On the Design of Dynamic Reconfiguration Policies for Broadcast WDM Networks

Ilia Baldine[a] and George N. Rouskas[b]

[a]Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534

[b]Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534

ABSTRACT

We study the issues arising when considering the problem of reconfiguring broadcast optical networks in response to changes in the traffic patterns. Although the ability to dynamically optimize the network under changing traffic conditions has been recognized as one of the key features of multiwavelength optical networks, this is the first in-depth study of the tradeoffs involved in carrying out the reconfiguration process. We first identify the degree of load balancing and the number of retunings as two important, albeit conflicting, objectives in the design of reconfiguration policies. We then formulate the problem as a Markovian Decision Process and we develop a systematic and flexible framework in which to view and contrast reconfiguration policies. We also apply results from Markov Decision Process theory to obtain optimal reconfiguration policies even for networks of large size. The advantages of optimal policies over a class of threshold-based policies are also illustrated through numerical results.

Keywords: Broadcast optical networks, Wavelength division multiplexing (WDM), Reconfiguration policies, Markov decision process

1. INTRODUCTION

One of the key features of multiwavelength optical networks is *rearrangeability*,[1] i.e., the ability to dynamically optimize the network for changing traffic patterns, or to cope with failure of network equipment. This ability arises as a consequence of the independence between the logical connectivity and the underlying physical infrastructure of fiber glass. By employing tunable optical devices, the assignment of transmitting or receiving wavelengths to the various network nodes may be updated on the fly, allowing the network to closely track changing traffic conditions.

While the rearrangeability property makes it possible to design traffic-adaptive, self-healing networks, the reconfiguration phase will interfere with existing traffic and disrupt network performance, causing a degradation of the quality of service perceived by the users. The issues that arise in reconfiguring a lightwave network by retuning a set of slowly tunable transmitters or receivers have been studied in the context of multihop networks.[2–4] Specifically, the problem of obtaining a virtual topology that minimizes the maximum link flow, given a set of traffic demands, has been studied,[2] algorithms have been developed for minimizing the number of branch-exchange operations required to take the network from an initial to a target virtual topology, once the traffic pattern changes,[3] and near-optimal policies to dynamically determine when and how to reconfigure the network have been obtained.[4]

In this paper we study the reconfiguration issues arising in single-hop lightwave networks, an architecture suitable for Local and Metropolitan Area Networks (LANs and MANs).[5] The single-hop architecture employs wavelength division multiplexing (WDM) to provide connectivity among the network nodes. The various channels are dynamically shared by the attached nodes, and the logical connections change on a packet-by-packet basis creating all-optical paths between sources and destinations. Thus single-hop networks require the use of rapidly tunable optical lasers and/or filters that can switch between channels at high speeds.

When tunability only at one end, say, at the transmitters, is employed, each fixed receiver is permanently assigned to one of the wavelengths used for packet transmissions. In a typical near-term WDM environment, the number of channels supported within the optical medium is expected to be smaller than the number of attached nodes. As a result, each channel will have to be shared by multiple receivers, and the problem of assigning receive wavelengths arises. Intuitively, a wavelength assignment (hereafter referred to as WLA) must be somehow based on the prevailing traffic conditions. More specifically, the stability condition for the HiPeR-ℓ reservation protocol[6] for broadcast WDM

This work was supported by the NSF under grant NCR-9701113.

networks suggests that, in determining an appropriate WLA, the objective should be to balance the offered load across all channels, such that each channel carries an approximately equal portion of the overall traffic. But with fixed receivers, any WLA is permanent and cannot be updated in response to changes in the traffic pattern.

Alternatively, one can use *slowly tunable*, rather than fixed, receivers. We will say that an optical laser or filter is rapidly tunable if its tuning latency (i.e., the time it takes to switch from one wavelength to another) is in the order of a packet transmission time at the high-speed rates at which optical networks are expected to operate. Slowly tunable devices, on the other hand, have tuning times that can be significantly longer. As a result, these devices cannot be assumed "tunable" at the media access level (i.e., for the purposes of scheduling packet transmissions), as this requires fast tunability. Motivation for the use of slowly tunable lasers or filters is provided by two factors. First, they can be significantly less expensive than rapidly tunable devices, making it possible to design lightwave network architectures that can be realized cost effectively. Second, the variation in traffic demands is expected to take place over larger time scales (several orders of magnitude larger than a single packet transmission time). Hence, even very slow tunable devices will be adequate for updating the WLA over time to accommodate varying traffic demands.

Assuming an existing WLA and some information about the new traffic demands, a new WLA, optimized for the new traffic pattern, must be determined. We have considered this problem,[7] and we have proposed an approach to reconfiguring the network that is minimally disruptive to existing traffic. Specifically, we have developed the *GLPT* algorithm for obtaining a new WLA such that (a) the new traffic load is balanced across the channels, and (b) the number of receivers that need to be retuned to take the network from the old to the new WLA is minimized. The specifications of *GLPT* include a *knob* parameter which provides for tradeoff selection between load balancing and number of retunings. In terms of load balancing, the WLA obtained by *GLPT* is guaranteed to be no more than 50% away from the optimal one, in the worst case, regardless of the knob value used. *GLPT* also leads to a scalable approach to reconfiguring the network since it tends to select the less utilized receivers for retuning, and since for certain values of the knob parameter the expected number of retunings scales with the number of channels, *not* the number of nodes in the network.

During the reconfiguration phase, while the network makes a transition from one WLA to another, some cost is incurred in terms of packet delay, packet loss, packet desequencing, and the control resources involved in receiver retuning. Clearly, receiver retunings should not be very frequent, since unnecessary retunings affect the performance encountered by the users. Hence, it is desirable to minimize the number of network reconfigurations. However, postponing a necessary reconfiguration also has adverse effects on the overall performance. Since the network does not operate at an optimal point in terms of load balancing, it takes longer to clear a given set of traffic demands, causing longer delays and/or buffer overflows, as well as a decrease in the network's traffic carrying capacity. Similarly, if the decisions are made merely by considering the degree of load balancing, even tiny changes in the traffic demands can lead to constant reconfiguration, thereby significantly hurting network performance. Consequently, it is important to have a performance criterion which can capture the above tradeoffs in an appropriate manner and allow their simultaneous optimization.

In this paper we develop a novel, systematic, and flexible framework in which to view and contrast reconfiguration policies. Specifically, we formulate the problem as a Markovian Decision Process and we show how an appropriate selection of reward and cost functions can achieve the desired balance between various performance criteria of interest. However, because of the huge state space of the underlying Markov process, it is impossible to directly apply appropriate numerical methods to obtain an optimal policy. We therefore develop an approximate model with a manageable state space, which captures the pertinent properties of the original model. We also demonstrate that results from Markov Decision Process theory can be applied in an efficient way to obtain reconfiguration policies for networks of large size.

The rest of this paper is organized as follows. In Section 2 we present a model of the broadcast WDM network under study. In Section 3 we formulate the reconfiguration problem as a Markovian Decision Process, and we discuss the issues of obtaining an optimal policy. We present numerical results in Section 4, where we also compare the optimal policy against a class of threshold policies, and we conclude the paper in Section 5.

2. THE BROADCAST WDM NETWORK

We consider a packet-switched single-hop lightwave network with N nodes, and one transmitter-receiver pair per node. The nodes are physically connected to a passive broadcast optical medium that supports $C < N$ wavelengths, $\lambda_1, \cdots, \lambda_C$. Both the transmitter and the receiver at each node are tunable over the entire range of available wavelengths. However, the transmitters are *rapidly tunable*, while the receivers are *slowly tunable*. We will refer to this tunability configuration as *rapidly tunable transmitter, slowly tunable receiver* (RTT-STR). Although we will only consider RTT-STR networks in this paper, we note that all our results can be easily adapted to the dual configuration, STT-RTR.

We represent the current traffic conditions in the network by a $N \times N$ traffic demand matrix $\mathbf{T}= [t_{ij}]$. Quantity t_{ij} could be a measure of the average traffic originating at node i and terminating at node j, or it could be the effective bandwidth[8] of the traffic from i to j. As traffic varies over time, the elements of matrix $\mathbf{T}$ will change. This variation in traffic takes place at larger scales in time, for instance, we assume that changes in the traffic matrix $\mathbf{T}$ occur at connection request arrival or termination instants. We also assume that the current matrix $\mathbf{T}$ completely summarizes the entire history of traffic changes, so that future changes only depend on the current values of the elements of $\mathbf{T}$.

During normal operation, each of the slowly tunable receivers is assumed to be fixed to a particular wavelength. Let $\lambda(j) \in \{\lambda_1, \cdots, \lambda_C\}$ be the wavelength currently assigned to receiver j. A WLA is a partition $\mathcal{R} = \{R_c,\ c = 1, \cdots, C\}$ of the set $\mathcal{N} = \{1, \cdots, N\}$ of nodes, such that $R_c = \{j \mid \lambda(j) = \lambda_c\},\ c = 1, \cdots, C$, is the subset of nodes currently receiving on wavelength λ_c. This WLA is known to the network nodes, and it is used to determine the target channel for a packet, given the packet's destination. The network operates by having each node employ a media access protocol, such as HiPeR-ℓ, that requires tunability only at the transmitting end. Nodes use HiPeR-ℓ to make reservations, and can schedule packets for transmission using algorithms that can effectively mask the (relatively short) latency of tunable transmitters.[9]

We now define the *degree of load balancing (DLB)* $\phi(\mathcal{R}, \mathbf{T})$ for a network with traffic matrix $\mathbf{T}$ operating under WLA $\mathcal{R}$ as:

$$(1 \ + \ \phi(\mathcal{R}, \mathbf{T})) \ \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij}}{C} \ = \ \max_{c=1,\cdots,C} \left\{ \sum_{i=1}^{N} \sum_{j \in R_c} t_{ij} \right\} \tag{1}$$

The right hand side of (1) represents the bandwidth requirement of the dominant (i.e., most loaded) channel, while the second term in the left hand side of (1) represents the lower bound, with respect to load balancing, for any WLA for traffic matrix $\mathbf{T}$. Thus, the DLB is a measure of how far away WLA $\mathcal{R}$ is from the lower bound. If $\phi = 0$, then the load is perfectly balanced, and each channel carries an equal portion of the offered traffic, while when $\phi > 0$, the channels are not equally loaded. In other words, the DLB characterizes the efficiency of the network in meeting the traffic demands denoted by matrix $\mathbf{T}$ while operating under WLA $\mathcal{R}$: the higher the value of ϕ, the less efficient the WLA is.

2.1. The Transition Phase

In order to more efficiently utilize the bandwidth of the optical medium as traffic varies over time, a new WLA may be sought that distributes the new load more equally among the channels. We will refer to the transition of the network from one WLA to another as *reconfiguration*. In general, we assume that reconfiguration is triggered by changes in the traffic matrix $\mathbf{T}$. When such a change occurs, the following actions must be taken:

1. a new WLA for the new traffic matrix must be determined,
2. a decision must be made on whether or not to reconfigure the network by adopting the new WLA, and
3. if the decision is to reconfigure, the actual retuning of receivers must take place.

The *GLPT* algorithm[7] which takes as input the current WLA $\mathcal{R}$ and the new traffic matrix $\mathbf{T}'$, and determines the new WLA addresses the first issue. The rest of the paper addresses the second problem of determining whether the changes in traffic conditions warrant the reconfiguration of the network to the new WLA. We now discuss the third issue of receiver retuning.

Let $\mathcal{R}$ and $\mathbf{T}$ be the current WLA and traffic matrix, respectively, and let $\mathbf{T}'$ be the new traffic matrix. Let $\mathcal{R}'$ be the new WLA computed by the *GLPT* algorithm with $\mathcal{R}$ and $\mathbf{T}'$ as input. Assuming that a decision has been made to reconfigure, there will be a transition phase during which a set of receivers is retuned to take the network from the current WLA $\mathcal{R}$ to the new WLA $\mathcal{R}'$. Let us define the distance $\mathcal{D}$ between the two WLAs $\mathcal{R}$ and $\mathcal{R}'$ as:

$$\mathcal{D}(\mathcal{R}, \mathcal{R}') \quad = \quad N \; - \; \sum_{c=1}^{C} \mid R_c \cap R'_c \mid \tag{2}$$

The distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$ represents the number of receivers that need to be taken off-line for retuning during the transition phase.

There is a wide range of strategies for retuning the receivers, mainly differing in the tradeoff between the length of the transition period and the portion of the network that becomes unavailable during this period (similar issues also arise in multihop networks[3]). One extreme approach would be to simultaneously retune all the receivers which are assigned new channels under $\mathcal{R}'$. The duration of the transition phase is minimized under this approach (it becomes equal to the receiver tuning latency), but a significant fraction of the network may be unusable during this time. At the other extreme, a strategy that retunes one receiver at a time minimizes the portion of the network unavailable at any given instant during the transition phase, but it maximizes the length of this phase (which now becomes equal to the receiver tuning latency times the distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$). Between these two ends of the spectrum lie a range of strategies in which two or more receivers are retuned simultaneously.

While the receiver of, say, node j, is being retuned to a new wavelength, it cannot receive data, and thus, any packets sent to node j are lost. If, on the other hand, the network nodes are aware that node j is in the process of retuning its receiver, they can refrain from transmitting packets to it. In this case, packets destined to node j will experience longer delays while waiting for the node to become ready for receiving again. Moreover, packets for j arriving to the various transmitters during this time cannot be serviced, and may cause buffer overflows. This increase in delay and/or packet loss is the penalty incurred for reconfiguring the network.

We note that, in general, the reconfiguration penalty associated with retuning a given number D of receivers will depend on the actual retuning strategy employed (e.g., simultaneously retuning all D receivers versus retuning one receiver at a time). Furthermore, the relative penalty of the various retuning strategies is a function of system parameters such as the receiver tuning latency and the offered load. Determining the best retuning strategy for a given region of network operation is beyond the scope of this paper. In our work, we instead develop an abstract model that includes a cost function to account for the reconfiguration penalty. Our model is flexible in that the cost function can be appropriately selected to fit any given strategy.

3. MARKOV DECISION PROCESS FORMULATION

3.1. Reconfiguration Policies

We define the state of the network as a tuple $(\mathcal{R}, \mathbf{T})$. $\mathcal{R}$ is the current WLA, and $\mathbf{T}$ is a matrix representing the prevailing traffic conditions. Changes in the network state occur at instants when the matrix $\mathbf{T}$ is updated. Since we have assumed that future traffic changes only depend on the current values of the elements of $\mathbf{T}$, the process $(\mathcal{R}, \mathbf{T})$ is a semi-Markov process. Let $\mathcal{M}$ be the process embedded at instants when the traffic matrix changes. Then, $\mathcal{M}$ is a discrete-time Markov process. Our formulation is in terms of the Markov process $\mathcal{M}$.

A network in state $(\mathcal{R}, \mathbf{T})$ will enter state $(\mathcal{R}', \mathbf{T}')$ if the traffic matrix changes to $\mathbf{T}'$. Implicit in the state transition is that the system makes a decision to reconfigure to WLA $\mathcal{R}'$. In order to completely define the Markovian state transitions associated with our model, we need to establish *next WLA* decisions. The decision is a function of the current state and is denoted by $d[(\mathcal{R}, \mathbf{T})]$. Setting $d[(\mathcal{R}, \mathbf{T})] = \mathcal{R}_{next}$ implies that if the system is in state $(\mathcal{R}, \mathbf{T})$ and the traffic demands change, the network should be reconfigured into WLA $\mathcal{R}_{next}$. Note that WLA $\mathcal{R}_{next}$ can be the same as $\mathcal{R}$, in which case the decision is not to reconfigure. Therefore, for each state $(\mathcal{R}, \mathbf{T})$ there are two alternatives: either the network reconfigures to WLA $\mathcal{R}'$ obtained by the *GLPT* algorithm with $\mathcal{R}$ and $\mathbf{T}'$ as inputs (in which case the new state will be $(\mathcal{R}', \mathbf{T}')$), or it maintains the current WLA (in which case the new state will be $(\mathcal{R}, \mathbf{T}')$). The set of decisions for all network states defines a *reconfiguration policy.*

To formulate the problem as a Markov Decision Process, we need to specify reward and cost functions associated with each transition. Consider a network in state $(\mathcal{R}, \mathbf{T})$ that makes a transition to state $(\mathcal{R}', \mathbf{T}')$. The network acquires an *immediate expected reward* equal to $\alpha[\phi(\mathcal{R}', \mathbf{T}')]$, where $\alpha(\cdot)$ is a non-increasing function of $\phi(\mathcal{R}', \mathbf{T}')$, the DLB of WLA $\mathcal{R}'$ with respect to the new traffic matrix $\mathbf{T}'$. Also, if $\mathcal{R}' \neq \mathcal{R}$, a *reconfiguration cost* equal to

$\beta[\mathcal{D}(\mathcal{R},\mathcal{R}')]$ is incurred, where $\beta(\cdot)$ is an non-decreasing function of the number of receivers that have to be retuned to take the network to the new WLA $\mathcal{R}'$. In other words, a switching cost is incurred each time the network makes a decision to reconfigure. We assume that the rewards and costs are bounded, i.e.:

$$\alpha_{min} \ \leq \ \alpha[\phi(\mathcal{R}',\mathbf{T}')] \ \leq \ \alpha_{max} \quad \text{and} \quad 0 \ \leq \ \beta_{min} \ \leq \beta[\mathcal{D}(\mathcal{R},\mathcal{R}')] \ \leq \ \beta_{max} \tag{3}$$

where α_{min}, α_{max}, β_{min} and β_{max} are real numbers.

The problem is how to reconfigure the network sequentially in time, so as to maximize the expected reward minus the reconfiguration cost over an infinite horizon. Let $(\mathcal{R}^{(k)},\mathbf{T}^{(k)})$ denote the state of the network immediately after the k-th transition, $k = 1, 2, \cdots$. Let also Z be the set of admissible policies. The network reconfiguration problem can then be formally stated as follows (note that $\mathcal{D}(\mathcal{R},\mathcal{R}) = 0$):

PROBLEM 3.1. *Find an optimal policy $z^\star \in Z$ that maximizes the expected reward*

$$F \ = \ \lim_{k\to\infty} \frac{1}{k} E\left\{\sum_{l=1}^{k} \alpha[\phi(\mathcal{R}^{(l)},\mathbf{T}^{(l)})] \ - \ \beta[\mathcal{D}(\mathcal{R}^{(l-1)},\mathcal{R}^{(l)})]\right\} \tag{4}$$

The first term in the right hand side of (4) is the reward obtained by using a particular WLA, and the second term is the cost incurred at each instant of time that reconfiguration is performed. The presence of a reward which increases as the DLB ϕ decreases (i.e., as the load is better balanced across the channels) provides the network with an incentive to associate with a WLA that performs well for the current traffic load. On the other hand, the introduction of a cost incurred at each reconfiguration instant discourages frequent reconfigurations. Thus, the overall reward function captures the fundamental tradeoff between the DLB and frequent retunings involved in the reconfiguration problem.

For the case $\beta_{max} = 0$, the problem of finding an optimal policy is trivial, since it is optimal for the network to associate with the WLA which best balances the offered load at each instant in time. This is because the evolution of the traffic matrix $\mathbf{T}$ is not affected by the network's actions and reconfigurations are free. However, when $\beta_{max} > 0$, there is a conflict between *future reconfiguration costs incurred* and *current reward obtained*, and it is not obvious as to what constitutes an optimal policy. We also note that as $\beta_{min} \to \infty$, the optimal policy would be to never reconfigure, since this is the only policy for which the expected reward in (4) would be non-negative. Again, however, the point (i.e., the smallest value of β_{min}) at which this policy becomes optimal is not easy to determine, as it depends on the transition probabilities of the underlying Markov chain.

Consider an ergodic, discrete-space discrete-time Markov process with rewards and a set of alternatives per state that affect the probabilities and rewards governing the process. The *policy-iteration* algorithm[10] can be used to obtain a policy that maximizes the long-term reward in (4) for such a process. Initially, an arbitrary policy is specified from which all state transition rates are determined. The algorithm then enters its basic iteration cycle which consists of two stages. The first stage is the *value-determination operation* which evaluates the current policy. In the second stage, the *policy-improvement routine* uses a set of criteria to modify the decisions at each state and obtain a new policy with a higher reward than the original policy. This new policy is used as the starting point for the next iteration. The cycle continues until the policies in two successive iterations are identical. At this point the algorithm has converged, and the final policy is guaranteed to be optimal with respect to maximizing the reward in (4).

A difficulty in applying the policy-iteration algorithm to the Markov process $\mathcal{M}$ is that its running time per iteration is dominated by the complexity of solving a number of linear equations in the order of the number of states in the Markov chain. Even if we restrict the elements of traffic matrix $\mathbf{T}$ to be integers * and impose an upper bound on the values they can take, the potential number of states $(\mathcal{R},\mathbf{T})$ is so large that the policy-iteration algorithm cannot be directly applied to anything but networks of trivial size. In the next subsection we show how to overcome this problem by making some simplifying assumptions that will allow us to set up a new Markov process whose state space is manageable.

*If the elements of $\mathbf{T}$ are real numbers, then $\mathcal{M}$ becomes a continuous-state process and the policy-iteration algorithm cannot be applied.

3.2. Alternative Formulation

Consider a network in state $(\mathcal{R}, \mathbf{T})$, and a new traffic matrix $\mathbf{T}'$ for which the WLA obtained with the *GLPT* algorithm is $\mathcal{R}'$. A closer examination of the reward function in (4) reveals that the immediate reward acquired when the network makes a transition does not depend on the actual values of the traffic elements or the actual WLAs involved, but only on the values of the DLBs $\phi(\mathcal{R}, \mathbf{T}')$ and $\phi(\mathcal{R}', \mathbf{T}')$, and the distance $\mathcal{D}(\mathcal{R}, \mathcal{R}')$. Thus, we make the simplifying assumption that the decision to reconfigure will also depend on the DLBs and the distance only. This is a reasonable assumption, since it is the DLB, not the actual traffic matrix or WLA that determine the efficiency of the network in satisfying the offered load. Similarly, it is the number of retunings that determines the reconfiguration cost, not the actual WLAs involved.

Based on these observations, we now introduce a new process embedded, as Markov process $\mathcal{M}$, at instants when the traffic matrix changes. The state of this process is defined to be the tuple (ϕ, D), where ϕ is the DLB achieved by the current WLA with respect to the current traffic matrix, and D is the number of retunings required if the network were to reconfigure. Transitions in the new process have the Markovian property, since they are due to changes in the traffic matrix which, in turn, are Markovian. However, as defined, the process is a continuous-state process since, in general, the DLB ϕ is a real number. In order to apply Howard's algorithm we need a discrete-state process. We obtain such a process by using discrete values for random variable ϕ as follows.

By definition (refer to expression (1)), the DLB ϕ can take any real value between 0 and $C-1$, where C is the number of channels in the network. We now divide the interval $[0, C-1]$ into a number $K+1$ of non-overlapping intervals $[\phi_0^{(l)}, \phi_0^{(u)}), [\phi_1^{(l)}, \phi_1^{(u)}), \cdots, [\phi_K^{(l)}, \phi_K^{(u)}]$, where $\phi_k^{(l)}$ and $\phi_k^{(u)}$ are the lower and upper values of interval $k, k = 0, \cdots, K$, and: $\phi_k^{(l)} < \phi_k^{(u)}$, $\phi_0^{(l)} = 0$, $\phi_k^{(u)} = \phi_{k+1}^{(l)}$, and $\phi_K^{(u)} = C-1$. Let ϕ_k denote the midpoint of interval k. We now define a new discrete-state process $\mathcal{M}'$ with state (ϕ_k, D). We will use state (ϕ_k, D) to represent any state (ϕ, D) of the continuous-state process such that $\phi_k^{(l)} \le \phi < \phi_k^{(u)}$. Clearly, the larger the number K of intervals, the better the approximation.

Before we proceed, we make one further refinement to the new discrete-state process $\mathcal{M}'$. We note that the *GLPT* algorithm[7] is an approximation algorithm for the load balancing problem, and it guarantees that the DLB of the WLA obtained using the algorithm will never be more than 50% away from the degree of load balancing of the optimal WLA. The importance of this result is as follows. Consider a network in which the traffic matrix changes in such a way that the current WLA provides a DLB ϕ for the new traffic matrix such that $\phi < 0.5$. Based on the guarantee provided by algorithm *GLPT*, we can safely assume that the load is well balanced and avoid a reconfiguration. This is because the network will incur a cost for reconfiguring, without any assurance that the new DLB will be less than ϕ. Therefore, we choose to let $\phi_0^{(u)} = 0.5$, and therefore the midpoint for the first interval is $\phi_0 = 0.25$. We will call any state (ϕ_0, D) a *balanced* state since the offered load is balanced within the guarantees of the *GLPT* algorithm.

We now specify decision alternatives, as well as reward and cost functions associated with each transition in the new process $\mathcal{M}'$. Consider a network in state (ϕ_k, D). At the instant the traffic matrix changes, the network has two options. It may maintain the current WLA, in which case it will make a transition into state (ϕ_l, D'), where ϕ_l is the DLB of the current WLA with respect to the new traffic matrix, and D' is the new distance. Or, it will reconfigure into a new WLA. In the latter case, the network will move into state (ϕ_0, D''), since its new DLB is guaranteed to be less than 0.5. When the network makes a transition into state $(\phi_l, D'), l \ge 0$, it acquires an immediate expected reward which is equal to $\alpha(\phi_l)$. In addition, if (ϕ_l, D') is a balanced state (i.e., if $l = 0$), a reconfiguration cost equal to $\beta(D)$ is incurred.

The transitions out of state (ϕ_k, D) and the corresponding rewards are illustrated in Figure 1. If the decision of the policy is not to reconfigure, then the process will take one of the transitions indicated by the solid arrows in Figure 1. Since the network does not incur any reconfiguration cost, the immediate reward acquired is a function of the new DLB in the new state. If, on the other hand, the decision is to reconfigure, the transition out of state (ϕ_k, D) will always take the network to a balanced state with a DLB equal to ϕ_0. These transitions are shown in dotted lines in Figure 1. A reconfiguration cost is incurred in this case, making the immediate reward equal to $\alpha(\phi_0) - \beta(D)$.

The new process $\mathcal{M}'$ is a discrete-space, discrete-time Markov process with rewards and two alternatives per state, and we can use the policy-iteration algorithm[10] to obtain an optimal policy off-line and cache its decisions. The optimal policy decisions can then be applied to a real network environment in the following way. Consider a network with traffic matrix $\mathbf{T}$ operating under WLA $\mathcal{R}$. Let $\mathbf{T}'$ be the new traffic matrix and $\mathcal{R}'$ be the WLA

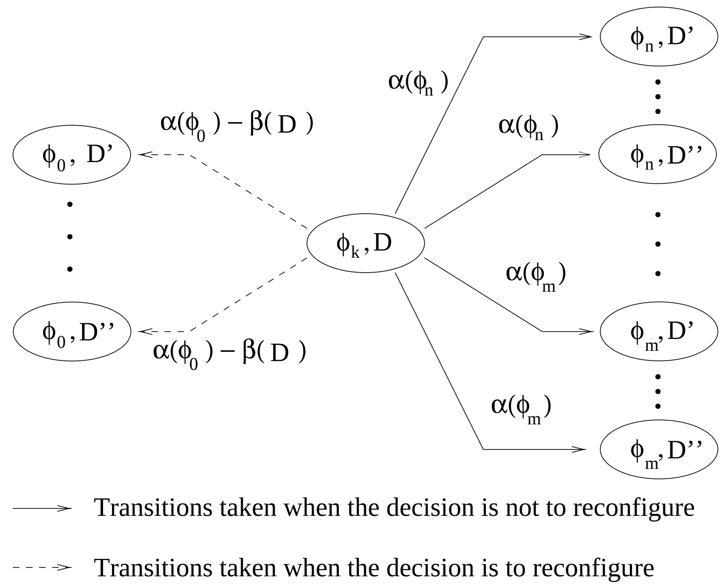

Figure 1. Transitions and rewards out of state (ϕ_k, D) of process $\mathcal{M}'$ under the two decision alternatives (*Note:* the labels along the transitions represent rewards, *not* transition probabilities)

constructed by algorithm $GLPT$[7] with $\mathcal{R}$ and $\mathbf{T}'$ as inputs. Let also $D = \mathcal{D}(\mathcal{R}, \mathcal{R}')$ be the number of receivers that need to be retuned to obtain WLA $\mathcal{R}'$ from WLA $\mathcal{R}$. To determine whether the network should reconfigure to the new WLA $\mathcal{R}'$, let $\phi(\mathcal{R}, \mathbf{T})$ be the current DLB for the network, and suppose that $\phi(\mathcal{R}, \mathbf{T})$ falls within the k-th interval, $0 \leq k \leq K$. By definition of the Markov process $\mathcal{M}'$, the current network state is modeled by state (ϕ_k, D) of this process. If, under the optimal policy, the decision associated with this state is to reconfigure, then the network must make a transition to the new WLA $\mathcal{R}'$; otherwise, the network will continue operating under the current WLA $\mathcal{R}$.

We note that the discrete-space Markov process (ϕ_k, D) is an approximation of the continuous-space process (ϕ, D), since, as discussed above, in general the DLB ϕ is a real number between 0 and $C - 1$. We also note that as the number of intervals $K \to \infty$, the discrete-state process approaches the continuous-state one. Therefore, we expect that as the number of intervals K increases, the accuracy of the approximation will also increase and the decisions of the optimal policy obtained through the process (ϕ_k, D) will "converge". This issue will be discussed in more detail in the next section, where numerical results to be presented will show that the decisions of the optimal policy "converge" for relatively small values of K. This is an important observation since the size of the state space of Markov process $\mathcal{M}'$ increases exponentially with K. By using a relatively small value for K we can keep the state space of the process to a reasonable size, making it possible to apply the policy-iteration algorithm.[10]

4. NUMERICAL RESULTS

In this section we demonstrate the properties of the optimal policies obtained by applying the policy-iteration algorithm[10] to the Markov decision process developed in the previous section. We also show how the optimal policy is affected by the choice of reward and cost functions, and we compare the long-term reward acquired by the network when the optimal policy is employed to the reward acquired by other policies. All the results presented in this section are for the approximate Markov process $\mathcal{M}'$ with state space (ϕ_k, D).

In this study, we consider a *near-neighbor* traffic model †. More specifically, we make the assumption that, if the network currently operates with a DLB equal to ϕ_k and no reconfiguration occurs, the next transition is more likely

†Other traffic models, including one derived experimentally from a client-server communication pattern, have been considered.[11] Although the optimal policy decisions obviously depend on the actual traffic patterns in the network, the overall results regarding the convergence of the optimal policy, the effects of different reward and cost functions, and the comparison to other policies are very similar to those shown here for the near-neighbor model.

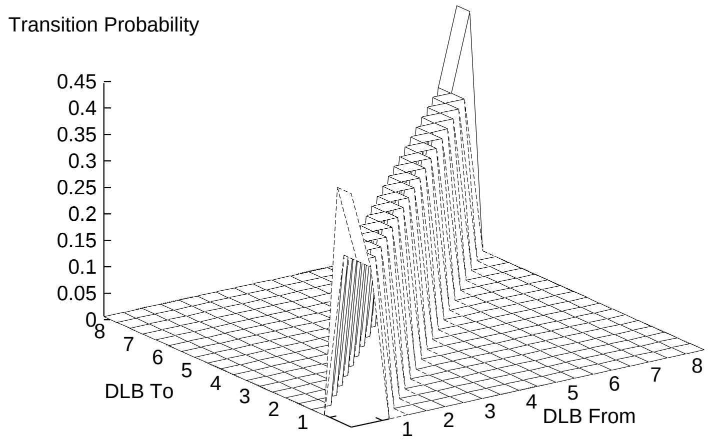

Figure 2. Near-neighbor model for $K = 20$

to take the network to the same DLB or its two nearest neighbors ϕ_{k-1} and ϕ_{k+1}, than to a DLB further away from ϕ_k. Specifically, we assume that

$$P[\phi_l \mid \phi_k] = \begin{cases} 0.3, & k = 1, \cdots, K-1, \; l = k-1, k, k+1 \\ 0.1/(K-2), & k = 1, \ldots, K-1, \; l \neq k-1, k, k+1 \\ 0.45, & k = 0, l = 1 \text{ or } k = K, l = K-1 \\ 0.1/(K-2), & k = 0, l = 2, \cdots, K \text{ or } k = K, l = 0, \cdots, K-2 \end{cases} \tag{5}$$

This traffic model is illustrated in Figure 2 which plots the conditional probability $P[\phi_k \mid \phi_l]$ that the next DLB will be ϕ_l given that the current DLB is ϕ_k, for $K = 20$ intervals. The near-neighbor model captures the behavior of networks in which the traffic matrix $\mathbf{T}$ changes slowly over time and abrupt changes in the traffic pattern have a low probability of occurring.

Given the probabilities in (5), we let the transition probability, *when no reconfiguration occurs*, from state (ϕ_k, D) to state (ϕ_l, D') be equal to:

$$P[(\phi_l, D') \mid (\phi_k, D)] = P[\phi_l \mid \phi_k] \, p_{D'} \tag{6}$$

where $p_{D'}$ is the probability that D' retunings will be required in the next reconfiguration. The probabilities p_D where measured experimentally, and we also observed that the probability that random variable D takes on a particular value is independent of the DLB ϕ_k, thus the expression (6).

We note that we need to obtain two different transition probabilities out of each state,[10] one for each of the two possible options: the do-not-reconfigure option and the reconfigure option. The above discussion explains how to obtain the transition probability matrix for the do-not-reconfigure option. The transition probability matrix for the reconfigure option is easy to determine since we know that regardless of the value ϕ_k of the current state, the next state will always be a balanced state, i.e., its DLB will be ϕ_0. The individual transition probabilities from a state (ϕ_k, D) to a state (ϕ_l, D') are then obtained by making the same assumption that all values of D have an equal probability of occurring. Therefore, the transition probabilities under the reconfigure option are:

$$P[(\phi_l, D') \mid (\phi_k, D)] = \begin{cases} p_{D'}, & l = 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

4.1. Convergence of the Optimal Policy

Let us consider the following reward and cost functions

$$\alpha[(\phi_k, D)] = \frac{A}{1 + \phi_k}, \qquad \beta(D) = BD \tag{8}$$

where A and B are weights assigned to the rewards and costs. In general, it is desirable to select rewards and costs that reflect performance measures such as throughput, delay, packet loss, or the control resources involved in receiver retuning. For example, a reward function of the form $A/(1+\phi)$ may, depending on the value of parameter A, capture either the throughput or average packet delay experienced while the network operates with a DLB equal to ϕ. On the other hand, using a cost which is proportional to the number $D = \mathcal{D}(\mathcal{R}, \mathcal{R}')$ of retunings (i.e., $\beta(\mathcal{D}(\mathcal{R}, \mathcal{R}')) = BD$) can account for the control requirements for retuning the receivers, or for the data loss incurred during reconfiguration. Furthermore, parameter B can be chosen based on which of the retuning strategies discussed in Section 2.1 is employed. Thus, network designers can select in a unified fashion appropriate rewards and costs to achieve the desired balance among the various performance criteria of interest.

We apply Howard's algorithm[10] to a network with $N = 20$ nodes and $C = 5$ wavelengths with a near-neighbor traffic model similar to the one shown in Figure 2. Our objective is to study the effect that the number of intervals K in the range $[0, C-1]$ of possible values of DLB ϕ has on the decisions of the optimal policy. As we mentioned in Section 3.2, we expect the decisions of the optimal policy to "converge" as $K \to \infty$. More formally, let φ be a real number such that $0 \leq \varphi \leq C-1$, and let k_K be the interval in which φ falls when the total number of intervals is K. Also let $d^{(K)}[(\phi_{k_K}, D)]$ be the decision of the optimal policy for state (ϕ_{k_K}, D) of Markov process $\mathcal{M}'$ when the number of intervals is K. We will say that the decisions of the optimal policy converge if

$$\lim_{K\to\infty} d^{(K)}[(\phi_{k_K}, D)] \quad = \quad d[(\varphi, D)] \qquad \forall\ \varphi,\ D \tag{9}$$

In Figures 3 to 5 we plot the decisions of the optimal policy for the 20-node, 5-wavelength network with a near-neighbor traffic model, and for three different values of K; the weights used in the functions (8) were set to $A = 30$ and $B = 1$. Figure 3 corresponds to the optimal policy for $K = 20$ intervals, while in Figures 4 and 5 we increase K to 30 and 40, respectively. The histograms shown in Figures 3 to 5, as well as in other figures in this section, should be interpreted as follows. In each figure, the x axis represents the DLB ϕ_k (with a number of intervals equal to the corresponding value of K), while the y axis represents the possible values of D. The vertical bar at a particular DLB value ϕ_k has a height equal to D_k^{thr} such that:

$$d^{(K)}[(\phi_k, D)] \quad = \quad \begin{cases} \text{reconfigure,} & D \leq D_k^{thr} \\ \text{do not reconfigure,} & D > D_k^{thr} \end{cases} \tag{10}$$

In other words, *for each value of* ϕ_k, there exists a *retuning threshold* value D_k^{thr} such that the decision is to reconfigure when the number of receivers to be retuned is less than D_k^{thr}, and not to reconfigure if it is greater than D_k^{thr}. Since the optimal policy had similar behavior for all the different reward and cost functions we considered, its decisions will be plotted as a histogram similar to those in Figures 3 to 5 [‡].

As we can see in Figures 3 to 5, the decisions of the optimal policy do converge (in the sense of expression (9)) as K increases. For instance, let us consider a DLB of 1, which falls in the fourth interval when $K = 20$ (in Figure 3), the sixth interval when $K = 30$ (in Figure 4), and the seventh interval when $K = 40$ (in Figure 5). In all three cases, the retuning threshold is equal to 9 for these intervals, therefore, the decisions of the optimal policy for the three values of K are the same. On the other hand, for a DLB of 2, the retuning threshold is 14 in Figure 3, but it drops to 13 in Figure 4, same as in Figure 5. In other words, for a DLB of 2, the decisions of the optimal policy are different when $K = 20$ than when $K = 30$ or 40 (in the former case, the decision is to reconfigure as long as the number of retunings is at most 14, while in the latter the decision is to reconfigure only when the number of retunings is at most 13). But the important observation is that the policy decisions do not change when the number K of intervals increases from 30 to 40, indicating convergence. In fact, there are no changes in the optimal policy for values of K greater than 40 (not shown here). We have observed similar behavior for a wide range of values for the weights A and B, for different network sizes, as well as for other reward and cost functions. These results indicate that a relatively small number of intervals is sufficient for obtaining an optimal policy.

Another important observation from Figures 3 to 5 is that the retuning threshold increases with the DLB values. This behavior can be explained by noting that, because of the near-neighbor distribution (refer to Figure 2), when

[‡]That the optimal policy was found to be a threshold policy (with a possibly different retuning threshold) for each value of ϕ_k, can be explained by the fact that we only consider cost functions that are non-decreasing functions of random variable D. As a result, if the decision of the optimal policy for a state (ϕ_k, D_1) is not to reconfigure, intuitively one expects the decision for state (ϕ_k, D_2), where $D_2 > D_1$ to also be not to reconfigure since the reconfiguration cost $\beta(D_2)$ for the latter state would be at least as large as the reconfiguration cost $\beta(D_1)$ for the former.

the network operates at states with high DLB values, it will tend to remain at states with high DLB values. Since the reward is inversely proportional to the DLB value, the network incurs small rewards by making transitions between such states. Therefore, the optimal policy is such that the network decides to reconfigure even when there is a large number of receivers to be retuned. By doing so, the network pays a high cost, which, however, is offset by the fact that the network makes a transition to the balanced state with a low DLB, reaping a high reward. On the other hand, when the network is at states with low DLB, it also tends to remain at such states where it obtains high rewards. Therefore, the network is less inclined to incur a high reconfiguration cost, and the retuning threshold for these states is lower.

4.2. Comparison to Threshold Policies

In this section we compare the optimal policy against a class of policies which we will call *DLB-threshold policies.* With such a policy, there exists a threshold DLB value ϕ_{max} such that, if the system is about to make a transition into a state $(\phi_k, D), \phi_k > \phi_{max}$, then the network will reconfigure and make a transition to a state with DLB ϕ_0, regardless of the reconfiguration cost involved. Otherwise, no reconfiguration occurs. This class of policies is not concerned with the reconfiguration cost incurred. Instead it ensures that the traffic carrying capacity of the network will never fall below the value $\gamma_{min} = C/(1 + \phi_{max})$.

We note that the DLB-threshold policies define Markov processes which are *outside* the class of Markovian Decision Processes considered in Section 3. In a Markovian Decision Process, there are several alternatives per state, but once an alternative has been selected for a state, then transitions from this state are always governed by the chosen alternative (refer also to Figure 1). In a DLB-threshold policy, on the other hand, the alternative selected does not depend on the *current* state, but rather on the *next* state. Therefore, the system may select different alternatives when at a particular state, depending on what the next state is. Since Howard's algorithm[10] is optimal only within the class of Markovian Decision Processes, it is possible that these threshold policies obtain rewards higher than the optimal policy determined by the algorithm.

All the results presented in this section are for a network with $N = 100$ nodes, $C = 20$ wavelengths, a near-neighbor traffic model, and $K = 20$ intervals. The reward and cost functions considered are those in expression (8), and we used $A = 50$ and $B = 1$ as the values for the weights in the reward and cost functions, respectively. In Figure 6 we compare the optimal policy to a number of DLB-threshold policies, each with a different threshold value. The figure plots the average long-term reward acquired by each of the policies against the retuning threshold ϕ_{max}. The horizontal line corresponds to the reward of the optimal policy, which, clearly, is independent of the retuning threshold. Each point of the second line in the figure corresponds to the reward of a DLB-threshold policy with the stated threshold value.

The most interesting observation from Figure 6 is that, for certain values of the DLB-threshold, the DLB-threshold policy achieves a higher reward than the optimal policy obtained through Howard's algorithm. This result is possible because, as we discussed earlier, the class of DLB-threshold policies is more general than the class of policies for which Howard's algorithm is optimal. On the other hand, we note that the reward of the DLB-threshold policy depends strongly on the DLB threshold used. Although within a certain range of these values the threshold policies perform better than the optimal policy, the latter outperforms the former for most threshold values. Therefore, threshold selection is of crucial importance for the threshold policies, but searching through the threshold space can be expensive. The optimal policy, however, guarantees a high overall reward and is also simpler to implement since the network does not need to *look ahead* to the next state to decide whether or not to reconfigure. We have also found that for other reward and cost functions, or for different reward and cost weights, the optimal policy is strictly better than DLB-threshold policies regardless of the threshold value.

These results demonstrate that DLB- or two-threshold policies do not always perform better than the optimal policy, and their performance depends on the system parameters and/or the reward and cost functions. Furthermore, it is not possible to know ahead of time under what circumstances the threshold policies will achieve a high reward. Equally important, if the network's operating parameters change, threshold selection must be performed anew.

Overall, we have found that the optimal policy obtained through Howard's algorithm can successfully balance the two conflicting objectives, namely the DLB and the number of retunings, and always achieves a high reward across the whole range of the network's operating parameters. On the other hand, pure threshold policies, although they can sometimes achieve high reward, they are less flexible, and they introduce an additional degree of complexity, namely, the problem of threshold selection.

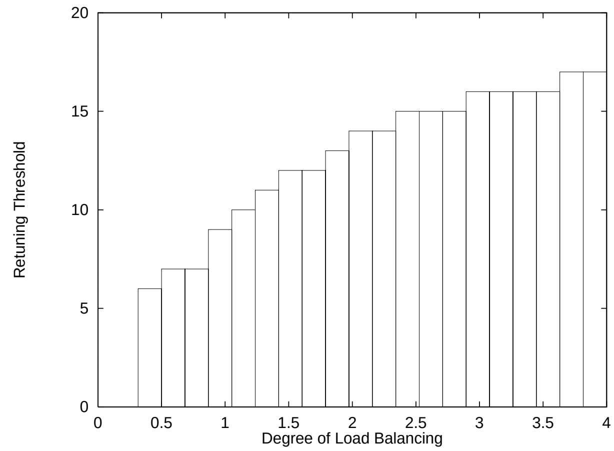

Figure 3. Optimal policy decisions for $N = 20$, $C = 5$, $K = 20$, $A = 30$, $B = 1$

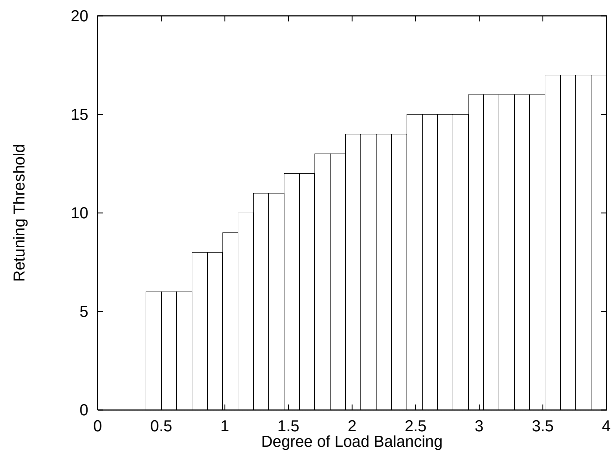

Figure 4. Optimal policy decisions for $N = 20$, $C = 5$, $K = 30$, $A = 30$, $B = 1$

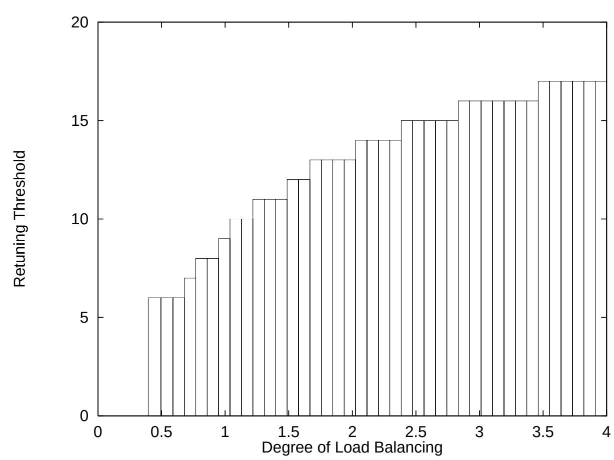

Figure 5. Optimal policy decisions for $N = 20$, $C = 5$, $K = 40$, $A = 30$, $B = 1$

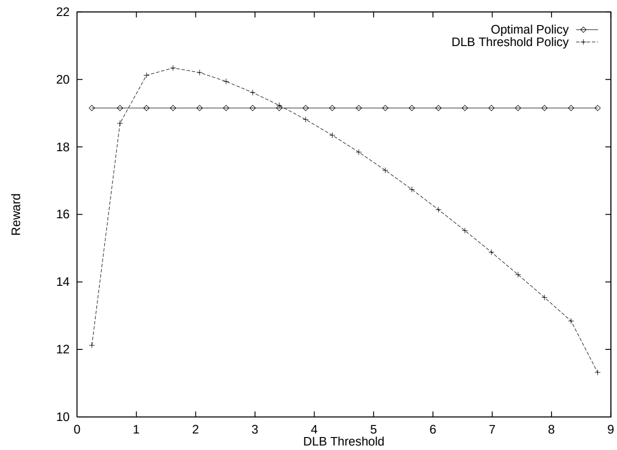

Figure 6. Policy comparison, $N = 100$, $C = 10$, $K = 20$, $A = 50$, $B = 1$

5. CONCLUDING REMARKS

We have studied the problem of reconfiguring broadcast multiwavelength optical networks so as to ensure that the traffic load remains balanced across the WDM channels under changing traffic conditions. We have used Markov Decision Process theory to obtain optimal reconfiguration policies. The formulation presented in this paper provides a unified framework for reconfiguration problems in optical networks, and provides further insight into the fundamental tradeoffs involved in the design of reconfiguration policies.

REFERENCES

1. J.-F. P. Labourdette, "Traffic optimization and reconfiguration management of multiwavelength multihop broadcast lightwave networks," *Computer Networks & ISDN Systems* , 1998. To appear.
2. J.-F. P. Labourdette and A. S. Acampora, "Logically rearrangeable multihop lightwave networks," *IEEE Transactions on Communications* **39**, pp. 1223–1230, August 1991.
3. J.-F. P. Labourdette, F. W. Hart, and A. S. Acampora, "Branch-exchange sequences for reconfiguration of lightwave networks," *IEEE Transactions on Communications* **42**, pp. 2822–2832, October 1994.
4. G. N. Rouskas and M. H. Ammar, "Dynamic reconfiguration in multihop WDM networks," *Journal of High Speed Networks* **4**(3), pp. 221–238, 1995.
5. B. Mukherjee, "WDM-Based local lightwave networks Part I: Single-hop systems," *IEEE Network Magazine* , pp. 12–27, May 1992.
6. V. Sivaraman and G. N. Rouskas, "HiPeR-ℓ: A High Performance Reservation protocol with ℓook-ahead for broadcast WDM networks," in *Proceedings of INFOCOM '97*, pp. 1272–1279, IEEE, April 1997.
7. I. Baldine and G. N. Rouskas, "Dynamic load balancing in broadcast WDM networks with tuning latencies," in *Proceedings of INFOCOM '98*, pp. 78–85, IEEE, March 1998.
8. H. G. Perros and K. M. Elsayed, "Call admission control schemes: A review," *IEEE Communications Magazine* **34**(11), pp. 82–91, 1996.
9. G. N. Rouskas and V. Sivaraman, "Packet scheduling in broadcast WDM networks with arbitrary transceiver tuning latencies," *IEEE/ACM Transactions on Networking* **5**, pp. 359–370, June 1997.
10. R. A. Howard, *Dynamic Programming and Markov Processes*, M.I.T. Press, Cambridge, 1960.
11. I. Baldine, *Dynamic Reconfiguration in Broadcast WDM Networks.* PhD thesis, North Carolina State University, Raleigh, NC, August 1998.