Dynamic Routing and Wavelength Assignment Using First Policy Iteration

Esa Hyytiä and Jorma Virtamo

Helsinki University of Technology
Laboratory of Telecommunications Technology
P.O.Box 3000, FIN-02150 HUT, Finland
E-mail: {esa.hyytia, jorma.virtamo}@hut.fi

December 10, 1999

Abstract

With standard assumptions the routing and wavelength assignment problem (RWA) can be viewed as a Markov Decision Process (MDP). The problem, however, defies an exact solution because of the huge size of the state space. Only heuristic algorithms have been presented up till now. In this paper we propose an approach where, starting from a given heuristic algorithm, one obtains a better algorithm by the first policy iteration. In order to estimate the relative costs of states, we make a simulation on the fly studying, at each decision epoch, the consequences of all the alternatives actions. Being computationally intensive, this method can be used in real time only for systems with slow dynamics. Off-line it can be used to assess how close the heuristic algorithms come to the optimal policy. Numerical examples are given about the policy improvement.

1 Introduction

The wavelength division multiplexing (WDM) is a promising technology for future all-optical networks. In WDM several optical signals using different wavelengths share same fibre. The capacity of such fibre links can be very huge, even terabits per second. The routing in network nodes is based on wavelengths of incoming signals [6][8][10].

Generally, the routing and wavelength assignment (RWA) problem in WDM networks consists of choosing a route and a wavelength for each connection so that no two connection using same wavelength share same fibre [1][2]. For example a simple form of RWA problem is a static traffic case with single fibre links. If the nodes are incapable to do a wavelength translations, an assumption made throughout this work, the problem can be mapped to a node coloring problem once routing is fixed (see e.g. [1]).

When the traffic is not static, lightpath requests arrive randomly following some traffic pattern. Connection requests between a given source destination pair constitute a *traffic class*, which we index by k, $k \in \mathcal{K}$, where $\mathcal{K}$ is the set of all source destination pairs. The RWA algorithm configures the lightpaths in the network unless there is no enough resources available and the request is blocked (see e.g. [3][5][7]).

The possible schemes considered under dynamical traffic can be divided in two cases. If it is possible to reconfigure the whole network when blocking would occur the blocking probability can be considerable reduced. Such an operation, however, interrupts all (or at least many) active lightpaths and requires a lot of coordination between all the nodes. In large networks the reconfiguration seems quite impossible. In any case the reconfiguration algorithm should try to minize the number of reconfigured lightpaths [4].

The other case is when active lightpaths may not be reconfigured. In this case it is important which route and wavelength are assigned to incoming connection requests in order to minimize the future congestion in the network.

Several heuristic algorithms have been proposed and studied (see e.g. [3][5][7]). In this paper we study this problem in the setting of Markov Decision Processes (MDP) and propose a new approach, where we try to improve any given heuristic algorithm by the first policy iteration [9][11]. The policy iteration, indeed, is known to lead to a new policy with better performance. In order to avoid dealing with huge size of the state space in calculating the relative state costs needed in the policy improvement step, we suggest to estimate these costs on the fly by simulations for the limited set of states that are relevant at any given decision epoch, i.e. when the route and wavelength assignment for an arriving call has to be made.

The rest of the paper is organized as follows. In section 2 we briefly review Markov Decision Processes and policy iteration in general, and the first policy iteration, in particular. In section 3, we consider the relative costs of states and how they are used in the policy iteration, and in the following section 4 we study how theses state costs can be estimated by simulations. Different heuristic RWA algorithms are presented in section 5. These are used as a starting point for policy iteration, and, in section 6 some numerical results obtained by simulations are presented. Finally, section 7 contains conclusions.

2 Policy Iteration

Routing and wavelength allocation constitute a typical decision making problem. When certain events occur, one has to decide on some action. In the RWA problem, in particular, upon arrival of a request for new connection one has to decide whether or not to accept the request, and if accepted which resources to allocate for it, i.e. which of available routes and wavelengths are used for that connection.

In general, one is interested in the optimal policy which maximizes or minimizes the expectation (infinite time horizon) of a given objective function. Here we assume that the objective is defined in terms of minimizing some cost function. The cost may represent e.g. the loss of revenue due to blocked calls, where different revenue may be associated to each type of call.

When the arrival process of type k calls is a Poisson process with intensity λ_k, the holding times of those calls are distributed exponentially with mean $1/\mu_k$ and the expected revenue per carried call is w_k, then the system constitues a Markov Process and the problem of determining the optimal policy belongs to the class of Markov Decision Processes (MDP) described e.g. in [9] and [11].

Three main approaches for solving the optimal policy in the MDP setting are the policy

iteration, value iteration and linear programming approach. In this paper, we concentrate on the iteration in the policy space, where, as the name says, one tries to find the optimal policy by starting from some policy and iteratively improving it. This policy iteration is known to converge rather quickly to the optimal policy. Even the first iteration yields a new policy which is rather close to the optimal one. In practice, it is seldom possible to go beyond the first iteration. Also in this work, we will restrict ourselves to the first policy iteration.

At each decision epoch, i.e. arrival of a new request, there is a finite set of possible actions: either reject the call or accept it and assign a feasible combination of route and wavelength (RW) to it. A feasible RW combination is such that along the route from the source to destination the wavelength is not being used on any of the links. If no feasible RW combination exists, the call is unconditionally rejected.

A *policy* defines for each possible state of the system and for each class k of an arriving call which of the possible actions is taken. Many heuristic policies have been proposed in the literature such as the first-fit wavelength and most-used wavelength policies combined with shortest path routing or near shortest path routing. Some of them work reasonably well. Common to all heuristic policies is that they are simple. The choice of the action to be taken at each decision epoch can usually be described in simple terms and does not require much computation. We take one of the heuristic politics as a starting point and call it the *standard policy*. The policy resulting from the first policy iteration we refer to as the *iteration policy*.

By doing the first policy iteration we have two goals in mind. 1) Finding a better RWA algorithm which, being computationally intensive, may or may not be calculable in real time, depending on the time scale of the dynamics of the system. 2) Even in the case the algorithm is not calculable in real time, estimating how far the performance of a heuristic algorithm is from the optimal one.

Briefly, as explained in more detail below, our idea in the policy iteration is the following: at each decision epoch we make a decision analysis of all the alternative actions. For each of the possible actions, i.e. decision alternatives, we estimate the future costs by simulation. Thus, assuming that a given action is taken we let the system proceed from the state where it is after that action and use the standard policy to make all the subsequent decisions. The iteration policy is the policy which is obtained when at each decision epoch the action is chosen for which the estimated cost is the minimum. It can be shown that the iteration policy is always better or at least as good a policy as the standard policy, and as said it often comes rather close to the optimal policy.

3 Relative costs of states

In the MDP theory, the first policy iteration consists of the following steps: With the standard policy one solves the Howard equations (see, e.g. [9][11]) to give the so called relative costs of the states, C_i, which for each possible state i of the system describe the difference in the expected cumulative cost from time 0 to infinity, given that the system starts from state i rather than from the equilibrium. Then, given that the current state of the system is j and a class-k call is offered, one calculates the cost $C_j + w_k$ for the action that the call is rejected, and the cost C_i, $i \in \mathcal{A}(j,k)$, for the case the call is accepted, where $\mathcal{A}(j,k)$ is the set of states reachable from state j by assigning call-k a feasible RW pair.

By choosing always the action which minimizes the cost, one gets the iteration policy, i.e. the policy resulting from the first policy iteration.

Though the Howard equations are just a set of linear equations for relative costs C_i and the average cost rate c of the standard policy (see below), their solution cannot be obtained because of the prohibitive size of the state space for any realistic system. However, at any decision epoch the relative costs C_i are needed only for the current state j and a small set of states $\mathcal{A}(j,k)$ reachable from the current state. We propose to estimate these values on the fly by means of simulations. To this end, it is useful to consider the physical interpretation of the relative costs C_i.

Given that the system starts from state i at time 0 and standard policy is applied for all decisions, the cumulative costs are accrued at the expected rate $c_t(i)$ at time t,

$$c_t(i) = \sum \lambda_k w_k \mathrm{P}\{I_t \in \mathcal{B}_k | I_0 = i\}, \tag{1}$$

i.e. the expected rate of lost revenue, where $\mathrm{P}\{I_t \in \mathcal{B}_k\}$ is the probability that at time t the state of the system I_t is a blocking state for class-k calls. When $I_t \in \mathcal{B}_k$ class-k calls arriving at time t are blocked by the standard policy because either no feasible RW pair exists or the policy otherwise deems the blocking to be advantageous in the long run. The expected cost rate $c_t(i)$ depends on the initial state i. However, no matter what the initial state is, as t tends to infinity, the expected cost rate tends to a constant c, which is specific to the standard policy, and corresponds to (1) with steady state blocking probabilities $\mathrm{P}\{I_t \in \mathcal{B}_k\}$.

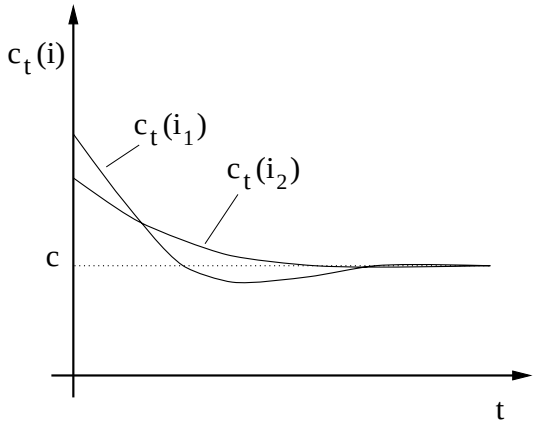

Figure 1: Expected costs with different initial choices as a function of time.

The behavior of the function $c_t(i)$ is depicted in Figure 1 for two different initial values i_1 and i_2. The relative costs C_i is defined as the integral

$$C_i = \int_0^\infty (c_t(i) - c)\, dt,$$

i.e. the area between the curve $c_t(i)$ and the line at level c. So we are interested in the transient behaviour of $c_t(i)$; after the transient no contribution comes to integral. The length of the transient is of the order $1/\mu$, where $1/\mu$ is the average holding time of a connection. After that the system essentially forgets the information about the initial state. So we can restrict ourselves to an appropriately chosen finite interval $(0, T)$. The actual choice of T is a tradeoff between different considerations as will be discussed later.

One easily sees that in the policy improvement step only the differences of the values C_i between different states are important. Therefore, we can neglect the c in the integral, as

it is common to all states, and end up for thus redefined C_i,

$$C_i \approx C_i(T) = \int_0^T c_t(i)\, dt, \tag{2}$$

which is simply the expected cumulative cost in interval $(0, T)$ starting from initial state i.

4 Estimation of the state costs by simulation

In practice, it is not feasible to calculate the cost rate function $c_t(i)$ analytically even for the simplest policies. Therefore, we estimate the state costs C_i by simulations. In each simulation the system is initially set in state i and then the evolution of the system is followed for the period of length T, making all the RWA decisions according to the standard policy.

4.1 Statistics collection: blocking time vs. blocking events

In collecting the statistics one has two alternatives. Either one records the time intervals when the system is in a blocking state of class-k calls, for all $k \in \mathcal{K}$. If the cumulative time within interval $(0, T)$ when the system is in the blocking state of class-k calls is denoted by $\tau_k(i)$, then the integral is simply

$$\hat{C}_i = \sum \lambda_k w_k \tau_k(i). \tag{3}$$

Alternatively, one records the number $\nu_k(i)$ of blocked calls of type k in interval $(0, T)$. Then we have

$$\hat{C}_i = \sum w_k \nu_k(i). \tag{4}$$

In these equations we have written explicitly $\tau_k(i)$ and $\nu_k(i)$ in order to emphasize that the system starts from the state i. Both (3) and (4) give an unbiased estimate for C_i. In either case, the simulation has to be repeated a number of times in order to get an estimator with small enough confidence interval.

To this, denote the estimates of future costs obtained in the jth simulation run by $\hat{C}_i^{(j)}$, using (3) or (4) as the case may be. Then our final estimator for C_i is

$$\hat{C}_i = \frac{1}{N} \sum_{j=1}^{N} \hat{C}_i^{(j)}, \tag{5}$$

where N is the number of simulation runs. In fact, for the policy improvement the interesting quantity is the difference

$$E_{i_1,i_2} = C_{i_2} - C_{i_1},$$

for which we have the obvious estimate

$$\hat{E}_{i_1,i_2} = \hat{C}_{i_2} - \hat{C}_{i_1}. \tag{6}$$

From the samples $\hat{C}_{i_1}^{(j)}$ and $\hat{C}_{i_2}^{(j)}$, $j = 1, \ldots, N$, we can also derive an estimate for the variance $\hat{\sigma}_{i_1,i_2}^2$ of the estimator $\hat{E}_{i_1,i_2}$

$$\hat{\sigma}_{i_1,i_2}^2 = \frac{N \sum_j (\hat{C}_{i_2}^{(j)} - \hat{C}_{i_1}^{(j)})^2 - \left(\sum_j \hat{C}_{i_2}^{(j)} - \hat{C}_{i_1}^{(j)} \right)^2}{N^2(N-1)}.$$

The choice between the alternative statistics collection methods is based on technical considerations. Though estimator (3) (blocking time) has a lower variance per one simulation run, it requires much more bookkeeping and the variance obtained with a given amount of computational effort may be lower for estimator (4) (blocking events).

In practice, when the blocking time is collected, we will have a binary state vector $\mathbf{b}$ with the kth component equal to one if the network is in the blocking state for class-k calls, and otherwise zero. Every time an arrival or departure occurs, the state vector $\mathbf{b}$ multiplied by the time elapsed since the last event is added to the vector $\boldsymbol{\tau}$, which contains the cumulative time spent in blocking state for each class k. Correspondingly, the components of $\mathbf{b}$ are updated, i.e. when a new connection is accepted, we have to check whether any of the previously non-blocked connection classes, turn blocked. Likewise, after a departure some previously blocked connection classes may turn to be non-blocked. This updating means that each accepted connection causes K checks on the average, where K is number of possible connections (node pairs), i.e. the number of elements in the set $\mathcal{K}$. This extra effort leads to longer running times.

4.2 Policy iteration with uncertain state costs

In order to deal with uncertainty of the estimators $\hat{C}_i$, we do not blindly accept the action with the smallest estimated cost, but give a special status for the decision which would be chosen by the standard policy. Let us give this policy the index 0. Based on the simulations we form estimates $\hat{E}_{0,i}$ for each possible action i. Then, as the decision we choose the action which minimizes the quantity

$$\hat{E}_{0,i} + \alpha \cdot \hat{\sigma}_{0,i}, \tag{7}$$

where α is an adjustable parameter. Note that for $i = 0$ this quantity is equal to 0. Thus, in order for another action i to replace action 0 of the standard policy, we must have $\hat{E}_{0,i} < -\alpha \cdot \hat{\sigma}_{0,i}$, i.e. we require a minimum level of confidence for the hypothesis $C_i < C_0$. An appropriate value for α has to be determined experimentally.

The important parameters of the simulation now are the length of the simulation period T and the number of simulation runs N used for the estimation of each C_i. In practice, we are interested in the smallest possible values of T and N in order to minimize the simulation time. However, making T and N too small increases the simulation noise, i.e. error in the estimates for C_i, occasionally leading to decisions that differ from that of the true iteration policy, consequently deteriorating the performance of the resulting algorithm.

4.3 Time complexity of iteration approach

Clearly the simulation of the future at each decision epoch makes this algorithm very time consuming. Assume, that a single decision of the standard policy takes a constant time u. Let N be the number of the simulations that are run for each alternative action, A average number of alternative actions per decision (possible RW pairs), λ the total arrival rate to the network (uniform load), and T the length of one simulation run. Then, the running time of each decision is on the average

$$u_i = A \cdot N \cdot (\lambda T)u = \lambda ANT \cdot u,$$

so the running time is λANT times longer than with the underlying algorithm. Neither λ nor A are parameters of the algorithm. Hence, the tradeoff between the goodness of solution and the running time is defined by choosing the value for product $N \cdot T$.

For example, to get decent results with a simple 11 node network (fig. 2) with moderate load ($\mu = 1$ and $\lambda_k = 0.4$ for all k), about 100 samples were required each $1/\mu$ time units long. So the increase in running time was order of $10^3 - 10^4$. It is essential that the decisions of the underlying standard policy can be determined quickly.

5 Heuristic Algorithms

Several quick heuristic algorithms have been proposed in the literature. Here we briefly present some of them and study how iteration approach works with them. The first set of algorithms assumes that a fixed set of possible routes for each connection is given in advance. Some papers refer to this as alternate routing. In practice this set usually consists shortest or nearly shortest path of routes. Each algorithm accepts the first feasible RW pair found (first-fit).

- *basic* algorithm goes through all the routes in a fixed order and for each route tries all the wavelengths in a fixed order.
- *porder* algorithm is similar to *basic*-algorithm but it goes through all the wavelengths in a fixed order and for each wavelength tries all the routes in a fixed order.
- *pcolor* algorithm works like *porder* but wavelengths are gone through in order of the usage instead of a fixed order, so that the most used wavelength is tried first.
- *lpcolor* algorithm also tries to pack colors, but the primary target is to minimize the number of used links. So the algorithm first tries the most used wavelength with all the shortest routes, then the next often used wavelength and so on. If no wavelength works, the set of routes is expanded to include routes having one link more and wavelengths are tried again in the same order.
- *ll* or least loaded algorithm (see [3] is similar to *pcolor* but here the chosen RW pair is the one which leaves most capacity free after the assignment, i.e. the minimal number of free wavelengths over the links used is maximized.

Another set of heuristic algorithms, adaptive unconstrained routing (AUR) algorithms, are described in [5]. These use dynamic routing instead of fixed set of routes, and are thus a little bit slower.

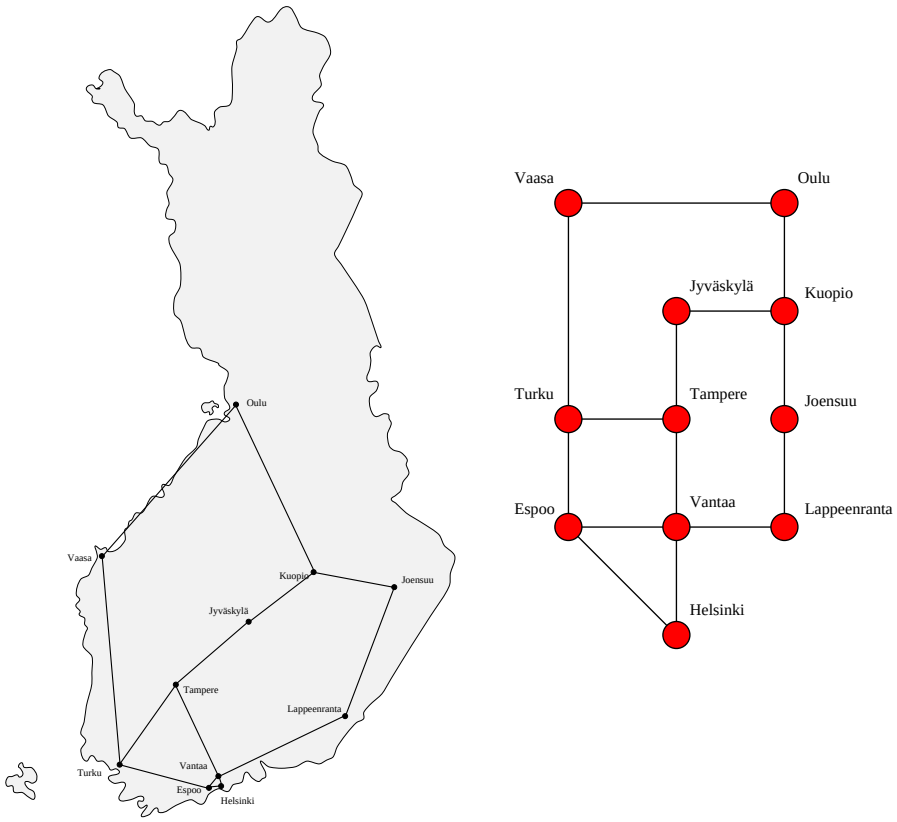

Figure 2: Hypothetical WDM-network residing in Finland.

- *aurpack* is similar to *pcolor*, but without the limitations of a fixed set of routes.
- *aurexhaustive* finds a route with each wavelength (if possible) and chooses the shortest among them, i.e. it is identical to *lpcolor* except that the set of possible routes is not limited.

Thus AUR-algorithms will search for a free route dynamically based on the current state of the network. So there is no need to store possible routes (which without any limitations can form a very large set) in advance.

Also other heuristics are given in [5] like *random* (tries wavelengths in random order) or *spread* (tries least used wavelength first), but they were reported to work worse than the ones described above, and are not further discussed here.

6 Simulation results

Next we will present some numerical results from simulations. All tests were run for the small network shown in figure 2. The network was assumed to have 8 wavelengths available on each link. All the links contained one fibre. The offered load was uniform among all traffic classes (node pairs) and each rejected call represents an equal cost, $w_k = 1$ for all $k \in \mathcal{K}$. These assumptions simplify formulas (3) and (4). Note that the assumption $w_k = 1$ for all k means that the the objective is to minimize the long term blocking rate, i.e. the blocking probability.

It should also be noted that the results for the iteration policy were obtained by two levels

algorithm	time block	Small Set of Routes				Larger Set of Routes			
		running time	time index	call blocking	time blocking	running time	time index	call blocking	time blocking
basic	no	1.5	1	5.225	-	1.6	1	4.035	-
basic	yes	29	20	5.225	5.214	36	20	4.035	4.040
pcolor	no	2.1	1.5	3.741	-	2.9	1.8	3.406	-
pcolor	yes	56	30	3.741	3.760	106	70	3.406	3.381
lpcolor	no	2.0	1.5	3.743	-	2.3	1.4	2.867	-
lpcolor	yes	54	30	3.743	3.715	67	40	2.867	2.920
aurpack	no	7.9	6	3.676	-	8.0	6	3.676	-
aurpack	yes	380	300	3.676	3.670	380	300	3.676	3.670

Table 1: Different heuristic algorithms compared.

of nested simulations. In order to assess the performance of the policy, an outer simulation is run, where connections arrive and leave the network and blocking times or events are recorded. Upon each arrival, a number of inner simulations are launched from the current state in order to make a comparison between different decision alternatives. Based on this comparison one alternative is chosen and used in the outer simulation, which then continues until upon the next arrival the decision analysis by the inner simulations is again started.

6.1 Heuristic algorithms

In table 1 running times and average results are given for the heuristic algorithms. The offered load per traffic class was $a = 0.4$. For non-*aur* algorithms two sets of routing parameters were used. For the smaller set of routes we had *ldelta*=1 and *rmax*=4, i.e. the routing parameters allowed routes with one more link than there are on the shortest path while still limiting the number of routes per traffic class pair to 4 at maximum. The other set of routing parameters used was *ldelta*=2 and *rmax*=16. The shown results are averages over 100000 arrivals. A warm-up period of $5/\mu$ was skipped before starting to record samples.

The column 'time block' (yes/no) indicates whether (3) or (4) was used for the cost estimation. The bookkeeping of the blocking time clearly raises the running time a lot. Thus it is unlikely that the benefit of smaller variance of cost estimates would outweight the longer running time.

6.2 Routing algorithm parameters

All non-*aur* algorithms assume that a predefined set of possible routes per traffic class is given. This raises the question, which set of routes is optimal? Clearly too small a set of routes limits how well any algorithm can perform. But, as can be seen from figure 3, also too large a set of routes can be a problem for some algorithms. Here, the set of routes is specified with two parameters *ldelta* and *rmax*. Parameter *ldelta* defines how many links longer routes than the shortest one are included in set of routes. The second parameter *rmax* limits the total number of routes, i.e. only the *rmax* first routes are included in set. For example, with *ldelta*=0 and *rmax*=10 only the shortest routes are included, and if there are more than 10 shortest routes for some node pair only the first 10 found are included.

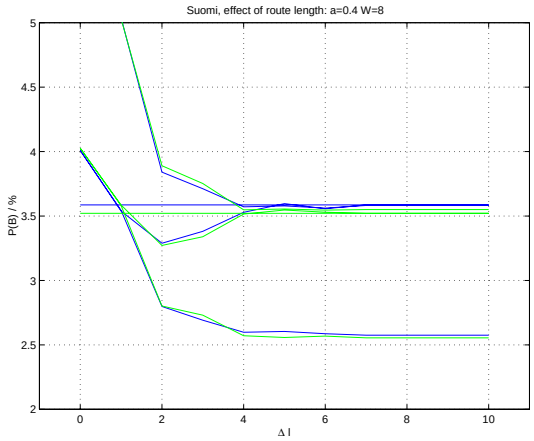

Figure 3: Effect of *ldelta* parameter on the results. At $\Delta l = 3$ the algorithms from worst to best are *basic*, *aurpack*, *pcolor* and *lpcolor*.

Third possible parameter to reduce the running time could be *maxtest*, which would be used to limit the number of alternative actions tested against the standard policy. This would be effective only when the load of the network is low and there are plenty of RW pairs available. In these tests such limit was not used.

6.3 Effect of simulation noise

Let T be the length of the transient period after which the cost rate $c_t(i)$ is probably very near to the long time average c of the standard policy. Simulating past T thus gives no new information, but actually only increases the noise resulting from the stochastic nature of the simulation. This can be seen from figure 4 where results clearly get worse as the simulation period T grows (diagrams from left to right) while keeping the number of simulation runs the same.

The figure presents blocking probability obtained with the first policy iteration in the network with a moderate load of $a = 0.4$ for each traffic class. On the x-axis of each diagram is the number of simulation runs N, i.e. samples of future costs of a given initial decision. The upper and lower rows in the figure represent results which were obtained using estimators (4) and (3) in the estimation of the state costs, respectively. The same realizations were used in both cases. The conclusion is that the longer the simulation period is, the smaller is the 'signal to noise ratio' and the more simulation runs are needed to 'recover the signal'. We cannot, however, make the simulation period T arbitrarily small, since if the whole transient period is not covered ´the signal becomes distorted'.

6.4 Iteration algorithm

The simulations were run for the same test network as was used before, i.e. the small network of figure 2. The network was assumed to have 8 wavelengths in each link and the offered load was uniform among all node pairs. Good running parameters for the inner simulations for this system were estimated from figure 4. Based on this we chose the simulation period $T = 1/\mu$ and $N = 50 \ldots 200$ simulation runs for each alternative action.

Simulations were run with quick heuristic algorithms as well as with the iteration algorithm

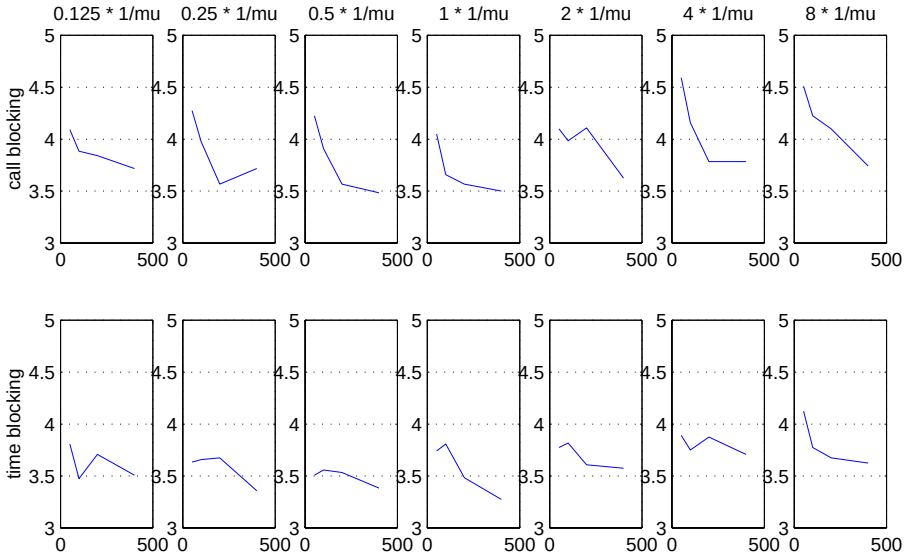

Figure 4: The effect of extending simulation period in the state cost estimation on the blocking probability of the iteration algorithm. The routing parameters are *ldelta*=0 and *rmax*=4, and *basic* is used as the standard policy. Load is $a = 0.4$ for each traffic class.

with different parameters. The resulting blocking probabilities are shown in figure 5. The upper part of the bars (light gray) represent two times the standard deviation and the mean value is in the middle of upper part. The routing parameters here were *ldelta*=1 and *rmax*=4 which clearly limit the set of routes. Bars 1-6 represent quick heuristics *basic*, *pcolor*, *porder*, *spread*, *ll* and *aurpack*. Bars 7-9 represent iteration policy with $N = 50, 100, 200$ using *basic* as the standard policy and bars 10-12 otherwise same but *pcolor* is used as the standard policy.

The improvement obtained by the first policy iteration starting with the *basic* algorithm was quite large, about 30%, while with *pcolor* the improvement is much less. Generally the results from iteration approach are always better than any of the heuristics which used same set of possible routes. The *aurpack* uses dynamic routing with routes of any length in the search space and is here only for comparison.

In another set of simulations the iteration approach was applied with different standard policies to get some idea about how important the underlying algorithm is. In the four diagrams of figure 6 the results can be seen with four different offered loads, with the blocking probability varying from quite a low to a really high value. The algorithms used were (in order) *basic*, *basic+iteration*, *pcolor*, *pcolor+iteration*, *lpcolor*, *lpcolor+iteration*, *aurexhaustive* and *aurpack*. In this simulation the routing parameters were *ldelta*=3 and *rmax*=30, so the set of routes is much larger than in previous case. This explains why *aurpack* loses its advantage over the other algorithms.

It can be seen from the figure that in each case the iteration algorithm indeed gives slightly better results, except with *lpcolor* with low load $a = 0.3$, when the blocking probability was about the same. It is also worth noting that *pcolor* performed worse than simpler *basic*, which must be due to a rather high value of *ldelta* in relation to the average route length.

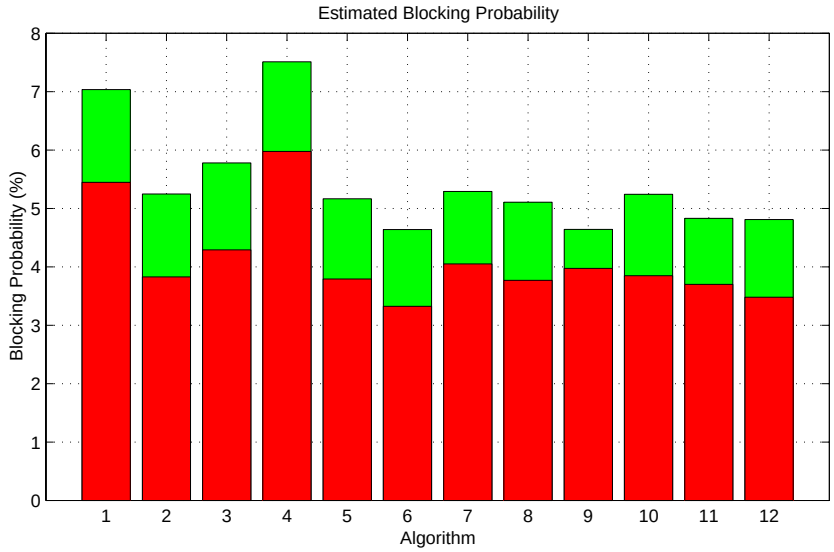

Figure 5: Results with quick heuristic algorithms and the first policy iteration. The routing parameters are: *ldelta*=1 and *rmax*=4. Algorithms are, from left to right, *basic*, *pcolor*, *porder*, *spread*, *ll*, *aurpack*, *basic+iteration*(50), *basic+iteration*(100), *basic+iteration*(200), *pcolor+iteration*(50), *pcolor+iteration*(100) and *pcolor+iteration*(200).

7 Conclusions

In this paper we have introduced the idea of applying the first iteration in the policy space to the RWA problem. With this method one can derive from a given heuristic policy an iteration policy, which theoretically always is a better policy, i.e. has lower average cost rate (e.g. blocking probability). The relative costs of states needed in the decision analysis are estimated on the fly by launching simulations from the current state of the system and trying different decision alternatives. The simulations introduce some noise in the cost estimates and careful control of the simulation parameters is required in order not to deteriorate the performance of the resulting iteration policy.

The performance improvement obtained by the policy iteration depends on the standard policy one starts with. The reduction of blocking probability in the numerical tests ranged from a few tens of percents to almost nothing. This suggests that algorithms like *lpcolor* and *aurexhaustive*, for which the improvement was small, are not far from optimal (with this particular network and setup). Also we can generally conclude that the order in which RW pairs are tried in a first-fit algorithm can be a very important factor for the performance. As the method of first policy iteration obviously is computationally intensive, it can be used in real time only if the dynamics of the system is slow. The inter-arrival times of the connection requests must be of the order of few seconds or more (depending on the number of possible decisions), but this may well be the case in WDM networks. However, the method can always be used off-line e.g. to evaluate heuristic algorithms to see how far they are from the optimum.

References

[1] D. Banaree and B. Mukherjee. A practical approach for routing and wavelength assignment in large wavelength-routed optical networks. *IEEE Journal on Selected*

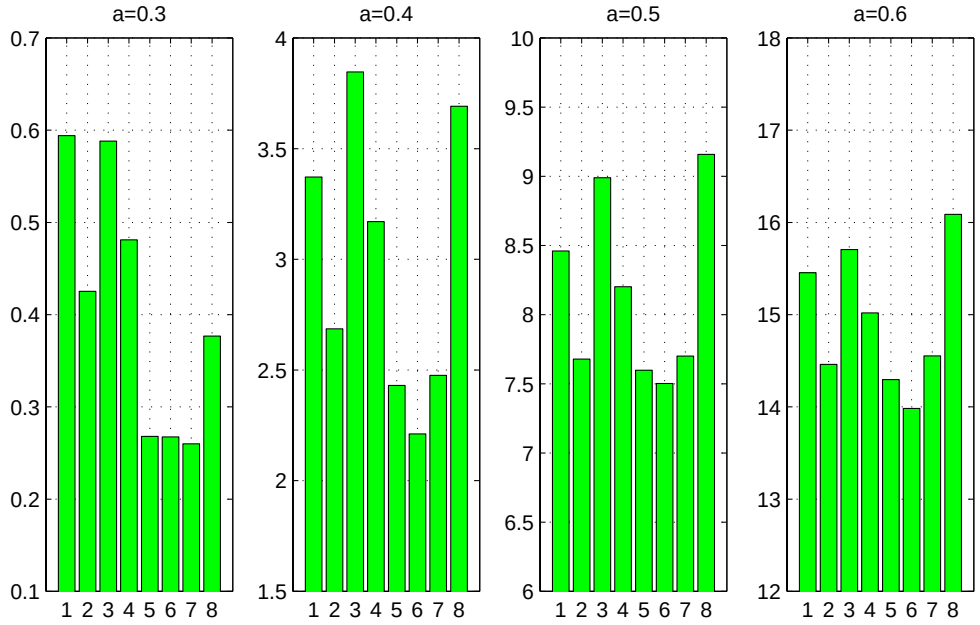

Figure 6: Blocking probability with loads ranging from $a = 0.3$ to $a = 0.6$. The set of routes were defined with *ldelta*=3 and *rmax*=30. Algorithms are, from left to right, *basic*, *basic+iteration*, *pcolor*, *pcolor+iteration*, *lpcolor*, *lpcolor+iteration*, *aurexhaustive* and *aurpack*.

Areas in Communications, 14(5), Jun 1996.

[2] S. Baroni. *Routing and wavelength allocation in WDM optical networks*. PhD thesis, University College London, May 1998.

[3] E. Karasan and E. Ayanoglu. Effects of wavelength routing and selection algorithms on wavelength conversion gain in wdm optical networks. *IEEE/ACM Transactions on Networking*, 6(2), Apr 1998.

[4] G. Mohan and S.R. Murthy. A time optimal wavelength rerouting algorithm for dynamic trafffic in wdm networks. *Journal of Lightwave Technology*, 17(3), Mar 1999.

[5] A. Mokhtar and M. Azizoğlu. Adaptive wavelength routing in all-optical networks. *IEEE/ACM Transactions on Networking*, 6(2), Apr 1998.

[6] Biswanath Mukherjee. *Optical Communication Networks*. McGraw-Hill series on computer communications. McGraw-Hill, 1997.

[7] R. Ramaswami and K.N. Sivarajan. Routing and wavelength assignment in all-optical networks. *IEEE/ACM Transactions on Networking*, 3(5), Oct 1995.

[8] R. Ramaswami and K.N. Sivarajan. *Optical Networks, A Practical Perspective*. Morgan Kaufmann Series in Networking. Morgan Kaufmann Publishers, 1998.

[9] Henk C. Tijms. *Stochastic Models, An Algorithmic Approach*. John Wiley & Sons Ltd, 1994.

[10] A.E. Willner. Mining the optical bandwidth for a terabit per second. *IEEE Spectrum*, Apr 1997.

[11] Dziong Zbigniew. *ATM Network resource management*. McGraw-Hill, 1997.