Theoretical Computer Science 255 (2001) 33–50

Theoretical Computer Science

www.elsevier.com/locate/tcs

The complexity of path coloring and call scheduling

Thomas Erlebach[a,*], Klaus Jansen[b]

[a] *Institut für Informatik, Technische Universität München, Arcisstraße 21, D-80290 München, Germany*

[b] *IDSIA Lugano, Corso Elvezia 36, CH-6900 Lugano, Switzerland*

Received July 1997; revised March 1999

Communicated by M.S. Paterson

Abstract

Modern high-performance communication networks pose a number of challenging problems concerning the efficient allocation of resources to connection requests. In all-optical networks with wavelength-division multiplexing, connection requests must be assigned paths and colors (wavelengths) such that intersecting paths receive different colors, and the goal is to minimize the number of colors used. This path coloring problem is proved $\mathcal{NP}$-hard for undirected and bidirected ring networks. Path coloring in undirected tree networks is shown to be equivalent to edge coloring of multigraphs, which implies a polynomial-time optimal algorithm for trees of constant degree as well as $\mathcal{NP}$-hardness and an approximation algorithm with absolute approximation ratio $\frac{4}{3}$ and asymptotic approximation ratio 1.1 for trees of arbitrary degree. For bidirected trees, path coloring is shown to be $\mathcal{NP}$-hard even in the binary case. A polynomial-time optimal algorithm is given for path coloring in undirected or bidirected trees with n nodes under the assumption that the number of paths touching every single node of the tree is $O((\log n)^{1-\varepsilon})$. Call scheduling is the problem of assigning paths and starting times to calls in a network with bandwidth reservation such that the maximum completion time is minimized. In the case of unit bandwidth requirements, unit edge capacities, and unit call durations, call scheduling is equivalent to path coloring. If either the bandwidth requirements or the call durations can be arbitrary, call scheduling is shown $\mathcal{NP}$-hard for virtually every network topology. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Call scheduling; Path coloring; Optical networks; Trees, Rings

1. Introduction

Modern high-performance communication networks pose a number of challenging research problems concerning the efficient allocation of resources to connection requests.

* Corresponding address: Institute TIK, ETH Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.
E-mail address: erlebach@in.tum.de (T. Erlebach), klaus@idsia.ch (K. Jansen).

In these networks, establishing a connection between two nodes requires selecting a path connecting the two nodes (routing) and allocating sufficient resources on all links along this path. We consider two such types of networks: *all-optical networks* and *networks with bandwidth reservation*.

In all-optical networks, data is transmitted on lightwaves through optical fiber, and several signals can be transmitted through a fiber link simultaneously provided that different wavelengths are used (wavelength-division multiplexing) [6, 1]. To establish a connection, it is necessary to reserve a wavelength on all links of its transmitter–receiver path. Since current all-optical networks do not support wavelength conversion, a connection must use the same wavelength on the whole path from transmitter to receiver. The number of wavelengths is a limited resource, and it is thus desirable to establish a given set of connection requests with a minimum number of wavelengths. We refer to wavelengths as colors and model the routing and wavelength assignment problem for all-optical networks as the *path coloring* problem.

In networks with bandwidth reservation, every link offers a certain bandwidth (capacity), and every connection request specifies its bandwidth requirement. A link can be shared by several connections provided that the sum of their bandwidth requirements does not exceed the link capacity. We refer to the problem of assigning paths and starting times to connection requests with the goal of minimizing the latest completion time as the *call scheduling* problem. A popular example of networks that support bandwidth reservation are ATM networks [40, 42].

In this paper, call scheduling and path coloring are considered as off-line minimization problems. We say that call scheduling or path coloring is $\mathcal{NP}$-hard in a certain setting if the decision version of the problem is $\mathcal{NP}$-complete. For a given instance I of a minimization problem, we let $OPT(I)$ denote the value of an optimal solution and $A(I)$ the value of the solution produced by algorithm A. A has *absolute approximation ratio* ρ if $A(I) \leqslant \rho OPT(I)$ for all I, and *asymptotic approximation ratio* ρ if $\limsup_{OPT(I)\to\infty} A(I)/OPT(I) \leqslant \rho$.

Part of our investigations will deal with tree networks. An undirected graph is a tree if it is connected and does not contain a cycle. The *diameter* of a tree is the maximum length (number of edges) of a simple path in the tree. A tree with diameter two is called a *star*; it consists of a central node and an arbitrary number of nodes that are adjacent to the central node but not to each other.

A path *touches* a node v if it begins at v, ends at v, or passes through v. For a given set of paths in a graph G, the *load* of an edge is the number of paths using that edge, and the *maximum load* among all edges of G is denoted by L. The *conflict graph* of a set of paths is the graph with one vertex for each path and an edge between two vertices if the corresponding paths share an edge. An *edge coloring* of a graph or multigraph G is an assignment of colors to the edges of G such that edges receive different colors if they share an endpoint. The minimum number of colors required in any edge coloring of G is denoted by $\chi'(G)$. The *line graph* of a graph or multigraph G is the graph with one vertex for each edge of G and an edge between two vertices if the corresponding edges in G share an endpoint.

1.1. The path coloring problem

The path coloring problem models routing and wavelength assignment in all-optical communication networks without wavelength converters. The network is represented by a connected graph $G=(V,E)$, where the nodes of the graph correspond to network switches and the edges correspond to fiber links between the switches. Connection requests are given by pairs of nodes. Establishing a connection from u to v requires assigning it a path p from u to v and a wavelength (color) such that no other connection whose path shares an edge with p is assigned the same wavelength. Given a connected graph G and a set R of connection requests, the path coloring problem is to assign paths and colors to all connection requests in R such that the number of different colors is minimized. For a coloring $C:R\rightarrow\mathbb{N}$, the number of distinct colors used is denoted by $|C|$. If $|C|\leqslant k$, C is called a k-coloring.

One can either study the case that G is an undirected graph and connection requests are assigned undirected paths in G, or the case that G is a bidirected graph (the directed graph obtained from an undirected graph by replacing each undirected edge by two directed edges with opposite directions) and connection requests are assigned directed paths in G. We refer to these variants as *undirected path coloring* and *directed path coloring*, respectively. Both variants have been studied in the literature, but directed path coloring appears to be a better model for current all-optical networks.

In tree networks, the paths for the connection requests are uniquely determined by their endpoints. Therefore, we refer to connection requests in tree networks simply as paths and denote the given set of connection requests by P instead of R. The maximum load L of the paths is a lower bound on the number of colors in an optimal solution.

1.2. The call scheduling problem

In networks with bandwidth reservation, it is desirable to complete a given set of calls in minimum time. The network is represented by a connected graph $G=(V,E)$ with edge capacities $c:E\rightarrow\mathbb{Q}_+$. The capacity of an edge represents the bandwidth of the corresponding link. A call c is specified by a pair (u_c,v_c) of nodes in G, a bandwidth requirement $b_c\in\mathbb{Q}_+$, and a duration $d_c\in\mathbb{Q}_+$. Establishing a call c from u_c to v_c at time $t_c\geqslant 0$ requires reserving bandwidth b_c on all edges along a path from u_c to v_c during the time interval $[t_c,t_c+d_c)$. The call is said to be *active* during that time interval. The *completion time* of the call is t_c+d_c.

A *schedule* for a set C of calls in G is an assignment of starting times and paths to the calls such that the sum of the bandwidth requirements of simultaneously active calls using the same edge does not exceed the capacity of that edge. The *makespan* (schedule length) of a schedule is the latest completion time of all calls, i.e., $\max_{c\in C} t_c+d_c$. Given a connected graph G with edge capacities and a set C of calls, the call scheduling problem is to compute a schedule for the calls such that the makespan is minimized. Like path coloring, call scheduling can be studied for undirected calls in undirected graphs (modeling, e.g., telephone conversations or video conferences) and for directed calls in bidirected graphs (modeling, e.g., movie transmissions).

We assume that all calls are available at time 0, can be delayed for an arbitrarily long time before they are established, and do not have to be completed by a certain deadline. There is no precedence relation among the calls. We consider the off-line version of the problem, where call durations are known in advance. Note that path coloring is equivalent to a special case of call scheduling. If all bandwidth requirements, edge capacities, and call durations are equal to 1, no two calls can use the same edge at the same time, and finding a schedule with minimum makespan is equivalent to finding an optimal assignment of paths and colors to the pairs (u_c, v_c): colors simply correspond to time steps.

1.3. Results

We study the complexity and the approximability of path coloring and call scheduling. Section 2 shows that undirected and directed path coloring are $\mathcal{NP}$-hard for ring networks. In Section 3 we consider path coloring in tree networks. First, in Section 3.1, we show that undirected path coloring in trees is equivalent to edge coloring of multigraphs. This has several implications: undirected path coloring can be solved optimally in polynomial time for trees of bounded degree, it is $\mathcal{NP}$-hard for trees of arbitrary degree (even if the tree has diameter two), approximating it with absolute approximation ratio $\frac{4}{3} - \varepsilon$ for any $\varepsilon > 0$ is $\mathcal{NP}$-hard, and any approximation algorithm for edge coloring of multigraphs can be translated into an approximation algorithm with the same (absolute and asymptotic) approximation ratio for undirected path coloring in trees of arbitrary degree (and vice versa). In Section 3.2, we show that directed path coloring is $\mathcal{NP}$-hard already for bidirected binary trees and that there can be no polynomial-time approximation algorithm (for bidirected trees of arbitrary degree) with absolute approximation ratio $\frac{4}{3} - \varepsilon$ for any $\varepsilon > 0$. Note that it remains a challenging open problem to prove lower bounds on the achievable asymptotic approximation ratio. In Section 3.3, we study the special case that the number of paths touching a node of the tree is bounded by $O((\log n)^{1-\varepsilon})$, where n is the number of nodes of the tree. We give polynomial-time algorithms to solve the (directed or undirected) path coloring problem to optimality using dynamic programming in this case.

Regarding call scheduling with arbitrary bandwidth requirements and call durations, we give some $\mathcal{NP}$-hardness results in Section 4. In particular, scheduling calls with arbitrary bandwidth requirements on a single link, scheduling calls with arbitrary durations in chain networks or in stars with degree at least three, and scheduling calls with either arbitrary bandwidth requirements or arbitrary durations in networks that contain at least two edge-disjoint paths between some pair of nodes are all $\mathcal{NP}$-hard.

Some of our results, which we have presented already at PASA'96 [11] and HICSS'97 [9], have been obtained independently by Kumar et al. [28]. They proved that undirected and directed path coloring can be solved optimally in polynomial time for networks of constant size; that undirected path coloring is $\mathcal{NP}$-hard for trees of diameter two, but can be solved optimally in polynomial time for trees of bounded degree; and that directed path coloring is $\mathcal{NP}$-hard for binary trees and for trees of diameter four with arbitrary degree.

1.4. Related work

A number of authors have studied path coloring problems. Early references consider the edge-intersection graphs of undirected paths in a tree, called EPT graphs. Vertex coloring of EPT graphs is obviously equivalent to undirected path coloring in trees. Golumbic and Jamison showed that vertex coloring for EPT graphs is $\mathcal{NP}$-hard [18]. Tarjan gave a (3/2)-approximation algorithm for coloring EPT graphs [39]. This approximation algorithm was rediscovered in the context of path coloring by Raghavan and Upfal [37] and improved to asymptotic approximation ratio $\frac{9}{8}$ by Mihail et al. [33].

Approximation algorithms for directed path coloring in trees were first presented in [33, 24, 30]. The best known algorithm [12, 25] uses at most $\lceil (5/3)L \rceil$ colors to color a set of paths with maximum load L. All approximation algorithms presented so far for directed path coloring in trees belong to a certain class of local greedy algorithms. It was shown that every deterministic local greedy algorithm uses at least $\lfloor (5/3)L \rfloor$ colors in the worst case, even for sets of paths that can be colored optimally with L colors [23]. Gargano et al. studied the all-to-all instance of path coloring in bidirected trees and showed that the optimal number of colors is equal to the maximum load [17].

For chain networks, the undirected and directed path coloring problems can be solved optimally in polynomial time, because the conflict graph of the paths is an interval graph and optimal vertex colorings for interval graphs can be computed efficiently [19].

For path coloring in ring networks, a 2-approximation algorithm can be obtained by cutting the ring at an arbitrary link and solving the path coloring problem on the resulting chain network. This approach works for undirected path coloring [37] and for directed path coloring [33]. Recently, Kumar gave a randomized approximation algorithm for undirected path coloring in rings that achieves asymptotic approximation ratio $1.5 + 1/2e \approx 1.68$ with high probability, if the optimal number of colors is $\omega(\ln n)$, where n is the number of nodes of the ring network [29]. Path coloring in bidirected ring networks was studied by Wilfong and Winkler [43]. They proved that there is a polynomial-time algorithm that computes a routing (assignment of paths to connection requests) that minimizes the maximum load. Denote the load of that routing by L_{opt}. They also showed that up to $2L_{\mathrm{opt}} - 1$ colors are required for some instances, and that it is $\mathcal{NP}$-hard to determine the minimum number of colors.

Path coloring is related to the edge-disjoint paths problem (i.e., deciding whether a set of pairs of nodes in a graph can be connected via edge-disjoint paths). In particular, an algorithm that computes an optimal solution to the path coloring problem can also be used to solve the edge-disjoint paths problem: a set of pairs of nodes can be connected via edge-disjoint paths if and only if the optimal number of colors used in a solution to the path coloring problem for these pairs of nodes is 1. Therefore, the path coloring problem is $\mathcal{NP}$-hard for all classes of graphs for which the edge-disjoint paths problem is $\mathcal{NP}$-complete, and no polynomial-time approximation algorithm with absolute approximation ratio $2 - \varepsilon$ can exist for path coloring in these classes of graphs. Examples for such classes of graphs are the planar graphs [32] and the partial k-trees [44]. Kramer and van Leeuwen [27] give an $\mathcal{NP}$-hardness proof for a wire routing problem that occurs in the context of VLSI theory, and simple modifications of that proof show that the edge-disjoint paths problem in undirected and bidirected mesh networks is $\mathcal{NP}$-complete. The best known approximation algorithm for undirected path coloring in meshes is due to Rabani and achieves approximation ratio polynomial in $\log\log n$, where n is the size of the mesh [36].

Good surveys on graph problems related to wavelength routing in all-optical networks can be found in [4] and [26], where path coloring is also discussed for all-to-all instances and permutation instances.

The on-line version of the undirected path coloring problem (where the connection requests are given to the algorithm one by one, and colors and paths must be assigned immediately without knowledge of future requests) was studied by Bartal and Leonardi [3]. They obtained deterministic on-line algorithms with competitive ratio $O(\log n)$ for networks with n nodes whose topology is that of a tree, a tree of rings, or a mesh. In addition, they presented a matching lower bound of $\Omega(\log n)$ for all on-line algorithms for undirected path coloring in meshes, and a lower bound of $\Omega(\log n/\log\log n)$ for trees. Leonardi and Vitaletti gave lower bounds for randomized on-line algorithms [31].

The call scheduling problem has not yet received comparable attention. Feldmann et al. [14] considered the on-line version of undirected call scheduling, where the scheduling algorithm does not have any knowledge about call durations or future calls. They studied chain networks and binary tree networks, assuming that all edges have the same capacity and that all calls are undirected. They showed that the greedy algorithm, i.e., the algorithm that always schedules the first call for which sufficient bandwidth is available, is $O(\log n)$-competitive for binary trees with n nodes provided that bandwidth requirements are either not too big or not too small, and they generalized this result to networks with small edge separators. For the case of unit bandwidth requirements and unit durations, they gave a set of calls in a binary tree such that the greedy algorithm produces a schedule that is longer than the optimal schedule by a factor of $\Omega(\log n)$. Furthermore, they presented a non-greedy algorithm for chain networks that is c-competitive for bandwidth requirements between $\frac{1}{c}$ and $\frac{1}{2}$. Additional results on on-line call scheduling in meshes and complete graphs can be found in [13]. Greedy algorithms for call scheduling in stars and trees were further studied in [10]. For calls with arbitrary bandwidth requirements and durations, algorithms with constant approximation ratio for stars and with approximation ratio $O(\log n)$ for trees were presented. A problem that is closely related to call scheduling is scheduling of file transfers. Complexity results and approximation algorithms for file transfer scheduling appeared in [7].

2. Path coloring in ring networks

The path coloring problem in ring networks is closely related to the arc-coloring problem, i.e., the problem of coloring circular-arc graphs. A graph $G=(V,E)$ is a circular-arc graph if its vertices can be represented by arcs of a circle such that there

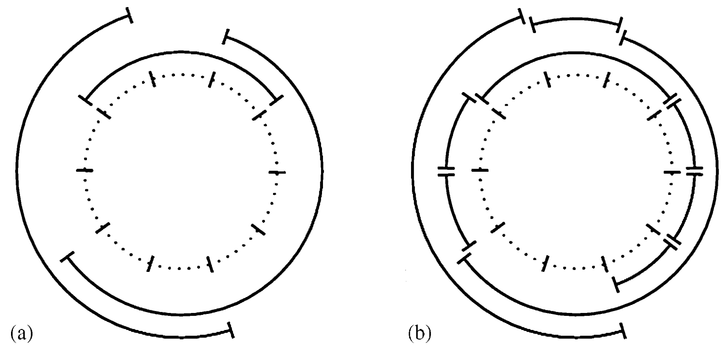

Fig. 1. (a) Instance of arc-coloring. (b) Adding short arcs.

is an edge between two vertices in G if and only if the corresponding arcs intersect. See part (a) of Fig. 1 for an example with three arcs that form a clique.

Given a family of circular arcs and a positive integer K, it is an $\mathcal{NP}$-complete problem to decide whether the arcs can be colored with K colors such that arcs with the same color do not intersect [16]. A 2-approximation algorithm for coloring circular-arc graphs was provided by Tucker [41]. Shih and Hsu improved this result and obtained a (5/3)-approximation algorithm [38]. A randomized arc-coloring algorithm that achieves asymptotic approximation ratio $1 + 1/e \approx 1.37$ with high probability under certain assumptions was given by Kumar [29]. Furthermore, it is known that *proper* circular-arc graphs (circular-arc graphs where no arc is contained within another arc) can be colored optimally in polynomial time [35].

If each connection request in an instance of the path coloring problem had to specify which of the two alternative paths in the ring it uses, the path coloring problem in ring networks would be equivalent to arc-coloring. Since we have the additional freedom to choose one of the two possible routes for each connection request, however, it is not immediately clear whether the path coloring problem for ring networks is $\mathcal{NP}$-hard. Nevertheless, we can show by a reduction from arc-coloring that path coloring is $\mathcal{NP}$-hard for undirected and for bidirected ring networks. (Another reduction that proves path coloring $\mathcal{NP}$-hard only for bidirected ring networks has been found independently by Wilfong and Winkler [43].)

Theorem 1. *Undirected and directed path coloring are $\mathcal{N}P$-hard in ring networks.*

Proof. (by reduction from arc-coloring). Let an instance I of the arc-coloring problem be given by a family of circular arcs and a positive integer K. We can assume that each point of the circle is contained in exactly K arcs (if some point is contained in fewer arcs, new short arcs can be introduced without changing the K-colorability; part (b) of Fig. 1 illustrates this for the case $K = 2$).

First, we transform I into an equivalent instance I' of arc-coloring in which every arc spans strictly less than half of the circle. For this purpose, we pick two antipodal points

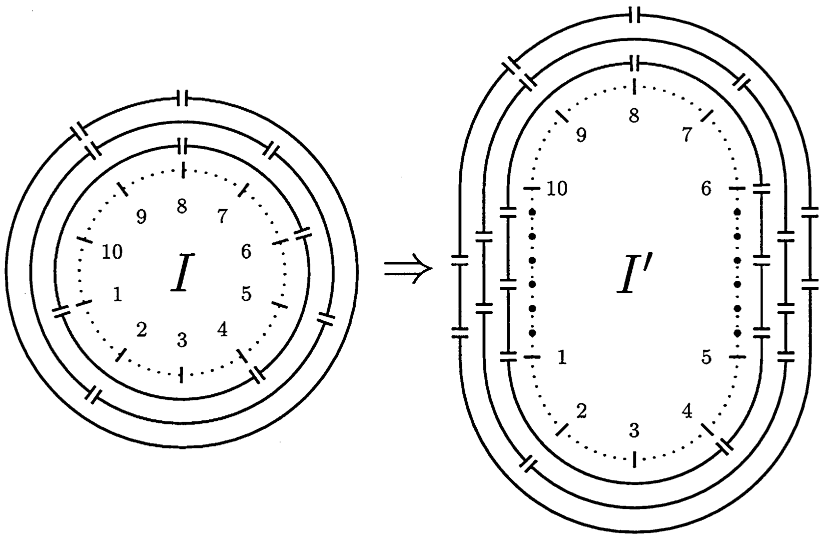

Fig. 2. Instances I and I' of the arc-coloring problem.

of the circle and insert $2K$ new points to the right of each of the two antipodal points. Within each of the two segments of new points, the jth arc, $1 \leqslant j \leqslant K$, containing that segment is split into three arcs by cutting it at the jth and at the $(K+j)$th new point. See Fig. 2 for an example with $K=3$. Obviously, the arcs in the resulting instance I' can be colored with K colors if and only if I can be colored with K colors.

If I' is interpreted as an instance I'' of path coloring in an undirected ring (i.e., every arc is viewed as a connection request between its endpoints), the connection requests in I'' can be routed and colored with K colors if and only if the arcs in I' can be colored with K colors. To see this, note that the load of all edges in the ring is K if every connection request in I'' is routed the short way; if at least one connection request is routed the long way, the load will be at least $K+1$ on some edge of the ring, and no K-coloring can exist.

Similarly, we can also obtain an instance of path coloring in bidirected rings from I': for every arc with endpoints a and b, we introduce two directed connection requests, one from a to b and one from b to a. □

3. Path coloring in tree networks

3.1. Undirected trees

For a set P of undirected paths (connection requests) in an undirected tree $G=(V,E)$, denote by P_v (for any $v \in V$) the subset of P that contains all paths touching node v. Paths in a set P_v intersect if and only if they share an edge incident to v. Hence, coloring the paths in a set P_v is equivalent to path coloring in a star. We call a coloring

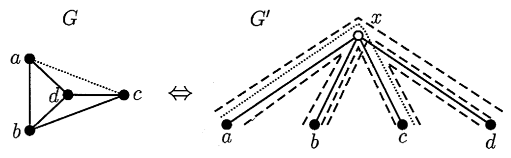

Fig. 3. The construction used in the proof of Theorem 3.

for the paths in a set P_v a *local coloring* and a coloring for all paths in P a *global coloring*. The difficulty of undirected path coloring in trees turns out to lie in the computation of optimal local colorings, while it is easy to combine local colorings for all sets P_v into a global coloring without increasing the number of colors.

Lemma 2. *Given local colorings $C_v : P_v \to \mathbb{N}$ for all $v \in V$, a global coloring $C : P \to \mathbb{N}$ with $|C| = \max_{v \in V} |C_v|$ can be computed in polynomial time.*

Proof. A simple merging process yields the desired coloring C. Initially, set $C = C_v$ for an arbitrary start node $v \in V$. Then visit all remaining nodes of the tree in depth-first search order. At each node w, rename the colors of C_w such that paths that are colored in C_w and in C receive the same color in both colorings. (Note that the node w is adjacent to exactly one previously processed node u, and that all paths that are colored in C and in C_w use the edge $\{u, w\}$.) Now the coloring C_w can be merged with C in the obvious way, and the algorithm terminates after $|V| - 1$ such merging steps. □

As a consequence, we get $OPT(P) = \max_{v \in V} OPT(P_v)$. Optimal local colorings can be merged into an optimal global coloring in polynomial time, but the following theorem implies that it is already $\mathcal{NP}$-hard to compute the optimal coloring for a set P_v. The theorem was proved in [18, Theorem 5], but we repeat the proof here because we will refer to the construction later on.

Theorem 3 (Golumbic and Jamison [18]). *Given a multigraph G, it is possible to compute in polynomial time a set P of paths in a star G' such that the conflict graph of P is* (*isomorphic to*) *the line graph of G, and vice versa.*

Proof. Given a multigraph $G = (V, E)$, the star $G' = (V', E')$ is constructed as follows. Set $V' = V \cup \{x\}$, where $x \notin V$, and $E' = \{\{v, x\} | v \in V\}$. Build P by including a path from v to w (in G') for every edge $\{v, w\} \in E$. Two paths in P intersect if and only if the corresponding edges in G share a vertex. Thus, the conflict graph of P is (isomorphic to) the line graph of G, as required. The other direction (constructing a multigraph from a given a set of paths in a star) is similar.

Fig. 3 illustrates the construction. The edges of the graph G on the left side correspond to paths in G' on the right side. The dotted path in G' corresponds to the dotted edge joining a and c in G. □

The theorem states that the class of graphs that can be obtained as conflict graphs of paths in a star is exactly the class of line graphs of multigraphs. Vertex coloring of line graphs is equivalent to edge coloring, and edge coloring was proved $\mathscr{NP}$-hard by Holyer [21]. Therefore, path coloring is $\mathscr{NP}$-hard for undirected paths in stars. This means that for given paths in a tree, not even optimal local colorings can be computed in polynomial time unless $\mathscr{P}=\mathscr{NP}$. On the other hand, the construction from the proof of Theorem 3 can be combined with Lemma 2 to show that approximation algorithms for edge coloring of multigraphs and for path coloring in trees are interchangeable. This relationship was used implicitly in [37, 33], but we consider it useful to state it explicitly. A function $f:\mathbb{N}_0 \to \mathbb{N}_0$ is called *non-decreasing* if $a \leqslant b$ implies $f(a) \leqslant f(b)$.

Theorem 4. *Let $f:\mathbb{N}_0 \to \mathbb{N}_0$ be an arbitrary non-decreasing function. If there is an approximation algorithm A for edge coloring of multigraphs that uses at most $f(\chi'(G))$ colors for any multigraph G, it is possible to derive an approximation algorithm B for path coloring in undirected trees such that B uses at most $f(OPT(P))$ colors for any given set P of paths, and vice versa.*

The theorem holds also if $f(OPT(P))$ and $f(\chi'(G))$ are replaced by $f(L)$ and $f(\Delta)$, respectively, where L is the maximum load of P and Δ is the maximum degree of G. Theorem 4 implies that approximation algorithms for edge coloring of multigraphs can be converted to approximation algorithms for path coloring in undirected trees with the same absolute and asymptotic approximation ratio, and vice versa. An approximation algorithm for path coloring in undirected trees with approximation ratio $\frac{4}{3}-\varepsilon$ for some $\varepsilon > 0$ could be used to decide whether the edges of a given multigraph can be colored with three colors or not. As the edges of a 3-regular simple graph can always be colored with either three or four colors and it is $\mathscr{NP}$-hard to decide whether three colors suffice [21], no polynomial-time approximation algorithm with approximation ratio $\frac{4}{3}-\varepsilon$ can exist for undirected path coloring in stars or trees unless $\mathscr{P}=\mathscr{NP}$.

In [34], Nishizeki and Kashiwagi presented an approximation algorithm for edge coloring that uses at most $\lfloor 1.1\cdot\chi'(G)+0.8\rfloor$ colors for any multigraph G. Furthermore, if the edges of a multigraph G can be colored with one or two colors, an optimal coloring can be obtained in polynomial time (by coloring the line graph of G, which is a bipartite graph in this case). If $\chi'(G)\geqslant 3$, we have $\lfloor 1.1\cdot\chi'(G)+0.8\rfloor \leqslant (4/3)\chi'(G)$. Therefore, there is a polynomial-time algorithm for edge coloring of multigraphs with absolute approximation ratio $\frac{4}{3}$ and asymptotic approximation ratio 1.1. Using Theorem 4, this implies the following corollary.

Corollary 5. *There is a polynomial-time approximation algorithm that colors a given set P of undirected paths in an undirected tree using $OPT(P)$ colors if $OPT(P)\leqslant 2$ and at most $\lfloor 1.1\cdot OPT(P)+0.8\rfloor$ colors otherwise. The algorithm has absolute approximation ratio $\frac{4}{3}$ and asymptotic approximation ratio* 1.1.

It is conjectured in [20, p. 394] that ultimately an approximation algorithm for edge coloring any multigraph G using at most $\chi'(G) + 1$ colors will be found; such an algorithm would immediately give an approximation algorithm that colors a set P of paths in an undirected tree using at most $OPT(P) + 1$ colors.

3.1.1. Undirected trees of bounded degree

Assume that the degree of the given tree is bounded by a constant c. Consider a node v of the tree and the set P_v of paths in P that touch v. The number of vertices (but not the number of edges) of the multigraph G constructed from P_v according to the proof of Theorem 3 is bounded by c as well. Therefore, the number of different types of color classes (sets of edges that can be assigned the same color, making no distinction between parallel edges) is bounded by a constant (dependent only on c), and an optimal edge coloring can be computed in polynomial time either by integer linear programming with fixed number of variables or by dynamic programming (cf. bin-packing with a bounded number of different item sizes [20, pp. 64–65], scheduling of multiprocessor tasks with bounded parallelism and unit execution time [5], or scheduling of multiprocessor tasks with unit execution time and prespecified processor allocations in the case that the number of processors is constant [22]). Thus, optimal local colorings can be computed in polynomial time and merged into an optimal global coloring by Lemma 2.

Theorem 6. *In trees whose degree is bounded by a constant, undirected path coloring can be solved optimally in polynomial time.*

If the maximum degree of the tree is bounded by 3, i.e., if the tree is a binary tree, a very simple algorithm (simpler than dynamic programming or ILP) can compute the optimal path coloring. The algorithm proceeds color by color and processes the nodes of the tree in depth-first search order for each color a. At a node v, it greedily selects uncolored paths that touch v and that do not intersect a path that is already colored with a, giving preference to paths that do not start or end at v; the selected paths are assigned color a.

3.2. Bidirected trees

Now we study the complexity of directed path coloring in trees. Interestingly, the directed path coloring problem can be solved efficiently in bidirected trees of diameter two (by reduction to the edge coloring problem in a bipartite graph [33]). The difficulty of directed path coloring in trees lies in combining the local colorings into a global coloring.

Unlike in the undirected case, even restricting the degree of the tree does not make path coloring in bidirected trees easier. We use a reduction from arc-coloring to show that directed path coloring is $\mathcal{NP}$-hard for binary trees.

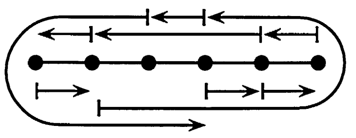

Fig. 4. Viewing arcs as directed paths in a bidirected chain.

Let an instance I of arc-coloring be given as in the proof of Theorem 1, with the same assumptions. One can view the given arcs as directed paths in a unidirectional ring network such that two paths share an edge if and only if the corresponding arcs intersect. Now, imagine the ring being "squashed" so as to yield a bidirected chain network: the directed edges from the left end of the chain to the right end, followed by the directed edges from the right end of the chain to the left end, correspond to the original unidirectional ring (Fig. 4 depicts the chain resulting from the circular arcs shown in part (b) of Fig. 1). However, the resulting paths in the chain are not necessarily simple; some may turn around at the left end or at the right end (or at both ends) of the chain. To prohibit paths from taking short-cuts, we split each problematic path into two or three simple paths, and ensure that all paths originating from the same original path are assigned the same color in any K-coloring. For this purpose, we extend the chain by K additional nodes on each side, and attach a distinct binary tree with three nodes to each of the $2K$ new nodes.

Original paths that do not turn around at either end of the chain need not be split. For each path (a,b) that turns around at the right end of the chain, we select a distinct binary subtree attached to the right side of the chain, and we split the path into a path from a to the left leaf of that binary tree and a path from the right leaf of that binary tree to b. In addition, we add $K-1$ paths (called *blockers*) from the right leaf to the left leaf of each binary subtree in order to ensure that the two paths turning around at that subtree receive the same color in any K-coloring. An analogous procedure is applied to paths turning around at the left end of the chain. Fig. 5 sketches the paths created for different cases of original paths.

It is easy to see that the resulting paths in the bidirected binary tree can be colored with K colors if and only if the original arcs can be colored with K colors. Hence, path coloring is $\mathscr{NP}$-hard for bidirected binary trees.

The following argument shows that no polynomial-time algorithm can achieve absolute approximation ratio $\frac{4}{3}-\varepsilon$ for any $\varepsilon > 0$ if $\mathscr{P} \neq \mathscr{NP}$. Recall that it is $\mathscr{NP}$-complete to decide whether a 3-regular simple graph can be edge-colored with three colors or not [21]. For a given 3-regular simple graph G, one can construct an instance of path coloring in a bidirected tree G' such that the paths can be colored with three colors if and only if G can be edge-colored with three colors. The basic idea of the construction is the same as in Theorem 3. However, we introduce a pair of directed paths in G' for each edge in G (instead of a single undirected path). In order to ensure that the directed paths in G' created for one edge of G receive the same color, blockers in attached binary trees with three nodes are used in the same way as above (see Fig. 6).

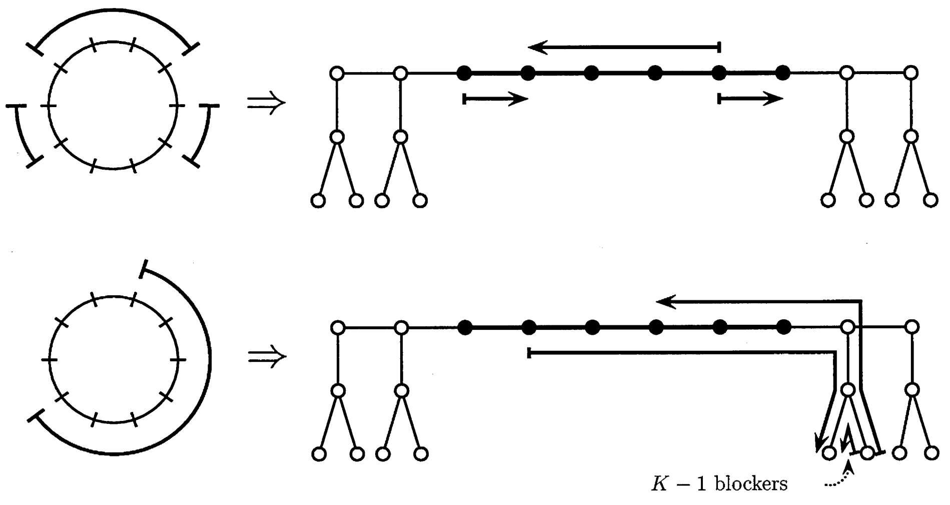

Fig. 5. Arcs and corresponding paths.

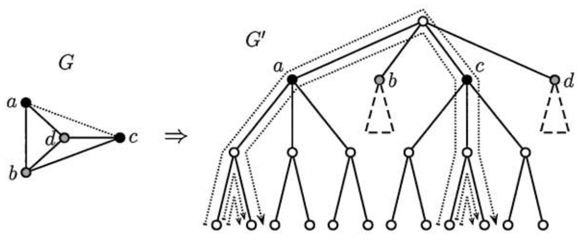

Fig. 6. Reduction from edge coloring. The subtrees of G' rooted at b and d are not shown. Only the six paths in G' corresponding to the edge $\{a,c\}$ in G are depicted.

Theorem 7. *Directed path coloring is $\mathcal{NP}$-hard even for binary trees. For a given set of paths in a bidirected tree of arbitrary degree, it is $\mathcal{N}P$-hard to decide whether three colors suffice, and there cannot be a polynomial-time algorithm with absolute approximation ratio $\frac{4}{3} - \varepsilon$ for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.*

3.3. Bounding the number of paths through a single node

In the following, we examine the case when the number of paths touching a single node of the tree is bounded. Such an assumption may be valid in real telecommunication applications if each customer submits only a limited number of connection requests to the network and if most calls are local calls.

Theorem 8. *Let $G = (V,E)$ be a tree network with n nodes, and let P be a set of undirected or directed paths in G. If the number of paths touching any single node of G is at most c, where $c = \mathrm{O}((\log n)^{1-\varepsilon})$ for some fixed $\varepsilon > 0$, an optimal coloring of the paths can be computed in polynomial time.*

Proof. Let L be the maximum load of the paths. Obviously, we have $L \leqslant c$. The optimal number of colors is at least L, and $2L$ is a trivial upper bound [37, 33]. The polynomial-time dynamic programming algorithm given below can be used to decide for any given $k \in \{L, L+1, \ldots, 2L\}$ whether P can be colored with k colors and, if so, produce a k-coloring. Linear or binary search for the optimal value of k in the range from L to $2L$ can then be used to find an optimal coloring in polynomial time.

For a node v of G, we call every k-coloring of the paths touching v a *local k-coloring at v*. The number of different local k-colorings at v is bounded by k^c, which is polynomial in n for $k \leqslant 2c$. To see this, note that $c = \mathrm{O}((\log n)^{1-\varepsilon})$ implies $c \leqslant d(\log n)^{1-\varepsilon}$ for some constant d. Hence, we get $(2c)^c \leqslant (2d(\log n)^{1-\varepsilon})^{d(\log n)^{1-\varepsilon}} = n^{(1+\log d+(1-\varepsilon)\log\log n)d(\log n)^{-\varepsilon}}$. Furthermore, as $\alpha := (1+\log d+(1-\varepsilon)\log\log n)d(\log n)^{-\varepsilon}$ approaches 0 as n goes to infinity, we have $\alpha \leqslant D$ for some constant D, and we obtain $(2c)^c \leqslant n^D$.

A local k-coloring at v is called *valid* if it can be extended to a k-coloring of all paths touching nodes in the subtree rooted at v. At a leaf v, all local k-colorings are valid. During a bottom-up traversal of the tree, the algorithm computes the set of all valid local k-colorings for all nodes. At each node v, it first computes the set of all local k-colorings at v; then it considers every local k-coloring C at v and discards it if there is a child w of v such that there is no valid local k-coloring C' at w that is consistent with C, where C and C' are consistent if they assign the same colors to the paths touching v and w. Exactly, the valid local k-colorings at v are not discarded. A k-coloring of all paths exists if and only if there is a valid local k-coloring at the root of the tree, and appropriate bookkeeping allows the algorithm to output a k-coloring of all paths in the end if it exists. □

In fact, the algorithm from the proof of Theorem 8 that computes a k-coloring, if it exists, can be modified so as to compute a coloring with the minimum number of colors, if any k-coloring exists. With this modification, it is sufficient to call the algorithm with $k = 2L$, and no search for the optimal value of k is necessary.

Furthermore, we observe that it is possible, for any given k and Δ such that $k\Delta = \mathrm{O}((\log n)^{1-\varepsilon})$, to decide in polynomial time whether a given set of paths in a tree with n nodes and with maximum degree at most Δ can be colored with k colors and, if so, to output a k-coloring. If $L > k$, no k-coloring can exist; otherwise, the number of paths touching a single node of the tree is bounded by $k\Delta$ (or $2k\Delta$, if the tree is bidirected), and Theorem 8 is applicable.

4. Call scheduling with arbitrary durations and bandwidths

As mentioned in the introduction, the path coloring problem is equivalent to call scheduling if all edge capacities, bandwidth requirements and call durations are equal to 1. Now we show that call scheduling with either arbitrary call durations or arbitrary bandwidth requirements is $\mathcal{NP}$-hard for virtually every network topology,

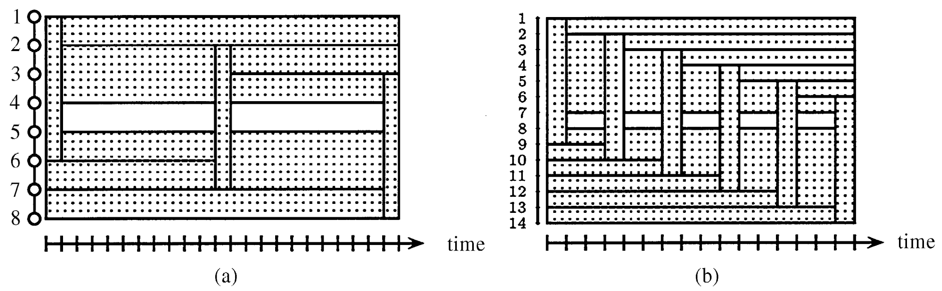

Fig. 7. (a) Reduction from PARTITION. (b) Reduction from 3-PARTITION.

both in the directed and in the undirected case and even if all edges have the same capacity.

The $\mathcal{NP}$-hardness is shown by reduction from PARTITION or 3-PARTITION. Recall that an instance of the $\mathcal{NP}$-complete problem PARTITION [15] is given by a finite set A with size $s(a) \in \mathbb{N}$ for each $a \in A$ such that $\sum_{a\in A} s(a)$ is even. The question is whether there is a subset $A' \subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A\setminus A'} s(a)$. Similarly, an instance of the strongly $\mathcal{NP}$-complete problem 3-PARTITION [15] is given by a set A of $3m$ items, a positive integer bound B, and a size $s(a) \in \mathbb{N}$ satisfying $B/4 < s(a) < B/2$ for each $a \in A$ such that $\sum_{a\in A} s(a) = mB$. Here, the question is whether A can be partitioned into m disjoint sets such that the sum of the item sizes in each set is equal to B.

Consider calls with unit bandwidth requirements, but with arbitrary durations. In this case, call scheduling in chain networks is equivalent to the one-dimensional layout compaction problem, whose decision version is strongly $\mathcal{NP}$-complete [8]. The proof of $\mathcal{NP}$-completeness can be adapted to call scheduling immediately. Given an instance of PARTITION or 3-PARTITION, it is possible to specify a set of frame-calls (shown dotted in Fig. 7) that can be scheduled optimally only such that one particular edge of the chain network remains idle during separate time intervals of length B. In addition, a call of duration $s(a)$ using only that idle link is generated for each $a \in A$. These calls can be scheduled within the gaps left by the frame-calls if and only if A can be partitioned into two (in the case of PARTITION, see part (a) of Fig. 7) or m (in the case of 3-PARTITION, see part (b) of Fig. 7) equal-size subsets. Therefore, we have $\mathcal{N}P$-hardness for chain networks with at least 8 nodes and strong $\mathcal{N}P$-hardness for arbitrary chain networks. This implies $\mathcal{NP}$-hardness for tree networks with diameter at least 8 and strong $\mathcal{NP}$-hardness for arbitrary tree networks.

Scheduling calls with unit bandwidth requirements in undirected stars with unit edge capacities is equivalent to scheduling multiprocessor tasks with prespecified processor allocations if each task requests one or two processors. (Processors correspond to edges of the star, tasks correspond to calls, and task durations correspond to call durations.) The computational complexity of this multiprocessor scheduling problem was investigated by Hoogeveen et al. [22]. They proved the problem strongly $\mathcal{NP}$-hard for three processors, which implies that call scheduling with unit edge capacities, unit bandwidth requirements, and arbitrary durations is strongly $\mathcal{NP}$-hard in a star of degree three.

If we have unit call durations but allow arbitrary bandwidth requirements, call scheduling is already $\mathcal{NP}$-hard for a chain network consisting of a single link, because it is equivalent to the bin-packing problem. In particular, PARTITION and 3-PARTITION can be reduced to the call scheduling problem with arbitrary bandwidth requirements on a single link, implying that the latter is strongly $\mathcal{NP}$-hard.

Finally, we consider networks that contain two nodes u and v with at least two edge-disjoint paths connecting u and v. The maximum number k of edge-disjoint paths between some nodes u and v can be computed in polynomial time with a maximum flow algorithm [2]. In a network G with k edge-disjoint paths between u and v, call scheduling is $\mathcal{NP}$-hard for calls with unit duration and arbitrary bandwidth requirements and for calls with arbitrary duration and unit bandwidth requirements. Given an instance of PARTITION, a call from u to v with bandwidth or duration $s(a)$ is generated for each $a \in A$, and $k-2$ additional calls that ensure that the former calls use exactly two of the edge-disjoint paths. Hence, these calls must be partitioned into two equal-size subsets in order to obtain an optimal schedule.

References

[1] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswami, B. Schieber, M. Sudan, Efficient routing in optical networks, J. ACM 46(6) (1996) 973–1001.

[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[3] Y. Bartal, S. Leonardi, On-line routing in all-optical networks, in: Proc. 24th Internat. Coll. on Automata, Languages and Programming ICALP'97, Lecture Notes in Computer Science, vol. 1256, pp. 516–526, Springer, Berlin, 1997.

[4] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Perennes, U. Vaccaro, Graph problems arising from wavelength-routing in all-optical networks, in: Proc. of IPPS'97, 2nd Workshop on Optics and Computer Science (WOCS), 1997.

[5] J. Blazewicz, M. Drabowski, J. Weglarz, Scheduling multiprocessor tasks to minimize schedule length, IEEE Trans. Comput. c-35(5) (1986) 389–393.

[6] N.K. Cheung, K. Nosu, G. Winzer (Eds.), Special issue on Dense Wavelength Division Multiplexing Techniques for High Capacity and Multiple Access Communication Systems, IEEE J. Selected Areas Commun. 8(6) IEEE Communications Society, August 1990.

[7] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, A.S. Lapaugh, Scheduling file transfers, SIAM J. Comput. 14(3) (1985) 744–780.

[8] J. Doenhardt, T. Lengauer, Algorithmic aspects of one-dimensional layout compaction, IEEE Trans. Computer-Aided Des. CAD-6(5) (1987) 863–878.

[9] T. Erlebach, K. Jansen, Call scheduling in trees, rings and meshes, in: Proc. 30th Hawaii Internat. Conf. on System Sciences HICSS-30, Vol. 1, IEEE Computer Society Press, 1997, pp. 221–222.

[10] T. Erlebach, K. Jansen, Off-line and on-line call-scheduling in stars and trees, in: Proc. 23rd Internat. Workshop on Graph-Theoretic Concepts in Computer Science WG'97, Lecture Notes in Computer Science, vol. 1335, pp. 199–213, Springer, Berlin, 1997.

[11] T. Erlebach, K. Jansen, Scheduling of virtual connections in fast networks, in: Proc. 4th Parallel Systems and Algorithms Workshop PASA'96, World Scientific Publishing, Singapore, 1997, pp. 13–32.

[12] T. Erlebach, K. Jansen, C. Kaklamanis, P. Persiano, An optimal greedy algorithm for wavelength allocation in directed tree networks, in: Proc. DIMACS Workshop on Network Design: Connectivity and Facilities Location, vol. 40 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, 1998, pp. 117–129.

[13] A. Feldmann, On-line call admission for high-speed networks (Ph.D. Thesis), Technical Report CMU-CS-95-201, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1995.

[14] A. Feldmann, B. Maggs, J. Sgall, D.D. Sleator, A. Tomkins, Competitive analysis of call admission algorithms that allow delay, Technical Report CMU-CS-95-102, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January 1995.

[15] M.R. Garey, D.S. Johnson, Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness, Freeman, New York-San Francisco, 1979.

[16] M.R. Garey, D.S. Johnson, G.L. Miller, C.H. Papadimitriou, The complexity of coloring circular arcs and chords, SIAM J. Algebraic Discrete Meth. 1(2) (1980) 216–227.

[17] L. Gargano, P. Hell, S. Perennes, Colouring paths in directed symmetric trees with applications to WDM routing, in: Proc. 24th Internat. Coll. on Automata, Languages and Programming ICALP'97, Lecture Notes in Computer Science, vol. 1256, pp. 505–515, Springer, Berlin, 1997.

[18] M.C. Golumbic, R.E. Jamison, The edge intersection graphs of paths in a tree, J. Comb. Theory Series B 38(1) (1985) 8–22.

[19] U.I. Gupta, D.T. Lee, J.Y.-T. Leung, Efficient algorithms for interval graphs and circular-arc graphs, Networks 12 (1982) 459–467.

[20] D.S. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, Boston, MA, 1997.

[21] I. Holyer, The NP-completeness of edge-coloring, SIAM J. Comput. 10(4) (1981) 718–720.

[22] J.A. Hoogeveen, S.L. van de Velde, B. Veltman, Complexity of scheduling multiprocessor tasks with prespecified processor allocations, Discrete Appl. Math. 55 (1994) 259–272.

[23] K. Jansen, Approximation results for wavelength routing in directed trees, in: Proc. IPPS'97, 2nd Workshop on Optics and Computer Science (WOCS), 1997.

[24] C. Kaklamanis, P. Persiano, Efficient wavelength routing on directed fiber trees, in: Proc. 4th Annual European Symp. on Algorithms ESA'96, Lecture Notes in Computer Science, vol. 1136, pp. 460–470, Springer, Berlin, 1996.

[25] C. Kaklamanis, P. Persiano, T. Erlebach, K. Jansen, Constrained bipartite edge coloring with applications to wavelength routing, in: Proc. 24th Internat. Coll. on Automata, Languages and Programming ICALP'97, Lecture Notes in Computer Science, vol. 1256, pp. 493–504, Springer, Berlin, 1997.

[26] R. Klasing, Methods and problems of wavelength-routing in all-optical networks, Technical Report CS-RR-348, Department of Computer Science, University of Warwick, September 1998, Presented as invited talk at the MFCS'98 Workshop on Communications.

[27] M.E. Kramer, J. van Leeuwen, The complexity of wire routing and finding the minimum area layouts for arbitrary VLSI circuits, in: F.P. Preparata (Ed.), Advances in Computing Research; VLSI Theory, Vol. 2, JAI Press Inc., Greenwich, CT-London, 1984, pp. 129–146.

[28] S.R. Kumar, R. Panigrahy, A. Russel, R. Sundaram, A note on optical routing on trees, Inf. Process. Lett. 62 (1997) 295–300.

[29] V. Kumar, Approximating circular arc coloring and bandwidth allocation in all-optical ring networks, in: Proc. Internat. Workshop on Approximation Algorithms for Combinatorial Optimization APPROX'98, Lecture Notes in Computer Science, vol. 1444, pp. 147–158, Springer, Berlin, 1998.

[30] V. Kumar, E.J. Schwabe, Improved access to optical bandwidth in trees, in: Proc. 8th Ann. ACM–SIAM Symp. on Discrete Algorithms SODA'97, 1997, pp. 437–444.

[31] S. Leonardi, A. Vitaletti, Randomized lower bounds for online path coloring, in: Proc. 2nd Internat. Workshop on Randomization and Approximation Techniques in Computer Science, Lecture Notes in Computer Science, vol. 1518, pp. 232–247, Springer, Berlin, 1998.

[32] M. Middendorf, F. Pfeiffer, On the complexity of the disjoint paths problem, Combinatorica 13(1) (1993) 97–107.

[33] M. Mihail, C. Kaklamanis, S. Rao, Efficient access to optical bandwidth. in: Proc. 36th Ann. Symp. on Foundations of Computer Science FOCS'95, 1995, pp. 548–557.

[34] T. Nishizeki, K. Kashiwagi, On the 1.1 edge-coloring of multigraphs, SIAM J. Disc. Math. 3(3) (1990) 391–410.

[35] J.B. Orlin, M.A. Bonuccelli, D.P. Bovet, An $O(n^2)$ algorithm for coloring proper circular arc graphs, SIAM J. Algebraic Discrete Meth. 2(2) (1981) 88–93.

[36] Y. Rabani, Path coloring on the mesh, in: Proc. 37th Ann. Symp. on Foundations of Computer Science FOCS'96, 1996, pp. 400–409.

[37] P. Raghavan, E. Upfal, Efficient routing in all-optical networks, in: Proc. 26th Ann. ACM Symp. on Theory of Computing STOC'94, 1994, pp. 134–143.
[38] W.-K. Shih, W.-L. Hsu, An approximation algorithm for coloring circular-arc graphs, in: SIAM Conf. on Discrete Mathematics, 1990.
[39] R.E. Tarjan, Decomposition by clique separators, Discrete Math. 55 (1985) 221–232.
[40] The ATM Forum, Upper Saddle River, NJ, ATM User-Network Interface (UNI) Specification Version 3.1., 1995.
[41] A. Tucker, Coloring a family of circular arcs, SIAM J. Appl. Math. 29(3) 1975, 493–502.
[42] R.J. Vetter, ATM concepts, architectures, and protocols, Commun. ACM 38(2) (1995) 30–38.
[43] G. Wilfong, P. Winkler, Ring routing and wavelength translation, in: Proc. 9th Ann. ACM-SIAM Symp. on Discrete Algorithms SODA'98, 1998, pp. 333–341.
[44] X. Zhou, T. Nishizeki, The edge-disjoint paths problem is $\mathcal{NP}$-complete for partial k-trees, in: K.-Y. Chwa, O.H. Ibarra (Eds.), Proc. 9th Ann. Internat. Symp on Algorithms and Computation ISAAC'98, Lecture Notes in Computer Science, vol. 1533, pp. 417–426, Springer, Berlin, 1998.