 1.0000	0.3771 0.9990	0.3851 0.9980	0.3822 0.9980	3.8746
worst	0.3815 1.0000	0.4073 0.9980	0.3768 0.9940	0.3702 0.9990	0.4091 0.9940	0.3977 0.9970	0.3898 0.9970	0.3770 0.9980	0.3852 0.9980	0.3824 0.9990	3.8731 99.96%
$Q_j = 0.3$											
best	2.5607 0.9999	3.0795 1.0000	2.2498 0.9999	1.3882 0.9998	3.0375 1.0000	1.4456 0.9487	3.0161 0.9999	1.0810 0.9872	2.3402 0.9998	1.6936 0.9999	15.3515
worst	2.5658 1.0000	3.0669 0.9999	2.2472 0.9999	1.3546 0.9950	3.0367 0.9999	1.4684 0.9506	3.0166 0.9999	1.1238 0.9949	2.3332 0.9998	1.6788 0.9996	15.3506 99.99%

### 9.3 Nonuniform QoS Requirements

In all examples discussed thus far, the QoS requirement was the same for all circuits, i.e.,  $Q_1 = \dots = Q_J$ . We now address the more general case, in which some circuits have more-stringent QoS requirements than others. In studying such examples, we demonstrate not only the capability of wireless networks to support such disparate QoS requirements, but also the capability of our algorithm to find good (and hopefully optimal) solutions in these more-difficult cases.

We consider examples in which the circuits must satisfy one of the following:

$$Q_1 = \dots = Q_{\frac{J}{2}} = 0.001; \quad Q_{\frac{J}{2}+1} = \dots = Q_J = 0.3 \quad (37)$$

$$Q_1 = \dots = Q_{\frac{J}{2}} = 0.3; \quad Q_{\frac{J}{2}+1} = \dots = Q_J = 0.001. \quad (38)$$

Thus, half the circuits must satisfy the more stringent constraint (QoS = 0.001), while the other half must satisfy the looser one (QoS = 0.3).

As in the other examples we have studied, we have attempted to solve this problem by using versions of the algorithm that use the projection, as well as those that do not. To accommodate the vastly different values of the QoS parameter, membership in the projection set  $\Sigma$  is based on the normalized value of blocking probability. Thus, when using the projection rule based on the parameter  $v$ , the decision on which circuits to include in  $\Sigma$  is made separately for the QoS = 0.001 group and separately for the QoS = 0.3 group. However, consistently poor results were obtained when any version of the projection was used; the best throughput values were approximately 5% less than those obtained without using the projection operation.<sup>41</sup> Apparently, as a result of the vast difference in QoS requirements for the different circuits, there is no well-defined contour that can be tracked by the projection vector  $D$ . In particular, small changes in some of the offered loads that have only minor impact on the contours of constant blocking probability for circuits in the QoS = 0.3 group can substantially alter the shape of the contours of constant blocking probability for the QoS = 0.001 group.<sup>42</sup> Consequently, these small changes can have a significant impact on the set of circuits from the QoS = 0.001 group that are included in  $\Sigma$ .

<sup>41</sup>Because of the high degree of consistency in throughput obtained among all runs for different versions of the algorithm for the many examples considered in this report, as well as the flatness of the throughput over a wide range of offered load, a difference in throughput of 5% is quite large.

<sup>42</sup>Small changes in one or more of the  $\lambda_j$ 's can have a significant impact on the contours of constant blocking probability, even when the QoS requirements are the same for all circuits.

We therefore use the algorithm with  $\Sigma = \emptyset$  to solve our problem. However, we have observed that it is necessary to use a larger stepsize for the circuits that are characterized by the larger value of QoS. This is consistent with the fact that the initial stepsize used for core run examples with  $Q_j = 0.3$  is much larger than that used for examples with  $Q_j = 0.001$ . Our approach for the choice of initial stepsize is quite similar to that used for the case of uniform QoS values. We perform a short pilot run in which the stepsizes are the same for all circuits, and find a stepsize value for which the inadmissible region is entered for the first time at about the fifth iteration. In the actual run, this stepsize value is used for the circuits for which QoS = 0.001; a stepsize value five times this size is used for the circuits for which QoS = 0.3.<sup>43</sup>

### Sample Runs

We have studied Networks 1, 2, and 3 with  $T_i = 6$  and  $X_j = 4$ . For each network we have considered the QoS requirements defined by Eqs. (37) and (38); thus six examples were studied. Figure 21 shows the evolution of admissible throughput and arrival rates for the example of Network 1 with QoS requirements defined by Eq. (37), i.e., the first five circuits have the more-stringent QoS requirement. Figure 22 shows a similar example for Network 2. In consideration of the disparate QoS requirements of the two groups and the anticipated resulting difficulty in achieving the optimal solution, a conservative stepsize rule was used. It appears, though, that this stepsize rule may have been overly conservative because little change occurs between iterations 200 and 400.

The milestones shown in Figs. 21 and 22 are based on achieving the indicated percentage of the best throughput value in each particular run. No benchmark of maximum achievable throughput is available for these examples because only a single run was performed for each. In each of the six examples we have investigated (three networks and two sets of QoS circuit groupings for each network), the 99% milestone was achieved before iteration 1100.

Table 10 summarizes the offered loads and corresponding normalized circuit blocking probabilities for the six runs. In many cases, the blocking probabilities of the circuits for which QoS = 0.001 are close to the maximum permissible values. However, only a small number of circuits in the QoS = 0.3 group (typically about 2, but in one case none) have blocking probabilities close to the QoS value. The achievable throughput can depend strongly on which circuits are in the two QoS groups and is difficult to predict. For example, the difference for the two examples involving Network 1 are modest, while that for Network 2 is significant, and that for Network 3 is dramatic. An extreme case is that of Network 3 for which the first four circuits must satisfy the 0.001 QoS constraint. In this case, the maximum circuit blocking probability among the QoS = 0.3 group is less than 1% of the permissible value. Thus, the requirement to satisfy the more-stringent QoS constraint on half of the circuits severely limits the offered loads of the remaining circuits as well. The throughput of 2.3459 in this example is only 4.5% greater than that for the case in which all the circuits must satisfy the 0.001 constraint (see Table 7).

In Section 8.4 we commented that it is not surprising that, for the case of Network 1 with large QoS values (such as 0.3), not all blocking probabilities were close to the maximum permitted value. We attributed this behavior to the higher level of interaction among the circuits that is characteristic of higher offered loads. In the current example in which half of the circuits are required to satisfy significantly more-stringent QoS requirements, it is therefore expected that even fewer of the circuits with less stringent QoS requirements will be able to support offered loads for which their blocking probabilities are at the maximum permitted values. As demonstrated here, such behavior does, in fact, occur. Thus, the circuits with less-stringent QoS requirements are often strongly affected by the circuits with more-stringent QoS requirements.

<sup>43</sup>This simple rule appears to work well for these QoS values in the examples we have studied; certainly no claim of optimality is made. The relative values of stepsizes for the two classes of traffic should depend on the values of QoS of the more- and less-stringent groups.

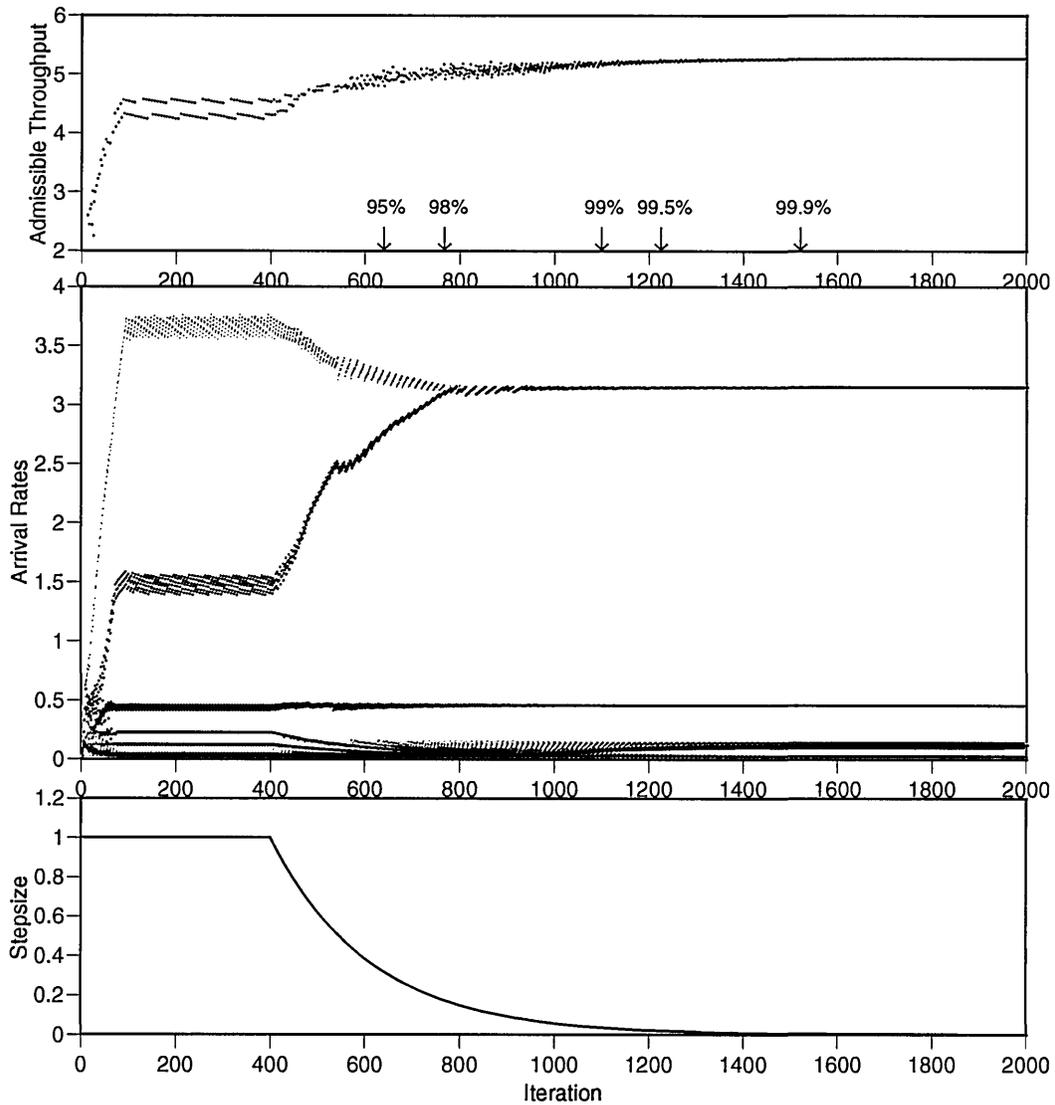


Fig. 21 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $Q_1 = \dots = Q_5 = 0.001$ ,  $Q_6 = \dots = Q_{10} = 0.3$

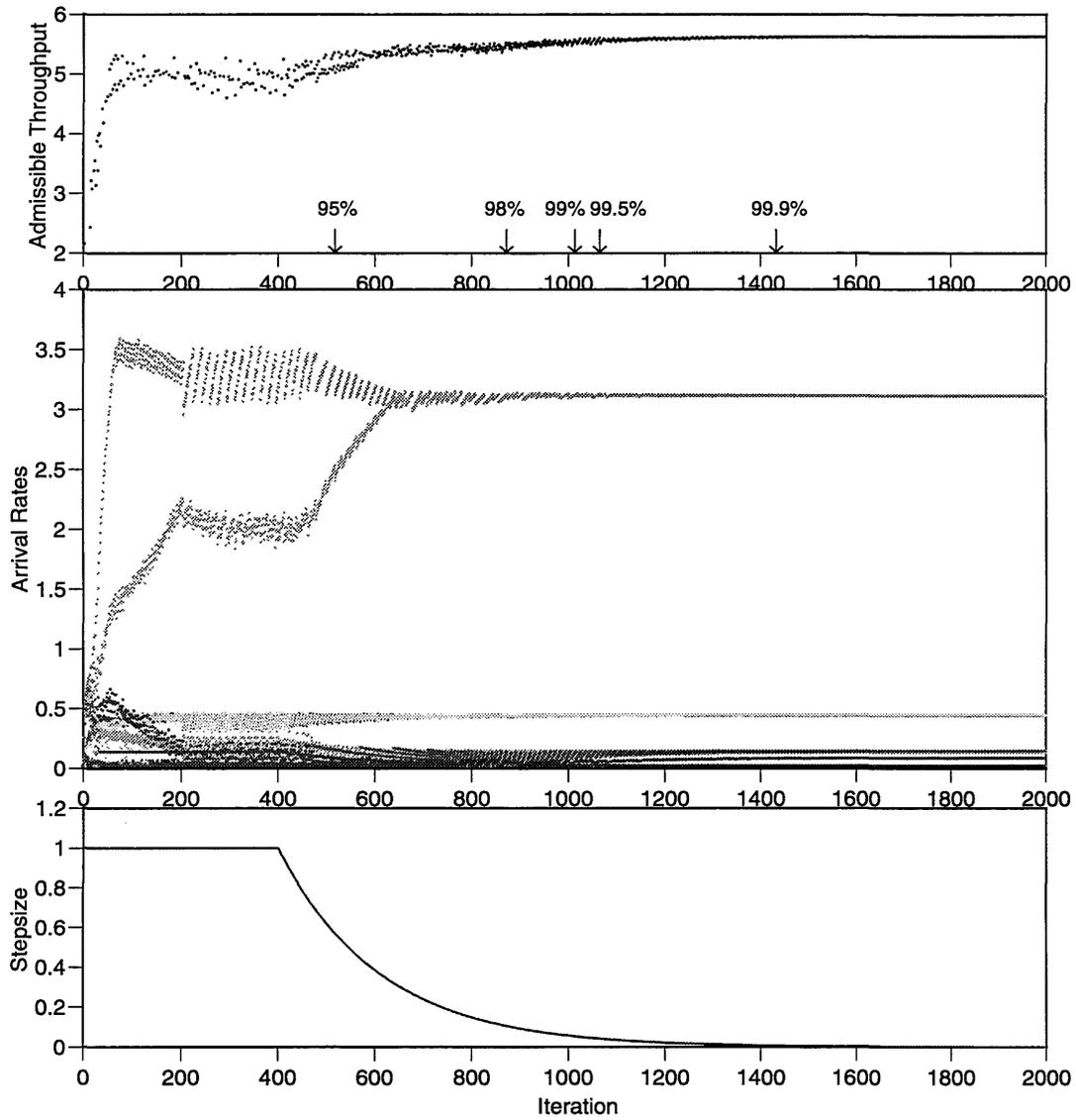


Fig. 22 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 2 with  $T_i = 6$ ,  $X_j = 4$ ,  $Q_1 = \dots = Q_5 = 0.001$ ,  $Q_6 = \dots = Q_{10} = 0.3$

Table 10 — Offered Loads and Normalized Circuit Blocking Probabilities for Examples with Two Different QoS Values (0.001 and 0.3);  $T_i = 6$ ,  $X_j = 4$

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	$S$
Network 1 ( $10^{-3}$ , 0.3)	0.4315 1.0000	0.0001 0.9980	0.0763 0.8290	0.1185 0.9990	0.0059 0.9980	0.1170 0.0035	3.1244 1.0000	0.0003 0.0034	0.0981 0.0042	3.1253 1.0000	5.2216
Network 1 (0.3, $10^{-3}$ )	0.0295 0.0017	3.8904 1.0000	0.0609 0.0018	0.0295 0.0017	3.8904 1.0000	0.0000 0.9990	0.0587 0.4990	0.0001 0.9990	0.0563 0.4620	0.0001 0.9990	5.6815
Network 2 ( $10^{-3}$ , 0.3)	0.0001 0.9990	0.1245 1.0000	0.4221 1.0000	0.4239 1.0000	0.0001 0.9990	3.0926 1.0000	0.1134 0.0028	3.0930 1.0000	0.1251 0.0032	0.0681 0.8055	5.5894
Network 2 (0.3, $10^{-3}$ )	0.0195 0.0035	3.5919 1.0000	0.0445 0.6044	2.2958 1.0000	3.5678 1.0000	0.0002 0.9980	0.0422 0.9990	0.4221 1.0000	0.4222 1.0000	0.0687 1.0000	7.6293
Network 3 ( $10^{-3}$ , 0.3)	0.1646 1.0000	0.0000 1.0000	0.1737 1.0000	0.3026 1.0000	0.3931 0.0049	0.3674 0.0050	0.5576 0.0097	0.3908 0.0050			2.3459
Network 3 (0.3, $10^{-3}$ )	0.0333 0.0036	3.8903 1.0000	0.0653 0.0037	3.8903 1.0000	0.0036 0.9990	0.0297 0.9980	0.0278 0.9970	0.0001 0.9990			5.6060

column 1: ( $10^{-3}$ , 0.3) means that  $Q_1 = \dots = Q_{j2} = 0.001$ ;  $Q_{j2+1} = \dots = Q_j = 0.3$ , etc.

columns 2 – 11: upper entries:  $\lambda_j$   
lower entries:  $\hat{P}_j$

## 10. AN ALTERNATIVE QOS CONSTRAINT: AVERAGE BLOCKING PROBABILITY

Thus far, we have required that each circuit satisfy the QoS constraint on blocking probability. In this section, we consider an alternative version of the QoS constraint in which we require only that the average blocking probability in the network satisfy this constraint. Whether the QoS constraint should be applied to the average blocking probability or to each individual circuit is a decision to be made by the network designer/administrator. We demonstrate in this section that relaxing the QoS constraint in this manner results not only in higher throughput values, but also in considerably faster convergence. To distinguish these two forms of the QoS constraint, we henceforth refer to them as  $P_{av}$  and  $P_{max}$  constraints, respectively. In addition, we demonstrate that in some cases the use of a QoS constraint based on the average blocking probability can provide a good initial point for a search based on the satisfaction of the QoS constraint on each individual circuit.

### 10.1 Mathematical Model

Recall that in Eq. (9) the overall (average) blocking probability  $P_{av}$  (i.e., the fraction of calls arriving to the network that are blocked, regardless of the circuit to which they arrive) was defined as the ratio of the expected number of blocked calls per unit time (summed over all circuits) to the expected total number of call arrivals per unit time:

$$P_{av} = \frac{1}{\Lambda} \sum_{j=1}^J \lambda_j P_j, \quad (39)$$

where  $\Lambda = \lambda_1 + \dots + \lambda_J$ . Thus, some circuits may be permitted relatively large blocking probabilities provided that the overall blocking probability satisfies the QoS constraint,<sup>44</sup> i.e., that  $P_{av} \leq Q = \text{QoS}$ .

<sup>44</sup>Circuits with very small offered loads do not affect  $P_{av}$  significantly, even when their blocking probabilities are high, because each value of  $P_j$  is weighted by the corresponding  $\lambda_j$ .

Following the approach of Section 4.1, the augmented Lagrangian function can now be written as

$$L = S - \gamma \max(0, P_{av} - Q) - \frac{d}{2} [\max(0, P_{av} - Q)]^2. \quad (40)$$

Note that this expression has the same form as Eq. (19), except that there is a single constraint on  $P_{av}$  rather than individual constraints on each of the  $J$  blocking probabilities  $P_j$ . The resulting partial derivatives are

$$\frac{\partial L}{\partial \lambda_i} = \frac{\partial S}{\partial \lambda_i} - \left[ \frac{\gamma + d(P_{av} - Q)}{\sqrt{Q}} \right] \frac{\partial P_{av}}{\partial \lambda_i} 1(P_{av} > Q), \quad (41)$$

where

$$\begin{aligned} \frac{\partial P_{av}}{\partial \lambda_i} &= \frac{(\lambda_1 + \dots + \lambda_J) \left[ P_i + \sum_{j=1}^J \lambda_j \frac{\partial P_j}{\partial \lambda_i} \right] - \sum_{j=1}^J \lambda_j P_j}{(\lambda_1 + \dots + \lambda_J)^2} \\ &= \frac{\Lambda P_i + \sum_{j=1}^J \lambda_j \left\{ \Lambda \frac{\partial P_j}{\partial \lambda_i} - P_j \right\}}{\Lambda^2}. \end{aligned} \quad (42)$$

The iterative algorithm proceeds in the same manner as for the case in which the QoS constraint must be satisfied on each individual circuit.

We again consider the use of the projection to guide the search. However, since only the average blocking probability  $P_{av}$  is of interest (and not the individual values of the  $P_j$ 's), we use  $P_{av}$  instead of  $P_\Sigma$  in the definition of  $D$ :

$$D = \begin{cases} \nabla S - \frac{\nabla S \cdot \nabla P_{av}}{\|\nabla P_{av}\|^2} \nabla P_{av}, & \text{if } \left\| \nabla S - \frac{\nabla S \cdot \nabla P_{av}}{\|\nabla P_{av}\|^2} \nabla P_{av} \right\| > \tau \|\nabla S\|, \\ \nabla S, & \text{otherwise} \end{cases} \quad (43)$$

where we have again used  $\tau = 0.1$ . Thus, the effect of the projection operation is to discourage the increase of the average blocking probability by removing the component of  $\partial S / \partial \lambda_i$  that results in an increase in  $P_{av}$ .

## 10.2 Performance Results

We tested this formulation on Networks 1, 2, and 3, and obtained rapid convergence to optimal solutions. Typically, the speed of convergence was similar, whether or not the projection was used. The figures shown in this section are for examples that do not use the projection (i.e.,  $\Sigma = \emptyset$ ). Figures 23 and 24 show the evolution of admissible throughput and offered load for Network 1 with  $T_i = 6$  and  $X_j = 4$ , for the cases of  $Q = 0.001$  and  $0.3$ , respectively.

Table 11 shows the optimal offered loads, the corresponding normalized blocking probabilities, and the throughput for Network 1 with  $T_i = 6$  and  $X_j = 4$ , for QoS values of  $0.001$  and  $0.3$ . Results are shown for both forms of the QoS constraint, namely the average ( $P_{av}$ ) and maximum individual circuit ( $P_{max}$ ) cases. The results for the  $P_{av}$  case are based on the use of no projection operation (results using the projection are virtually identical). The results for the  $P_{max}$  case are based on the best run among the 18 versions of the algorithm (i.e., the one that provided the highest throughput). For  $Q = 0.001$ , the relaxation of the QoS constraint has had a negligible impact on overall throughput, which has increased from 2.6645 to 2.6674.

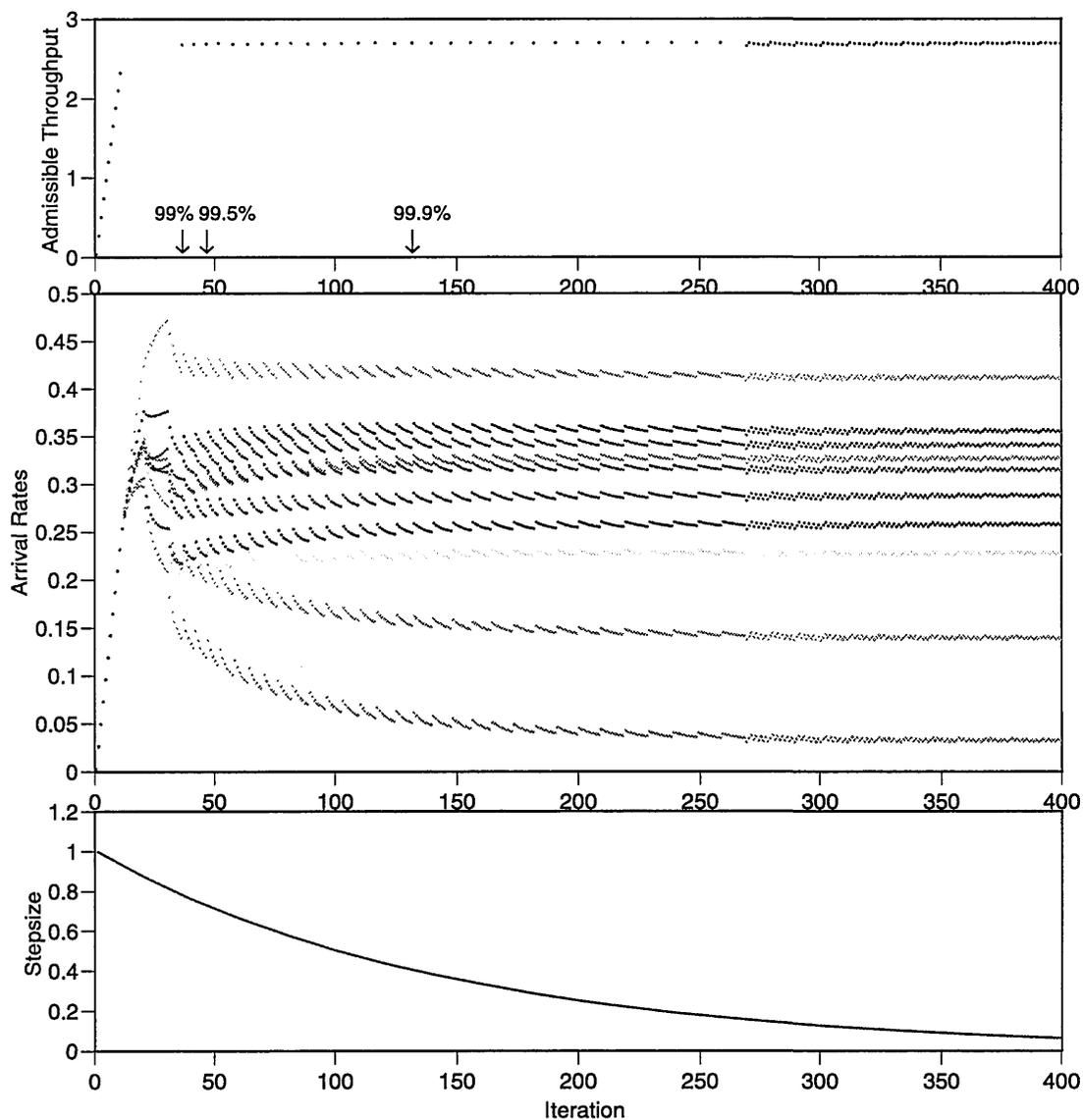


Fig. 23 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $Q = 0.001$ ;  $P_{av}$  form of QoS constraint; no projection used

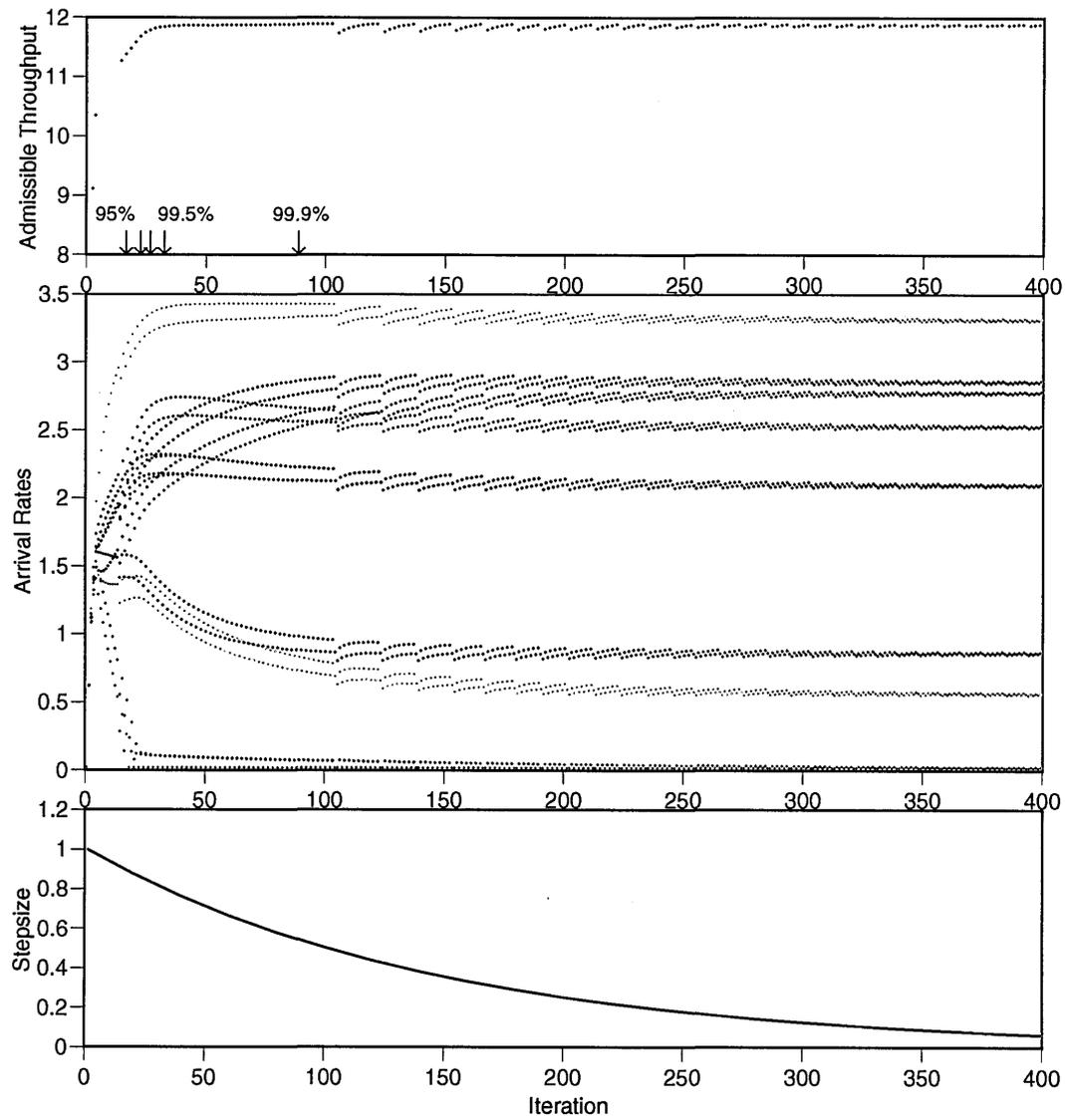


Fig. 24 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $Q = 0.3$ ;  $P_{av}$  form of QoS constraint; no projection used

Table 11 — Offered Loads and Normalized Circuit Blocking Probabilities for Network 1 under Both  $P_{av}$  and  $P_{max}$  Forms of the QoS Constraint on Average Blocking Probability;  $T_i = 6$ ,  $X_j = 4$

QoS	form of constr	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	$S$
0.001	Average	0.2557 0.9530	0.3248 0.9990	0.3126 0.9850	0.2255 0.9410	0.3530 1.0260	0.1360 0.9280	0.4087 1.0990	0.0289 0.9260	0.3385 1.0070	0.2863 0.9670	2.6674
	Max	0.2527 1.0000	0.3230 1.0000	0.3083 1.0000	0.2337 1.0000	0.3413 1.0000	0.1545 0.9790	0.3947 1.0000	0.0231 0.9300	0.3374 1.0000	0.2984 1.0000	2.6645
0.3	Average	2.7669 1.0131	2.8347 0.9921	2.0771 1.0014	0.5338 0.9823	2.5085 0.9669	0.0000 1.1590	3.2890 1.0353	0.0000 1.1484	2.0771 1.0014	0.8449 0.9486	11.8524
	Max	3.3160 0.9997	3.5230 0.9999	1.9599 0.8975	0.0009 0.8862	2.0194 0.7656	0.0008 0.9999	3.3179 1.0000	0.0005 1.0000	1.9675 0.9011	0.0036 0.7542	11.5380

column 2: form of constraint indicates whether the  $P_{av}$  or  $P_{max}$  case applies to the following columns  
columns 3 – 12:  $\lambda_j$  shows offered load to circuit  $j$  at best solution

$\hat{P}_j$  shows normalized circuit blocking probability at best solution

The individual offered loads and the corresponding normalized blocking probabilities are also only slightly changed from the values obtained for the  $P_{max}$  case (only two of the blocking probabilities exceed 0.001, the larger by 2.6%). For  $Q = 0.3$ , the use of the average QoS constraint results in significant changes in some of the offered load values, as well as an increase in throughput from 11.5380 to 11.8524 (an increase of 2.7%). The normalized blocking probabilities of circuits 6 and 8 are about 1.15, i.e., these blocking probabilities are about 15% greater than the specified QoS value.

Table 12 shows similar results for Network 2. When the constraint is relaxed to the average form, the impact on throughput is again negligible when QoS = 0.001. For QoS = 0.3, the offered loads change more than in the QoS = 0.001 case, but significantly less than for the case of Network 1 with QoS = 0.3. The increase in throughput is only 0.18%, which is considerably less than that observed for Network 1. It is also noteworthy that the maximum normalized circuit blocking probability for the  $P_{av}$  case is 1.048, as compared to 1.159 for Network 1.

Finally, Table 13 shows similar results for Network 3, for which the differences between the  $P_{av}$  and  $P_{max}$  cases are even smaller. The maximum normalized circuit blocking probability for the  $P_{av}$  case is 1.0094 for QoS = 0.3 and 1.0360 for QoS = 0.001.

Table 12 — Offered Loads and Normalized Circuit Blocking Probabilities for Network 2 under Both  $P_{av}$  and  $P_{max}$  Forms of the QoS Constraint on Average Blocking Probability;  $T_i = 6$ ,  $X_j = 4$

QoS	form of constr	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	$S$
0.001	Average	0.2153 0.9280	0.3839 1.0580	0.3061 0.9720	0.3658 1.0300	0.2439 0.9390	0.2869 0.9620	0.2858 0.9570	0.3601 1.0280	0.4001 1.0840	0.2279 0.9330	3.0726
	Max	0.2326 1.0000	0.3724 1.0000	0.3129 0.9990	0.3571 0.9990	0.2460 1.0000	0.2819 1.0000	0.2941 0.9990	0.3546 0.9990	0.3856 1.0000	0.2357 1.0000	3.0700
0.3	Average	0.0000 1.0482	3.4830 1.0385	0.9704 0.9528	1.9350 0.9765	2.5908 1.0008	1.1287 0.9865	1.2356 0.9619	2.4761 0.9713	3.5499 1.0300	1.8258 0.9901	13.4366
	Max	0.0002 0.9999	3.2507 0.9999	1.1397 0.9999	2.0615 1.0000	2.3653 0.9819	1.3159 1.0000	1.2968 0.9999	2.5453 0.9999	3.4367 0.9999	1.7296 0.9989	13.4129

Table 13 — Offered Loads and Normalized Circuit Blocking Probabilities for Network 3 under Both  $P_{av}$  and  $P_{max}$  Forms of the QoS Constraint on Average Blocking Probability;  $T_i = 6$ ,  $X_j = 4$

QoS	form of constr	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$S$
0.001	Average	0.2390 0.9730	0.2473 0.9820	0.2835 0.9940	0.3332 1.0360	0.2790 0.9930	0.2783 0.9940	0.3204 1.0240	0.2655 0.9880	2.2441
	Max	0.2481 0.9990	0.2529 1.0000	0.2852 1.0000	0.3242 1.0000	0.2808 1.0000	0.2777 1.0000	0.3122 1.0000	0.2646 1.0000	2.2436
0.3	Average	1.0953 0.9961	1.2371 1.0069	1.4422 0.9809	2.2493 0.9979	1.6959 0.9998	1.7494 1.0045	2.2185 1.0017	17606 1.0094	9.4138
	Max	1.1174 1.0000	1.2697 0.9998	1.5078 0.9999	2.2527 1.0000	1.6666 0.9999	1.7254 1.0000	2.1911 1.0000	1.7158 0.9999	9.4128

A potentially useful technique that exploits the similarity between the  $P_{av}$  and  $P_{max}$  solutions is discussed in the next subsection.

### 10.3 Combined Use of Average and Individual QoS Constraints

The results of Section 10.2 indicate that the use of the average blocking probability as the QoS constraint typically provides a solution that is “similar” to that obtained using the individual blocking probabilities, especially for small QoS values (i.e., 0.001). Thus, it brings us close to the neighborhood of the optimal solution of our original problem. This similarity has led us to consider the use of an alternative approach during phase 1 of the algorithm. Instead of using the projection in conjunction with QoS constraints on each individual circuit, we have considered using the QoS constraint on the average blocking probability without use of the projection operation. Then, at the beginning of phase 2 the QoS constraint is applied to each individual circuit as in the original formulation. Thus, the formulation based on the average blocking probability is used to determine an initial condition for the problem in which the QoS constraint must be satisfied on every circuit.

Figure 25 shows the evolution of admissible throughput, offered load, and stepsize for the example of Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and QoS = 0.001. The average form of the QoS constraint is used for 100 iterations; after this point, the QoS constraint must be satisfied on each individual circuit. Note that the definition of admissibility changes at iteration 100. Solutions classified as admissible during the first phase are typically not admissible when the  $P_{max}$  criterion is applied, and few admissible solutions (that satisfy  $P_{max}$ ) are observed during the first phase. The admissible solutions and milestones shown on the graph are based only on solutions that are admissible under the  $P_{max}$  criterion. The first admissible solution (based on  $P_{max}$ ) is at the 98% milestone level (based on the benchmark throughput achieved for this network example, as discussed in Section 8.2), which occurs at iteration 112. The 99% and 99.5% milestones are reached shortly thereafter (at iterations 127 and 149, respectively).

Figure 26 shows curves for the same network example, but for QoS = 0.3. Again, the 98% milestone is achieved rapidly; however, the higher milestones are not reached. Apparently, at iteration 100 (when the performance criterion changes to  $P_{max}$ ) the stepsize is too small to make sufficient progress toward the optimal point in this example because the  $P_{av}$  and  $P_{max}$  solutions are relatively far apart. Some experimentation with larger stepsizes when the criterion changes to  $P_{max}$  (e.g., increasing it from 0.05 to 0.2 at iteration 100, and then letting it decrease exponentially) have produced solutions that reach the 99% milestone relatively rapidly. However, the need to use a more-conservative stepsize rule when switching between  $P_{av}$  and  $P_{max}$  for some network examples may make it difficult to reach the higher milestones rapidly. In such cases, the combined  $P_{av}/P_{max}$  method may not provide faster convergence to good solutions than those that use the  $P_{max}$  criterion throughout.

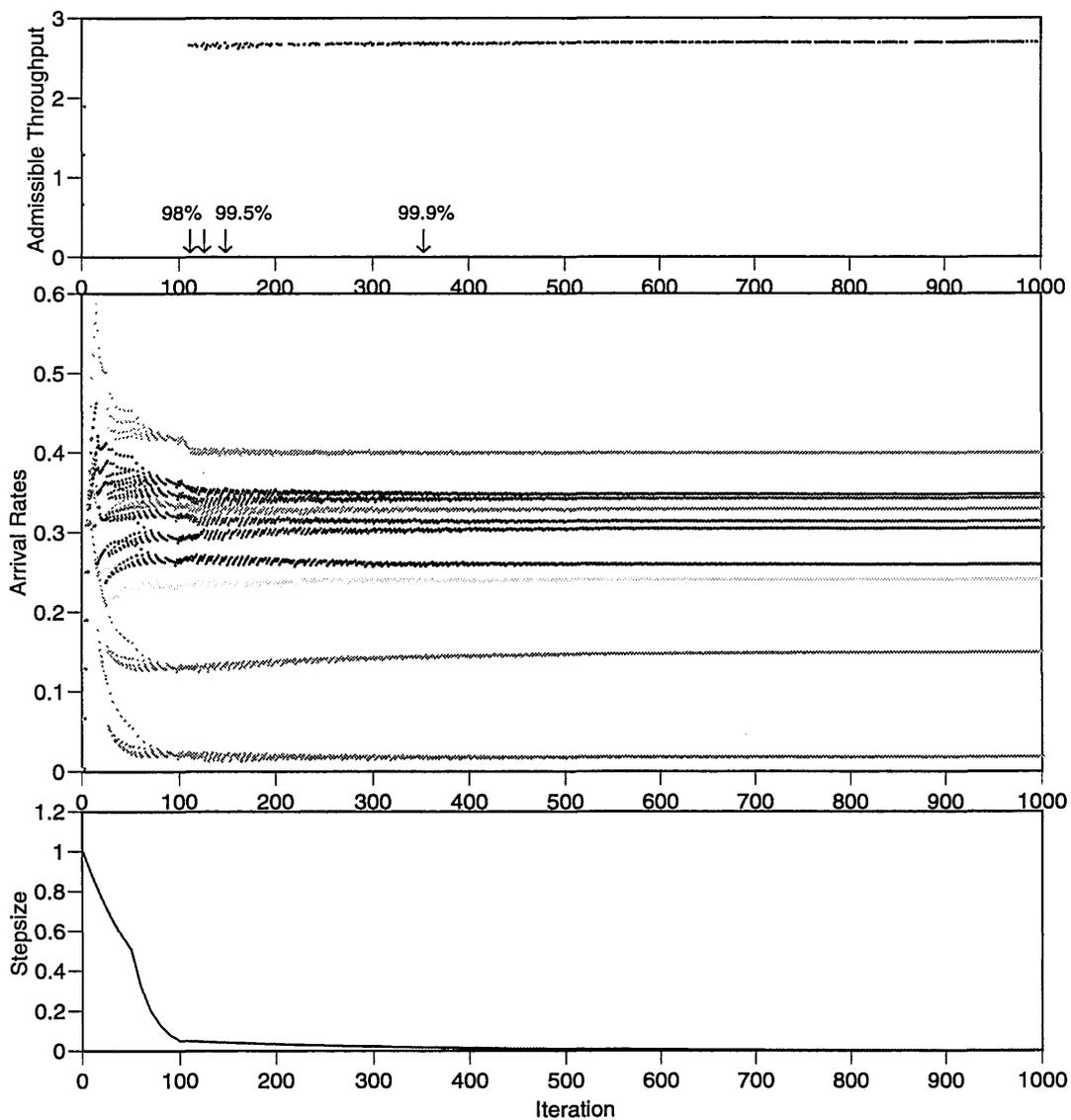


Fig. 25 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $QoS = 0.001$ ;  $P_{av}$  form of QoS constraint used for first 100 iterations,  $P_{max}$  thereafter; no projection used

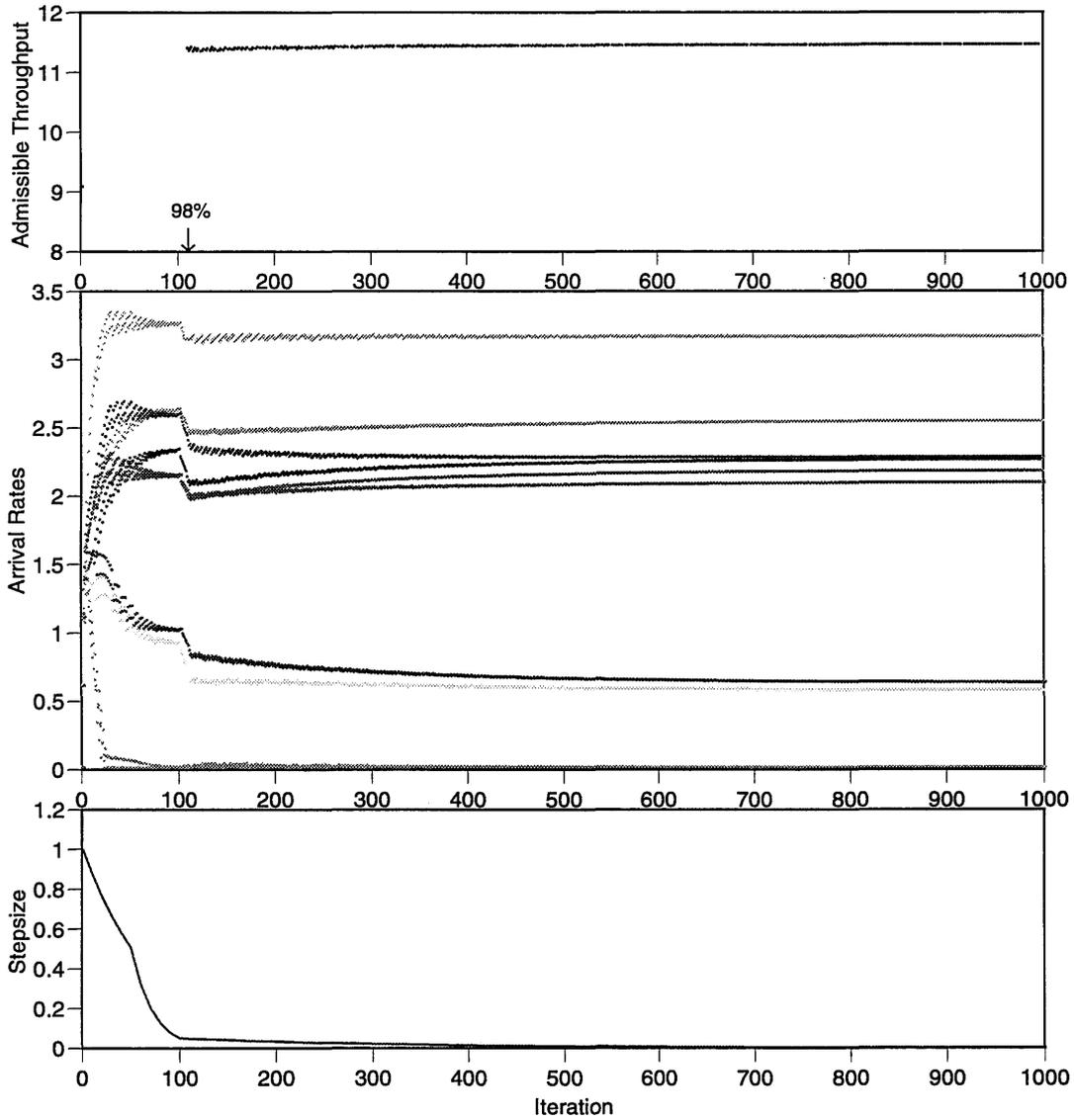


Fig. 26 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $QoS = 0.3$ ;  $P_{av}$  form of QoS constraint used for first 100 iterations,  $P_{max}$  thereafter; no projection used

Figures 27 and 28 show curves for Network 2 with the same parameters, for QoS = 0.001 and 0.3, respectively. In both cases, convergence to the 99.5% milestone is extremely fast, and convergence to the 99.9% milestone is also among the fastest observed for these network examples (see Tables A4 and A5).

#### *Comments on the Effectiveness of the Combined $P_{av}/P_{max}$ Approach*

For three of the four examples shown here, the performance obtained by using the combined approach is comparable to that of the better versions that are based on the use of the projection in one or more phases. We hypothesize that this approach would be most effective when the solution based on the  $P_{av}$  QoS criterion is most similar to that based on the  $P_{max}$  QoS criterion. Of course, it is not possible to determine with certainty that the solutions are similar without solving both problems. However, the circuit blocking probabilities shown in Tables 11-13 may provide a helpful clue. For example, Table 11 shows that, for  $Q = 0.001$ , the maximum normalized blocking probability is 1.099, and the minimum is 0.926. Thus, little has to be done to coax the offered load values to a point that maximizes throughput while satisfying the  $P_{max}$  QoS criterion. Comparing the offered loads under the  $P_{av}$  and  $P_{max}$  QoS constraints shows little difference between these two cases; admissible throughput under the  $P_{av}$  and  $P_{max}$  QoS constraints shows that they differ by 0.1%. By contrast, the same table shows that, for  $Q = 0.3$ , two of the normalized blocking probabilities are approximately 1.15. Thus, a significant change in the offered load vector is required to satisfy the  $P_{max}$  QoS constraint. In this example, a comparison of the offered loads under the  $P_{av}$  and  $P_{max}$  QoS constraints shows that there is significant difference; furthermore, the throughput for the  $P_{av}$  case is 2.7% greater than for the  $P_{max}$  case.

Table 12 shows similar results for Network 2. Results for  $Q = 0.001$  are comparable to those for Network 1. However, results for  $Q = 0.3$  show that the maximum normalized circuit blocking probability associated with the solution based on the  $P_{av}$  QoS constraint is 1.0482 (the deviation from 0.3 is thus about one third of that observed for Network 1). Thus, it is expected that the solutions for the  $P_{max}$  and  $P_{av}$  cases would be more similar than those for Network 1. In fact, the difference in throughput between the  $P_{av}$  and  $P_{max}$  cases is 0.18% (considerably less than the 2.7% for Network 1), and although the offered load values for the  $P_{max}$  case are not as close to the  $P_{av}$  case as they were for QoS = 0.001, they are considerably closer (percentagewise) than they were for Network 1.

Similar observations apply to the case of Network 3 (see Table 13).

Thus, we feel that the combined use of the  $P_{av}$  and  $P_{max}$  constraints may provide an efficient method for determining optimal offered loads for some problems. We hypothesize that this method will work well when none of the individual circuit blocking probabilities in the  $P_{av}$  solution exceeds the QoS constraint value by a "significant" amount, although we have not quantified the tolerable amount by which QoS can be exceeded. It appears that an aggressive stepsize rule works well when this condition is satisfied, and that a more conservative stepsize rule is appropriate when there is more asymmetry in the values of the blocking probabilities.

In Section 8.4 we commented that the algorithm provides more-robust performance for Networks 2 and 3 than for Network 1 under the  $P_{max}$  constraint. We hypothesized that one reason for this might be that for these two cases there seems to be relatively little interaction among the circuits. This is suggested by the fact that all of the circuit blocking probabilities are very close to the QoS value. Perhaps the fact that the  $P_{av}$  solution is so close to the  $P_{max}$  solution is another example of little interaction. However, further research is needed to establish any definite conclusions on this matter.

We now summarize our tentative conclusions on the combined  $P_{av}/P_{max}$  approach. When the normalized blocking probabilities based on the  $P_{av}$  form of the QoS constraint are all close to 1.0, it appears that the  $P_{av}/P_{max}$  approach provides rapid convergence to nearly optimal solutions, i.e., to the 99.5% milestone. Another advantage of this approach is that experimentation is not needed to determine a good set  $\Sigma$  to be included in the projection operation (or equivalently the value of the parameter  $v$ ) since this method appears to work well whether or not the projection is used. However, further testing is needed to determine with more certainty the class of problems for which the combined  $P_{av}/P_{max}$  approach works well.

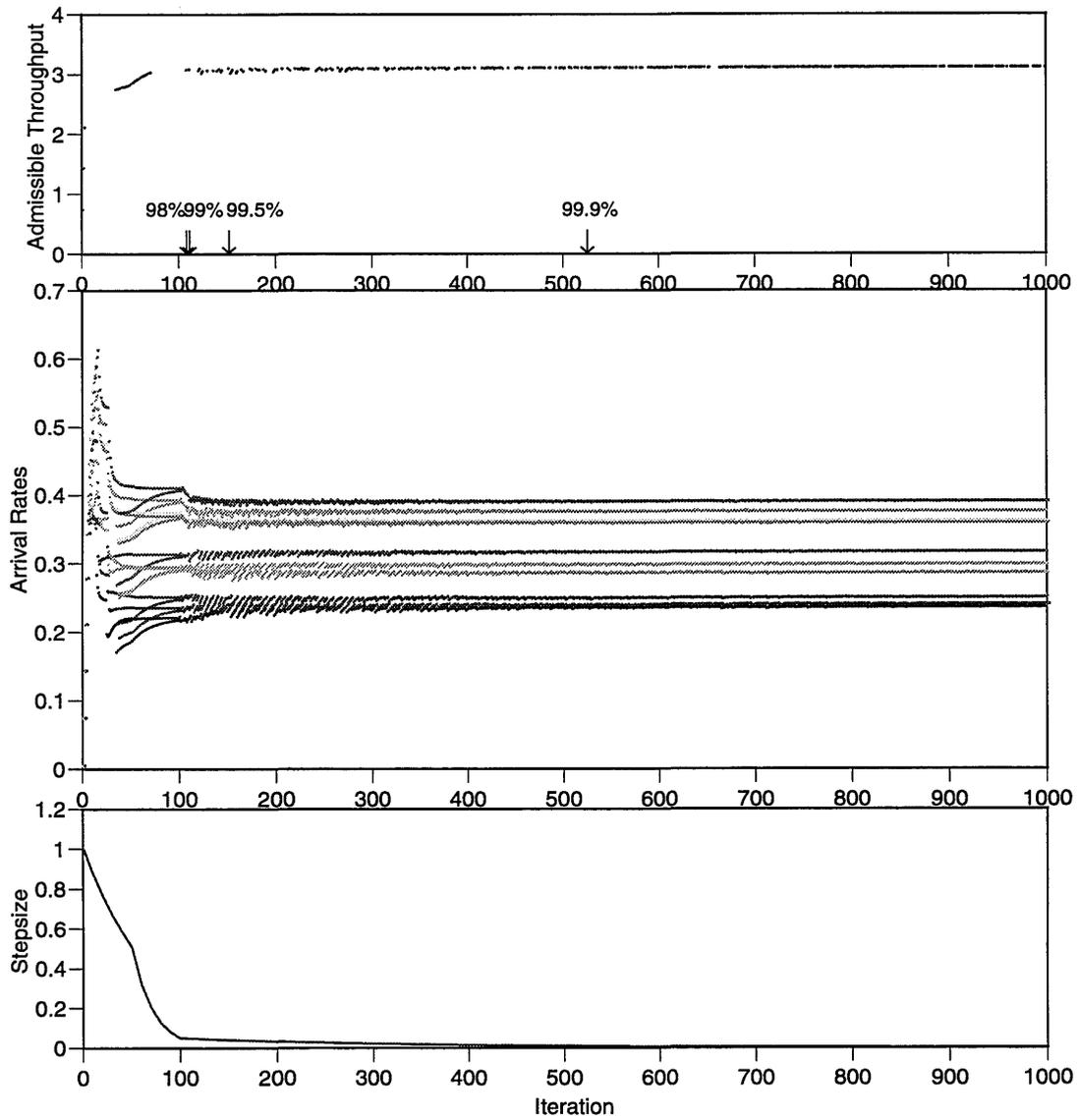


Fig. 27 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 2 with  $T_i = 6$ ,  $X_j = 4$ ,  $QoS = 0.001$ ;  $P_{av}$  form of QoS constraint used for first 100 iterations,  $P_{max}$  thereafter; no projection used

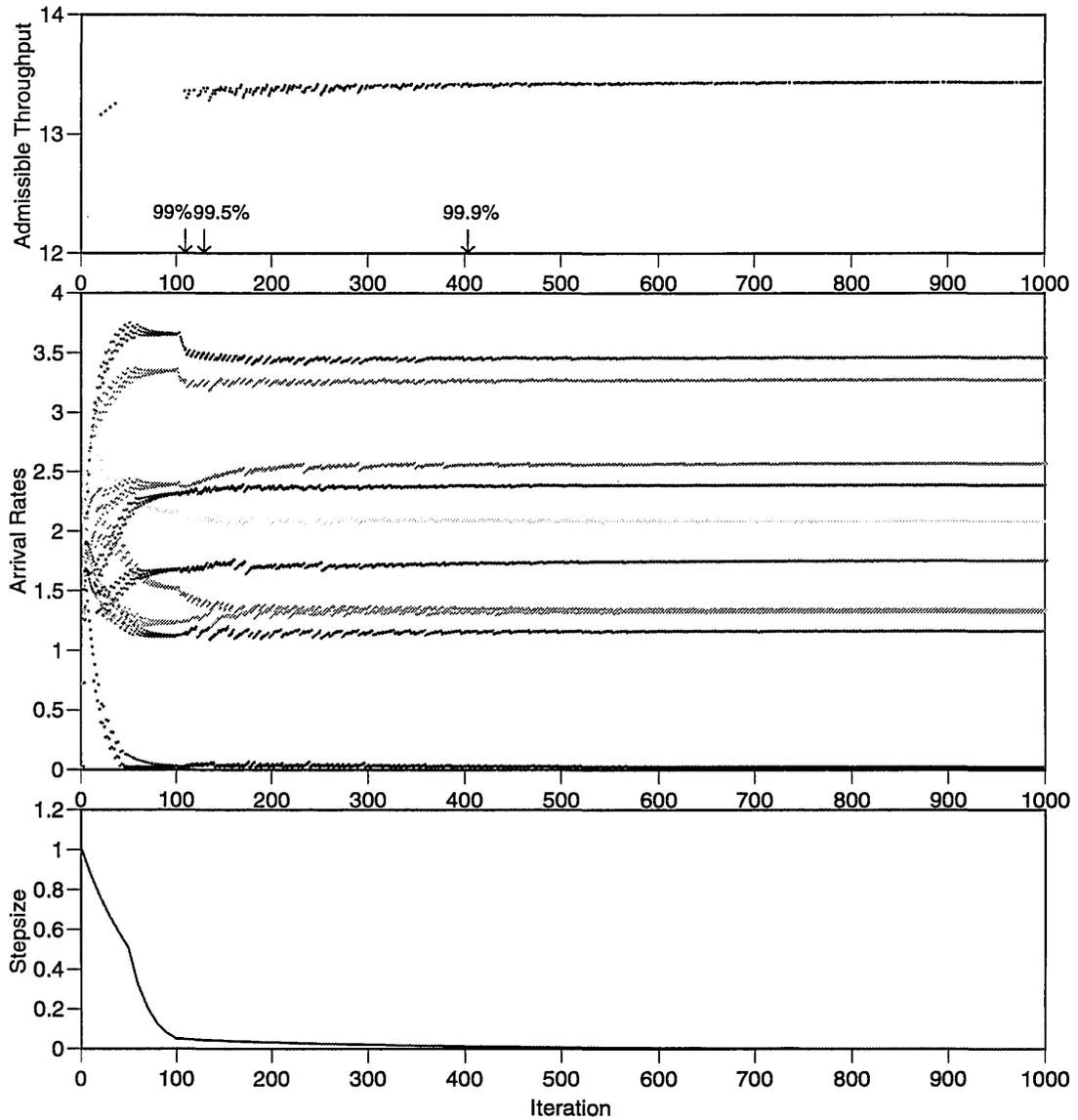


Fig. 28 — Evolution of admissible throughput, offered load, and normalized stepsize for Network 2 with  $T_i = 6$ ,  $X_j = 4$ ,  $QoS = 0.3$ ;  $P_{av}$  form of QoS constraint used for first 100 iterations,  $P_{max}$  thereafter; no projection used

## 11. THROUGHPUT GAIN FROM NETWORK OPTIMIZATION

Thus far, we have focused our attention on the convergence properties of our optimization algorithm. However, from the perspective of the network user/manager, a more important issue is the degree of performance improvement that can be obtained when the offered loads are optimized. In particular, in this section we compare the performance obtained by using our algorithm to that obtained under “uniform loading,” under which the  $\lambda_j$ 's are all equal. In the latter case, the search for the optimal solution is a simple one-dimensional search.

We first consider examples based on our core runs, and show that the degree of improvement is rather modest for these cases. We then consider some examples for which considerable increase in throughput is, in fact, possible by means of our optimization techniques.

### 11.1 Core Runs

To assess the benefits achievable through optimal loading, we first consider Networks 1, 2, and 3 with six transceivers at each node ( $T_i = 6, i = 1, \dots, N$ ) and at most four calls on any circuit ( $X_j = 4, j = 1, \dots, J$ ). Figure 29 shows throughput in Network 1 as a function of  $Q_j$  (i.e., of the maximum permitted blocking probability on any individual circuit) for both uniform and optimal loading. The “optimal loading” curve represents the higher throughput obtained by our optimization algorithm, and demonstrates the degree of improvement that has been achieved by our multidimensional search. For  $Q_j = 0.001$ , the improvement achieved by optimizing the offered load is 14.3%. The improvement is greatest at about  $Q_j = 0.1$ , where it is 19.6%. At very large values of  $Q_j$  the improvement decreases significantly; it is 8.9% at  $Q_j = 0.9$  and 3.5% at  $Q_j = 0.99$ . Similarly, Figs. 30 and 31 show the performance achievable for Networks 2 and 3 as  $Q_j$  is increased to a maximum value of 0.6. Results for Network 2 show that the degree of improvement achievable by using the optimization technique is similar to that achieved for Network 1. However, little improvement is obtained for Network 3, apparently because the offered loads at the optimal point are more nearly symmetrical than they were for the other two network examples.

In view of the highly asymmetrical nature of the optimal offered load (particularly for Network 1), it is perhaps surprising that only “modest” increase in throughput was obtained by the optimization process. The inability to produce more substantial gains can be explained, at least in part, by the highly coupled nature of the circuits in the network. We can view this as a form of “balanced coupling,” in which a relatively large region of the offered-load space produces similar values of throughput.

The characteristics of these networks that produce the high degree of insensitivity to loading appear to include:

- interaction (i.e., contention for resources) among many circuits,
- equal QoS requirements for each circuit, and
- equal number of transceivers at each node.

We now explore some network examples that do not possess all of these characteristics, and demonstrate the capability of our optimization techniques to provide significant increase in throughput.

### 11.2 Networks 1, 2, and 3 with Different QoS Values

The results of Section 11.1 indicated that only modest increase in throughput is achievable for Networks 1, 2, and 3 for examples in which the number of transceivers is the same at all nodes and the QoS constraint is the same for all circuits. We now look at examples in which the blocking probability of half of the circuits must satisfy a QoS constraint of 0.001 and half must satisfy 0.3.<sup>45</sup> In all cases,  $T_i = 6$  and  $X_j = 4$ . Two examples were considered for each case, as summarized below:

- (a) Network 1:  $Q_1 = \dots = Q_5 = 0.001$ ;       $Q_6 = \dots = Q_{10} = 0.3$   
 (b) Network 1:  $Q_1 = \dots = Q_5 = 0.3$ ;       $Q_6 = \dots = Q_{10} = 0.001$

<sup>45</sup>The convergence properties of our algorithm for examples of this type were discussed in Section 9.3.

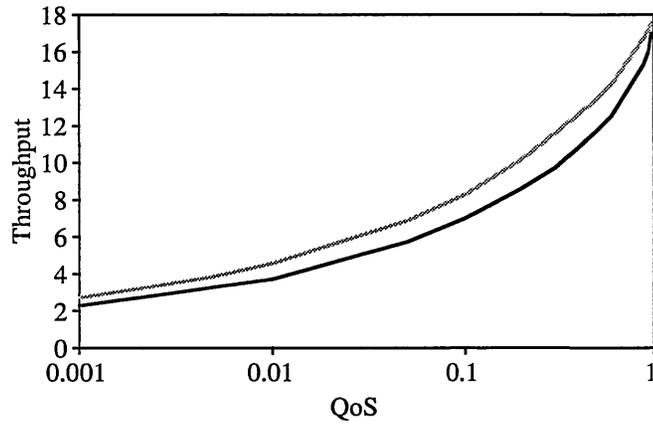


Fig. 29 — Throughput achievable in Network 1 using uniform (lower curve) and optimal loading (upper curve)

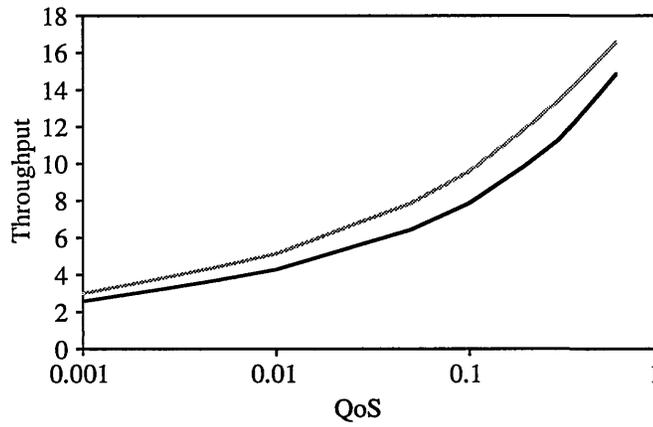


Fig. 30 — Throughput achievable in Network 2 using uniform (lower curve) and optimal loading (upper curve)

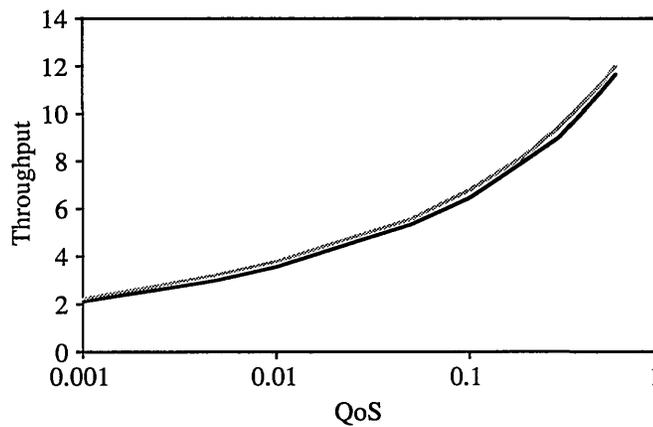


Fig. 31 — Throughput achievable in Network 3 using uniform (lower curve) and optimal loading (upper curve)

- (c) Network 2:  $Q_1 = \dots = Q_5 = 0.001$ ;  $Q_6 = \dots = Q_{10} = 0.3$   
 (d) Network 2:  $Q_1 = \dots = Q_5 = 0.3$ ;  $Q_6 = \dots = Q_{10} = 0.001$   
 (e) Network 3:  $Q_1 = \dots = Q_4 = 0.001$ ;  $Q_5 = \dots = Q_8 = 0.3$   
 (f) Network 3:  $Q_1 = \dots = Q_4 = 0.3$ ;  $Q_5 = \dots = Q_8 = 0.001$ .

Table 14 summarizes the results of these examples. The columns a through f refer to the six network examples listed above. The first row shows the maximum throughput that can be obtained when all of the  $\lambda_j$ 's are constrained to be equal. We also considered an alternative form of loading in which all of the circuits in the QoS = 0.001 group are constrained to have the same offered load value (call it  $\lambda_{0.001}$ ), and all those in the QoS = 0.3 group are constrained to have the same offered load value (call it  $\lambda_{0.3}$ ). The throughput associated with the best values of  $\lambda_{0.001}$  and  $\lambda_{0.3}$  that we found are shown in the second row. Little improvement over the case of uniform offered load is observed. Apparently, this is because even small increases of the offered load on some circuits in the QoS = 0.3 group (above the optimum uniform value) result in the violation of the QoS constraint on one or more circuits in the QoS = 0.001 group. The third row shows the throughput achieved using our optimization algorithm. The fourth row shows the improvement when the throughput under optimal loading is compared to that under uniform loading.<sup>46</sup> The degree of improvement ranges from modest to dramatic, and depends strongly on network characteristics and on which circuits are in the two QoS groups.

Table 14 — Throughput Improvement Achieved by Optimization of Offered Load in Network Examples with Different Values of the QoS Constraint  $Q_j$  ( $T_i = 6$ , and  $X_j = 4$ )

Network example	a	b	c	d	e	f
Throughput (all $\lambda_j$ 's equal)	2.34	2.34	2.63	2.63	2.17	2.17
Throughput (2 equal groups)	2.45	2.36	2.70	2.78	2.17	2.25
Throughput (optimal load)	5.22	5.68	5.59	7.63	2.35	5.61
Improvement	123%	142%	112%	190%	8%	158%

### 11.3 Unequal Number of Transceivers

We examined the degree of improvement that can be achieved in networks in which the number of transceivers is not the same at all nodes. In particular, we considered Network 1 with  $T_1 = T_2 = \dots = T_{12} = 3$  and  $T_{13} = T_{14} = \dots = T_{24} = 6$ . We considered an uncontrolled system, i.e., one with  $X_j = 6$ . Table 15 shows the throughput achieved using uniform and optimal loading for this example, for  $Q_j = 0.001, 0.3$ , and  $0.9$ . A significant degree of improvement is achieved, except for very high values of QoS.

### 11.4 The Multicross Network

We now consider a totally different network topology, which we first studied in Refs. 1 and 2, namely the "multicross network" (Fig. 32). This network has the property that the horizontal circuit  $c_0$  intersects each of the remaining  $N$  circuits,  $c_1, \dots, c_N$ . However, circuits  $c_1$  through  $c_N$  are mutually disjoint and share

<sup>46</sup>The results presented here and in the following two subsections are each based on a single run of 2000 iterations. Thus, further improvement over the throughput values shown for the optimal load may be possible. The values shown for the degree of improvement that is achieved through the optimization of offered load are therefore conservative.

Table 15 — Throughput Improvement Achieved by Optimization of Offered Load in Example with Non-uniform Number of Transceivers; Network 1 ( $T_1 = T_2 = \dots = T_{12} = 3$ ,  $T_{13} = T_{14} = \dots = T_{24} = 6$ , and  $X_j = 6$ )

	$Q_j = 0.001$	$Q_j = 0.3$	$Q_j = 0.9$
Throughput (all $\lambda_j$ 's equal)	0.37	3.82	7.88
Throughput (optimal load)	0.51	4.92	8.52
Improvement	37.8%	28.8%	8.1%

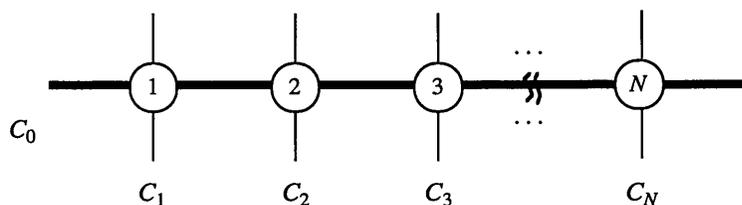


Fig. 32 — Example of a multicross network

resources only with circuit  $c_0$ . In Refs. 1 and 2 this class of networks was investigated from the perspective of optimal admission control. The goal was to determine the admission-control policy that minimizes the overall blocking probability for a specified offered load vector. It was shown that at high offered loads or at high values of  $N$  (the number of mutually disjoint circuits intersecting circuit  $c_0$ ), optimal admission control restricts the number of calls allowed on circuit  $c_0$ , ultimately not admitting any calls on this circuit. However, calls on the cross circuits ( $c_1, \dots, c_N$ ) are never blocked if resources are available.

In the present study we used our optimization algorithm to determine the optimal offered load for a multicross circuit with  $N = 5$  (i.e., there are five nodes, five cross circuits, and one horizontal circuit). The number of transceivers at each node is  $T_i = 6$ , and the circuit threshold is  $X_j = 6$ . Since  $X_j = T_i$ , this example corresponds to an uncontrolled mode of operation, in which calls are admitted as long as resources are available. Table 16 shows that significant increase in admissible throughput is achieved when the offered load is optimized. The improvement is greatest at low values of  $Q_j$ .

### 11.5 Discussion

We have demonstrated that significant and often dramatic performance improvement can be obtained by using the optimization algorithm developed in this study. The degree of improvement appears to be greatest in examples that have inherent asymmetry. Several forms of such asymmetry have been identified. For example, in the multicross network, the cross circuits do not share any resources with each other (although they do share resources with a common circuit). Similarly, examples in which either the QoS value or the number of transceivers is not uniform can benefit greatly from the optimization of offered load.

## 12. MISCELLANEOUS PERFORMANCE CONSIDERATIONS

Thus far, when assessing the performance improvement achieved by means of the optimization of offered load, our only performance index has been the total throughput (subject, of course, to the QoS constraints). In this section we take a more detailed look at performance by examining the offered load permitted to each individual circuit. In particular, we observe that the presence of “underloaded circuits” (defined in Section 9.1) can reduce overall throughput, even though they offer negligible load to the network. In

Table 16 — Throughput Improvement Achieved by Optimization of Offered Load in Multicross Network Example ( $N = 5$ ,  $T_i = 6$ , and  $X_j = 6$ )

	$Q_j = 0.001$	$Q_j = 0.3$	$Q_j = 0.9$
Throughput (all $\lambda_j$ 's equal)	2.55	10.69	21.50
Throughput (optimal load)	4.16	15.30	24.07
Improvement	63.1%	43.1%	12.0%

addition, we address the issue of fairness, which is implemented in two ways, namely by either guaranteeing each circuit at least a specified level of offered load or by limiting the maximum value of offered load permitted to any circuit.

### 12.1 The Impact of “Underutilized Circuits” on Performance

In several of our examples, the optimal offered-load vector contains one or more entries that are either zero or close to it. We refer to circuits whose offered load at the optimal point is less than 1% of the average offered load as “underloaded circuits.” For example, consider Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $\lambda_{\min} = 0$ , and  $Q_j = 0.3$ . The optimal throughput value for this network is 11.538. It is interesting to note that this value was achieved when three of the offered load values ( $\lambda_4$ ,  $\lambda_6$ , and  $\lambda_8$ ) were approximately equal to zero (see Table 7). To assess the impact of these underloaded circuits on overall network performance, the algorithm was rerun by setting  $\lambda_4 = \lambda_6 = \lambda_8 = 0$ , and ignoring the requirement that these underloaded circuits satisfy the QoS constraint.<sup>47</sup> The resulting optimal throughput value increased to 11.83. Thus, a modest (2.5%) increase in throughput was achieved by ignoring the requirement to satisfy the QoS constraint on circuits that make a negligible impact on the total network throughput. One way to view this situation is to say that the nonunderloaded circuits are being penalized by the need to satisfy the QoS constraint on underloaded circuits. They are forced to reduce their offered load to accommodate the QoS requirements of unlikely events (the occurrence of arrivals on underloaded circuits).

The issue of whether the QoS constraint should be guaranteed for all circuits, or for only those with significant offered loads, is a topic for future study. There are implications on pricing, e.g., whether the underloaded circuits should be required to pay more than nonunderloaded circuits to receive QoS guarantees. Here, the requirement that the QoS constraint be satisfied by the underloaded circuits has a greater deleterious impact on the other circuits in the network than does the blocking caused by the underloaded circuits (which is negligible).

### 12.2 Fairness Considerations: A Guaranteed Offered Load on Each Circuit

Another way to address the issue of underloaded circuits is to prevent their occurrence by guaranteeing a nonnegligible offered load  $\lambda_{\min}$  to all circuits. The goal here is to provide some degree of fairness to all users. Of course, to maintain satisfaction of the QoS constraint, some of the offered load values will have to be decreased, resulting in decreased overall throughput. We again consider the same network example of Network 1 with  $T_i = 6$ ,  $X_j = 4$ ,  $\lambda_{\min} = 0$ , and  $Q_j = 0.001$  and 0.3 (and return to our original requirement that the QoS constraint must be satisfied on all circuits, including the underloaded ones). Figure 33 shows that the optimal admissible throughput decreases when  $\lambda_{\min}$  increases, as is certainly expected.

<sup>47</sup>The blocking probability of a circuit can be high (relative to the specified QoS constraint value) even when the offered load to it is zero. The blocking probability for circuit  $j$  is the probability that the network is in a state in which a call to circuit  $j$  would be blocked, should it arrive.

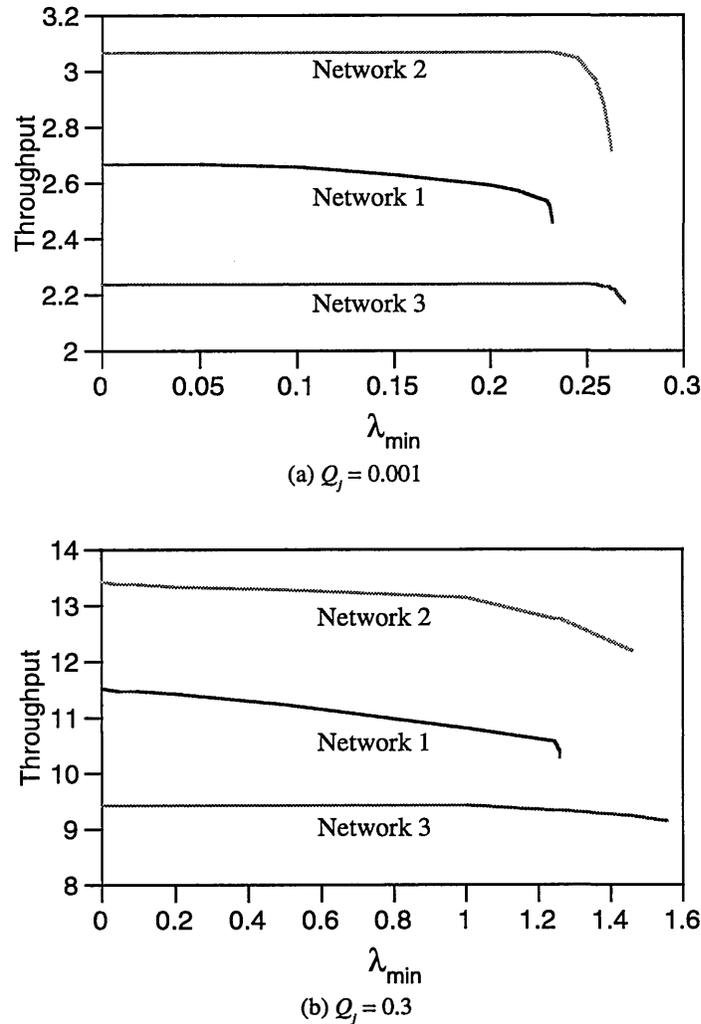


Fig. 33 — Maximum throughput achievable on Networks 1, 2, and 3 when each circuit is guaranteed an offered load of at least  $\lambda_{\min}$

For example, consider first the case of  $Q_j = 0.001$ . Beginning at about  $\lambda_{\min} = 0.1$ , Network 1 at first experiences a gradual perceptible decrease in admissible throughput as  $\lambda_{\min}$  is increased. However, at  $\lambda_{\min} = 0.23$  the decrease in throughput becomes precipitous, and the largest value of  $\lambda_{\min}$  for which an admissible solution can be found is about 0.234; the curve stops at this point. This value of  $\lambda_{\min}$  is equal to the value obtained by assuming that all of the  $\lambda_j$  are equal, and solving for the largest value of  $\lambda_j$  that provides an admissible solution. Network 2 is not perceptibly affected by the  $\lambda_{\min}$  constraint until  $\lambda_{\min} = 0.23$ ; similarly, Network 3 is not affected until  $\lambda_{\min} = 0.25$ . The relative insensitivity of Networks 2 and 3 to  $\lambda_{\min}$  is a consequence of the more nearly symmetrical nature of these networks, as is evidenced by the more nearly equal values of the  $\lambda_j$ 's at the optimal point.

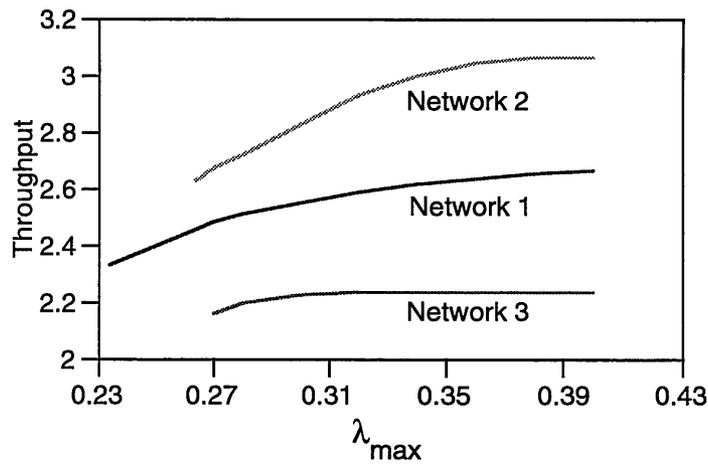
Results are qualitatively similar when  $Q_j = 0.3$ , although the effects of  $\lambda_{\min}$  are felt at values of  $\lambda_{\min}$  that are a smaller fraction of the maximum value it can take on. For example, when each of the circuits is guaranteed an offered load of at least  $\lambda_{\min} = 0.5$ , the new optimal throughput value decreases, from its previous optimal value of 11.538 (when  $\lambda_{\min} = 0$ ) to 11.24.<sup>48</sup> Whether or not the decrease in overall throughput, for the sake of providing each of the users a guaranteed level of offered load, is an acceptable tradeoff is, of course, the decision of the network designer/manager.

<sup>48</sup>The largest value of  $\lambda_{\min}$  for which the QoS constraint can be satisfied is 1.268.

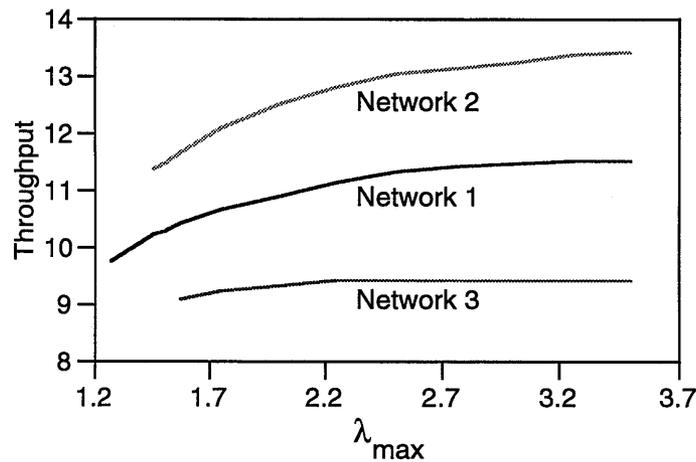
### 12.3 An Upper-Bounded Offered Load on Each Circuit

Alternatively, fairness can be implemented by imposing an upper limit  $\lambda_{\max}$  on the offered load permitted to each circuit. By doing so, individual circuits are prevented from grabbing an excessive amount of system resources. Figure 34 shows the effect on throughput of imposing this constraint for the same examples considered in Section 12.2. When  $\lambda_{\max}$  is sufficiently large, its imposition has no impact on throughput.

When the value of  $\lambda_{\max}$  is lowered, the overall throughput is decreased. However, the impact on fairness is more difficult to assess, because individual circuits are not guaranteed a specified level of offered load. Figure 35 shows the smallest value of the  $\lambda_j$ 's that results from the imposition of the  $\lambda_{\max}$  constraint for  $Q_j = 0.001$  and 0.3. When  $\lambda_{\max}$  is large, the smallest value of  $\lambda_j$  is the same as that for the case in which no bounds are placed on any of the offered loads. As  $\lambda_{\max}$  decreases, the smallest value of  $\lambda_j$  increases, thus providing a more-equitable distribution of offered load to the circuits. However, because use of the  $\lambda_{\max}$  constraint has only an indirect effect on the offered loads of the "deprived" circuits, it is felt that use of the  $\lambda_{\min}$  criterion of Section 12.2 is a better choice.



(a)  $Q_j = 0.001$



(b)  $Q_j = 0.3$

Fig. 34 — Maximum throughput achievable on Networks 1, 2, and 3 when each circuit is permitted at most  $\lambda_{\max}$

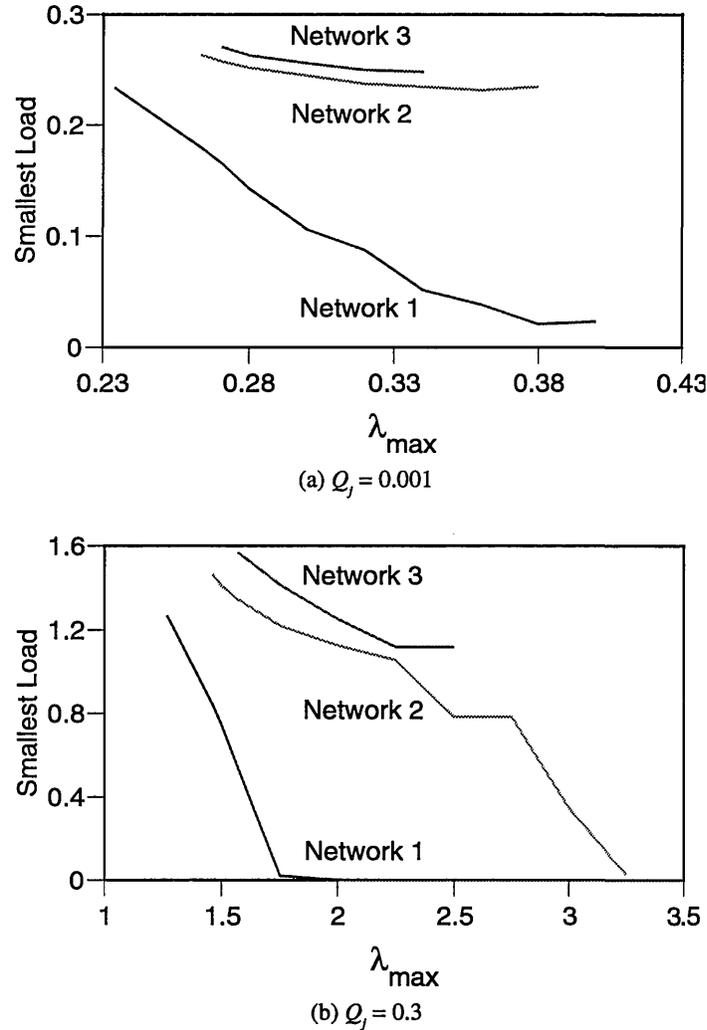


Fig. 35 — Smallest offered load on Networks 1, 2, and 3 when each circuit is permitted at most  $\lambda_{\max}$

### 13. RELATIONSHIPS BETWEEN ADMISSION CONTROL AND OPTIMIZATION OF OFFERED LOAD

We pointed out in Section 1 that network performance depends strongly on the highly interdependent issues of admission control, routing, offered load, and channel access. We have focused on the optimization of offered load under fixed admission control and routing schemes. In this section we address the impact of admission control on achievable throughput as well as on the performance of our optimization algorithm.

Most of our examples have addressed the case of  $T_i = 6$  (i.e., six transceivers at each node) and  $X_j = 4$  (i.e., at most four active calls permitted on any circuit). Had we set  $X_j = T_i = 6$ , the resulting system would be an “uncontrolled” network, in which all calls would be accepted as long as network resources (i.e., transceivers) are available to service them; this form of admission control is often referred to as complete sharing (see Section 2.1). As  $X_j$  is lowered further, the admission-control policy approaches that of complete partitioning.

We first address the impact of the imposition of admission control (i.e., setting  $X_j \leq T_i$ ) on achievable throughput. Table 17 shows the benchmark throughput values for Network 1 with  $T_i = 6$  for  $X_j = 3, 4,$  and  $6$ . Results for  $Q_j = 0.001$  and  $0.3$  are shown. For  $Q_j = 0.3$ , there is little difference in benchmark throughput as  $X_j$  is varied from 3 to 6 (an increase of only 2.4% as it is increased from 3 to 4, and only 0.66% as it is

Table 17 — Benchmark Throughput Values for Network 1 with  $T_i = 6$

	$Q_j = 0.001$	$Q_j = 0.3$
$X_j = 3$	1.8391	11.2686
$X_j = 4$	2.6645	11.5380
$X_j = 6$	2.9095	11.6147

increased from 4 to 6). However, the increase is much more substantial for  $Q_j = 0.001$  (where the increases are 44.9% and 9.2%, respectively).

Next, we address the convergence properties of the 18 versions of our optimization algorithm. A milestone table for the case of  $X_j = 6$  and  $Q_j = 0.001$  is shown in Table 18, and an offered-load table is shown in Table 19. The most striking feature of the milestone table is that there is a great deal more variation in performance than for the case of  $X_j = 4$ . For example, only seven versions of the algorithm reached the 99.9% milestone, in contrast to all 18 when  $X_j = 4$ . In fact, three versions failed to reach even the 98% milestone. It is apparent from Table 18 that Stepsize Rule 4 is too aggressive for this example. The differences in achieved throughput among the other stepsize rules are not as dramatic.

Tables 20 and 21 provide results for the same network example, but with  $Q_j = 0.3$ . Again, there is a great deal more variation in performance than for the cases of  $X_j = 4$  and  $X_j = 3$ ; e.g., only Version 2.4 reached the 99.9% milestone, and only eight versions reached the 99% milestone. In fact, the throughput found by only seven of the 18 versions was as high as the benchmark throughput level for the case of  $X_j = 4$ . By contrast, for the same problem with  $X_j = 4$  (see Table 4) 13 versions of the algorithm reached the 99.9% milestone. For  $X_j = 3$ , all 18 versions reached the 99.9% milestone.

A comparison of the performance of the algorithm for different values of  $X_j$  raises the question of why the uncontrolled system typically seems to offer more resistance to our algorithm than a controlled one.<sup>49</sup> Some insight into this question is provided by Table 19, which shows the offered loads and normalized blocking probabilities for the versions of the algorithm that provide the highest and lowest throughput for Network 1 with  $X_j = T_i = 6$  and  $Q_j = 0.001$ . These results should be compared with those shown in Table 7 for the case of  $X_j = 4$  and  $T_i = 6$ . Perhaps the most striking observation from Table 19 is that the optimal set of offered loads is much more asymmetrical than in the case of  $X_j = 4$ . For example, two of the offered-load values in the best solution are more than twice the value of the maximum offered load for the best solution when  $X_j = 4$ . The presence of these high values of offered load naturally forces some of the other offered loads to decrease significantly. In fact, the best solution for the uncontrolled system includes four underloaded circuits, whereas the minimum offered-load value in the best solution for  $X_j = 4$  is 0.0123 and all other loads are much higher (see Table 7).

Apparently, the use of higher values of  $X_j$  results in a larger Erlang capacity region (see Section 3), and hence the possibility of more-extreme values for the offered loads. Note that two of the normalized blocking probabilities in the best solution are significantly lower than any of those observed for  $X_j = 4$ . Thus the solution is also more extreme by this criterion as well. To perform a reliable search of this larger region, the use of conservative stepsize rules appears to be appropriate.

<sup>49</sup>Similar results were observed for the case of Network 1 with  $X_j = T_i = 4$  for QoS = 0.3; e.g., only two versions reached the 99.9% milestone, four reached the 99.5% milestone, eight reached the 99% milestone, and all 18 reached at least 98.6% of the benchmark throughput. For QoS = 0.001, eight reached the 99% milestone, 12 reached the 98% milestone, and all provided at least 96.75% of the benchmark throughput.

Table 18 — Milestone Table for Network 1 with  $T_i = 6$ ,  $X_j = 6$ ,  
and  $Q_j = 0.001$  (benchmark throughput: 2.9095)

algo	adm thruput at first exit	best adm thruput	last adm thruput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4 61.86%	990 99.99%	997 99.98%	450	521	638	672	820
1.1	8 76.13%	988 100.00%	1000 99.99%	380	499	595	654	739
1.2	15 89.84%	980 100.00%	1000 99.98%	148	233	614	694	783
2.0	4 61.86%	995 99.89%	999 99.88%	165	375	565	631	
2.1	8 76.13%	974 99.99%	1000 99.98%	250	408	509	543	752
2.2	15 89.84%	966 99.97%	1000 99.96%	292	418	541	605	876
2.3	9 80.97%	957 99.69%	999 99.68%	210	459	519	738	
2.4	15 89.84%	997 99.99%	1000 99.98%	174	393	484	563	815
3.0	4 61.71%	991 99.96%	1000 99.96%	279	347	407	473	793
3.1	8 75.68%	991 99.64%	999 99.64%	219	357	412	470	
3.2	15 88.49%	945 99.85%	998 99.84%	109	300	394	440	
3.3	9 80.38%	999 99.11%	999 99.11%	224	345	580		
3.4	15 88.49%	995 99.81%	1000 99.80%	109	300	367	538	
4.0	4 61.86%	995 97.54%	999 97.53%	148				
4.1	8 76.13%	982 97.50%	999 97.49%	165				
4.2	15 89.84%	978 98.03%	1000 98.02%	144	729			
4.3	9 80.97%	936 97.32%	999 97.32%	173				
4.4	15 89.84%	995 98.15%	1000 98.14%	127	570			

Table 19 — Offered-Load Table for Network 1 with  $T_i = 6$ ,  $X_j = 6$ ,  
and  $Q_j = 0.001$  (benchmark throughput: 2.9095)

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	S
best	0.7484 0.5160	0.9218 0.5800	0.0001 1.0000	0.1269 1.0000	0.1269 1.0000	0.0001 1.0000	0.8835 0.9500	0.0000 1.0000	0.0001 0.9990	0.1038 0.9990	2.9095
worst	0.5227 0.7650	0.5955 0.7430	0.2602 0.9980	0.1088 0.8610	0.3148 0.7850	0.0001 1.0000	0.6645 1.0000	0.0001 1.0000	0.2458 0.9180	0.1214 0.8570	2.8315 97.32%

Table 20 — Milestone Table for Network 1 with  $T_i = 6$ ,  $X_j = 6$ , and  $Q_j = 0.3$   
(benchmark throughput: 11.6147)

algo	adm thruput at first exit	best adm thruput	last adm thruput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4 83.77%	985 99.15%	1000 99.15%	217	562	677		
1.1	6 84.18%	971 98.09%	998 98.09%	414	729			
1.2	57 96.36%	969 99.81%	1000 99.80%	51	510	564	696	
2.0	4 83.77%	979 98.92%	1000 98.91%	110	409			
2.1	6 84.18%	983 98.34%	997 98.34%	105	463			
2.2	57 96.36%	974 99.47%	1000 99.46%	51	429	570		
2.3	7 86.76%	998 98.51%	1000 98.50%	48	463			
2.4	57 96.36%	987 100.00%	999 100.00%	51	145	501	532	719
3.0	4 83.70%	982 98.72%	999 98.72%	45	354			
3.1	6 84.04%	997 98.37%	1000 98.37%	66	392			
3.2	61 96.27%	981 99.81%	1000 99.80%	54	294	376	446	
3.3	7 86.59%	996 98.35%	1000 98.34%	34	382			
3.4	61 96.27%	978 99.78%	1000 99.78%	54	294	358	488	
4.0	4 83.77%	999 98.18%	999 98.18%	105	348			
4.1	6 84.18%	985 98.05%	999 98.04%	125	517			
4.2	57 96.36%	998 99.06%	1000 99.06%	51	203	658		
4.3	7 86.76%	980 99.09%	998 98.09%	48	361			
4.4	57 96.36%	986 99.47%	1000 99.46%	51	140	343		

Table 21 — Offered-Load Table for Network 1 with  $T_i = 6$ ,  $X_j = 6$ ,  
and  $Q_j = 0.3$  (benchmark throughput: 11.6147)

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	$S$
best	3.5162 0.6788	4.1701 0.7253	0.4680 0.9993	0.6934 0.9999	0.8057 0.8812	0.0008 0.9998	4.9307 0.9999	0.0004 0.9999	0.3725 0.9068	0.6030 0.9993	11.6147
worst	1.9072 0.8124	1.9925 0.7604	2.2631 0.9998	0.8009 0.8599	2.2658 0.7872	0.0002 1.0000	3.1314 1.0000	0.0000 0.9999	2.2627 0.9996	0.9671 0.8450	11.3879 98.05%

Another complicating factor is the irregular nature of the QoS constraint contours (since the blocking probability is a complicated function of all the offered load values). It appears that, for some network examples, the admissible throughput may be a multimodal function of the offered load values; thus, the solution may be trapped in one of several local maxima that arise because of the QoS constraints. However, the depth of these local maxima (if they do, indeed, exist) does not appear to be great because little difference is typically observed among the various versions of the optimization algorithm. Another possible explanation for the inability of some versions to converge to the optimal solution is that for some problem instances<sup>50</sup> our algorithm may not be robust with respect to the choice of system parameters. This could possibly result in failure to converge to the optimal point even when the throughput is a unimodal function of the offered loads. Because of the limited number of iterations (1000 in most of our examples), as well as the large number of dimensions (8 or 10 offered load values), it is not surprising that the true optimal point is not always reached.

### 13.1 Discussion

We have observed that, for large values of the QoS constraint parameter, there is little difference in throughput between uncontrolled systems (i.e., systems for which calls are admitted as long as resources are available) and only “slightly” controlled systems (i.e., systems in which  $X_j$  is only slightly smaller than  $T_j$ ). At lower QoS values, by contrast, the degree of improvement in throughput achievable by using an uncontrolled system can be significant.

We have also observed that our algorithm tends to be less robust for uncontrolled systems than for controlled systems, in the sense that the choice of stepsize rule and projection rule are more critical in such cases. Apparently, the reason for this reduced robustness is the more-extreme nature of the optimal solution, which results in a larger Erlang capacity region, and hence in a larger space to search. Thus, the optimal solution can take on more-extreme values, and it is more likely to include blocking probabilities that are substantially smaller than the maximum permitted value (i.e., the QoS constraint value).

Despite the added difficulties associated with uncontrolled systems, the algorithm continues to perform well. However, some guidelines emerge from the observations made in this section. Perhaps our most important recommendation, based primarily on the case of  $Q_j = 0.001$ , is that the use of relatively conservative stepsize rules is to be preferred for uncontrolled systems. The particular projection rule makes relatively little difference for this QoS value; Version 3.3 is the only version among the first three stepsize rules that failed to reach the 99.5% milestone. For  $Q_j = 0.3$ , we observed that all of the X.2 and X.4 versions reached at least the 99% milestone,<sup>51</sup> indicating a high level of robustness for these versions. Taking into consideration the results presented in this section for the two QoS values, it appears that use of a conservative stepsize rule (any of the first three would be acceptable, although Rule 1 may be overly conservative), coupled with Projection Rules 2 or 4, would be appropriate for most applications.

Although in the  $X_j = 6, T_i = 6$  case it has not been possible to achieve the uniformly high level of performance achieved for the  $X_j = 4, T_i = 6$  case, we consider our results to be quite acceptable, considering the difficulty of the problem. In both cases, a search of limited duration (in our case, a self-imposed limitation of 1000 iterations) must be performed over 10 highly dependent variables, subject to 10 nonlinear constraints. When  $X_j$  is increased from 4 to 6, the admissible search space (the Erlang capacity region) in the offered-load domain increases significantly even though the admissible state space (in the domain of active calls) increases by only 6.7% (from 284,115 to 303,248). We believe that it should be possible to develop stepsize rules (possibly requiring a larger number of iterations) that, when used with Projection Rules 2 and/or 4 (or similar), would result in more-reliable convergence to possibly higher values of admissible throughput. However, it is questionable as to whether the additional effort required would be worth the expected small increase in throughput.

<sup>50</sup>Such as those in which not all blocking probabilities are close to the QoS value.

<sup>51</sup>This was also true for the case of  $X_j = T_i = 4$  for QoS = 0.3.

We observed above that the solutions for  $X_j = T_i = 6$  (see Tables 19 and 21) include some values of offered load that are considerably higher than those for the  $X_j = 4, T_i = 6$  case (see Table 7). A consequence of the higher offered load values is a decrease in the level of fairness, which was discussed in Section 12. In particular, in Section 12.3 we discussed the impact of placing an upper bound on the offered-load values. The effect of reducing  $X_j$  from 6 to 4 appears to be similar to that of setting a maximum permitted value on the offered loads.

## 14. SUMMARY AND CONCLUSIONS

In this study, we have investigated the optimization of offered load in multihop, circuit-switched networks. The goal here has been to provide the maximum possible throughput, subject to constraints on blocking probability. Although the examples in this report are based on wireless networks, our techniques are equally applicable to “wired” networks. In addition to providing improved performance, the techniques presented in this report are useful for “sizing” a network’s capability, and thereby providing a measure of “network capacity.” Our techniques provide a valuable new network evaluation tool because it is generally difficult to estimate the traffic loads (or the resulting throughput) that a network can support.

The problem we have investigated is characterized by a nonlinear objective function and nonlinear constraints. In most of our examples we have sought the optimization of offered load on eight or 10 circuits, subject to the same number of constraints on circuit blocking probabilities. Thus, we have addressed a difficult multidimensional optimization problem. In problems such as these, the available mathematical theory provides the basic principles for solution, but no guarantee of convergence to the optimal point.

In product-form circuit-switched networks, it appears that throughput is a reasonably well-behaved function of offered load, and is thus amenable to Lagrangian optimization techniques. The circuit blocking probability, however, (in examples where the QoS constraint is based on the maximum blocking probability among the individual circuits in the network, which is the case considered in most of this report) is characterized by irregularly shaped contours of the admissible region.<sup>52</sup> Thus, the Lagrangian methods require modifications that permit them to bypass sharp edges during the search.<sup>53</sup> It is well known that constrained optimization problems can be challenging, and that the convergence of iterative algorithms to an optimal solution in such problems can be slow unless the algorithm’s parameters (most importantly, stepsize) are chosen carefully. As in many studies of optimization problems, we have observed that a relatively large stepsize is useful at the early stages of the search so that sufficient progress is made toward the optimal solution, and that damping is useful to reduce the incidence of oscillations as the optimal solution is approached. Furthermore, we have developed a heuristic “projection” technique that guides the search more directly toward the optimal solution, thereby resulting in faster and more-reliable convergence. This technique is based on gradient information that is available at each step of the search.

A high degree of robustness with respect to parameter values has been observed, in the sense that little experimentation with parameter values was needed for network examples with widely differing characteristics. Good results were obtained when using the same parameter values in a variety of applications. The algorithm incorporates a mechanism to choose the initial stepsize and to adjust the relative weights of the throughput-gradient term and the constraint-violation terms, thus permitting adaptability at the time of execution. It would be useful to develop a mechanism to adaptively choose the stepsize damping rule and projection rule (perhaps based on artificial-intelligence- or neural-network-based techniques) with the goal of speeding up convergence while further increasing the reliability of achieving the optimal solution, but this is beyond the scope of the present effort.

<sup>52</sup>These irregularly shaped contours result from the dependence of the blocking probability of any individual circuit on several (possibly all) of the offered loads, and the fact that the identity of the circuit with the largest offered load changes as the offered load varies.

<sup>53</sup>When the QoS constraint is based on the average blocking probability, blocking probability is well behaved, and such difficulties are not encountered.

We have observed that throughput is generally a “flat” function of the offered loads near the optimal point, in the sense that the region of offered-load vectors that produce nearly optimal throughput (e.g., 95%, 98%, or even 99% of the benchmark throughput value) can be large. Thus, offered-load vectors that differ greatly from each other can often provide nearly identical overall throughput. Therefore, nearly optimal performance can be obtained even when the offered loads are not close to their optimal values.

The problem becomes considerably simpler when an alternative form of the QoS constraint is used, in which the average blocking probability  $P_{av}$  (rather than the largest individual circuit blocking probability  $P_{max}$ ) must not exceed a specified value. We have shown that convergence to the optimal solution is rapid and robust under the  $P_{av}$  form of the QoS constraint. Furthermore, in some cases, the solution based on the  $P_{av}$  form of the QoS constraint is similar to that based on the  $P_{max}$  form of the constraint.<sup>54</sup> In such cases, the  $P_{av}$  solution can provide a good initial condition for the original problem based on the  $P_{max}$  form of the QoS constraint. The choice of the form of the QoS constraint (i.e.,  $P_{max}$  vs  $P_{av}$ ) to be used in practical applications would be made by the network manager.

The significance of the problem we have studied and the value of our solution method lie in the need to assess the network’s ability to provide high throughput under strict QoS constraints. A quick overall estimation of the network’s capability is usually referred to as “sizing,” and is of crucial importance for trading off communication volume against quality. It provides a useful benchmark for making further studies into questions of fairness with respect to the different customer classes (i.e., the different circuits) and into questions of “pricing” the offered services. In commercial networks it is important for the service provider to know the maximum achievable network throughput subject to QoS constraints in order to “price” the service to the customer classes sensibly and profitably. In military networks the same is true, except that the notion of pricing should be interpreted as a tool for prioritizing service in relation to the needs of the application and to the general operations doctrine.

Our method permits the network designer or network operator to quickly make a fairly accurate sizing estimate. As the stopping rule tables in Section 8.3 and Appendixes A and B indicate, it is only necessary to set a predetermined acceptable value to the maximum number of iterations (which follows from the available computing power and the available time limitations) and to what constitutes satisfactory indication of convergence (the value of  $\delta$  in Section 8.3). Thus the algorithm is armed with a flexible stopping criterion, and permits routine experimentation with different QoS values.

In conclusion, we consider this work to have led to the development and evaluation of a useful tool for addressing some subtle aspects of overall network design.

## ACKNOWLEDGMENTS

Mr. Craig M. Barnhart (a former NRL employee, now with TRW Data Technologies Division, Aurora, CO) participated in the early stages of this study, and contributed significantly to the development of the preliminary form of the iterative algorithm described in this report.

This work was supported in part by a grant of HPC time from the DoD HPC Shared Resource Center, Naval Research Laboratory, Washington, D.C., on their Thinking Machines Corp. CM-5E Connection Machine.

## REFERENCES

1. C.M. Barnhart, J.E. Wieselthier, and A. Ephremides, “Admission-Control Policies for Multihop Wireless Networks,” *Wireless Net.* 1(4) 373-387 (1995).
2. C.M. Barnhart, J.E. Wieselthier, and A. Ephremides, “Admission Control in Integrated Voice/Data Multihop Radio Networks,” NRL/MR/5521--93-7196, Naval Research Laboratory, January 1993.

<sup>54</sup>We have identified a property that seems to indicate when this is true, without having to solve the  $P_{max}$  form of the problem.

3. C.M. Barnhart, J.E. Wieselthier, and A. Ephremides, "Constrained Optimization Methods for Admission Control and Offered Load in Communication Networks," *Proceedings of the Thirtieth Annual Conference on Information Sciences and Systems (CISS)*, Princeton University, Princeton, NJ, March 1996, pp. 686-691.
4. J.E. Wieselthier, G.D. Nguyen, C.M. Barnhart, and A. Ephremides, "A Problem of Constrained Optimization for Bandwidth Allocation in High-Speed and Wireless Communication Networks," *Proceedings of the 35th IEEE Conference on Decision and Control*, Kobe, Japan, December 1996, pp. 1347-1348.
5. G.D. Nguyen, J.E. Wieselthier, and A. Ephremides, "Lagrangian Techniques for Optimizing Throughput in Wireless Communication Networks Subject to QoS Constraints," *Proceedings of the 1997 Conference on Information Sciences and Systems (CISS)*, Johns Hopkins University, Baltimore, MD, March 1997, pp. 405-410.
6. J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, "Algorithms for Finding Optimal Offered Load in Wireless Communication Networks," *Conference Record of IEEE MILCOM'97*, Monterey, CA, November 1997.
7. S. Jordan and P. Varaiya, "Control of Multiple Service, Multiple Resource Communication Networks," *IEEE Trans. Commun.* **42**(11), 2979-2988 (1994).
8. J.M. Aein, "A Multi-User-Class, Blocked-Calls-Cleared, Demand Access Model," *IEEE Trans. Commun.* **COM-26**(3), 378-385 (1978).
9. S. Jordan and P. Varaiya, "Throughput in Multiple Service, Multiple Resource Communication Networks," *IEEE Trans. Commun.* **39**(8), 1216-1222 (1991).
10. D.Y. Burman, J.P. Lehoczky, and Y. Lim, "Insensitivity of Blocking Probabilities in a Circuit-Switching Network," *J. Appl. Prob.* **21**, 850-859 (1984).
11. J.E. Wieselthier, C.M. Barnhart, and A. Ephremides, "Data-Delay Evaluation in Integrated Wireless Networks Based on Local Product-Form Solutions for Voice Occupancy," *Wireless Net.* **2**(4), 297-314 (1996).
12. J.E. Wieselthier, C.M. Barnhart, and A. Ephremides, "A Mini-Product-Form-Based Solution to Data-Delay Evaluation in Wireless Integrated Voice/Data Networks," *Proceedings of IEEE INFOCOM'95*, Boston, MA, April 1995, pp. 1044-1052.
13. J.E. Wieselthier, C.M. Barnhart, and A. Ephremides, "Novel Techniques for the Analysis of Wireless Integrated Voice/Data Networks," NRL Memorandum Report NRL/MR/5521--95-7744, Naval Research Laboratory, July 1995.
14. J.E. Wieselthier, C.M. Barnhart, and A. Ephremides, "Standard Clock Simulation and Ordinal Optimization Applied to Admission Control in Integrated Communication Networks," *J. Discrete Event Dyn. Sys.: Theory Appl.* **5**, 243-279 (1995).
15. C.G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis* (R.D. Irwin, Inc. and Aksen Associates, Inc., Homewood, IL 1993).
16. R.G. Gallager, *Discrete Stochastic Processes* (Kluwer Academic Publishers, Boston, MA, 1996).
17. G. Louth, M. Mitzenmacher, and F. Kelly, "Computational Complexity of Loss Networks," *J. Theor. Comp. Sci.* **125**, 45-59 (1994).
18. A.E. Conway and N.D. Georganas, *Queueing Networks—Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation* (The MIT Press, Cambridge, MA, 1989).

19. D. Mitra and J.A. Morrison, "Erlang Capacity and Uniform Approximations for Shared Unbuffered Resources," *IEEE/ACM Trans. Net.* **2**(6), 558-570 (1994).
20. J.E. Wieselthier, C.M. Barnhart, and A. Ephremides, "Ordinal Optimization of Admission Control in Wireless Multihop Integrated Networks via Standard Clock Simulation," NRL Report NRL/FR/5521--95-9781, Naval Research Laboratory, August 1995.
21. D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods* (Athena Scientific, Belmont, MA (originally published by Academic Press in 1982), 1996).
22. R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows* (Prentice Hall, Englewood Cliffs, NJ, 1993).

## **Appendix A**

### **TABLES FOR THE CORE RUNS**

Tables summarizing the behavior of several of the core runs were included in the body of this report. This appendix contains tables for the others. These tables provide a great deal of insight into the behavior of the various versions of the optimization algorithm. Three types of tables are shown.

“Milestone” tables (described in Section 8.2) indicate the number of iterations required by each version of the algorithm to reach specified percentages of the benchmark throughput value for each network example. Other useful information in the milestone tables includes the percentage of benchmark throughput obtained before exiting the admissible region for the first time and the percentage of the benchmark throughput obtained at the best point of each run.

“Offered-load” tables (described in Section 8.4) show the offered loads and normalized blocking probabilities for the best version of the algorithm (i.e., the version that provides the highest throughput) and the worst version (the version that provides the lowest throughput). These tables show the variation among the offered loads in the best and worst solutions, as well as the closeness with which the blocking probabilities on each circuit approach the maximum permitted QoS value. The rightmost column shows the throughput achieved by the best and worst versions of the algorithm, along with the percentage of benchmark throughput achieved by the worst solution.

“Stopping-rule” tables (described in Section 8.3) show the percentage of benchmark throughput that would be achieved under the 18 versions of the algorithm when the search is stopped based on the satisfaction of a convergence criterion, rather than after a fixed number (in our case 1000) of iterations.

Table A1 — Network 1 with  $T_i = 6$ ,  $X_j = 3$ , and  $Q_j = 0.001$  (benchmark throughput: 1.8391)

(a) Milestone Table

algo	adm throughput at first exit	best adm throughput	last adm throughput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4 78.25%	993 99.97%	1000 99.92%	549	657	705	765	906
1.1	4 70.24%	1000 99.81%	1000 99.81%	701	706	769	771	
1.2	8 80.80%	995 99.98%	998 99.94%	252	692	751	824	958
2.0	4 78.25%	989 100.00%	999 99.97%	434	536	688	746	880
2.1	4 70.24%	974 99.77%	1000 99.64%	462	698	852	924	
2.2	8 80.80%	1000 99.98%	1000 99.98%	431	443	510	785	930
2.3	7 77.45%	998 99.98%	1000 99.92%	423	577	686	721	888
2.4	8 80.80%	961 100.00%	1000 99.92%	383	445	625	702	926
3.0	4 78.07%	929 99.99%	1000 99.95%	365	393	438	476	775
3.1	4 70.08%	967 100.00%	1000 99.91%	375	401	476	517	760
3.2	8 80.02%	991 99.97%	999 99.95%	257	414	431	480	824
3.3	8 87.41%	958 99.98%	1000 99.92%	126	390	396	451	795
3.4	8 80.02%	903 99.97%	999 99.96%	257	385	385	422	750
4.0	4 78.25%	949 100.00%	1000 99.91%	177	390	402	629	795
4.1	4 70.24%	938 99.82%	1000 99.46%	285	287	729	830	
4.2	8 80.80%	1000 99.97%	1000 99.97%	181	208	373	546	825
4.3	7 77.45%	1000 99.98%	1000 99.98%	186	230	307	449	847
4.4	8 80.80%	1000 99.99%	1000 99.99%	147	194	365	646	896

(b) Offered-Load Table

	$\lambda_1$ $F_1$	$\lambda_2$ $F_2$	$\lambda_3$ $F_3$	$\lambda_4$ $F_4$	$\lambda_5$ $F_5$	$\lambda_6$ $F_6$	$\lambda_7$ $F_7$	$\lambda_8$ $F_8$	$\lambda_9$ $F_9$	$\lambda_{10}$ $F_{10}$	S
best	0.1797	0.1845	0.1841	0.1797	0.1883	0.1838	0.1930	0.1794	0.1841	0.1844	1.8391
	1.0000	0.9990	1.0000	1.0000	1.0000	0.9980	0.9990	0.9990	0.9980	0.9980	
worst	0.1799	0.1836	0.1837	0.1787	0.1882	0.1833	0.1917	0.1789	0.1843	0.1843	1.8349
	1.0000	0.9860	0.9940	0.9850	0.9990	0.9900	0.9810	0.9910	0.9990	0.9950	99.77%

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\bar{s} = 0.1$	0.01	0.001	0.1	0.01	0.001	0.01	0.001	0.001
1.0	300	300	300	479	600	600	666	1000	1000
	82.168	82.168	82.168	82.168	97.527	97.527	98.020	99.858	99.969
1.1	300	300	300	531	600	600	692	881	1000
	81.455	81.455	81.455	90.954	90.954	90.954	90.954	99.710	99.806
1.2	300	300	300	410	600	600	691	895	1000
	95.179	95.179	95.179	95.179	95.179	95.179	97.142	99.834	99.981
2.0	300	300	300	352	558	600	558	854	1000
	82.168	82.168	82.168	86.247	98.166	98.913	98.166	99.828	99.998
2.1	300	300	300	358	596	600	596	844	1000
	92.682	92.682	92.682	92.682	96.977	96.977	96.977	98.948	99.772
2.2	300	300	300	350	539	600	539	854	1000
	91.872	91.872	91.872	91.872	99.218	99.218	99.218	99.819	99.976
2.3	105	300	300	105	552	600	552	855	1000
	77.449	85.195	85.195	77.449	97.156	98.722	97.156	99.874	99.979
2.4	180	300	300	180	538	600	538	781	936
	94.134	94.134	94.134	94.134	98.599	98.851	98.599	99.690	99.903
3.0	300	300	300	321	445	600	445	747	1000
	83.993	83.993	83.993	84.634	99.014	99.844	99.014	99.869	99.994
3.1	300	300	300	336	439	600	439	690	1000
	89.369	89.369	89.369	89.369	98.700	99.740	98.700	99.842	100.000
3.2	108	300	300	108	409	600	409	716	1000
	93.420	96.078	96.078	93.420	97.891	99.792	97.891	99.808	99.969
3.3	24	300	300	24	436	600	436	720	923
	87.409	97.663	97.663	87.409	99.016	99.888	99.016	99.888	99.947
3.4	108	300	300	108	422	600	422	728	1000
	93.420	96.078	96.078	93.420	99.561	99.854	99.561	99.872	99.973
4.0	156	300	300	156	380	600	380	796	1000
	88.524	97.622	97.622	88.524	97.622	99.470	97.622	99.920	99.995
4.1	177	300	300	177	375	600	375	754	1000
	83.256	98.177	98.177	83.256	98.177	98.177	98.177	99.174	99.817
4.2	152	300	300	152	380	600	380	777	920
	91.872	98.689	98.689	91.872	99.101	99.658	99.101	99.873	99.924
4.3	135	300	300	135	332	600	332	771	893
	85.214	98.943	98.943	85.214	99.019	99.630	99.019	99.813	99.950
4.4	154	217	300	154	217	600	217	772	1000
	95.674	98.830	98.830	95.674	98.830	99.342	98.830	99.821	99.986

Table A2 — Network 1 with  $T_i = 6$ ,  $X_j = 3$ , and  $Q_j = 0.3$  (benchmark throughput: 11.2686)

(c) Stopping-Rule Table

algo	max it. = 300				max it. = 600				max it. = 1000			
	$\delta = 0.1$	0.01	0.001	0.0001	0.1	0.01	0.001	0.0001	0.01	0.001	0.0001	0.0001
1.0	15	300	300	300	15	535	600	535	741	933	0.0001	
	92.470	95.842	95.842	95.842	92.470	98.992	99.175	98.992	99.821	99.974		
1.1	9	300	300	300	9	530	600	530	700	907	0.0001	
	91.653	97.152	97.152	97.152	91.653	99.302	99.811	99.302	99.856	99.982		
1.2	11	152	300	300	11	152	600	152	685	938	0.0001	
	84.174	98.451	98.822	98.822	84.174	98.451	99.565	98.451	99.858	99.973		
2.0	15	300	300	300	15	362	600	362	613	910	0.0001	
	92.470	98.040	98.040	98.040	92.470	98.854	99.853	98.854	99.867	99.978		
2.1	9	300	300	300	9	366	600	366	603	856	0.0001	
	91.653	99.060	99.060	99.060	91.653	99.060	99.904	99.060	99.904	99.987		
2.2	11	300	300	300	11	421	600	421	648	918	0.0001	
	84.174	98.451	98.451	98.451	84.174	98.865	99.698	98.865	99.816	99.976		
2.3	10	300	300	300	10	381	600	381	588	865	0.0001	
	95.217	98.461	98.461	98.461	95.217	98.770	99.839	98.770	99.839	99.984		
2.4	11	138	300	300	11	138	600	138	654	907	0.0001	
	84.174	98.607	99.208	99.208	84.174	98.607	99.757	98.607	99.878	99.978		
3.0	15	300	300	300	15	356	600	356	463	653	0.0001	
	92.603	97.621	97.621	97.621	92.603	98.962	99.912	98.962	99.912	99.973		
3.1	9	300	300	300	9	337	600	337	448	640	0.0001	
	91.501	98.228	98.228	98.228	91.501	98.989	99.892	98.989	99.892	99.977		
3.2	11	101	300	300	11	101	489	101	489	825	0.0001	
	83.876	98.403	98.963	98.963	83.876	98.403	99.880	98.403	99.880	99.978		
3.3	10	300	300	300	10	323	450	323	450	714	0.0001	
	94.984	99.118	99.118	99.118	94.984	99.118	99.893	99.118	99.893	99.981		
3.4	11	101	300	300	11	101	400	101	400	494	0.0001	
	83.876	98.403	98.963	98.963	83.876	98.403	99.736	98.403	99.736	99.771		
4.0	15	178	300	300	15	178	452	178	452	858	0.0001	
	92.470	99.430	99.790	99.790	92.470	99.430	99.910	99.430	99.910	99.973		
4.1	9	155	265	265	9	155	265	155	265	723	0.0001	
	91.653	98.849	99.691	99.691	91.653	98.849	99.691	98.849	99.691	99.985		
4.2	11	168	300	300	11	168	473	168	473	869	0.0001	
	84.174	99.187	99.776	99.776	84.174	99.187	99.851	99.187	99.851	99.979		
4.3	10	156	300	300	10	156	457	156	457	851	0.0001	
	95.217	98.375	99.783	99.783	95.217	98.375	99.831	98.375	99.831	99.972		
4.4	11	153	200	200	11	153	200	153	200	850	0.0001	
	84.174	99.503	99.698	99.698	84.174	99.503	99.698	99.503	99.698	99.979		

(a) Milestone Table

algo	adm throughput at first exit	best adm throughput	last adm throughput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4	983	1000	113	492	596	651	805
	83.87%	99.99%	99.98%					
1.1	7	932	999	39	452	463	550	710
	91.65%	99.99%	99.99%					
1.2	20	999	999	20	76	557	595	791
	95.16%	99.98%	99.98%					
2.0	4	998	1000	102	188	386	475	653
	83.87%	99.98%	99.98%					
2.1	7	976	999	39	116	193	418	569
	91.65%	100.00%	99.99%					
2.2	20	952	1000	20	76	456	520	720
	95.16%	99.99%	99.98%					
2.3	10	992	998	10	101	382	410	676
	95.22%	99.99%	99.99%					
2.4	20	997	1000	20	76	290	550	697
	95.16%	99.99%	99.98%					
3.0	4	846	999	35	314	360	373	450
	83.79%	99.99%	99.98%					
3.1	7	988	1000	48	299	347	368	459
	91.50%	99.99%	99.98%					
3.2	21	937	999	21	39	348	422	512
	96.74%	99.99%	99.98%					
3.3	10	982	1000	19	141	171	364	469
	94.98%	99.99%	99.98%					
3.4	21	971	1000	21	39	314	331	559
	96.74%	99.98%	99.98%					
4.0	4	974	1000	109	139	163	202	445
	83.87%	99.99%	99.98%					
4.1	7	976	998	39	136	164	188	317
	91.65%	100.00%	100.00%					
4.2	20	988	999	20	76	154	197	493
	95.16%	99.99%	99.98%					
4.3	10	994	1000	10	138	160	188	565
	95.22%	99.99%	99.98%					
4.4	20	995	999	20	76	151	152	433
	95.16%	99.98%	99.97%					

(b) Offered-Load Table

	$\lambda_1$ $\hat{F}_1$	$\lambda_2$ $\hat{F}_2$	$\lambda_3$ $\hat{F}_3$	$\lambda_4$ $\hat{F}_4$	$\lambda_5$ $\hat{F}_5$	$\lambda_6$ $\hat{F}_6$	$\lambda_7$ $\hat{F}_7$	$\lambda_8$ $\hat{F}_8$	$\lambda_9$ $\hat{F}_9$	$\lambda_{10}$ $\hat{F}_{10}$	S
best	1.4945	1.7551	2.1608	1.4944	2.2920	0.4398	2.5648	0.0001	2.2691	1.4929	11.2686
	1.0000	0.9288	1.0000	1.0000	1.0000	0.9979	0.9999	0.9999	0.9997	0.8747	
worst	1.4945	1.6209	2.1597	1.4945	2.2925	0.4384	2.5642	0.0001	2.2683	1.6209	11.2667
	0.9996	0.8997	0.9997	0.9996	1.0000	0.9973	0.9996	0.9998	0.9998	0.8997	99.98%

Table A3 — Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ ; smaller stepsize (benchmark throughput: 2.6646)

(a) Milestone Table

algo	adm throughput at first exit	best adm throughput	last adm throughput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	8	975	1000	131	508	571	643	764
1.1	13	995	999	25	481	533	593	797
1.2	28	970	1000	38	506	535	624	741
2.0	8	984	997	131	448	505	591	787
2.1	13	890	998	25	329	475	544	757
2.2	28	970	999	38	443	510	609	754
2.3	17	982	998	35	447	502	560	704
2.4	28	965	997	38	406	540	654	807
3.0	8	975	991	62	177	382	382	472
3.1	14	899	998	28	235	372	400	456
3.2	29	981	1000	36	91	380	405	477
3.3	18	990	999	41	292	332	408	476
3.4	29	914	995	36	91	341	354	517
4.0	8	994	1000	126	178	178	224	327
4.1	13	988	999	25	139	170	240	445
4.2	28	953	1000	38	168	193	253	426
4.3	17	969	1000	35	170	179	179	545
4.4	28	836	1000	38	149	149	205	425

(b) Offered-Load Table

	$\lambda_1$ $\hat{F}_1$	$\lambda_2$ $\hat{F}_2$	$\lambda_3$ $\hat{F}_3$	$\lambda_4$ $\hat{F}_4$	$\lambda_5$ $\hat{F}_5$	$\lambda_6$ $\hat{F}_6$	$\lambda_7$ $\hat{F}_7$	$\lambda_8$ $\hat{F}_8$	$\lambda_9$ $\hat{F}_9$	$\lambda_{10}$ $\hat{F}_{10}$	S
best	0.2512	0.3230	0.3075	0.2321	0.3395	0.1603	0.3941	0.0229	0.3379	0.2987	2.6646
worst	1.0000	1.0000	1.0000	1.0000	1.0000	0.9910	1.0000	0.9270	1.0000	1.0000	
	0.2549	0.3236	0.3093	0.2353	0.3431	0.1478	0.3954	0.0213	0.3373	0.2988	2.6641
	0.9990	1.0000	1.0000	0.9990	0.9990	0.9660	0.9990	0.9300	1.0000	1.0000	99.98%

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.1	0.01	0.001
1.0	55	300	300	55	600	600	633	822	1000
1.1	15	300	300	15	581	600	581	809	988
1.2	88.125	97.013	97.013	88.125	99.211	99.535	99.211	99.947	99.997
2.0	56.996	97.738	97.738	56.996	97.738	99.410	97.738	99.943	99.988
2.1	55	300	300	55	592	600	592	837	1000
2.2	92.887	96.641	96.641	92.887	99.556	99.556	99.556	99.962	99.982
2.3	15	300	300	15	539	600	539	791	1000
2.4	80.387	97.699	97.699	80.387	99.496	99.799	99.496	99.958	99.993
3.0	54	300	300	54	372	571	372	571	997
3.1	15	300	300	15	380	498	380	498	908
3.2	56.247	98.237	98.434	56.247	99.454	99.958	99.454	99.958	99.997
3.3	15	300	300	15	389	510	389	510	1000
3.4	79.320	98.501	98.501	79.320	99.208	99.961	99.208	99.961	99.992
4.0	55	170	300	55	170	600	170	625	1000
4.1	15	201	300	15	201	530	201	530	981
4.2	88.125	99.397	99.734	88.125	99.397	99.930	99.397	99.930	99.998
4.3	15	207	300	15	207	600	207	623	962
4.4	56.996	99.441	99.869	56.996	99.441	99.940	99.441	99.940	99.998

Table A4 — Network 2 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$  (benchmark throughput: 3.0700)

(a) Milestone Table

algo	adm thrupt at first exit	best adm thrupt	last adm thrupt	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4	981	1000	533	620	679	755	873
1.1	6	980	99.96%	486	520	630	681	839
1.2	12	988	99.99%	19	578	588	747	863
2.0	4	982	99.98%	327	449	483	582	765
2.1	6	955	99.97%	296	431	431	615	778
2.2	12	980	99.99%	19	491	552	650	816
2.3	11	929	1000	109	516	568	671	864
2.4	12	943	99.98%	19	449	521	521	793
3.0	4	864	1000	323	361	430	430	571
3.1	6	972	99.97%	343	363	374	431	545
3.2	12	819	99.98%	53	369	402	414	522
3.3	11	995	99.97%	200	360	423	447	634
3.4	12	955	99.94%	53	334	372	399	551
4.0	4	996	1000	163	199	277	476	757
4.1	6	976	99.92%	142	190	223	223	567
4.2	12	1000	99.99%	19	178	256	256	709
4.3	11	985	99.99%	111	198	251	391	724
4.4	12	975	99.96%	19	192	288	499	696

(b) Offered-Load Table

	$\lambda_1$ $F_1$	$\lambda_2$ $F_2$	$\lambda_3$ $F_3$	$\lambda_4$ $F_4$	$\lambda_5$ $F_5$	$\lambda_6$ $F_6$	$\lambda_7$ $F_7$	$\lambda_8$ $F_8$	$\lambda_9$ $F_9$	$\lambda_{10}$ $F_{10}$	S
best	0.2326	0.3724	0.3129	0.3571	0.2460	0.2819	0.2941	0.3546	0.3856	0.2357	3.0700
worst	1.0000	1.0000	0.9990	0.9990	1.0000	1.0000	0.9990	0.9990	1.0000	1.0000	
	0.2323	0.3723	0.3129	0.3570	0.2458	0.2818	0.2942	0.3548	0.3856	0.2355	3.0693
	0.9970	0.9990	0.9990	0.9980	0.9970	0.9970	0.9990	1.0000	1.0000	0.9970	99.98%

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.1	0.01	0.001
1.0	300	300	300	497	600	600	679	880	1000
1.1	300	300	300	450	600	600	618	848	1000
1.2	89.780	89.780	89.780	94.744	98.090	98.090	98.575	99.948	100.00
2.0	13	300	300	13	600	600	648	874	1000
2.1	93.737	96.261	96.261	93.737	99.151	99.151	99.151	99.921	99.987
2.2	186	300	300	186	538	600	538	816	1000
2.3	94.259	94.259	94.259	94.259	99.103	99.715	99.103	99.947	99.993
2.4	105	300	300	105	541	600	541	797	1000
2.5	88.831	95.032	95.032	88.831	99.323	99.467	99.323	99.904	99.994
2.6	13	300	300	13	585	600	585	862	1000
2.7	93.737	96.223	96.223	93.737	99.431	99.431	99.431	99.934	99.977
2.8	117	300	300	117	581	600	581	842	1000
2.9	95.280	95.280	95.280	95.280	99.130	99.422	99.130	99.897	99.986
3.0	13	300	300	13	584	600	584	821	1000
3.1	93.737	96.250	96.250	93.737	99.604	99.604	99.604	99.935	99.996
3.2	224	300	300	224	431	600	431	640	1000
3.3	91.320	93.926	93.926	91.320	99.570	99.925	99.570	99.952	99.995
3.4	298	300	300	298	394	532	394	532	947
3.5	92.571	92.571	92.571	92.571	99.066	99.881	99.066	99.881	99.991
3.6	13	300	300	13	433	600	433	606	940
3.7	92.780	97.874	97.874	92.780	99.550	99.958	99.550	99.958	99.986
3.8	158	300	300	158	431	600	431	682	1000
3.9	93.209	97.628	97.628	93.209	99.023	99.872	99.023	99.933	99.989
4.0	13	300	300	13	408	600	408	694	1000
4.1	92.780	97.874	97.874	92.780	99.551	99.911	99.551	99.950	99.999
4.2	151	274	300	151	274	600	274	751	1000
4.3	94.259	98.801	99.218	94.259	98.801	99.781	98.801	99.892	99.986
4.4	139	295	300	139	295	600	295	657	1000
4.5	92.836	99.512	99.512	92.836	99.512	99.917	99.512	99.917	99.997
4.6	13	300	300	13	318	600	318	762	1000
4.7	93.737	99.668	99.668	93.737	99.668	99.783	99.668	99.927	99.987
4.8	145	266	300	145	266	600	266	670	1000
4.9	95.264	99.054	99.054	95.264	99.054	99.824	99.054	99.898	99.997
5.0	13	300	300	13	356	600	356	648	1000
5.1	93.737	99.016	99.016	93.737	99.161	99.871	99.161	99.871	99.996

Table A5 — Network 2 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$  (benchmark throughput: 13.4129)

(a) Milestone Table

algo	adm throughput at first exit	best adm throughput	last adm throughput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4 86.06%	991 100.00%	1000 99.99%	13	489	569	586	740
1.1	7 92.83%	989 100.00%	998 100.00%	70	485	545	603	711
1.2	31 97.58%	989 100.00%	1000 99.99%	26	45	539	616	736
2.0	4 86.06%	954 100.00%	998 100.00%	13	150	406	466	633
2.1	7 92.83%	990 100.00%	1000 100.00%	70	227	319	434	589
2.2	31 97.58%	936 100.00%	999 100.00%	26	45	375	446	620
2.3	9 94.71%	990 99.99%	1000 99.99%	16	78	371	497	624
2.4	31 97.58%	997 100.00%	1000 99.99%	26	45	171	443	616
3.0	4 85.99%	938 99.99%	999 99.99%	13	267	354	379	502
3.1	7 92.68%	984 100.00%	999 99.99%	34	234	352	398	481
3.2	31 98.03%	991 100.00%	997 99.99%	27	31	237	383	474
3.3	9 94.50%	991 99.99%	1000 99.99%	16	100	326	399	522
3.4	31 98.03%	996 99.99%	996 99.99%	27	31	237	376	426
4.0	4 86.06%	996 99.99%	999 99.98%	13	154	189	369	650
4.1	7 92.83%	1000 99.94%	1000 99.94%	70	135	218	326	697
4.2	31 97.58%	1000 100.00%	1000 100.00%	26	45	179	220	411
4.3	9 94.71%	939 100.00%	1000 99.99%	16	78	175	216	433
4.4	31 97.58%	957 99.99%	1000 99.99%	26	45	155	184	490

(b) Offered-Load Table

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	S
best	0.0002	3.2507	1.1397	2.0615	2.3653	1.3159	1.2968	2.5453	3.4367	1.7296	13.4129
worst	0.9999	0.9999	0.9999	1.0000	0.9819	1.0000	0.9999	0.9999	0.9999	0.9989	0.9989
	0.0298	3.2340	1.1582	2.0825	2.3093	1.3500	1.2961	2.5239	3.4291	1.7127	13.4050
	0.9998	1.0000	0.9998	0.9999	0.9761	1.0000	0.9997	0.9999	0.9999	0.9998	99.94%

(c) Stopping-Rule Table

algo	$\delta = 0.1$			max it = 300			max it = 600			max it = 1000		
	12	300	0.001	0.01	0.001	0.001	0.01	0.001	0.001	0.01	0.001	0.001
1.0	12	300	300	300	300	300	486	600	486	600	486	600
1.1	28	300	300	300	300	300	86.469	97.890	86.469	97.890	86.469	97.890
1.2	12	31	300	300	300	300	92.835	96.942	92.835	96.942	92.835	96.942
2.0	12	300	300	300	300	300	82.152	97.577	82.152	97.577	82.152	97.577
2.1	28	279	300	300	300	300	86.469	98.419	86.469	98.419	86.469	98.419
2.2	12	31	300	300	300	300	92.835	98.849	92.835	98.849	92.835	98.849
2.3	9	300	300	300	300	300	82.152	97.577	82.152	97.577	82.152	97.577
2.4	12	31	300	300	300	300	94.707	98.529	94.707	98.529	94.707	98.529
3.0	12	300	300	300	300	300	82.152	97.577	82.152	97.577	82.152	97.577
3.1	33	300	300	300	300	300	86.558	98.176	86.558	98.176	86.558	98.176
3.2	12	16	300	300	300	300	94.623	98.149	94.623	98.149	94.623	98.149
3.3	9	300	300	300	300	300	81.844	99.068	81.844	99.068	81.844	99.068
3.4	12	16	300	300	300	300	94.502	98.780	94.502	98.780	94.502	98.780
4.0	12	175	300	300	300	300	81.844	99.068	81.844	99.068	81.844	99.068
4.1	28	159	300	300	300	300	86.469	98.435	86.469	98.435	86.469	98.435
4.2	12	31	300	300	300	300	92.835	98.192	92.835	98.192	92.835	98.192
4.3	9	152	300	300	300	300	82.152	97.577	82.152	97.577	82.152	97.577
4.4	12	31	292	292	292	292	94.707	98.303	94.707	98.303	94.707	98.303
	82.152	97.577	99.840	99.840	99.840	99.840	82.152	97.577	82.152	97.577	99.840	99.840

Table A6 — Network 3 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$  (benchmark throughput: 2.2436)

(a) Milestone Table

algo	adm throughput at first exit	best adm throughput	last adm throughput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4	962	998	122	546	592	647	745
1.1	6	1000	1000	139	522	569	663	775
1.2	17	1000	1000	17	204	574	654	788
2.0	4	946	1000	346	371	464	522	721
2.1	6	995	999	117	356	415	508	667
2.2	17	1000	1000	17	402	472	517	653
2.3	8	955	1000	81	410	460	536	662
2.4	17	984	998	17	259	260	494	703
3.0	4	947	1000	76	358	395	428	515
3.1	6	958	1000	140	339	386	407	483
3.2	17	994	999	33	126	385	405	502
3.3	8	987	1000	19	181	390	408	514
3.4	17	895	1000	33	126	355	399	483
4.0	4	798	999	117	184	196	246	495
4.1	6	916	1000	120	154	196	231	546
4.2	17	892	997	17	182	209	224	517
4.3	8	997	999	81	183	210	255	513
4.4	17	995	999	17	160	188	201	603

(b) Offered-Load Table

	$\lambda_1$ $\hat{F}_1$	$\lambda_2$ $\hat{F}_2$	$\lambda_3$ $\hat{F}_3$	$\lambda_4$ $\hat{F}_4$	$\lambda_5$ $\hat{F}_5$	$\lambda_6$ $\hat{F}_6$	$\lambda_7$ $\hat{F}_7$	$\lambda_8$ $\hat{F}_8$	S
best	0.2481	0.2529	0.2852	0.3242	0.2808	0.2777	0.3122	0.2646	2.2436
worst	0.9990	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	0.9990	1.0000	0.9990	0.9990	0.9990	1.0000	1.0000	1.0000	99.99%

(c) Stopping-Rule Table

algo	max. it = 300			max. it = 600			max. it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.01	0.001	0.001
1.0	300	300	300	515	600	600	685	893	1000
1.1	95.912	95.912	95.912	96.058	99.096	99.096	99.675	99.963	99.996
1.2	300	300	300	471	600	600	629	814	1000
2.0	95.481	95.481	95.481	96.686	99.123	99.123	99.193	99.940	99.999
2.1	13	145	300	13	145	600	145	906	1000
2.2	81.757	97.782	98.122	81.757	97.782	99.381	97.782	99.982	99.992
2.3	300	300	300	368	600	600	627	862	1000
2.4	93.951	93.951	93.951	96.124	99.782	99.782	99.804	99.973	99.992
3.0	112	300	300	112	525	600	525	784	1000
3.1	94.007	97.262	97.262	94.007	99.802	99.802	99.802	99.965	99.998
3.2	13	300	300	13	600	600	622	904	1000
3.3	81.757	97.782	97.782	81.757	99.814	99.814	99.814	99.977	99.999
3.4	300	300	300	368	600	600	620	866	1000
4.0	97.200	97.200	97.200	97.987	99.824	99.824	99.824	99.966	99.997
4.1	13	300	300	13	600	600	639	898	955
4.2	81.757	99.473	99.473	81.757	99.720	99.720	99.845	99.992	99.992
4.3	300	300	300	336	451	600	451	758	1000
4.4	96.071	96.071	96.071	96.071	99.735	99.735	99.735	99.990	100.000
4.5	179	300	300	179	387	585	387	585	1000
4.6	97.612	97.612	97.612	97.612	99.106	99.977	99.106	99.977	99.997
4.7	13	185	300	13	185	524	185	524	524
4.8	80.960	98.351	98.351	80.960	98.351	99.923	98.351	99.923	99.923
4.9	12	300	300	12	440	441	440	441	1000
5.0	88.757	98.467	98.467	88.757	99.634	99.634	99.634	99.634	99.997
5.1	13	185	300	13	185	600	185	710	962
5.2	80.960	98.351	98.351	80.960	98.351	99.953	98.351	99.966	99.996
5.3	165	300	300	165	433	600	433	755	1000
5.4	96.399	99.673	99.673	96.399	99.844	99.928	99.844	99.968	99.995
5.5	153	242	300	153	242	600	242	624	995
5.6	96.962	99.544	99.697	96.962	99.544	99.932	99.544	99.955	99.996
5.7	13	257	300	13	257	600	257	798	1000
5.8	81.757	99.633	99.668	81.757	99.633	99.931	99.633	99.984	99.996
5.9	164	275	275	164	275	275	275	275	1000
6.0	97.154	99.545	99.545	97.154	99.545	99.545	99.545	99.545	99.995
6.1	13	294	294	13	294	294	294	294	1000
6.2	81.757	99.712	99.712	81.757	99.712	99.712	99.712	99.712	99.998

Table A7 — Network 3 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$  (benchmark throughput: 9.4128)

(a) Milestone Table

algo	adm thput at first exit	best adm thput	last adm thput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4	997	1000	47	499	524	596	730
1.1	8	999	99.99%	8	24	420	570	700
1.2	31	987	100.00%	30	47	50	580	724
2.0	4	997	1000	47	118	361	484	669
2.1	8	989	99.99%	8	24	111	371	624
2.2	31	995.99%	100.00%	30	47	50	456	608
2.3	10	966	999	10	59	342	459	649
2.4	31	972	1000	30	47	50	251	641
3.0	4	987	1000	83	256	340	365	454
3.1	8	912	1000	8	24	305	375	425
3.2	37	942	1000	32	69	71	254	484
3.3	10	992	99.99%	10	67	145	398	445
3.4	37	938	1000	32	69	71	254	408
4.0	4	962	999	47	150	170	201	417
4.1	8	980	1000	8	24	163	192	340
4.2	31	993	1000	30	47	50	198	471
4.3	10	990	1000	10	59	164	210	422
4.4	31	971	1000	30	47	50	161	349

(b) Offered-Load Table

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	S
best	1.1174	1.2697	1.5078	2.2527	1.6666	1.7254	2.1911	1.7158	9.4128
worst	1.0000	0.9998	0.9999	1.0000	0.9999	1.0000	1.0000	0.9999	0.9999
	1.1127	1.2428	1.5010	2.2480	1.6754	1.7293	2.2056	1.7274	9.4116
	0.9998	0.9959	1.0000	0.9999	0.9997	0.9999	1.0000	0.9999	99.99%

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.1	0.01	0.001
1.0	12	13	300	12	13	600	12	13	632
1.1	72	300	300	72	532	600	72	532	99.986
1.2	10	27	300	10	27	600	10	27	902
2.0	12	13	300	12	13	600	12	13	917
2.1	72	300	300	72	372	600	72	372	99.990
2.2	10	27	300	10	27	600	10	27	99.994
2.3	11	300	300	11	422	600	11	422	885
2.4	10	27	300	10	27	600	10	27	99.995
3.0	12	13	300	12	13	600	12	13	918
3.1	33	300	300	33	341	473	33	341	99.992
3.2	10	52	300	10	52	600	10	52	848
3.3	11	300	300	11	367	472	11	367	99.984
3.4	10	52	300	10	52	600	10	52	99.993
4.0	12	13	300	12	13	600	12	13	843
4.1	72	184	300	72	184	409	72	184	99.990
4.2	10	27	300	10	27	600	10	27	576
4.3	11	178	300	11	178	485	11	178	99.950
4.4	10	27	300	10	27	600	10	27	801

Table A8 — Network 3 with  $T_i = 8$ ,  $X_j = 6$ , and  $Q_j = 0.001$  (benchmark throughput: 4.3486)

(a) Milestone Table

algo	adm thrupt at first exit	best adm thrupt	last adm thrupt	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4	981	999	11	11	602	650	804
1.1	8	959	997	60	496	543	608	770
1.2	23	1000	1000	23	78	589	653	821
2.0	4	985	998	11	11	467	529	777
2.1	8	999	999	60	240	399	489	690
2.2	23	983	1000	23	78	463	517	790
2.3	10	982	999	32	419	463	529	763
2.4	23	999	999	23	78	482	538	738
3.0	4	912	999	9	9	390	421	581
3.1	8	992	999	27	328	365	375	478
3.2	23	998	1000	39	90	387	442	563
3.3	10	959	1000	23	171	389	440	553
3.4	23	994	999	39	90	337	384	489
4.0	4	992	1000	11	11	205	267	604
4.1	8	990	997	60	150	201	201	505
4.2	23	990	999	23	78	210	262	627
4.3	10	998	1000	32	178	210	256	587
4.4	23	974	1000	23	78	137	205	611

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.1	0.01	0.001
1.0	300	300	300	484	600	600	669	902	1000
1.1	56	300	300	56	505	600	505	822	1000
1.2	11	300	300	11	600	600	673	908	1000
2.0	300	300	300	379	600	600	622	872	1000
2.1	56	300	300	56	500	600	500	768	1000
2.2	11	300	300	11	574	600	574	871	1000
2.3	55	300	300	55	577	600	577	874	1000
2.4	11	300	300	11	573	600	573	875	1000
3.0	300	300	300	316	413	600	413	755	1000
3.1	28	300	300	28	391	535	391	535	962
3.2	11	300	300	11	462	600	462	736	1000
3.3	84	300	300	84	447	600	447	780	1000
3.4	11	300	300	11	379	494	379	494	1000
4.0	145	300	300	145	332	600	332	802	1000
4.1	56	228	300	56	228	566	228	566	984
4.2	11	300	300	11	342	600	342	789	1000
4.3	55	300	300	55	346	600	346	802	1000
4.4	11	300	300	11	343	600	343	800	1000

(b) Offered-Load Table

	$\lambda_1$ $\hat{F}_1$	$\lambda_2$ $\hat{F}_2$	$\lambda_3$ $\hat{F}_3$	$\lambda_4$ $\hat{F}_4$	$\lambda_5$ $\hat{F}_5$	$\lambda_6$ $\hat{F}_6$	$\lambda_7$ $\hat{F}_7$	$\lambda_8$ $\hat{F}_8$	S
best	0.4010	0.4163	0.5232	0.6949	0.5303	0.5483	0.6937	0.5451	4.3486
worst	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	
	0.3974	0.3991	0.5180	0.6927	0.5364	0.5508	0.7042	0.5529	4.3471
	0.9980	0.9840	0.9980	1.0000	0.9990	0.9990	1.0000	0.9990	99.97%

Table A9 — Network 3 with  $T_i = 8$ ,  $X_j = 6$ , and  $Q_j = 0.3$  (benchmark throughput: 13.9482)

(a) Milestone Table

algo	adm throughput at first exit	best adm throughput	last adm throughput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4	989	999	4	475	531	590	720
1.1	7	1000	1000	7	200	200	562	702
1.2	24	989	999	21	34	37	579	712
2.0	4	965	999	4	103	351	470	612
2.1	7	995	999	7	107	107	425	615
2.2	24	993	999	21	34	37	447	569
2.3	10	989	1000	9	42	347	472	599
2.4	24	979	1000	21	34	37	470	622
3.0	4	999	999	4	93	308	359	470
3.1	7	896	1000	7	98	198	368	456
3.2	24	962	999	21	38	50	200	456
3.3	10	986	1000	9	47	241	358	457
3.4	24	991	1000	21	38	50	200	414
4.0	4	965	1000	4	152	170	182	437
4.1	7	874	999	7	110	143	183	387
4.2	24	963	999	21	34	37	195	427
4.3	10	990	1000	9	42	169	200	389
4.4	24	998	1000	21	34	37	166	380
4.4	24	998	1000	21	34	37	166	380

(b) Offered-Load Table

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	S
best	1.7132	1.8659	2.2145	3.2701	2.4814	2.5598	3.2469	2.5737	13.9482
worst	1.0000	0.9999	0.9998	1.0000	1.0000	0.9999	0.9999	1.0000	100.00%
	1.7123	1.8649	2.2150	3.2691	2.4813	2.5595	3.2469	2.5741	13.9476
	0.9996	0.9997	0.9999	0.9996	0.9998	0.9998	0.9997	0.9999	100.00%

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.1	0.01	0.001
1.0	12	300	300	12	552	600	552	738	937
1.1	59	155	300	96.330	99.325	99.524	96.330	99.325	99.990
1.2	10	19	300	97.828	99.075	99.822	97.828	99.828	99.993
2.0	12	300	300	12	405	600	12	607	910
2.1	59	300	300	96.330	99.397	99.893	96.330	99.397	99.993
2.2	10	19	300	97.828	99.014	99.731	97.828	99.731	99.991
2.3	9	300	300	81.531	94.559	99.925	81.531	94.559	99.997
2.4	10	19	300	95.486	98.503	99.214	95.486	99.214	99.996
3.0	12	300	300	12	355	463	12	463	692
3.1	35	300	300	96.272	98.732	99.131	96.272	99.131	99.987
3.2	10	19	300	97.902	99.328	99.895	97.902	99.328	99.988
3.3	9	300	300	81.247	94.142	99.674	81.247	94.142	99.984
3.4	10	19	300	95.286	99.385	99.891	95.286	99.552	99.989
4.0	12	169	300	12	169	437	12	169	796
4.1	59	166	300	96.330	98.889	99.910	96.330	98.889	99.991
4.2	10	19	300	97.828	99.463	99.846	97.828	99.463	99.996
4.3	9	176	300	81.531	94.559	99.887	81.531	94.559	99.983
4.4	10	19	300	95.486	99.206	99.785	95.486	99.206	99.995
4.4	81.531	94.559	99.830	81.531	94.559	99.830	81.531	94.559	99.830

## APPENDIX B

### Tables for Runs with $\lambda_{\min} > 0$

This appendix consists of milestone tables, offered-load tables, and stopping-rule tables for three examples in which each of the circuits is guaranteed an offered-load of at least  $\lambda_{\min}$ . These tables are discussed in Section 9.1.

Table B1 — Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$ ;  $\lambda_{\min} = 0.5$  (benchmark throughput: 11.2368)

(a) Milestone Table

algo	adm throughput at first exit	best adm throughput	last adm throughput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	86.55%	99.26%	99.26%	75	459	595		
1.1	91.45%	99.32%	99.31%	30	480	621		
1.2	91.45%	99.32%	99.31%	17	76	579	641	775
2.0	97.40%	99.98%	99.98%	75	356	482		
2.1	91.45%	99.30%	99.30%	30	281	452		
2.2	97.40%	100.00%	100.00%	17	76	362	484	628
2.3	91.13%	99.36%	99.36%	17	69	457		
2.4	97.40%	99.98%	99.98%	17	76	202	492	659
3.0	86.48%	99.28%	99.28%	20	339	385		
3.1	91.30%	99.35%	99.34%	31	326	389		
3.2	97.74%	99.43%	99.43%	17	56	408		
3.3	90.97%	99.27%	99.27%	21	51	398		
3.4	97.74%	99.47%	99.46%	17	56	387		
4.0	86.55%	99.27%	99.26%	75	171	223		
4.1	91.45%	99.29%	99.28%	30	136	201		
4.2	97.40%	99.99%	99.99%	17	76	166	324	538
4.3	91.13%	99.28%	99.28%	17	69	217		
4.4	97.40%	99.94%	99.93%	17	76	190	359	762

(b) Offered-Load Table

	$\lambda_1$ $\hat{F}_1$	$\lambda_2$ $\hat{F}_2$	$\lambda_3$ $\hat{F}_3$	$\lambda_4$ $\hat{F}_4$	$\lambda_5$ $\hat{F}_5$	$\lambda_6$ $\hat{F}_6$	$\lambda_7$ $\hat{F}_7$	$\lambda_8$ $\hat{F}_8$	$\lambda_9$ $\hat{F}_9$	$\lambda_{10}$ $\hat{F}_{10}$	S
best	3.0490	3.3214	1.1858	0.5003	1.4128	0.5005	3.5801	0.5002	1.1149	0.5004	11.2368
worst	0.9999	0.9998	0.8298	0.9044	0.7935	1.0000	1.0000	0.9998	0.7782	0.7734	
	1.2679	1.3580	2.4087	0.7387	2.3995	0.5001	2.7534	0.5001	2.4312	1.0532	11.1540
	0.7798	0.7876	0.9998	0.7984	0.8594	1.0000	0.9999	1.0000	0.9999	0.8422	99.26%

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	$\delta = 0.1$	0.01	0.001	$\delta = 0.1$	0.01	0.001
1.0	12	300	300	12	567	600	12	567	600
1.1	9	300	300	9	523	600	9	523	600
1.2	10	300	300	10	438	600	10	438	600
2.0	12	146	300	12	146	521	12	146	521
2.1	9	300	300	9	366	600	9	366	600
2.2	10	110	300	10	110	600	10	110	600
2.3	10	181	300	10	181	565	10	181	565
2.4	10	188	300	10	188	600	10	188	600
3.0	12	300	300	12	368	490	12	368	490
3.1	9	300	300	9	339	457	9	339	457
3.2	10	109	300	10	109	469	10	109	469
3.3	10	206	300	10	206	395	10	206	395
3.4	10	109	300	10	109	440	10	109	440
4.0	12	179	247	12	179	247	12	179	247
4.1	9	174	300	9	174	399	9	174	399
4.2	10	117	226	10	117	226	10	117	226
4.3	10	172	208	10	172	208	10	172	208
4.4	10	158	213	10	158	213	10	158	213

Table B2 — Network 1 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.001$ ;  $\lambda_{\min} = 0.001$ ; (benchmark throughput: 2.6537)

(a) Milestone Table

algo	adm thrupt at first exit	best adm thrupt	last adm thrupt	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	6	940	1000	41	487	591	610	701
1.1	11	971	1000	42	501	513	578	757
1.2	24	982	1000	24	98	518	640	815
2.0	6	988	1000	108	387	394	500	758
2.1	11	983	999	42	339	401	442	696
2.2	24	988	1000	24	98	429	559	737
2.3	12	1000	1000	35	408	420	544	717
2.4	24	972	1000	24	98	424	522	779
3.0	6	949	998	31	363	380	383	484
3.1	12	991	996	56	288	354	399	457
3.2	26	916	999	26	98	251	394	478
3.3	12	973	1000	35	157	359	386	535
3.4	26	971	998	26	98	251	375	470
4.0	6	943	1000	116	157	157	238	560
4.1	11	894	1000	42	137	175	229	364
4.2	24	980	1000	24	98	208	236	536
4.3	12	970	996	35	168	185	253	605
4.4	24	995	1000	24	98	173	189	552

(b) Offered-Load Table

	$\lambda_1$ $\hat{P}_1$	$\lambda_2$ $\hat{P}_2$	$\lambda_3$ $\hat{P}_3$	$\lambda_4$ $\hat{P}_4$	$\lambda_5$ $\hat{P}_5$	$\lambda_6$ $\hat{P}_6$	$\lambda_7$ $\hat{P}_7$	$\lambda_8$ $\hat{P}_8$	$\lambda_9$ $\hat{P}_9$	$\lambda_{10}$ $\hat{P}_{10}$	$S$
best	0.2376	0.3071	0.3057	0.2147	0.3387	0.1839	0.3925	0.1001	0.3257	0.2505	2.6537
worst	1.0000	0.9780	1.0000	0.9960	1.0000	1.0000	1.0000	1.0000	0.9990	0.9070	0.9070
	0.2389	0.2974	0.3055	0.2127	0.3382	0.1847	0.3922	0.1000	0.3258	0.2605	2.6531
	0.9990	0.9540	1.0000	0.9990	1.0000	1.0000	1.0000	1.0000	0.9990	0.9280	99.98%

(c) Stopping-Rule Table

algo	max it = 300			max it = 600			max it = 1000		
	$\delta = 0.1$	0.01	0.001	0.1	0.01	0.001	0.1	0.01	0.001
1.0	300	300	300	432	600	600	610	600	610
1.1	11	300	300	11	578	600	578	600	578
1.2	7	300	300	7	600	600	607	600	607
2.0	80	300	300	80	439	600	439	600	439
2.1	11	300	300	11	472	600	472	600	472
2.2	7	300	300	7	461	600	461	600	461
2.3	10	300	300	10	474	600	474	600	474
2.4	7	300	300	7	491	600	491	600	491
3.0	69	300	300	69	392	470	392	470	392
3.1	11	300	300	11	335	495	335	495	335
3.2	7	145	300	7	145	490	145	490	145
3.3	10	300	300	10	371	482	371	482	371
3.4	7	145	300	7	145	436	145	436	145
4.0	80	201	300	80	201	600	201	600	201
4.1	11	192	300	11	192	335	192	335	192
4.2	7	226	300	7	226	504	226	504	226
4.3	10	233	300	10	233	535	233	535	233
4.4	7	180	300	7	180	449	180	449	180

Table B3 — Network 2 with  $T_i = 6$ ,  $X_j = 4$ , and  $Q_j = 0.3$ ;  $\lambda_{\min} = 0.5$  (benchmark throughput: 13.2950)

(a) Milestone Table

algo	adm thput at first exit	best adm thput	last adm thput	95.0%	98.0%	99.0%	99.5%	99.9%
1.0	4	997	997	46	485	526	562	738
1.1	6	966	998	11	98	534	585	696
1.2	32	997	999	26	31	63	566	720
2.0	4	998	1000	46	135	368	426	658
2.1	6	964	1000	11	98	113	429	603
2.2	32	937	998	26	31	63	545	708
2.3	8	988	995	13	121	416	438	648
2.4	32	992	999	26	31	63	485	634
3.0	4	985	1000	32	83	351	386	686
3.1	6	987	994	11	241	350	387	497
3.2	34	974	1000	27	32	74	74	451
3.3	8	1000	1000	13	65	269	387	497
3.4	34	998	1000	27	32	74	74	514
4.0	4	928	998	46	147	168	206	398
4.1	6	959	999	11	98	168	204	410
4.2	32	994	999	26	31	63	192	391
4.3	8	989	1000	13	123	155	228	440
4.4	32	996	1000	26	31	63	166	441

(b) Offered-Load Table

	$\lambda_1$ $F_1$	$\lambda_2$ $F_2$	$\lambda_3$ $F_3$	$\lambda_4$ $F_4$	$\lambda_5$ $F_5$	$\lambda_6$ $F_6$	$\lambda_7$ $F_7$	$\lambda_8$ $F_8$	$\lambda_9$ $F_9$	$\lambda_{10}$ $F_{10}$	S
best	0.5002	2.9095	1.4482	2.4442	1.4953	1.9257	1.3360	2.1995	3.3015	1.3763	13.2950
worst	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	1.0000	0.9981	13.2864
	0.5344	2.8754	1.4530	2.4838	1.4640	1.9876	1.3586	2.1946	3.2834	1.2840	13.2864
	0.9994	1.0000	0.9998	0.9998	0.9186	0.9999	1.0000	0.9996	0.9997	0.9828	99.94%

(c) Stopping-Rule Table

algo	$\delta = 0.1$		max it = 300		max it = 600		max it = 1000	
	$\delta = 0.1$	0.01	0.001	0.01	0.01	0.01	0.01	0.001
1.0	12	300	300	12	553	600	553	0.001
1.1	10	300	300	10	90.788	99.309	99.309	0.001
1.2	10	300	300	10	90.174	98.863	98.863	0.001
2.0	12	300	300	12	74.737	98.914	98.914	0.001
2.1	10	300	300	10	90.788	98.865	98.865	0.001
2.2	10	300	300	10	90.174	99.224	99.224	0.001
2.3	10	279	300	10	74.737	98.914	98.914	0.001
2.4	10	300	300	10	93.941	98.802	98.802	0.001
3.0	12	300	300	12	74.737	98.914	98.914	0.001
3.1	10	300	300	10	90.904	98.687	98.687	0.001
3.2	10	300	300	10	90.253	98.672	98.672	0.001
3.3	10	300	300	10	74.522	98.542	98.542	0.001
3.4	10	300	300	10	93.776	99.263	99.263	0.001
4.0	12	168	300	12	74.522	98.542	98.542	0.001
4.1	10	161	300	10	90.788	99.150	99.150	0.001
4.2	10	300	300	10	90.174	98.750	98.750	0.001
4.3	10	189	300	10	74.737	98.914	98.914	0.001
4.4	10	300	300	10	93.941	99.306	99.306	0.001