ELSEVIER

Theoretical Computer Science 221 (1999) 19–39

Theoretical Computer Science

www.elsevier.com/locate/tcs

On-line routing in all-optical networks ☆

Yair Bartal[a, 1], Stefano Leonardi[b, *,2]

[a] *U.C. Berkeley and International Computer Science Institute (ICSI), Berkeley, CA 94704, USA*
[b] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, I-00198 Roma, Italy*

Abstract

The paper deals with *on-line* routing in wavelength division multiplexing (WDM) optical networks. A sequence of requests arrives over time, each is a pair of nodes to be connected by a path. The problem is to assign a *wavelength* and a *path* to each pair, so that no two paths sharing a link are assigned the same wavelength. The goal is to minimize the number of wavelengths used to establish all connections. Raghavan and Upfal (Proc. 26th Annual Symp. on Theory of Computing, 1994, pp. 133–143) considered the off-line version of the problem, which was further studied in Aumann and Rabani (Proc. 6th ACM-SIAM Symp. on Discrete Algorithms, 1995, pp. 567–576), Kaklamanis and Persiano (Proc. 4th Annual European Symp. on Algorithms, Lecture Notes in Computer Science, vol. 1136, Springer, Berlin, 1996, pp. 460–470), Mihail et al. (Proc. 36th IEEE Annual Symp. on Foundations of Computer Science, 1995, pp. 548–557), Rabani, (Proc. 37th Annual Symp. on Foundations of Computer Science, 1996, pp. 400–409). For a line topology, the problem is the well-studied interval graph coloring problem. On-line algorithms for this problem have been analyzed in Kierstead and Trotter (Congr. Numer. 33 (1981) 143–153). We consider trees, trees of rings, and meshes topologies, previously studied in the off-line case. We give on-line algorithms with competitive ratio $O(\log n)$ for all these topologies. We give a matching $\Omega(\log n)$ lower bound for meshes. We also prove that any algorithm for trees cannot have competitive ratio better than $\Omega(\log n/\log\log n)$. We also consider the problem where every edge is associated with parallel links. While in WDM technology, a fiber link requires different wavelengths for every transmission, space division multiplexing technology allows parallel links for a single wavelength, at an additional cost. Thus, it may be beneficial in terms of network economics to combine between the two technologies (this is indeed done in

☆ A preliminary version of this paper has appeared in the Proceedings of the *24th* International Colloquium on Automata, Languages, and Programming (ICALP 97), pp. 516–526, Lecture Notes in Computer Science 1256, Springer, Berlin, 1997.

* Corresponding author. Tel.: 39-6-49918341; fax: 39-5-85300849.

E-mail addresses: yairb@icsi.berkeley.edu (Y. Bartal), leon@dis.uniromal.it (S. Leonardi)

[1] Research supported in part by the Rothschild Postdoctoral fellowship and by the National Science Foundation operating grants CCR-9304722 and NCR-9416101.

[2] This work was partially done while the author was a post-doc at the International Computer Science Institute (ICSI), Berkeley. This work is partly supported by EU ESPRIT Long Term Research Project ALCOM-IT under contract no. 20244, and by Italian Ministry of Scientific Research Project 40% "Algoritmi, Modelli di Calcolo e Strutture Informative".

PII: S0304-3975(99)00025-0

practice). For *arbitrary* networks with $\Omega(\log n)$ parallel links we give an on-line algorithm with competitive ratio $O(\log n)$.

Keywords: Online algorithms; Competitive analysis; Routing; Optical networks

1. Introduction

All-optical networks promise data transmission rates several orders of magnitude higher than current networks. The high speeds in these networks arise from maintaining signals in optical form throughout a transmission thereby avoiding the overhead of conversions to and from electrical form (see [14] for an overview of the topic). *Wavelength division multiplexing* (WDM) supports the propagation of multiple laser beams of distinct wavelengths through an optic fiber. Thus, the high bandwidth of the WDM network is utilized by partitioning it in many "channels", each at a different optical *wavelength*. Intuitively, we may think of wavelengths as light rays of different colors.

A major algorithmic problem for optical networks is that of routing. Each routing request consists of a pair of nodes in the network, and requires the assignment of a path and a wavelength (color). The key restriction is that two requests with equal wavelength cannot be routed through the same link. The main goal is to minimize the number of wavelengths for certain routing requests.

Many of the applications for high speed optical networks are real-time. It is therefore very natural to consider the problem of routing in an on-line setting where routing requests appear over time.

1.1. The path coloring Problem

The routing problem on a WDM network with generalized switches is referred to as *path coloring*. More formally, let $G=(V,E)$ be a graph representing the network, with $|V|=n$. We are given a sequence of routing requests consisting of pairs $p_i=(s_i,t_i)$ of nodes in G. The algorithm must assign a path connecting s_i and t_i and a color, so that no two paths sharing an edge are assigned the same color. The goal is to minimize the number of colors. The performance measure for an on-line algorithm is the *competitive ratio* [29] defined as the worst-case ratio over all request sequences between the number of colors used by the on-line algorithm and the optimal number of colors necessary on the same sequence.

While in WDM technology, a fiber link requires different wavelengths for every transmission, space division multiplexing (SDM) technology allows parallel links for a single wavelength, at an additional cost. This can be profitable since only a limited number of wavelengths are available in practice. The two technologies are then combined to find an efficient trade off between the two approaches. This motivates considering a generalization of the path coloring problem where a link of a color is

replaced with a number of parallel links. We will alternatively model this case with a bandwidth B available on a link for any color, meaning that B paths of the same color may be routed through a link (that is in the basic path coloring problem $B = 1$).

1.1.1. Related previous work

The *off-line* path coloring problem has been studied by Raghavan and Upfal [27] who give constant approximation algorithms for undirected trees and trees of rings. Further results for trees were given in a sequence of papers [16, 21, 23, 17]. Rings have been recently addressed in [12] and meshes were studied in [3, 20, 27]. Kleinberg and Tardos [20] give an $O(\log n)$ approximation algorithm for meshes and certain "nearly Eulerian planar graphs". Rabani [26] improves the bound for meshes to $O(\text{poly}(\log \log n))$.

The *on-line* path coloring problem has been studied in the case of a line topology in the context of *interval graph coloring* by Kierstead and Trotter [19]. They give an optimal 3-competitive algorithm for the line [19]. Ślusarek [28] proved the same bound for circular arc graphs.

The path coloring problem is closely related to the *virtual circuit routing* problem, motivated by its application to ATM networks. The *load* version of this problem is where every requested pair must be assigned a path as to minimize the maximum number of paths crossing a given edge. Aspnes et al. [2] give an $O(\log n)$ competitive algorithm for the load version. Most of the work has concentrated on the *throughput* version of the problem, where every requested pair may be either accepted or rejected. The basic problem, also referred to as *call-control*, is where the paths of all accepted pairs must be edge-disjoint. This can also be generalized to the case where edges may have a given bandwidth B (that can be viewed as having B parallel edges). Awerbuch et al. [5] prove that if $B = \Omega(\log n)$ then there is an $O(\log n)$ competitive algorithm (for throughput). They also give a lower bound of $\Omega(n)$ for deterministic algorithms in the general case. Randomized algorithms have been first studied by Awerbuch et al. [6, 7] giving an $O(\log \Delta)$ competitive algorithm for trees, where Δ is the diameter of the tree. They also show a matching lower bound. Kleinberg and Tardos [20] give $O(\log n)$ competitive algorithms for meshes (and some generalization), improving upon a previous result of [7].

Bartal et al. [10] prove that for various routing problems including the throughput version of virtual circuit routing and the path coloring problem there exist networks where the competitive ratio is $\Omega(n^{\varepsilon})$ (for some fixed ε) for any randomized algorithm. Finally, the on-line version of maximizing the throughput in optical networks was addressed in [4].

1.1.2. Contributions of this paper

We consider the *on-line* path coloring problem on trees, trees of rings, and meshes topologies:

- We present an $O(\log n)$ competitive deterministic algorithm for path coloring on *meshes*.

- We prove a matching $\Omega(\log n)$ lower bound for the mesh. The lower bound holds for randomized algorithms for the load version of the virtual circuit problem which immediately extends to the path coloring problem.

  We comment that this also provides the first lower bound for the load version of the virtual circuit routing problem in undirected networks with unit edge capacities [2].
- We give an $O(\log n)$ competitive algorithm for path coloring on *arbitrary* networks with bandwidth $\Omega(\log n)$ (the actual statement is somewhat more general). This algorithm is also used as a building block for our algorithm for path coloring on meshes. This result can be viewed as a balanced combination of WDM and SDM technologies.
- We give an $O(\log n)$ competitive algorithm for *trees* and *trees of rings*. We also prove that any deterministic algorithm for trees cannot have competitive ratio better than $\Omega(\log n/\log\log n)$ (even for trees with $\Delta = O(\log n)$). A logarithmic upper bound and an $\Omega(\sqrt{\log n})$ lower bound for trees have been independently obtained by Borodin, Kleinberg, and Sudan [11].

1.1.3. *Paper structure*

Section 2 contains the results for path coloring with more bandwidth on arbitrary networks, that are also used in Section 3 for the $O(\log n)$ competitive algorithm for path coloring on meshes. Section 4 contains the lower bound for meshes. Upper and lower bounds for trees are in Section 5.

2. Path coloring with more bandwidth

Let $G=(V,E)$ be a network with $|V|=n$ vertices and $|E|=m$ edges. We consider the *path coloring* problem with bandwidth B on the edges. At the jth step, call j, with endpoints (s_j,t_j), is presented to the algorithm that must assign a color $c(j)$ and a path $P(j)$. The goal of the on-line algorithm is to use a set of colors of cardinality at most C under the constraint that the bandwidth on any edge does not exceed B.

We give an algorithm for general networks for this problem. The algorithm fixes a set $\mathscr{C}$ of C colors that it may choose from, at the beginning, based on an estimate for the optimal performance. The basic algorithm chooses, at every step, one path connecting the endpoints of the current call and one of the colors in $\mathscr{C}$ according to some optimization criterion. This criterion assigns to any edge of any color an exponential function of the current load. Our goal is to prove that the algorithm never exceeds a certain bandwidth on every edge.

A variant for this algorithm proves to be useful (see Section 3) in obtaining an algorithm for path coloring on meshes (with edge bandwidth $=1$).

In this variant we restrict the choice of the on-line algorithm for call j to a subset $\mathscr{C}(j)$ of $\mathscr{C}$ (that may be chosen according to some arbitrary rule) whose cardinality is at most αC.

We thus state our results in terms of this parameter α. However, for the scope of this section alone one can focus the attention to the case where $\alpha = 1$.

Let B^* be the bandwidth used by an offline algorithm on any of C^* colors to accomodate the whole set of calls in the sequence.

We compare our algorithm to a *stronger adversary* that uses a bandwidth $\Lambda^* \leqslant B^* C^*$ on a single color, rather than being restricted to using C^* colors and bandwidth B^* for every color.

We assume that the on-line algorithm knows a value Λ such that $\Lambda^* \leqslant \Lambda \leqslant 2\Lambda^*$. This is performed by applying a doubling technique that results in increasing the competitive ratio at most by a factor of 4.

Let the load on edge e for color c, denoted by $\lambda_e^c(j)$, be the number of calls assigned color c and a path crossing edge e when call j is presented. For a parameter $\beta > 0$, let $a = 2^\beta$. Call j is assigned a color $c(j)$ and a path $P(j)$ which achieve the minimum, over all the colors in $\mathscr{C}(j)$ and all paths connecting s_j and t_j, of the following "exponential cost":

$$\sum_{e \in P(j)} a^{\lambda_e^{c(j)}(j)}.$$

Theorem 1. *If the number of colors used by the algorithm is* $C = 2\Lambda(1/\alpha)(2^\beta - 1)$ *where* $\Lambda \geqslant \Lambda^*$ *then the bandwidth is* $B \leqslant 1 + (1/\beta)\log(2m/\alpha)$.

Proof. Let $\bar{\lambda}$ be the maximum load on any edge for any color in the solution of the on-line algorithm at the end of the sequence. Thus there exists an edge such that $\bar{\lambda}$ calls are assigned a path crossing it and with the same color. When the last such path $P(k)$ is assigned to a call k, its exponential cost is at least $a^{\bar{\lambda}-1}$. By definition of the algorithm, the chosen path is the minimum cost path over all paths and colors in $\mathscr{C}(k)$. Therefore, at the time this call arrived, for $|\mathscr{C}(k)| \geqslant \alpha C$ colors, any path connecting the same pair of vertices has a cost of at least $a^{\bar{\lambda}-1}$. Let $Z(j)$ denote the sum of the exponential costs over all edges and all colors at step j. That is

$$Z(j) = \sum_{c \in \mathscr{C}} \sum_{e \in E} a^{\lambda_e^c(j)}.$$

Then it follows that

$$Z(f) \geqslant \alpha C a^{\bar{\lambda}-1}, \tag{1}$$

where $X(f)$ indicates the value of a function X after the last step in the sequence.

Let $l_e^*(j)$ be the number of calls in the adversary solution assigned path crossing edge e when call j is presented.

We use the following potential function:

$$\Phi(j) = \sum_{c \in \mathscr{C}} \sum_{e \in E} a^{\lambda_e^c(j)} \left(1 - \frac{l_e^*(j)}{2\Lambda}\right).$$

The sum of the exponential costs of the on-line algorithm at the end of the sequence is also bounded by the following:

$$\begin{aligned} Z(f) &= \sum_{c \in \mathscr{C}} \sum_{e \in E} a^{\lambda_e^c(f)} \\ &\leqslant 2 \sum_{c \in \mathscr{C}} \sum_{e \in E} a^{\lambda_e^c(f)} \left(1 - \frac{l_e^*(f)}{2\Lambda}\right) \\ &\leqslant 2(\Phi(f) - \Phi(0)) + 2mC, \end{aligned} \tag{2}$$

where the first inequality follows since $\Lambda \geqslant \Lambda^* \geqslant l_e^*(f)$.

In the following we prove that for the claimed choice of C, the potential function does not increase after each step of the algorithm. Therefore, $\Phi(f) \leqslant \Phi(0)$, and thus Eq. (1) and (2) can be combined to achieve

$$B \leqslant \bar{\lambda} \leqslant 1 + \frac{1}{\beta} \log \frac{2m}{\alpha}.$$

To complete the proof we prove that if $C = 2\Lambda(1/\alpha)(2^\beta - 1)$ then for every j, $\Phi(j+1) - \Phi(j) \leqslant 0$. Let $P^*(j)$ be the path assigned by the adversary for call j. The change in the potential function due to call j is

$$\begin{aligned} \Phi(j+1) - \Phi(j) &\leqslant \sum_{e \in P(j)} (a^{\lambda_e^{c(j)}(j+1)} - a^{\lambda_e^{c(j)}(j)}) - \frac{1}{2\Lambda} \sum_{c \in \mathscr{C}} \sum_{e \in E} (a^{\lambda_e^c(j+1)} l_e^*(j+1) \\ &\quad - a^{\lambda_e^c(j)} l_e^*(j)) \leqslant \sum_{e \in P(j)} (a-1) a^{\lambda_e^{c(j)}(j)} - \frac{1}{2\Lambda} \sum_{c \in \mathscr{C}} \sum_{e \in P^*(j)} a^{\lambda_e^c(j)} \end{aligned}$$

since the load on an edge is nondecreasing and for $e \in P^*(j)$ we have $l_e^*(j+1) = l_e^*(j) + 1$.

Observe that for any color $c \in \mathscr{C}(j)$, the cost of any path P connecting s_j to t_j is not less than the cost of the path $P(j)$ on color $c(j)$ chosen by the on-line algorithm for call j. Therefore, we get for any $c \in \mathscr{C}(j)$:

$$\sum_{e \in P} a^{\lambda_e^c(j)} \geqslant \sum_{e \in P(j)} a^{\lambda_e^{c(j)}(j)}.$$

The above inequality also holds for $P = P^*(j)$, and hence

$$\Phi(j+1) - \Phi(j) \leqslant \left((a-1) - \frac{\alpha C}{2\Lambda}\right) \sum_{e \in P(j)} a^{\lambda_e^{c(j)}(j)}.$$

Recall that $a = 2^\beta$. Thus, by choosing $C = 2\Lambda(1/\alpha)(2^\beta - 1)$ we have that the potential function does not increase. □

We have assumed in the analysis that the algorithm knows a value Λ such that $\Lambda^* \leqslant \Lambda \leqslant 2\Lambda^*$. In the following, we explain how to deal with this assumption.

We start running a copy of the algorithm with $\Lambda = 1$. If the algorithm is going to increase the load over B on some edge, then we run a new copy of the algorithm

with value of Λ doubled. Once $\Lambda^* \leqslant \Lambda \leqslant 2\Lambda^*$, it follows from Theorem 2 that the load on any edge does not exceed B. In that case the number of colors used is at most $C = 2\Lambda(1/\alpha)(2^\beta - 1) \leqslant 4\Lambda^*(1/\alpha)(2^\beta - 1)$.

Every time a new copy of the algorithm is run, a completely new set of colors is used. Observe that the cardinality of the set of colors is doubled with respect to the cardinality of the set of colors of the previous run of the algorithm. Therefore, the number of colors used by the algorithm is at most two times the number of colors used in the last run of the algorithm.

Hence we get the following theorem:

Theorem 2. *The number of colors used by the algorithm is* $C \leqslant 8\Lambda^*(1/\alpha)(2^\beta - 1)$ *and the bandwidth is* $B \leqslant 1 + (1/\beta)\log(2m/\alpha)$.

As an application we get the following result for the on-line load balancing problem [2], in which only one color is available and the goal is to minimize the number of paths assigned to a single edge of the network.

By applying Theorem 2 with $\beta = 1$ and $\alpha = 1$ we get the following.

Corollary 3. *There exists an algorithm for on-line load balancing that uses* $O(\Lambda^*)$ *colors with bandwidth* $O(\log n)$.

Note that Corollary 3 gives a stronger result than that of [2] that only shows that the on-line load is bounded by $O(\Lambda^* \log n)$.

Finally, going back to the path coloring problem, recall that $\Lambda^* \leqslant C^* B^*$. We can reformulate Theorem 2 to obtain the following (by choosing $\beta = 2/\gamma\delta$ where γ and δ are defined below).

Corollary 4. *Let* δ *be such that* $B^* = \delta \log(2m/\alpha)$, *and let* γ *be some positive coefficient (satisfying* $\gamma B^* \geqslant 2$). *The algorithm for on-line path coloring with more bandwidth uses* $C \leqslant 8C^*(\delta/\alpha)\log(2m/\alpha)(2^{2/\gamma\delta} - 1)$ *colors with bandwidth* $B \leqslant \gamma B^*$,

The above corollary shows that if the bandwidth is $\Omega(\log n)$, then the on-line algorithm does not exceed the bandwidth by using $O(\log n)$ more colors. We thus obtain the result for optical networks with general topology when the technologies WDM and SDM are combined in a network that contains $\Omega(\log n)$ parallel fiber optic links on each connection.

3. Path coloring on meshes

In this section we present an $O(\log n)$ competitive algorithm for path coloring on meshes.

$G = (V, E)$ denotes the $\sqrt{n} \times \sqrt{n}$ two-dimensional mesh. We consider $\sqrt{n}$ to be a power of 2. Let $|E| = m$ be the number of edges of the mesh. The vertex of the mesh

with row i and column j is denoted with $G[i,j]$. Given two vertices $v=G[i,j], v'=G[i',j']$ we denote by $d(v,v')=|i-i'|+|j-j'|$ the length of the shortest path connecting v and v'.

Calls are divided into *short* calls and *long* calls. Let α and σ be parameters that will be fixed later. They will satisfy the condition that $\sigma \log(2m/\alpha)$ is a power of two. A call (s,t) is long if $d(s,t) > 2\sigma \log(2m/\alpha)$, and short if $d(s,t) \leqslant 2\sigma \log(2m/\alpha)$.

We use two different algorithms for long calls and short calls. A different set of colors will be used for long calls and short calls. The algorithm for long calls translates the problem in a mesh, to a problem of coloring with more bandwidth in a simulated network that is also a mesh. Theorem 2 allows a logarithmic competitive ratio with a logarithmic bandwidth on any edge. The route obtained in the simulated network is later translated into a route in the original mesh, satisfying the constraint that paths associated to calls with the same color are disjoint. We describe in Section 3.1 the construction of the simulated network, and in Section 3.2 how a route in the simulated network is transformed into a route in the original mesh.

The algorithm for short calls classifies the calls on the basis of their length, and applies a greedy algorithm within each class. We present the algorithm for short calls in Section 3.3.

Both algorithms for long and short calls have competitive ratio $O(\log n)$. Therefore, we can state the following theorem.

Theorem 5. *There exists an* $O(\log n)$ *competitive algorithm for path coloring on meshes.*

3.1. The construction of the simulated network

In this section we describe the algorithm for coloring and routing long calls. We transform this problem into a problem on a simulated network of a mesh of size $\sqrt{n} \times \sqrt{n}$, with logarithmic bandwidth on the edges.

The algorithm divides the mesh into $[\sqrt{n}/(\sigma \log(2m/\alpha))] \times [\sqrt{n}/(\sigma \log(2m/\alpha))]$ squares of size $\sigma \log(2m/\alpha) \times \sigma \log(2m/\alpha)$. Square $S[p,q]$, $p,q=1,\ldots,\sqrt{n}/[\sigma \log(2m/\alpha)]$, is the subgraph of G induced by the set of vertices $\{G[i,j] | i=(p-1)\sigma \log(2m/\alpha)+1,\ldots,p\sigma \log(2m/\alpha); j=(q-1)\sigma \log(2m/\alpha)+1,\ldots,q\sigma \log(2m/\alpha)\}$.

Long calls have their endpoints in different squares, since the distance between the endpoints is bigger than $2\sigma \log(2m/\alpha)$.

The *simulated network* N of the mesh $G=(V,E)$ is a mesh of size $[\sqrt{n}/(\sigma \log(2m/\alpha))] \times [\sqrt{n}/(\sigma \log(2m/\alpha))]$.

Let m' be the number of edges of the simulated network. Every edge of N is associated with a bandwidth equal to $\sigma \log(2m/\alpha)=\delta \log(2m'/\alpha)$. (Observe that $\log m' = \log m - \Theta(\log \log m)$. Hence $\sigma \approx \delta$ for large m.)

This mesh corresponds to the network obtained from the original mesh by representing every square of G with a vertex and connecting every pair of vertices

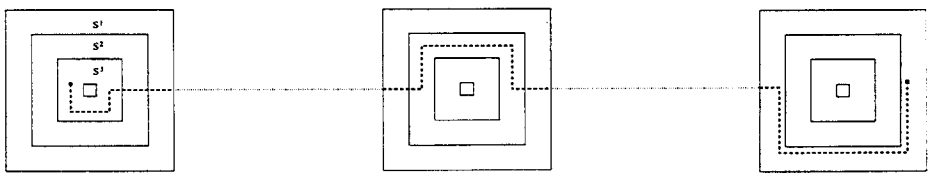

Fig. 1. The routing of a long call.

representing adjacent squares with an edge. The bandwidth of the edges models the fact that at most $\sigma \log(2m/\alpha)$ edge-disjoint paths can be routed between two adjacent squares.

The basic idea is to color and route long calls in the simulated network using the algorithm of Section 2 for path coloring with more bandwidth, and then translate the assigned paths into an appropriate routing in the original network.

The sequence of long calls in the mesh G is transformed into a sequence of calls in the simulated network in the most natural way: Each long call (s,t) is replaced by a call between the two vertices of N representing the two squares containing s and t.

The path obtained for a call in the simulated network is transformed into a path in the original mesh G respecting the following rule: The path in G will cross between adjacent squares in G where the path in N passes through the edge connecting the corresponding nodes in N.

However, we need that the paths with same color crossing any square are edge-disjoint. For this purpose we will restrict the set of candidate colors for each call to a constant fraction of the overall number of colors. (Observe that the design of the algorithm of Section 3.2 includes this feature.)

For this purpose we distinguish between the two squares that include the endpoints of a call, and the squares that are crossed by the path connecting the endpoints. We say that a call is *internal* to a square if one of its endpoints belongs to the square. A call is called *external* to a square if it is not internal to the square and the path derived by the routing in the simulated network crosses the square.

We furthermore define in any square S of the mesh, three concentric regions: S^1, S^2 and S^3 (see Fig. 1). Each region contains $2\log(2m'/\alpha)$ concentric rings of the square. S^1 is the most external region, S^2 is internal to S^1 and S^3 is internal to both S^1 and S^2. Finally, the area surrounded by S^3 is called the *central region* of the square.

The set of colors $\mathscr{C}$ used by the on-line algorithm is partitioned into three sets $\mathscr{C}^1, \mathscr{C}^2, \mathscr{C}^3$ of *equal size*. We impose a set of constraints on the color assignment and the routing strategy. These constraints help in avoiding intersection between paths associated with calls assigned the same color.

1. If a call is assigned a color $c \in \mathscr{C}^i$, $i = 1,2,3$, then its two endpoints must lie on a region different from S^i.

2. The path assigned to a call with color $c \in C^i$, $i = 1,2,3$, will cross any square of the mesh not containing an endpoint of the call using a ring of region S^i.
3. For any square, at most one internal call is assigned a given color.

Consider the jth long call (s_j, t_j). Let $S(s_j)$ and $S(t_j)$ be the squares containing s_j and t_j, respectively. The algorithm of Section 2 asks for each call to define a set of candidate colors. The set $\mathscr{C}(j)$ of candidate colors for call j is defined as follows. A color $c \in \mathscr{C}^i$ is in $\mathscr{C}(j)$ if the two following conditions hold:

1. $s_j \notin S^i(s_j)$ and $t_j \notin S^i(t_j)$, i.e. both endpoints are not in region i of their corresponding squares.
2. No call with an endpoint in $S(s_j)$ or $S(t_j)$ has been previously assigned color c.

The algorithm for path coloring with more bandwidth in the simulated network is run with parameters satisfying: $\alpha \leqslant \frac{1}{9}$; $\delta \geqslant 13$; and $\gamma = 1/\delta$. The value σ that defines the size of each square is chosen in order to satisfy $\sigma \log(2m/\alpha) = \delta \log(2m'/\alpha)$.

The choice of the parameters is such that the adversary bandwidth $B^* = \delta \log(2m'/\alpha)$ is equal to the maximum number of calls that can be routed through two adjacent squares, and the width $\delta \log(2m'/\alpha)$ of a square is equal to 13 times the maximum bandwidth $B = \log(2m'/\alpha)$ used by the on-line algorithm for routing between two adjacent squares.

To apply the result of Corollary 4 we need the following lemma.

Lemma 6. *The set of feasible colors $\mathscr{C}(j)$ for a call (s_j, t_j) has size at least αC.*

Proof. If the endpoints s_j, t_j belong to two differently indexed regions (for instance $s_j \in S^1(s_j)$ and $t_j \in S^3(t_j)$) then one of sets C^i, $i = 1,2,3$, meets Condition 1, that is both endpoints of the call are outside region S^i.

If the endpoints belong to the same indexed region or at least one of them is in the central region, then at least two sets among C^1, C^2, C^3 satisfy Condition 1.

Thus we conclude that in all cases, at least $\frac{1}{3}$ of the colors satisfy Condition 1. Let us compute which fraction of these colors satisfy Condition 2 as well, that is no call, internal to the square, has been previously assigned the color.

For this purpose we observe that if the optimal number of colors is C^* then the maximum number of calls internal to a specific square is bounded by $C^* 12\delta \log(2m'/\alpha)$. Thus the maximum number of calls internal to one of the two squares containing s_j and t_j is $I \leqslant 8\delta C^* \log(2m'/\alpha)$.

Recall that the set of colors used in one run of the algorithm is at most $C = 4C^*(\delta/\alpha)(\log(2m'/\alpha) + 1)$. Thus $I \leqslant \frac{2}{3}\alpha C$.

Therefore, the set of colors that can be used for a given call has size at least

$$|\mathscr{C}(|)| \geqslant \tfrac{1}{3}C - I \geqslant \tfrac{1}{3}C - \tfrac{2}{3}\alpha C \geqslant \alpha C,$$

for $\alpha \leqslant \frac{1}{9}$, thus proving the claim. □

From Corollary 4 we then derive the following corollary, that bounds the number of colors and the bandwidth used by the on-line algorithm for path coloring with more bandwidth in the simulated network.

Corollary 7. *The algorithm for on-line path coloring with more bandwidth in the simulated network N uses $C = 24C^*(\delta/\alpha)\log(2m'/\alpha)$ colors with bandwidth $B \leqslant \log(2m'/\alpha)$.*

3.2. Routing of long calls

In this section we describe how to transform a path in the simulated network N into a path in the mesh G, so that the paths associated to calls with same color are mutually edge-disjoint. Each call is assigned a path in the simulated network. This path indicates the squares in the original mesh to cross to connect the two endpoints of a call. We are left to describe the route followed by the path within each square.

The run of the algorithm for path coloring with more bandwidth ensures that the maximum bandwidth of the on-line algorithm in the simulated network is $B = (\sigma/13)\log(2m/\alpha)$. It follows that at most B calls are assigned paths crossing the boundary between two adjacent squares, and then at most $2B$ external calls cross each square.

We will maintain inductively the following property: A call crosses the boundary between two squares on a row or on a column connecting the central regions of the two squares. The central region of a square has size $B \times B$. Since B is the maximum number of calls routed between adjacent squares, a distinct row or column can be associated with any call.

We first consider external calls. By induction, each external call enters the square on a row or on a column leading to the central area. We route it towards the central area until a free ring of region S^i is reached. This is always the case since there are at most $2B$ external calls and $2B$ available rings in each region S^i. The call then follows the ring until it reaches a free row or a free column connecting the central region of the square to the central region of the adjacent square to which the call is directed. The route follows such row or such column until the adjacent square.

Finally, we consider the routing of the possible single internal call. The endpoint of the internal call is outside the area S^i. If it is originated in the central area, then it can follow any path within the central area until it reaches a free row or column that connects to the central area of the adjacent square to which the internal call is directed. The route goes through such row or column until the adjacent square is reached. If the endpoint of the internal call is outside the central area and outside region S^i, then the call is routed through the ring that contains the endpoint, until a free row or column connecting to the central area of the adjacent square to which the call is directed is met. This row or column is followed until the border of the appropriate adjacent square.

The routing of a call assigned a color of set $\mathscr{C}^2$ is shown in Fig. 2. The figure describes the route followed in the two squares where the call is internal, and in one square where the call is external. The call has one endpoint in a region S^3 and one endpoint in a region S^1. The call is routed on a ring of region S^2 in all the squares where it is external.

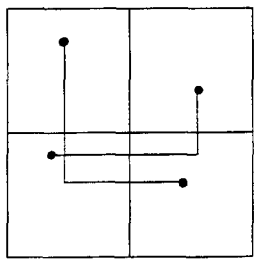

Fig. 2. The routing of short calls.

3.3. Coloring short calls

In this section we consider the routing and coloring of short calls. We remind that a call (u,v) is short if $d(u,v) \leqslant L = 2\sigma \log(2m/\alpha)$.

We define a partition of level l of the mesh, for each $l = 1,\ldots,\log L$, as follows. The mesh is divided into $\sqrt{n}/2^l \times \sqrt{n}/2^l$ squares of size $2^l \times 2^l$. Each square $S[p,q]$, $p,q = 1,\ldots,\sqrt{n}/2^l$, is the subgraph of $G = (V,E)$ induced by the set of vertices $\{G[i,j] \mid i = (p-1)2^l + 1,\ldots,p2^l;\ j = (q-1)2^l + 1,\ldots,q2^l\}$.

We define the set $\mathscr{C}_l$ of calls of level l as those calls that have the two endpoints within the same square in any partition of level higher than l, but in two different squares of level l. The level of a call is $\log L$ if the two endpoints are in different squares of the partition of level $\log L$.

The algorithm uses a distinct set of colors for each set $\mathscr{C}_l$. The routing assignment for a call (u,v) will always follow a shortest path from u to v. Each color is denoted by a positive integer. The color selected for a call (u,v) of level l is that one of minimum integer value such that there exists a path of minimum length connecting u to v, with this color available on all the edges of the path.

Theorem 8. *The algorithm for on-line coloring of short calls is* $O(\log n)$*-competitive.*

Proof. Let ON_l and OPT_l, be the number of colors used by the on-line algorithm and by the optimal off-line algorithm respectively, for calls of set l. Let C_l be the maximum number of calls that have both endpoints within the same square of level $l+1$, but in different squares of level l. Clearly, $ON_l \leqslant C_l$.

Let us consider the optimal number of colors for a set of calls of level l with both endpoints within the same square of level $l+1$. Each of these calls is routed either through the border of the square of level $l+1$, or through the border separating two of the four squares of level l contained in the square of level $l+1$. Since the size of the border of the square of level $l+1$ is $4 \times 2^{l+1}$ and the size of the borders separating the four squares of level l is 4×2^l, then at most $6 \cdot 2^{l+1}$ calls of level l with both endpoints in the same square of level $l+1$ can be assigned the same color. Therefore, the number of colors required by the optimal solution for calls of level l is at least $OPT_l \geqslant C_l/6 \cdot 2^{l+1}$.

Summing over all the levels, we get

$$\mathrm{ON} \leqslant \sum_{l=0}^{\log L} \mathrm{ON}_l \leqslant \sum_{l=0}^{\log L} C_l \leqslant \sum_{l=0}^{\log L} 6 \cdot 2^{l+1}\mathrm{OPT}_l \leqslant 6 \cdot \mathrm{OPT} \sum_{l=0}^{\log L} 2^{l+1}$$

$$= O(\log n)\mathrm{OPT}. \quad \square$$

4. Lower bounds on meshes

In this section we present a randomized lower bound of $\Omega(\log n)$ for the path coloring problem on meshes. The lower bound also applies to the load balancing problem on meshes, where we seek for minimizing the maximum number of paths crossing a link, irrespective of the color [2].

The lower bound is based on an application of Yao's Lemma to on-line algorithms [30]. We construct a distribution over request sequences, such that the number of colors used by an optimal algorithm is always bounded by a constant, while the expected on-line load (i.e., the maximum number of paths crossing an edge) of a deterministic algorithm is $\Omega(\log n)$. We recall that the load of a path coloring algorithm is bounded above by the number of colors and thus the lower bound follows.

Consider a $N \times N$ 2-dimensional mesh, with $N = \sqrt{n}$. The distribution over request sequences is defined recursively in $L = \log_4 N$ stages as follows. At the ith stage of the recursion, $i = 1, 2, \ldots, L$, we define a probability distribution for an $4^{L-i+1} \times 4^{L-i+1}$ square S_i of the mesh. We consider a partition of S_i into 16 subsquares of size $4^{L-i} \times 4^{L-i}$. The internal part of the square S_i is defined as the square I consisting of the 4 internal subsquares in the above partition. $S_i \backslash I$ is called the external part of the square. Let $I[x, y]$ denote the vertex with row x and column y in the submesh defined by I where $0 \leqslant x, y < 2 \cdot 4^{L-i}$. We now give for each $0 \leqslant y < 2 \cdot 4^{L-i}$ a set of 12 vertical calls from $I[0, y]$ to $I[2 \cdot 4^{L-i} - 1, y]$. Then choose at random one of the 16 subsquares and proceed with the $(i+1)$st stage of the probability distribution for that subsquare recursively.

The next two claims give bounds on the optimal and the on-line solutions.

Claim 9. *The number of colors used by an optimal algorithm for the above probability distribution is at most* 12.

Proof. We prove the claim by induction on i. If the subsquare of size $4^{L-i} \times 4^{L-i}$ chosen in the probability distribution is not in the internal part I, then we route the calls given in the ith stage through the internal part of the square (see Fig. 3a), and otherwise we route the calls through the external part of the square (see Fig. 3b) so that none of the routes will cross the routes for calls in stages $j > i$. This can be done so that calls with distinct source and destination have disjoint paths and thus the number of colors is 12. $\square$

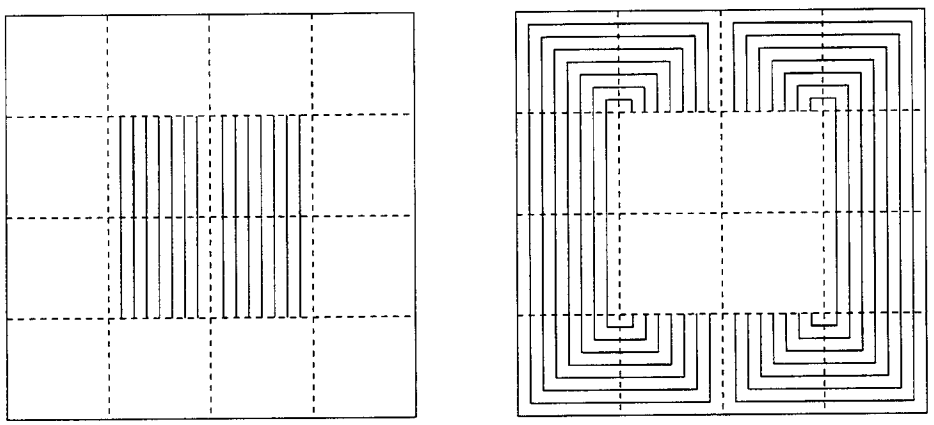

Fig. 3. The routing for the lower bound on meshes.

Claim 10. *Let A_i be the expected average load of the on-line algorithm on the edges in the square S_i. Then $A_i \geqslant i$.*

Proof. We first prove that the average increase in the load of the edges of the square S_i due to the requests given at the ith stage is at least 1. The number of edges of the mesh S_i is $2 \times 4^{L-i+1} \times (4^{L-i+1} - 1)$. At the ith stage, $12 \times 2 \times 4^{L-i}$ calls are presented. Every call of stage i is connected on a path containing at least $2 \times 4^{L-i} - 1$ edges of S_i (even if the path passes outside S_i). The sum for every call of stage i of the number of edges of S_i included in the call is then at least $2 \times 4^{L-i+1} \times (4^{L-i+1} - 1)$. This results into an increase of the average load on the edges of S_i of at least 1.

We now prove by induction that $A_i \geqslant i$. For $i = 1$ it follows from the above claim. We assume the claim holds for i and prove it for $i + 1$. Since the subsquare for the $(i + 1)$'st stage is chosen at random the expected average load of the edges of S_{i+1} is equal to A_i. Since the average increase in the load of the edges of S_{i+1} is at least 1 we have $A_{i+1} \geqslant i + 1$. □

We conclude the following.

Theorem 11. *The competitive ratio of any on-line randomized path coloring algorithm on meshes is $\Omega(\log n)$ against oblivious adversaries. The same lower bound holds for load balancing on meshes.*

5. Path coloring on trees and trees of rings

In this section we consider the on-line path coloring problem on trees and on trees of rings. We give an algorithm with competitive ratio $O(\log n)$ for both trees and trees

of rings, and prove an $\Omega(\log n/\log\log n)$ deterministic lower bound on the competitive ratio of any algorithm on trees. We remark that the lower bound is achieved on a tree of diameter $O(\log n)$ (in contrast to the $\Omega(\log n)$ randomized lower bound for the call-control problem which is achieved on a tree of diameter n).

5.1. An upper bound for path coloring on trees and trees of rings

We show in this section how to obtain an on-line path coloring algorithm for trees. A simple modification of the algorithm also applies to trees of rings.

A natural approach is to use an algorithm based on a First Fit strategy. Colors are indicated by positive integers. Every pair is assigned the minimum integer not assigned to an intersecting pair.

The First Fit algorithm is proved to be $O(\log n)$ competitive if the number of pairs in the instance is polynomial in the number n of vertices of the tree. The problem is reduced to coloring a d-inductive graph. A graph is d-inductive if its vertices can be numbered in a way that each vertex has at most d links to vertices with higher number. Irani [15] gives an algorithm that colors on-line a d-inductive graph of n vertices with $O(d\log n)$ colors.

The path coloring problem on trees corresponds to the problem of coloring an intersection graph where each vertex represents a pair, and two vertices are connected if the two corresponding pairs are intersecting. Let C be the maximum number of paths that are connected through a given edge. C is a lower bound on the clique number of the intersection graph and then a lower bound on the chromatic number of the intersection graph.

Lemma 12. *The intersection graph of an instance of the path coloring problem on trees is* $2(C-1)$*-inductive, where* C *is the clique number of the intersection graph.*

Proof. We show that we can number all pairs of vertices in a way that each pair is conflicting with at most $2(C-1)$ pairs with higher number.

The numbering is obtained by rooting the tree at any vertex, then ordering the pairs in a way that pair (s,t) precedes pair (s',t') only if $\mathrm{LCA}(s',t')$ is not a descendant of $\mathrm{LCA}(s,t)$. This ordering can be easily produced by visiting the tree from the leaves to the root. It follows that a pair (s,t) can intersect a pair (s',t') with higher number only on one of the two edges of (s,t) that are adjacent to $\mathrm{LCA}(s,t)$.

Since there are at most $C-1$ pairs distinct from (s,t) intersecting on an edge of the tree, it follows that any pair can intersect at most $2(C-1)$ pairs with higher number. □

Combining the above lemma with the result of [15], we obtain the following.

Theorem 13. *The algorithm First Fit is* $O(\log n)$*-competitive for on-line path coloring on trees of n vertices, if the number of pairs is polynimially bounded in n.*

If the number of pairs is not bounded by a polynomial in n, we present an algorithm with competitive ratio $O(\log n)$ based on a variation of the First Fit strategy. A similar result has independently been obtained in [1].

All the vertices of the tree are partitioned into $O(\log n)$ different classes, by recursively finding a balanced tree separator. A *balanced tree separator* [22] is a vertex whose removal splits the tree into pieces of at most $\frac{2}{3}n$ vertices. The tree separator of the original tree is assigned *level* 0. Removing the level-0 node splits the tree into subtrees *of level* 1. In general, the tree separators of the level-j trees are assigned *level* j and removing them creates subtrees of *level* $j+1$. After a logarithmic number of recursions the trees obtained are single vertices and the procedure stops. A similar technique is also used in [6] for the online call-control problem on trees.

The classification of the calls is done as follows. Every edge is assigned the lowest of the levels associated with its two endpoints. Every call is assigned the class indicated by the lowest level of an edge included in the call.

Every time a new pair is presented, the algorithm assigns the color with minimum integer value that is not assigned to any intersecting call of the same level or to *any* call of other level.

Theorem 14. *The algorithm for on-line path coloring on a tree of n vertices is* $O(\log n)$*-competitive.*

Proof. Let $C_l \leqslant C$ be the maximum number of calls including the same edge of level l. For every class l, we clearly have that C_l is a lower bound on the optimal solution.

The proof uses the two following facts:

(i) Every call of level l contains at most one vertex of level l;

(ii) If two calls of level l intersect, then they cross an identical edge of level l.

Both facts are easily proved observing that otherwise a call of level l would include a vertex of level $l-1$, a contradiction to level l being the minimal level of an edge included in the call.

A call of level l intersects at most $2(C_l-1)$ calls of level l. Therefore, the maximum number of colors used by the greedy algorithm for calls of level l is bounded by $2C_l-1$. Calls of distinct levels are assigned disjoint sets of colors, thus implying that at most $O(\log n)(2C-1)$ colors are used by the algorithm for the whole sequence of calls. □

For trees of rings we apply the same algorithm after using the idea [16, 27] of removing one edge in every ring, thus obtaining a tree. Since the maximum number of calls that cross an edge of the obtained tree is at most twice the maximum number of calls that cross an edge of the original tree of rings, we obtain the following.

Theorem 15. *The algorithm for on-line path coloring on trees of rings of n vertices is* $O(\log n)$*-competitive.*

5.2. A lower bound for path coloring on trees

We prove that for any on-line ρ-competitive algorithm for path coloring on trees $\rho = \Omega(\log n / \log\log n)$.

The lower bound is established on a complete binary tree of L levels, where L is a power of 2. Level 0 denotes the root while level $L-1$ denotes the leaves. The number of vertices at level l is 2^l, and the number of vertices in the tree is $n = 2^L - 1$.

The input sequence for the lower bound is generated in stages. The optimal number of colors is 2 for the entire sequence. We will prove for any on-line algorithm that the number of necessary colors will increase by 1 at each stage. The sequence will be iterated for $\Omega(L/\log L)$ stages, thus obtaining the lower bound.

At stage i of the sequence we will give a lower bound on the number of existing disjoint paths from a vertex of a level $\bar{l}_i$ to a leaf, such that for each color in a specific set of i colors there exists an edge in the path that is included in a call assigned this color.

More precisely, we maintain the following invariant at the end of any stage $i \geqslant 0$ of the sequence.

There exists a set C_i of i colors, a level $\bar{l}_i$, $l_i \leqslant \bar{l}_i \leqslant L - 1 - i$, where $l_i = L - 1 - i\log 8\rho L$, such that there are at least $p_i = 2^{L-1}/(8\rho L)^i$ pairs of paths with the following properties:

1. Each pair is formed by two paths from two leaves to their least common ancestor (LCA) at level $\bar{l}_i$.
2. Each vertex of level $\bar{l}_i$ is the LCA of at most one pair of paths.
3. For any path and for any color $c \in C_i$, there is one edge in the path included in a call with color c.
4. Any edge of a path is included in at most one call.

The invariant is maintained in the following way: the basis case of our construction is as follows. At stage 0, $\bar{l}_0 = l_0 = L-1$, and $C_0 = \emptyset$. We associate a set of $p_0 = 2^{L-1}$ pairs of *empty* paths, two with each leaf, with both endpoints equal to the leaf itself. No calls are presented. Hence, all four properties trivially hold.

At stage $i+1$, p_i new calls are presented, one for each pair of paths. Let u_1, u_2 be the two leaves that are endpoints of the two paths of a pair, and let $\mathrm{LCA}(u_1, u_2)$ be the LCA at level $\bar{l}_i$ of these two leaves. Let v be the direct ancestor of $\mathrm{LCA}(u_1, u_2)$. For any pair of paths we present a call having as endpoints one of the two leaves (arbitrarily chosen), say u_1, and v.

The on-line algorithm cannot use colors of C_i for these calls. We will show that the optimal number of colors at any stage of the sequence is at most 2. Hence, in order for the on-line algorithm to be ρ-competitive, it must use less than 2ρ colors for this set of calls.

Therefore, there must be a set of calls S_i of cardinality at least $p_i/2\rho$ assigned the same color. Call this color c_{i+1}. Let $C_{i+1} = C_i \cup \{c_{i+1}\}$. In the following we concentrate on this set of calls.

We first show a set of paths that satisfy conditions 3 and 4.

Each call in S_i is from a leaf u_1 to a node v. This is constructed from some level i pair of paths from leaves u_1 and u_2 to LCA(u_1,u_2), which is a child of v. For any pair only one call in S_i is constructed. Therefore, for any call in S_i, conditions 3 and 4 are satisfied at stage $i+1$ for the path connecting the leaf u_2 to v or any ancestor of v. In fact, at most one call includes any edge from u_2 to LCA(u_1,u_2) and any color $c \in C_i$ is associated with a call that crosses only one edge in the path, using the invariant for level i. Moreover, the edge from LCA(u_1,u_2) to v is associated with only one call with color c_{i+1}.

We thus have a set of $p_i/2\rho$ paths satisfying conditions 3 and 4 of the invariant. We call this set of paths P_{i+1}.

The level l_{i+1} is such that $2^{l_{i+1}+1} \leqslant p_i/2\rho$. We derive from P_{i+1} a new set P'_{i+1} as follows: we consider each path in P_{i+1} according to its ancestor in level l_{i+1}. If a vertex in level l_{i+1} is an ancestor of an odd number of paths we exclude one of these paths. Since the number of vertices of level l_{i+1} is at most $\frac{1}{2}p_i/2\rho$, the cardinality of P'_{i+1} is at least $\frac{1}{2}p_i/2\rho$.

We now scan paths in P'_{i+1} from left to right, following the order of the leaves that are endpoints of those paths (we assume that there is some left-to-right order between the leaves, i.e., their order in a pre-order traversal). We associate each pair of successive leaves with their LCA, that is a vertex of level between l_{i+1} and $L-i-1$.

We need the following simple property of binary trees.

Lemma 16. *Each vertex in a binary tree is the LCA of at most one pair of successive leaves.*

Proof. The LCA of two leaves in a binary tree is the only vertex with one leaf in the right subtree and the other leaf in the left subtree. Therefore, by contradiction, if two pairs of successive leaves have the same LCA, then both the right and the left subtree contain one leaf for both pairs. It follows that two leaves in a pair are not successive leaves in the order. □

Finally, let $\overline{l_{i+1}}$ be a level between $L-i-1$ and l_{i+1}, achieving the maximum cardinality set of pairs of successive paths that have LCA at that level. We define the set of pairs of paths for the stage $i+1$ to be the set of pairs of successive paths that have LCA at level $\overline{l_{i+1}}$.

Since the number of levels is L, it follows that the number of pairs of paths at stage $i+1$ is at least $\frac{1}{4}p_i/2\rho L = 2^{L-1}/(8\rho L)^{i+1} = p_{i+1}$. From the above construction, it follows that both conditions 1 and 2 hold for this set of pairs.

The next lemma gives the size of the optimal solution on the input sequence.

Lemma 17. *The optimal solution uses at most* 2 *colors.*

Proof. Any edge is included in at most 2 calls of the input sequence, and all calls are directed from a leaf to an ancestor. We color calls in a top-to-bottom way. Consider

a vertex v and an edge e connecting v to a descendant. If no call including e has v as an endpoint, the claim holds. If a call including e has v as an endpoint, then we use for that call a color that is not used for the other call including e. Since an edge is included in at most two calls, the claim is proved. □

The following theorem states the lower bound for path coloring on trees.

Theorem 18. *Any algorithm for path coloring on trees of n vertices has a competitive ratio of* $\Omega(\log n/\log\log n)$.

Proof. The on-line algorithm uses at least i colors after i stages of the construction above. Hence, by Lemma 17 the competitive ratio is $\rho \geqslant i/2$.

The lower bound is thus obtained by computing the maximum number of stages the sequence can be repeated. We iterate the sequence while $l_i = L - 1 - i \log 8\rho L \geqslant 1$. Since $\rho \geqslant i/2$, at the end of the sequence we have that $\rho \geqslant (L-2)/2 \log 8\rho L$. Since $n = 2^L - 1$, we obtain that $\rho = \Omega(\log n/\log\log n)$. □

6. Conclusions

In this paper we consider the on-line path coloring problem on trees and meshes topologies, motivated by its applications to Wavelength division multiplexing optical networks. We also study general networks when a logarithmic number of parallel links is available, thus modeling Space division multiplexing optical networks.

We present deterministic algorithms with logarithmic competitive ratio. For meshes, we give a matching randomized lower bound. For trees topologies, it is not known if a randomized algorithm can achieve a better competitive ratio. This is not even known for a line network topology, where deterministic algorithms have competitive ratio of 3 [19].

The off-line approximation of these problems is also an interesting open problem. For meshes, recently Rabani has achieved a poly $(\log\log n)$ approximation [26]. For trees, when each link is formed by two directed links with opposite directions, an algorithm that uses a number of colors that is at most $\frac{5}{3}$ times the maximum number of paths crossing a link has recently been proposed [17]. There are no reasons to think that such bounds cannot be improved.

Acknowledgements

We would like to thank Yossi Azar, Allan Borodin, Amos Fiat, Sandy Irani, Hal Kierstead and Gerhard Woeginger for useful discussions. We would also like to thank Thomas Erlebach for useful suggestions.

References

[1] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswami, B. Schieber, M. Sudan, Efficient routing and scheduling algorithms for optical networks, Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms, 1994, pp. 412–423.

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing, Proc. 25th Annual ACM Symp. on Theory of Computing, 1993, pp. 623–631.

[3] Y. Aumann, Y. Rabani, Improved bounds for all-optical routing, Proc. 6th ACM-SIAM Symp. on Discrete Algorithms, 1995, pp. 567–576.

[4] B. Awerbuch, Y. Azar, A. Fiat, S. Leonardi, A. Rosén, On-line competitive algorithms for call admission in optical networks, Proc. 4th Annual European Symp. on Algorithms, Lecture Notes in Computer Science, vol. 1136, Springer, Berlin, 1996, pp. 431–444.

[5] B. Awerbuch, Y. Azar, S. Plotkin, Throughput competitive on-line routing, Proc. 34th Annual Symp. on Foundations of Computer Science, 1993, pp. 32–40.

[6] B. Awerbuch, Y. Bartal, A. Fiat, A. Rosén, Competitive non-preemptive call control, Proc. SODA'94, Proc. 5th ACM-SIAM Symp. on Discrete Algorithms, 1994, pp. 312–320.

[7] B. Awerbuch, R. Gawlick, F.T. Leighton, Y. Rabani, On-line admission control and circuit routing for high performance computing and communication, Proc. 35th IEEE Annual Symp. on Foundations of Computer Science, 1994, pp. 412–423.

[8] R.A. Barry, P.A. Humblet, Bounds on the Number of Wavelengths Needed in WDM Networks, LEOS'92 Summer Topical Mtg. Digest, 1992, pp. 114–127.

[9] R.A. Barry, P.A. Humblet, On the number of wavelengths and switches in all-optical networks, IEEE Trans. Commun. 1994, pp. 567–576.

[10] Y. Bartal, A. Fiat, S. Leonardi, Lower bounds for on-line graph problems with application to on-line circuit and optical routing, Proc. 28th ACM Symp. on Theory of Computing, 1996, pp. 531–540.

[11] A. Borodin, J. Kleinberg, M. Sudan, Personal communication.

[12] O. Gerstel, S. Kutten, Dynamic wavelength allocation in WDM ring networks, Proc. ICC '97, 1997.

[13] O. Gerstel, G.H. Sasaki, R. Ramaswami, Dynamic channel assignment for WDM optical networks with little or no wavelength conversion, Proc. 34th Allerton Conf. on Communication, Control and Computation, 1996.

[14] P.E. Green, Fiber-Optic Communication Networks, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[15] S. Irani, Coloring inductive graphs on-line, Proc. 31st IEEE Annual Symp. on Foundations of Computer Science, 1990, pp. 470–479.

[16] C. Kaklamanis, P. Persiano, Efficient wavelength routing in directed fiber trees, Proc. 4th Annual European Symp. on Algorithms, Lecture Notes in Computer Science, vol. 1136, Springer, Berlin, 1996, pp. 460–470.

[17] C. Kaklamanis, P. Persiano, T. Erlebach, K. Jansen, Constrained bipartite edge coloring with applications to wavelength routing, Proc. 24th Int. Colloquium on Automata, Languages and Programming, 1997, pp. 493–504.

[18] H.A. Kierstead, The linearity of first-fit coloring of interval graphs, Siam J. Discrete Math. 1(4) (1988) 526–530.

[19] H.A. Kierstead, W.T. Trotter, An extremal problem in recursive combinatorics, Congr. Numer. 33 (1981) 143–153.

[20] J. Kleinberg, E. Tardos, Disjoint paths in densely embedded graphs, Proc. 36th IEEE Annual Symp. on Foundations of Computer Science, 1995, pp. 52–61.

[21] V. Kumar, E. Schwabe, Improved access to optical bandwidth in trees. Proc. 8th ACM-SIAM Symp. on Discrete Algorithms, 1997, pp. 437–444.

[22] J. Van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity, The MIT Press, Cambridge, MA, 1990.

[23] M. Mihail, C. Kaklamanis, S. Rao, Efficient access to pptical bandwidth, Proc. of the 36th IEEE Annual Symp. on Foundations of Computer Science, 1995, pp. 548–557.

[24] R.K. Pankaj, Architectures for linear light-wave networks, Ph.D. Thesis, MIT, 1992.

[25] G.R. Pieris, G.H. Sasaki, A linear lightwave benes network, IEEE/ACM Trans. Networking (1993).

[26] Y. Rabani, Path-coloring on the mesh, Proc. 37th Annual Symp. on Foundations of Computer Science, 1996, pp. 400–409.

[27] P. Raghavan, U. Upfal, Efficient routing in all-optical networks, Proc. 26th Annual Symp. on Theory of Computing, 1994, pp. 133–143.
[28] M. Ślusarek, Optimal online coloring of circular arc graphs, RAIRO J. Inform. Theor. Appl. 29 (5) 423-429.
[29] D. Sleator, R.E. Tarjan, Amortized efficiency of list update and paging rules, Communi. ACM 28 (1985).
[30] A.C. Yao, Probabilistic computations: towards a unified measure of complexity, Proc. 17th Annual Symp. on Foundations of Computer Science, 1977, pp. 222–227.