Algorithmica (2005) 43: 43–61
DOI: 10.1007/s00453-005-1157-y

Algorithmica

Wavelength Conversion in All-Optical Networks with Shortest-Path Routing[1]

Thomas Erlebach[2] and Stamatis Stefanakos[3]

Abstract. We consider all-optical networks with shortest-path routing that use wavelength-division multiplexing and employ wavelength conversion at specific nodes in order to maximize their capacity usage. We present efficient algorithms for deciding whether a placement of wavelength converters allows the network to run at maximum capacity, and for finding an optimal wavelength assignment when such a placement of converters is known. Our algorithms apply to both undirected and directed networks. Furthermore, we show that the problem of designing such networks, i.e., finding an optimal placement of converters, is MAX SNP-hard in both the undirected and the directed case. Finally, we give a linear-time algorithm for finding an optimal placement of converters in undirected triangle-free networks, and show that the problem remains $\mathcal{NP}$-hard in bidirected triangle-free planar networks.

Key Words. Wavelength assignment, Wavelength converter placement, Graph algorithm, Edge coloring, Bipartization, Sufficient set, L-reduction.

1. Introduction.

All-optical networks are emerging as a promising solution for meeting the rapidly increasing bandwidth demand. In these networks optical switches are employed to avoid the bottleneck of opto-electronic conversions, and wavelength-division multiplexing is used to partition the optical bandwidth into channels that carry data at rates manageable by electronic network elements. A connection between two nodes must be carried through a single channel that operates on a different wavelength than any other channel with which it shares a fiber. Wavelength converters are being used to exploit fully the available capacity of the network. A converter, when placed at some node of the network, has the ability of altering the operating wavelength of any channel that goes through that node.

Several interesting algorithmic problems arise in such networks. The wavelength assignment problem asks for an assignment of a minimum number of wavelengths to a given set of connections such that no conflicts occur. If wavelength converters are used, the wavelength of a connection can change whenever its route goes through a converter,

[1] This research was partially supported by the Swiss National Science Foundation under Contract No. 21-63563.00 (Project AAPCN) and the EU Thematic Network APPOL II (IST-2001-32007), with funding provided by the Swiss Federal Office for Education and Science (BBW). An extended abstract of this paper has appeared in the *Proceedings of the* 14*th International Symposium on Algorithms and Computation* (*ISAAC* 2003), pp. 595–604, LNCS 2906, Springer-Verlag, Berlin, 2003.
[2] Department of Computer Science, University of Leicester, Leicester LE1 7RH, England. t.erlebach@mcs.le.ac.uk.
[3] Computer Engineering and Networks Laboratory (TIK), ETH Zürich, CH-8092 Zürich, Switzerland. stefanak@tik.ee.ethz.ch.

Received January 2004; revised August 2004. Communicated by L. Zhang.
Online publication June 14, 2005.

otherwise each connection must be carried through the same wavelength all the way from sender to receiver. The use of wavelength converters can decrease the number of necessary wavelengths for routing a given set of connections. Ideally, one should be able to route any set of connections that induce congestion L (i.e., at most L connections share a common fiber) with L wavelengths. This is the case when all the nodes of the network are equipped with converters. Due to the high cost of such devices, however, one is faced with a network design problem, namely, to achieve the optimal capacity usage by placing a minimum number of converters on the network.

In this paper we study issues related to the design of networks with wavelength converters, and to the problem of wavelength assignment in networks that already have conversion capabilities. We restrict ourselves to the practical scenario where shortest-path routing is used in the network; if one does not take into account the specific routing algorithm that will be used and allows arbitrary routings, the requirements of converters turn out to be unduly pessimistic. We address the following questions:

(i) Can we efficiently decide whether a given placement of wavelength converters results in optimal capacity usage?
(ii) Can we efficiently find an optimal wavelength assignment when such a placement of converters is known?
(iii) Can we efficiently find a placement of a minimum number of converters that results in optimal capacity usage?

We answer the first two questions in the affirmative. For the third question we are able to do so only for a special case; for arbitrary networks we show that the problem is MAX SNP-hard, while for a different special case we show that it remains $\mathcal{NP}$-hard.

Preliminaries. We model the network by a graph $G = (V, E)$. Throughout the paper, G is assumed to be connected. We consider both undirected and directed graph models since both are of interest for optical networks. If the fiber allows two-way communication then we model the network as an undirected graph, while if the fiber allows one-way communication, we model the network as a directed graph. A special case of directed graphs that are of particular interest are *bidirected graphs*: in these graphs, for every directed edge the oppositely directed edge also exists. Connections in the network can be seen as paths in G, and wavelengths can be regarded as colors. If the network does not employ wavelength conversion then a *wavelength assignment* or *coloring* for a set $\mathcal{P}$ of paths is an assignment of a color to each path in $\mathcal{P}$. A *valid coloring* is one in which no two paths that use the same edge get assigned the same color. In the presence of wavelength converters, a coloring is an assignment of a color to every edge of each path. In that case we say that a coloring is *valid with respect to* a vertex-set S (on which the converters are placed) if it satisfies the additional constraint that the color assignments to two consecutive edges of a path differ only if their incident vertex is in S. We denote the *load* or congestion of the network that is induced by a set of paths $\mathcal{P}$ by $L(\mathcal{P}) = \max_{e \in E} L_e(\mathcal{P})$, where $L_e(\mathcal{P})$ is the number of paths in $\mathcal{P}$ that use edge e. We denote the degree of a vertex v by $\deg(v)$ and the maximum degree of G by $\Delta(G)$ (in a directed graph, the degree of a vertex is its degree in the underlying simple undirected graph). The *diameter* of a graph is the number of edges on the longest shortest path. For any graph-theoretic terms or notation not defined here we refer the reader to [4].

Most networks that do not employ wavelength conversion are destined to waste capacity. Consider an undirected graph with a vertex v of degree at least 3 and assume that only two wavelengths are available. The paths between three neighbors of v that go through v have load 2, but require three colors for a valid coloring if v is not equipped with a converter. Therefore, not all three paths can be simultaneously routed, and, furthermore, two of the edges incident to v will only use one of the two available wavelengths resulting in a 50% waste of capacity. To avoid this, for a graph $G = (V, E)$ we need a placement of wavelength converters on a subset of its vertices $S \subseteq V$ such that any set of paths $\mathcal{P}$ in G can be colored with $L(\mathcal{P})$ colors with respect to S. A set of vertices S that has this property is called a *sufficient set*. If the network uses shortest-path routing, we require that any set $\mathcal{P}$ of shortest paths can be colored with $L(\mathcal{P})$ colors with respect to S. We refer to this kind of sufficient sets for shortest-path routings as *SP-sufficient sets*. Note that an SP-sufficient set might be substantially smaller than a sufficient set. A vertex of degree greater than or equal to 3, for example, might not require a converter if its neighborhood does not contain an independent set of size 3. We say that an induced $K_{1,3}$ is a *claw*; the vertex adjacent to the independent set of size 3 is its *center*.

Previous Work. Wilfong and Winkler [17] showed that the only bidirected graphs that admit the empty sufficient set are spiders, i.e., trees with at most one vertex of degree greater than 2, and that rings (cycles) admit a sufficient set of size 1. They provided an efficient way of determining whether a set S is sufficient for a bidirected graph G: One modifies G by "exploding" each node $s \in S$ into degree-of-s-many copies, each of which is made adjacent to one of the old neighbors of s. The set S is sufficient for G if and only if every component of the graph obtained after this modification is a spider. Concerning the problem of finding a minimum sufficient set (MIN SUFFICIENT SET), they showed that it is $\mathcal{NP}$-hard even for planar bidirected graphs. Kleinberg and Kumar [12] gave a 2-approximation algorithm for arbitrary directed graphs and a polynomial-time approximation scheme for directed planar graphs. Their approach can be extended to give a linear-time algorithm for MIN SUFFICIENT SET in directed graphs of bounded treewidth [5]. Sufficient sets in undirected graphs were studied in [5]. Therein, it was shown that lines (paths) are the only undirected graphs that do not need converters, and an optimal polynomial-time algorithm for finding a minimum sufficient set was given. Networks with shortest-path routings were considered in [6]. A complete characterization of the undirected graphs which admit the empty SP-sufficient set, as well as efficient optimal coloring algorithms for this class of graphs, were given.

Wavelength converters of bounded degree have also been studied in the literature. A converter of bounded degree does not have full conversion capabilities (as in the case we study here), but it can transform wavelength i to only a few other specified wavelengths. We do not discuss these results here, but refer the reader to [1] and the references therein for more details.

Motivation. The characterization of the undirected graphs which admit the empty SP-sufficient set given in [6] implies that the restriction to shortest-path routings can reduce the converter requirements of a given network significantly. Furthermore, the known algorithms for MIN SUFFICIENT SET from [5] and [12] indicate that by allowing arbitrary routings we end up with unduly pessimistic placements of converters since they can result in picking half or even all vertices of degree greater than 2. Because of the high

cost of wavelength converters, and since optical networks typically employ simple fixed-routing strategies (i.e., for every pair of nodes there is a prespecified path through which all traffic between these nodes will be carried; see [18]), it is important to consider the problem of converter placement under such practical scenarios and avoid the traditional worst-case analysis for arbitrary paths. In this paper we mainly consider networks with shortest-path routing, but as we will see the algorithms we propose can be adapted to work for other fixed routings as well. From a theoretical point of view, it is interesting to observe how the restriction to shortest-path routings changes the nature of the problem, thus requiring completely different methods for tackling it than the ones used for MIN SUFFICIENT SET in [5], [12], and [17]. For example, the "exploding" technique from [17] cannot be used in our setting, since it modifies the distances of the vertices.

Our Results. In Section 2 we give a polynomial-time algorithm for deciding whether for a given graph $G = (V, E)$, a subset $S \subseteq V$ is SP-sufficient, and we give a polynomial-time algorithm for optimally coloring any set of shortest paths in G with respect to a valid SP-sufficient set. Both results apply to directed and undirected graphs. This generalizes the result of [6], which only allows us to decide whether the empty set is SP-sufficient for a given undirected graph, and also extends it to the directed case. We note that the proof given here, although for a much more general result, is substantially simpler and more elegant than the one given in [6]. In their most general form, our algorithms apply to networks with other routings as well (for example, arbitrary routings). The only restriction is that the set of allowed paths in the network must be closed under taking subpaths of length 2 and 3. We are thus able to unify the results for identifying sufficient sets in undirected and directed graphs from [5], [12], and [17].

In Section 3 we turn to the problem of finding a minimum SP-sufficient set (MIN SP-SUFFICIENT SET). We show that the problem is MAX SNP-hard for undirected and directed graphs by providing L-reductions from EDGE-DELETION BIPARTIZATION (EBIP) and VERTEX-DELETION BIPARTIZATION (VBIP), respectively, i.e., the problems of deleting a minimum set of edges or vertices in order to make a given graph bipartite. We also show that MIN SP-SUFFICIENT SET can be solved optimally in linear time in undirected triangle-free graphs, but remains $\mathcal{NP}$-hard for bidirected triangle-free planar graphs.

We conclude in Section 4 with a discussion on possible directions for future research.

2. Networks with Given Placements of Converters. In this section we consider (undirected and directed) networks that are equipped with wavelength converters. We present algorithms for deciding whether the given placement is SP-sufficient, and for optimally assigning wavelengths when such a placement is known. The main idea behind our proposed algorithms is to employ an auxiliary graph that captures the structure of the existing shortest paths in the given network.

Before we begin, we give some definitions and notation that we need to present our results. For a graph $G = (V, E)$, let $\mathcal{P}_G$ be the set of all shortest paths of length 2 in G. For a set $\mathcal{P}$ of paths in G and a set $S \subseteq V$, let $\mathcal{P}(S)$ be the set of paths obtained from $\mathcal{P}$ after cutting every path at the vertices of S that it contains. (Cutting a path p at an internal vertex v means replacing the path by two new paths, one consisting of the first part of p ending at v and the other consisting of the second part of p starting at v.)

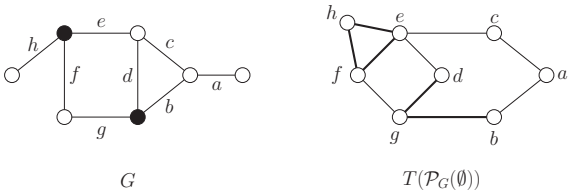

Fig. 1. An example of a graph G along with $T(\mathcal{P}_G(S))$ for $S = \emptyset$. Notice that both conditions of Theorem 1 are violated: the graph $T(\mathcal{P}_G(\emptyset))$ is not bipartite, and both edges corresponding to the subpaths of a shortest path of length 3 (e.g., the path using edges a, c, e) lie consecutively on the same cycle. If we let S consist of the two black vertices then $T(\mathcal{P}_G(S))$ is obtained from $T(\mathcal{P}_G(\emptyset))$ by removing the bold edges and S is indeed an SP-sufficient set.

Clearly, coloring a set $\mathcal{P}$ of paths with respect to $S \subseteq V$ is equivalent to coloring $\mathcal{P}(S)$. For a set $\mathcal{P}$ of shortest paths in G, let $T(\mathcal{P})$ be the multigraph on E with an edge-set containing for every path $p \in \mathcal{P}$ an edge for each of the subpaths of length 2 of p.

2.1. *Identifying Valid Placements of Converters*. The following theorem characterizes valid SP-sufficient sets.

THEOREM 1. *Let $G = (V, E)$ be a (possibly directed) graph. A set $S \subseteq V$ is SP-sufficient for G if and only if the following two conditions hold*:

(i) *$T(\mathcal{P}_G(S))$ is bipartite, and*
(ii) *not both edges corresponding to the two subpaths of length 2 of a shortest path of length 3 in G lie consecutively on the same cycle in $T(\mathcal{P}_G(S))$.*

An example of the construction of $T(\mathcal{P}_G(S))$ is shown in Figure 1. Before we proceed to the proof of Theorem 1, we show how a polynomial-time algorithm for deciding whether a given subset $S \subseteq V$ is SP-sufficient can be derived from it.

THEOREM 2. *There exists an algorithm for deciding whether a given subset $S \subseteq V$ is SP-sufficient for a (possibly directed) graph $G = (V, E)$ in $O(|V|^2 \cdot |E|)$ time.*

PROOF. The graph $T(\mathcal{P}_G(S))$ can be constructed in $O(|V| \cdot |E|)$ time by examining for each edge uv the edges incident to u, if $u \notin S$, and the edges incident to v, if $v \notin S$, and using the adjacency matrix to check in constant time if there is a triangle. This is optimal since there can exist $\Theta(|V|^3)$ shortest paths of length 2 in G: consider, for example, a graph consisting of three independent sets of vertices A, B, and C of the same cardinality, where $G[A \cup B]$ and $G[B \cup C]$ form complete bipartite graphs, and $A \cup C$ is an independent set. It follows that we can list all shortest paths of length 3 in G in $O(|V|^2 \cdot |E|)$ time: we first compute all distances in G in $O(|V| \cdot |E|)$ time, and then check for each shortest path of length 2 whether it can be extended into a shortest path of length 3. This is also optimal since we can have $\Theta(|V|^4)$ shortest paths of length 3 in G as in the case of a graph consisting of four independent sets of vertices A, B, C, and D of the same cardinality, where $G[A \cup B]$, $G[B \cup C]$, and $G[C \cup D]$ form complete bipartite graphs, and $A \cup C$, $A \cup D$, $B \cup D$ are independent sets.

The two conditions of Theorem 1 can, therefore, be checked as follows. Since $T(\mathcal{P}_G(S))$ has size $O(|V| \cdot |E|)$, we can check bipartiteness in $O(|V| \cdot |E|)$ time. To verify the second condition, we first compute the biconnected components of $T(\mathcal{P}_G(S))$ in time $O(|V| \cdot |E|)$ using Tarjan's algorithm [16], and then check for each shortest path p of length 3 that does not go over a vertex in S, whether the edges corresponding to the two subpaths of length 2 of p lie in different biconnected components of $T(\mathcal{P}_G(S))$. If this is not the case then the two edges lie consecutively on some cycle. Since there are $O(|V|^2 \cdot |E|)$ shortest paths of length 3 in G, we have an overall running time $O(|V|^2 \cdot |E|)$. □

We proceed to prove Theorem 1. It is easy to see that the conditions of the statement are indeed necessary. First assume that $T(\mathcal{P}_G(S))$ is not bipartite. Then $T(\mathcal{P}_G(S))$ contains an odd cycle whose edges correspond to a set of shortest paths in G with load 2 that require three colors for a valid coloring. To see that (ii) is necessary, assume that $T(\mathcal{P}_G(S))$ is bipartite and that there is a shortest path p of length 3 in G containing two paths of length 2 whose corresponding edges lie on the same (even) cycle C in $T(\mathcal{P}_G(S))$. Let e be the middle edge of p in G. The paths corresponding to the edges of C that are not incident to e, along with p, form a set of paths of load 2 that require three colors for a valid coloring. Sufficiency will follow from the wavelength-assignment algorithm given in the following section.

2.2. *Wavelength Assignment in Networks with Converters.* The following theorem shows that the conditions of Theorem 1 are indeed sufficient.

THEOREM 3. *Let $G = (V, E)$ be a (possibly directed) graph and let $S \subseteq V$ be such that the conditions of Theorem* 1 *are satisfied. Any set $\mathcal{P}$ of shortest paths on G can be colored with $L(\mathcal{P})$ colors with respect to S.*

PROOF. Let $G = (V, E)$ be a (possibly directed) graph, and let $S \subseteq V$ be such that conditions (i) and (ii) hold. Let $\mathcal{P}$ be a set of shortest paths in G. To simplify notation we let $\mathcal{P}$ be equal to $\mathcal{P}(S)$. Also, we remove all paths of length one from $\mathcal{P}$ since these can be colored greedily afterwards. The coloring of $\mathcal{P}$ will be obtained by edge-coloring $T(\mathcal{P})$. The edge-coloring of $T(\mathcal{P})$ will be done by computing many local edge-colorings and then merging them in order to obtain a coloring for $T(\mathcal{P})$ with $L(\mathcal{P})$ colors such that edges in $T(\mathcal{P})$ that correspond to the same path in $\mathcal{P}$ get the same color.

Two edges in $T(\mathcal{P})$ that originate from the same path in $\mathcal{P}$ are said to be *relatives*. An edge-coloring of $T(\mathcal{P})$ is *valid* if all edges incident to the same vertex get different colors, except for relative edges which are required to have the same color. Let $B_1, \ldots, B_k$ be the blocks (biconnected components) of $T(\mathcal{P})$ ordered according to a breadth-first search traversal of the block graph of $T(\mathcal{P})$, i.e., the bipartite graph on $C \cup \mathcal{B}$ and edges cB for $c \in C$, $B \in \mathcal{B}$ if $c \in B$, where C is the set of cut-vertices of $T(\mathcal{P})$ and $\mathcal{B}$ is the set of blocks of $T(\mathcal{P})$. Let $\mathcal{P}_i \subseteq \mathcal{P}$ be the set of paths that correspond to an edge in B_i. Since condition (i) of Theorem 1 is satisfied, B_i is bipartite and can be edge-colored with $\Delta(B_i)$ colors in polynomial time [13]. Since condition (ii) is satisfied, B_i may contain

only one edge from each path and hence the edge-coloring is valid for B_i. Furthermore, the degree of every vertex of B_i is equal to the load of the corresponding edge in G that is induced by $\mathcal{P}_i$.

Let A_i be a valid edge-coloring for B_i that uses $L(\mathcal{P}_i)$ colors. We will show how we can merge colorings $A_1, \ldots, A_k$ into a valid coloring for $\bigcup_{1\le i\le k} B_i$ that uses $L(\mathcal{P})$ colors. We begin by merging A_1 with A_2. After the merging, A_2 is the new coloring we obtain. After having merged colorings $A_1, \ldots, A_{i-1}$ we proceed to the merging of A_{i-1} with A_i. Define $B'_i := B_1 \cup \cdots \cup B_i$. Let $\mathcal{P}'_i \subseteq \mathcal{P}$ be the set of paths that correspond to edges in B'_i. During the whole merging process we maintain the following invariants for A_i: (a) A_i uses no more than $L(\mathcal{P}'_i)$ colors, and (b) A_i is valid for B'_i. After $k-1$ mergings, $B'_k = T(\mathcal{P})$ and hence we will have obtained a valid coloring of $\mathcal{P}$ with $L(\mathcal{P})$ colors.

Assume we have merged the first $i-1$ colorings and the invariants (a) and (b) hold for A_{i-1}. We show how to merge A_{i-1} with A_i such that (a) and (b) are maintained. Let x be the common cut-vertex of B_i with a block whose coloring has already been merged. In order to merge the two colorings and maintain the invariants we have to ensure that (1) relative edges incident to x get the same color in A_{i-1} and A_i, and (2) non-relative edges incident to x get different colors in A_{i-1} and A_i. In order to ensure (1) we permute A_i so that every edge incident to x that is colored in A_i and is relative to an edge f colored in A_{i-1}, gets the color that f has in A_{i-1}. In order to ensure (2) we need to introduce new colors. Assume there are two edges e, colored in A_{i-1}, and f, colored in A_i, that are incident to x, have the same color, and correspond to different paths. In B'_i these edges contribute 2 to the degree of x (and their corresponding paths contribute 2 to the load of edge x in G); however, in each of B'_{i-1}, B_i only one of these edges is present. Therefore, there is at least one color among $1, \ldots, L(\mathcal{P}'_i)$ that is not used by the edges incident to x in B'_i. Let c be such a color. We permute coloring A_i so that all edges that were previously assigned color c get the color that f had and vice versa. Examples of the necessary modifications are shown in Figure 2.

Since both colorings were valid, they remain valid after these modifications. After permuting A_i, the two colorings become compatible and can be merged in the obvious way. After $k-1$ such mergings, A_k is a valid coloring for $\mathcal{P}$ that uses $L(\mathcal{P})$ colors. □

The next theorem shows how an algorithm for computing such a coloring can be implemented efficiently.

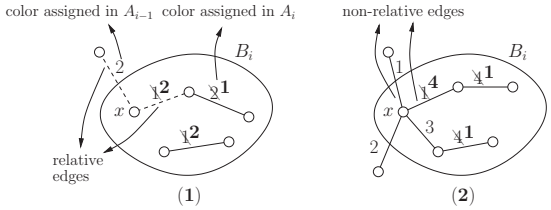

Fig. 2. Example of the two types of permutations needed to merge local colorings in the proof of Theorem 3. Colors assigned in A_i that change are crossed out and the new ones appear in bold.

THEOREM 4. *Let $G = (V, E)$ be a (possibly directed) graph and let $S \subseteq V$ be such that the conditions of Theorem* 1 *are satisfied. There exists an algorithm for coloring any set $\mathcal{P}$ of shortest paths with $L(\mathcal{P})$ colors with respect to S in $O(|E|+|V|\cdot|\mathcal{P}|\cdot\log L(\mathcal{P}))$ time.*

PROOF. The proof of Theorem 3 provides explicitly such a coloring algorithm. We argue that this algorithm can be implemented to run in $O(|E| + |V| \cdot |\mathcal{P}| \cdot \log L(\mathcal{P}))$ time.

We assume that each path $p \in \mathcal{P}$ is given as a list of edges $e_1, e_2, \ldots, e_{|p|}$, ordered so that e_i and e_{i-1} are incident to the same vertex. We can construct $T(\mathcal{P})$ in $O(|E|+|V|\cdot|\mathcal{P}|)$ time since the length of a path is bounded by $|V|$. The biconnected components (blocks) of $T(\mathcal{P})$ can be computed in $O(|V| \cdot |\mathcal{P}|)$ time using Tarjan's algorithm [16]. A block B with edge-set E_B can be edge-colored in time $O(|E_B| \cdot \log L(\mathcal{P}))$ using the algorithm of Cole et al. [3], since $L(\mathcal{P})$ is an upper bound on the maximum degree of any block of $T(\mathcal{P})$. Since $T(\mathcal{P})$ has $O(|V| \cdot |\mathcal{P}|)$ edges and the edge-sets of different blocks are disjoint, all edge-colorings can be performed in total time $O(|V| \cdot |\mathcal{P}| \cdot \log L(\mathcal{P}))$. In the remainder of the proof we argue that the merging of the colorings can be done in $O(|V| \cdot |\mathcal{P}|)$ time.

Before we begin the merging process we initialize some necessary data-structures. We maintain two arrays π, π^{-1} of size $L(\mathcal{P})$ which are initialized by setting $\pi(j) := j$ and $\pi^{-1}(j) := j$ for all $1 \leq j \leq L(\mathcal{P})$. At each merging, π holds a permutation to be applied to the coloring of the next block, while π^{-1} holds its inverse (where we need it). Only after the correct π is computed, we traverse all edges of the next block and rename their colors according to π. Furthermore, we use a binary array U of size $L(\mathcal{P})$ which we initialize by $U(j) := 0$ for all $1 \leq j \leq L(\mathcal{P})$; at a merging of two colorings, U indicates which colors are used by the edges that are incident to the cut-vertex "connecting" the two colorings and that belong to the previously merged coloring.

We also use a set F which at each merging holds all free colors at the edges incident to the "connecting" cut-vertex. The set F initially contains all colors $1, \ldots, L(\mathcal{P})$. We store F using a set data-structure that can maintain arbitrary subsets of $\{1, \ldots, L(\mathcal{P})\}$ and that supports insertion, deletion, membership checking, and returning an arbitrary element in constant time. Such a data-structure can be implemented using a combination of a doubly linked list of elements and an array of size $L(\mathcal{P})$; the jth element in the array is a pointer to the corresponding list element, if j is in the set, and a NIL pointer otherwise.

The data-structures for π, π^{-1}, U, and F can be initialized in time $O(L(\mathcal{P})) = O(|V| \cdot |\mathcal{P}|)$. Note that in order to save time, after each merging we do not re-initialize these data-structures from scratch but re-set only the values that have changed in the merging.

In addition, for each edge incident to a cut-vertex we store a pointer to its relative incident to the same cut-vertex, if there is any. These pointers can be computed in $O(|V| \cdot |\mathcal{P}|)$ time.

Let $B_1, \ldots, B_k$ be the blocks of $T(\mathcal{P})$ and let $x_1, \ldots, x_{k'}$ be the cut-vertices of $T(\mathcal{P})$, ordered according to a breadth-first search traversal of the block-graph of $T(\mathcal{P})$ starting at an arbitrary block. For $1 \leq i \leq k$, let A_i denote the computed initial edge-coloring of B_i. To perform the merging we proceed through the cut-vertices in the order $x_1, \ldots, x_{k'}$.

Consider cut-vertex x_j. Assume that the parent of x_j in the breadth-first search tree of the block graph of $T(\mathcal{P})$ is $B_{i'}$, where $i' \leq i-1$, and its children are blocks $B_i, \ldots, B_l$. At the time when we process x_j, we have already merged the colorings $A_1, \ldots, A_{i-1}$, and A_{i-1} holds the merged coloring. While processing x_j, we are going to merge colorings $A_i, \ldots, A_l$ one by one with A_{i-1}, and then we proceed to the next cut-vertex x_{j+1}. While π and π^{-1} will be re-set after each individual merging, U and F will be maintained during all the mergings of $A_i, \ldots, A_l$ and re-set only when we proceed to the next cut-vertex.

Before we begin with the merging of $A_i, \ldots, A_l$, we examine the colors on the edges incident to x_j. For each such edge that is in $B_{i'}$, we remove its color from F and mark it in U. In order to determine how the colors in A_i must be renamed to make A_i compatible with A_{i-1}, we make two passes over the edges incident to x_j contained in B_i. Note that we do not need to go through all edges incident to x_j in $T(\mathcal{P})$ for implementing such a pass, since we have computed the blocks of $T(\mathcal{P})$ beforehand. In each pass, we compute a permutation and apply it to the colors of the edges in B_i.

In the first pass we deal with edges that have relatives in $B_{i'}$. We process an edge e in B_i that is incident to x_j and has received color a in A_i as follows:

- If e does not have a relative in $B_{i'}$, we proceed to the next edge incident to x_j in B_i.
- If e has a relative in $B_{i'}$ that has received color b in A_{i-1}, we need to ensure that $\pi(a) = b$. If this is already satisfied, we do nothing. Otherwise, i.e., if $\pi(a) \neq b$, let $z := \pi(a)$ and $y := \pi^{-1}(b)$. We set $\pi(a) := b$, $\pi(y) := z$, $\pi^{-1}(b) := a$, and $\pi^{-1}(z) := y$. Intuitively, the effect of this operation is the same as applying the original permutation π to the colors on the edges of B_i, exchanging the color of e, i.e., color z, with color b on all edges in B_i, and letting the new π represent the resulting permutation.

At the end of the first pass, we traverse all edges of B_i and apply the permutation π to their colors. The resulting coloring of B_i, which we denote again by A_i, has the property that edges incident to x_j with a relative in $B_{i'}$ have the same color as that relative has in A_{i-1}. Now we re-set π and π^{-1} to the initial status by taking back all changes that we have made to these arrays in the first pass. Then we go once again through all edges of B_i that are incident to x_j and, for each such edge e that has color a in the new coloring A_i, we remove a from F (if it is still in F). Note that now the status of U and F is such that the colors marked in U are those that are used by A_{i-1} on edges incident to x_j in $B_{i'}$, while the set F contains all colors except those that are used by A_{i-1} or A_i on edges incident to x_j in $B_{i'}$ or B_i. This completes the first pass.

In the second pass over the edges of B_i incident to x_j, we deal with edges that have no relative in $B_{i'}$ but use the same color as an edge incident to x_j in $B_{i'}$. We process an edge e in B_i that is incident to x_j and has received color a in the new A_i as follows:

- If e has a relative colored in A_{i-1}, we do nothing.
- If e does not have a relative colored in A_{i-1}, we check whether a is marked in U. If $U(a) = 0$, we do nothing. If $U(a) = 1$, we pick a new color y from F (note that F must be non-empty in this case) and exchange colors a and y in a similar way as above: we set $\pi(a) := y$ and $\pi(y) := a$. Note that we do not need to use π^{-1} here since the pairs of colors that are exchanged are disjoint from each other. Furthermore, we delete y from F, as y is now no longer a free color.

At the end of the second pass, we go through all edges of B_i and apply π to their colors. The resulting coloring of B_i is compatible with A_{i-1} and can be merged with A_{i-1} without further modifications. Let A_i denote the merged coloring. To update U, we go through the edges incident to x_j in B_i and mark the colors that they receive in the new A_i in U. Finally, we re-set π by taking back all changes we made to this array in the second pass. This completes the second pass and also the merging of A_{i-1} with A_i.

Now we proceed to the merging of A_i with A_{i+1}. This merging is done in the same way as the merging of A_{i-1} with A_i (i.e., we use two passes over the edges in B_{i+1} incident to x_j in order to compute two permutations, and we apply each of these permutations to the colors on the edges of B_{i+1}). This is repeated for colorings $A_{i+2}, \ldots, A_l$. After we finish with the merging of these colorings we re-set F and U by going through the colors on all edges incident to x_j: we insert all colors on these edges into F and unmark them in U. Then we proceed to the next cut-vertex x_{j+1}.

At each cut-vertex x we traverse each edge incident to x a constant number of times. When merging a new block, we traverse its edge-set a constant number of times. Therefore, altogether we have a running time of $O(|V| \cdot |\mathcal{P}|)$ for the merging process. □

We remark that the term $|V| \cdot |\mathcal{P}|$ in the running-time given in Theorem 4 can actually be replaced by the sum of the lengths of all paths in $\mathcal{P}$. In the proof of the theorem we have simply used $|V| \cdot |\mathcal{P}|$ as an upper bound on the value of this sum.

2.3. *Generalizations to Other Routings.* We now discuss how our algorithms can be adapted to work on networks with other routings as well. Notice that the fact that we are dealing with shortest paths does not play a role in the proof of Theorem 3. What we need is that the set of all admissible paths is closed under taking subpaths of length 2 and 3: these are the paths needed for the necessity of the conditions in Theorem 1, since they provide the routings that witness the non-existence of a coloring with $L(\mathcal{P})$ colors. It is also interesting to observe that we can always find such a witness routing that has load 2 and requires three colors for a valid coloring, and that if no such routing exists then we can find a "good" coloring of any set of paths just by merging colorings of paths of length 2.

Therefore, our results can be applied to any network that uses a routing closed under taking subpaths of length 2 and 3. If, for example, we allow arbitrary routings, we choose $\mathcal{P}_G$ as the set of all simple paths of length 2, thus providing an algorithm to unify the results in [5], [12], and [17] for identifying sufficient sets and finding an optimal wavelength assignment with respect to a valid sufficient set in both undirected and directed networks. Observe that in this case $T(\mathcal{P}_G)$ is simply the line graph of G.

We also give an alternative formulation of Theorems 1 and 3 that might find applications in other areas as well.

COROLLARY 5. *There exists a polynomial-time algorithm to decide whether for a given multiset $\mathcal{P}$ of paths, that is closed under taking subpaths of length 2 and 3, all subsets $\mathcal{P}' \subseteq \mathcal{P}$ can be colored with $L(\mathcal{P}')$ colors, and, if the answer is yes, a polynomial-time algorithm to find such a color assignment for any given $\mathcal{P}' \subseteq \mathcal{P}$.*

That is, given a set $\mathcal{P}$ of paths, with the property that for every path $p \in \mathcal{P}$ all its subpaths of length 2 and 3 are also in $\mathcal{P}$, we construct $T(\mathcal{P})$ and check whether the

conditions of Theorem 1 are satisfied. If not then there exists some $\mathcal{P}' \subseteq \mathcal{P}$ that cannot be colored with $L(\mathcal{P}')$ colors. If the conditions are satisfied, any $\mathcal{P}' \subseteq \mathcal{P}$ can be colored with $L(\mathcal{P}')$ colors using the algorithm described in the proof of Theorem 3.

3. Complexity and Hardness of Placing Converters. We now consider the problem of placing as few converters as possible on a given network in order to optimize its capacity usage. In order to show that this problem is hard, even to approximate, we provide L-reductions (see [15]) from already known hard problems. (Recall that we will use the EDGE-DELETION BIPARTIZATION (EBIP) and VERTEX-DELETION BIPARTIZATION (VBIP) problems, i.e., the problems of deleting a minimum set of edges or vertices in order to make a given graph bipartite.) For completeness we give here the definition of an L-reduction.

DEFINITION 1 (L-reducibility). An optimization problem Π *L-reduces* to an optimization problem Π' if there are polynomial-time algorithms f, g, and positive constants c_1, c_2 such that for each instance I of Π the following properties hold:

(i) Algorithm f produces an instance $I' = f(I)$ of Π', such that the cost of the optimal solutions of I and I', $OPT(I)$ and $OPT(I')$, respectively, satisfy $OPT(I') \leq c_1 OPT(I)$.
(ii) Given any solution of I' with cost $SOL(I')$, algorithm g produces a solution of I with cost $SOL(I)$ such that $|SOL(I) - OPT(I)| \leq c_2 |SOL(I') - OPT(I')|$.

3.1. *The Undirected Case*

THEOREM 6. *There exists an L-reduction from* EDGE-DELETION BIPARTIZATION *to the undirected* MIN SP-SUFFICIENT SET *problem.*

PROOF. Let $G = (V, E)$ be an instance of EBIP, and assume, without loss of generality, that G does not contain any vertices of degree 1. We first show how to construct an undirected instance $G' = (V', E')$ of MIN SP-SUFFICIENT SET. The idea behind the construction is to map odd cycles of G onto "bad" induced odd cycles in G', i.e., cycles on which the set of all shortest paths of length 2 corresponds to an odd cycle in $T(\mathcal{P}_G(\emptyset))$. Since single triangles do not allow the construction of such a set of shortest paths, but have to be hit by any solution of EBIP, we replace each edge with a longer path in order to "blow up" the length of the cycles of G. Moreover, in order to avoid claws in G' we use a special gadget for every vertex of G. Finally, we eliminate all shortest paths of length 3 from G' so that no other configuration apart from odd induced cycles of length greater than 3 will need to be hit by a solution for MIN SP-SUFFICIENT SET.

We construct G' as follows. For each vertex v of degree $\deg(v)$ in G, we have a clique on $\deg(v)$ vertices. For each edge uv in G, we have a simple path on three vertices. The endpoints of the path are identified with two distinct vertices in the cliques that correspond to u and v. The construction of G' is completed by adding a claw with center x and making x adjacent to all other vertices. Observe that the parity of each cycle in

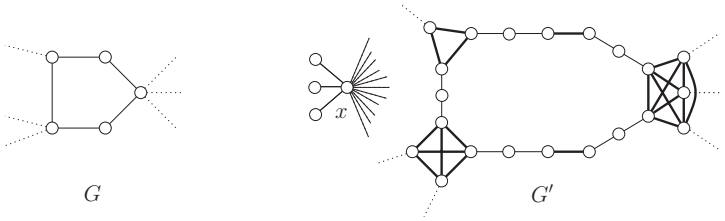

Fig. 3. The transformation used in the reduction in the proof of Theorem 6. Bold edges are in cliques that correspond to vertices of G.

G is preserved in G': a cycle on m edges in G is mapped onto an induced cycle on $3m$ edges in G'. An example is shown in Figure 3.

Consider an optimal solution to the instance of EBIP, namely a set of edges $F \subseteq E$ of minimum cardinality OPT_{BIP} such that $G - F$ is bipartite. We construct an SP-sufficient set S of size $OPT_{\text{BIP}} + 1$ for G' as follows. For every edge $e \in F$, we take in S one of the endpoints of its corresponding path in G'. Furthermore, we take x into S since it is the center of a claw. We will show that S is indeed an SP-sufficient set. Since G' has diameter 2, according to Theorem 1 we only need to show that $T(\mathcal{P}_{G'}(S))$ is bipartite, or equivalently that the edges of G' can be 2-colored such that no shortest path of length 2 that does not go through a converter uses edges of the same color. We exhibit such a coloring. Consider a bipartition of $G - F$. For every vertex that is in the first side of the partition, we color the edges of its corresponding clique in G' with color α. All edges in cliques corresponding to vertices in the other side of the partition are colored with color β. We extend this coloring to paths corresponding to edges in $E \setminus F$. Such paths connect cliques whose edges get different colors. Therefore, we can assign to each of the two edges of such a path a different color than the one that its incident clique already has. Paths corresponding to edges in F connect cliques assigned the same color. Consider a path connecting two cliques with the same color, say α. Since one of the endpoints of the path is in S, we assign color α to the edge of the path incident to the converter and color β to the other edge. Finally, we assign the same color, say α, to all edges incident to x. It is easy to see that the only shortest paths of length 2 that use edges of the same color are the ones that go over converters. The set S is, therefore, SP-sufficient.

Now, assume we are given an SP-sufficient set S' and consider a 2-coloring of the edges of G', with colors α and β, such that no shortest path of length 2 that does not go over a converter uses edges of the same color. We can assume that in each clique, all edges corresponding to a vertex in G have the same color. If this is not the case we can transform the coloring into one that has this property as follows. Consider a clique K in G', corresponding to some vertex in G, whose edges are not monochromatic. If all the vertices of the clique are in S' then we can simply modify the coloring of the edges of K so that they all have the same color. The resulting coloring still has the property that no shortest path of length 2 that does not go over a converter uses edges of the same color. Otherwise, there is some vertex $v \in K$ without a converter. Say that its incident edge outside K has color α. Then all its incident edges in K must have color β. Consider now some edge uw in K that has color α. Let u', w' be

the neighbors of u and w, respectively, outside K. The paths vuu', $u'uw$ are shortest and whatever the color assigned to edge uu', one of the two paths will use two edges of the same color. The vertex u is, therefore, in S'. Similarly, because of the paths vww', $w'wu$, w is also in S'. Hence, if we change the color of uw to β the coloring still has the desired property. By repeating this modification we can obtain a coloring such that all cliques are monochromatic, and thus our assumption is without loss of generality.

Construct a solution F' to EBIP by taking all the edges whose corresponding path in G' contains a vertex in S'. We show that $G - F'$ is indeed bipartite. Construct a bipartition by taking in one side vertices corresponding to cliques in G' whose edges are colored with color α. The other side contains the remaining vertices. Assume for a contradiction that there is an edge $e \in E \setminus F'$ whose endpoints are in the same side. Then there cannot exist a valid 2-coloring of the edges of the path corresponding to e, a contradiction. Note that $|F'| \leq |S'| - 1$, since $x \in S'$ but no edge in G has a corresponding path in G' containing x.

Let OPT_{SPSS} denote the size of an optimal SP-sufficient set for G'. Since from a given SP-sufficient set S' we can obtain a solution F' to EBIP of size at most $|S'| - 1$, and from an optimal solution to EBIP we can obtain an SP-sufficient set of size $OPT_{\text{BIP}} + 1$, it follows that $OPT_{\text{SPSS}} = OPT_{\text{BIP}} + 1$. Furthermore, $|F'| - OPT_{\text{BIP}} \leq |S'| - OPT_{\text{SPSS}}$ and therefore the transformation is an L-reduction. □

Since EBIP is known to be MAX SNP-hard [15], and since the constructed instance of MIN SP-SUFFICIENT SET has diameter 2, we obtain the following corollary.

COROLLARY 7. *The undirected* MIN SP-SUFFICIENT SET *problem is MAX SNP-hard even when restricted to graphs of diameter* 2.

Observe that the presence of triangles in the proof of Theorem 6 is essential in order to avoid claws. In fact, if we restrict to triangle-free graphs then the problem can be solved optimally in polynomial time as the following theorem illustrates.

THEOREM 8. *There exists a linear-time algorithm for finding a minimum SP-sufficient set in undirected triangle-free graphs*.

PROOF. Consider an undirected triangle-free graph $G = (V, E)$, and assume, without loss of generality, that G is connected and is not a cycle (in the latter case, a single converter at an arbitrary node of the cycle is SP-sufficient except for a cycle on four vertices where the empty set suffices, see [5]). Denote by $N(v)$ the subgraph induced by the vertices adjacent to v. Since G is triangle-free, $N(v)$ is an independent set for all $v \in V$. Furthermore, any SP-sufficient set must contain all vertices $v \in V$ with $\deg(v) \geq 3$ since otherwise there would be a claw without a converter. Thus, $S = \{v \in V \mid \deg(v) \geq 3\}$ is a minimum SP-sufficient set: $V \setminus S$ contains vertices of degree at most 2, hence if we explode all vertices in S, as described in Section 1, the exploded graph is a collection of lines. By [5], S is a sufficient set, therefore also SP-sufficient and the statement is proved. □

3.2. *The Directed Case*. The directed MIN SP-SUFFICIENT SET problem gives us more freedom in designing a reduction since claws do not necessarily require a converter. This is in accord with our expectation that the directed case is more difficult (recall that MIN SUFFICIENT SET is $\mathcal{NP}$-hard in directed graphs but polynomial in undirected). In this case we are able to give a simple approximation preserving reduction from VBIP. In [14] Lund and Yannakakis note that edge-deletion problems tend to be computationally easier than vertex-deletion problems.

THEOREM 9. *There exists an L-reduction from* VERTEX-DELETION BIPARTIZATION *to the directed* MIN SP-SUFFICIENT SET *problem*.

PROOF. The idea behind the reduction is the same as in the undirected case. The main difference, which simplifies the construction and allows us to reduce a vertex-deletion problem to MIN SP-SUFFICIENT SET, is that claws do not necessarily require a converter in the directed case; hence, we do not need to use a special gadget for every vertex.

Let $G = (V, E)$ be an instance of VBIP. The directed instance $G' = (V', E')$ of MIN SP-SUFFICIENT SET is constructed as follows. For every vertex v in G, we have a vertex v' in G'; the vertex v' corresponds to v. For every edge uv in G we have a bidirected path of length 3 between u' and v' corresponding to edge uv; each of the two directed paths (i.e., from u' to v' and from v' to u') will be called a *blow-up path*. Finally, we add a new vertex x which is made adjacent to all other vertices via two oppositely directed edges.

Consider an optimal solution to the instance of VBIP, namely a set of vertices $U \subseteq V$ of minimum cardinality OPT_{BIP} such that $G - U$ is bipartite. We construct an SP-sufficient set S of size $|U|$ for G' by taking all vertices corresponding to vertices in U. We show that S is indeed an SP-sufficient set by exhibiting a 2-coloring of the edges of G', as in the proof of Theorem 6. Consider a bipartition of $G - U$ into a left and a right part. Vertices in G' that correspond to vertices in the left side of the partition are called *left* vertices. Vertices in G' that correspond to vertices in the right side of the partition are called *right* vertices. Vertices in G' that correspond to vertices in U are called *deleted* vertices. Assign color α to edges directed out of left vertices and color β to edges directed out of right vertices. Now, simply extend this coloring by assigning alternating colors to the edges of all blow-up paths that have one edge already colored. The only edges remaining uncolored are edges incident to x and edges in blow-up paths directed out of deleted vertices. We color all edges directed into x with color α and all edges directed out of x with color β. All edges directed out of deleted vertices which are in blow-up paths that end up at a left vertex are colored with color β. The remaining edges directed out of deleted vertices are colored with color α. The remaining blow-up paths that have uncolored edges (these are paths directed out of some deleted vertex) can be colored by assigning alternating colors to the uncolored edges depending on the color assigned to the edge incident to the deleted vertex. It is easy to see that the only shortest paths of length 2 that use edges of the same color are paths that go over vertices which correspond to vertices in U. Since these vertices are equipped with converters, it follows that S is SP-sufficient.

Now, assume we are given an SP-sufficient set S' for G'. We can assume that S' consists only of vertices corresponding to vertices in G (i.e., not x and not any internal

vertex of a blow-up path). Construct a solution U' to vBIP by taking all vertices that correspond to vertices in S'. To see that $G - U'$ is bipartite, assume for a contradiction that it contains an odd cycle. Consider the blow-up paths in G' that correspond to the edges of the cycle: they form two oppositely directed odd cycles in G' which can be used for a construction of a set of paths with load 2 that requires three colors for a valid coloring. Hence, at least one of the vertices of the cycle is in S', and therefore its corresponding vertex in G is deleted, a contradiction.

Let OPT_{SPSS} denote the size of an optimal SP-sufficient set for G'. Since from a given SP-sufficient set for G' we can construct a valid solution to vBIP of the same size it follows that $OPT_{\text{SPSS}} \geq OPT_{\text{BIP}}$ and hence $|S| = OPT_{\text{SPSS}} = OPT_{\text{BIP}}$. Furthermore, $|U'| - OPT_{\text{BIP}} \leq |S'| - OPT_{\text{SPSS}}$ and hence the transformation is an L-reduction. □

Since vBIP is known to be MAX SNP-hard [14], and our constructed instance of MIN SP-SUFFICIENT SET is bidirected and has diameter 2, Theorem 9 implies the following.

COROLLARY 10. *The directed* MIN SP-SUFFICIENT SET *problem is MAX SNP-hard even when restricted to bidirected graphs of diameter* 2.

The following theorem shows that MIN SP-SUFFICIENT SET is $\mathcal{NP}$-hard even on bidirected triangle-free planar graphs. The reduction is from PLANAR 3-SAT (see [8]) and is similar to the one given in [17] for showing the hardness of MIN SUFFICIENT SET. Our reduction shows that, in contrast to the undirected case, MIN SP-SUFFICIENT SET remains $\mathcal{NP}$-hard even on triangle-free bidirected graphs.

THEOREM 11. *The problem of determining whether for a given directed graph G and an integer k there exists an SP-sufficient set for G of size k is $\mathcal{NP}$-complete even when restricted to bidirected triangle-free planar graphs.*

PROOF. We refer to the decision version of MIN SP-SUFFICIENT SET as SP-SUFFICIENT SET. For a directed graph $G = (V, E)$ it is easy to check in polynomial time if $S \subseteq V$ is SP-sufficient as is shown in Theorem 1. Therefore, SP-SUFFICIENT SET is in $\mathcal{NP}$. The $\mathcal{NP}$-hardness of SP-SUFFICIENT SET follows by a reduction from the PLANAR 3SAT problem [8]. An instance of PLANAR 3SAT is a set of clauses $\{C_1, \ldots, C_m\}$, each with exactly three literals, with the additional property that the bipartite graph with one vertex for each variable, one vertex for each clause, and with an edge between a variable vertex x and a clause vertex C when x or $\bar{x}$ is in the clause C is planar. Given an instance I of PLANAR 3SAT, we show how to construct an instance I' of SP-SUFFICIENT SET that is a YES instance if and only if I is satisfiable. We assume that the literals in each clause of I correspond to different variables. The construction of I' will be polynomial in the size of I. The construction is similar to the one used for SUFFICIENT SET in bidirected graphs [17]; the only difference is that we use a cycle of length 5 instead of a triangle for each clause because of the restriction to shortest paths.

Let I be an instance of PLANAR 3SAT with m clauses on n variables, each clause containing exactly three literals. Construct a bidirected graph G as follows. For ease of presentation the construction will be given for the skeleton of G, i.e., when we say

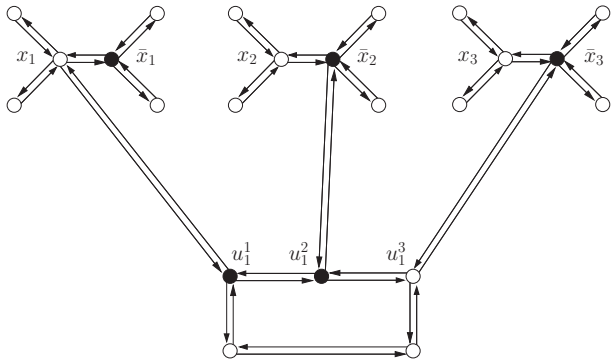

Fig. 4. The construction used in the hardness proof for the directed case.

"edge" (or "adjacent") we actually mean "(connected with) two oppositely directed edges." For each variable x_i, G contains two vertices $x_i, \bar{x}_i$ adjacent to each other. We refer to these vertices as *variable vertices*. For each variable vertex x, we have two new vertices adjacent to x. For each clause, G contains a cycle of length 5. Let u_j^1, u_j^2, u_j^3 be three consecutive vertices of the cycle corresponding to clause C_j. These three vertices correspond to the literal occurrences in C_j and are called *literal vertices*. If x_i occurs in C_j then an edge is added between x_i and the vertex corresponding to this literal occurrence; if $\bar{x}_i$ occurs then an edge is added between $\bar{x}_i$ and the corresponding vertex. These edges are called the *connecting edges*. An example of the construction for $C_1 = x_1 \vee \bar{x}_2 \vee \bar{x}_3$ is shown in Figure 4. Clearly, G is planar since I is an instance of PLANAR 3SAT, and does not contain any triangles.

We claim that there is an SP-sufficient set S for G of $2m + n$ vertices if and only if I is satisfiable.

Suppose that I is satisfiable and let T be a satisfying truth assignment. We call a variable vertex *true* (*resp. false*) if its corresponding literal evaluates to true (resp. false) according to T. Let S contain all true variable vertices and two out of the three literal vertices of each clause such that the literal vertex that is not taken in S corresponds to a true literal occurrence (such a vertex exists since T is satisfying). Notice that $|S| = 2m+n$. It is easy to see that after exploding all vertices in S, as described in Section 1, the resulting graph consists of spiders. Thus, by the result of Wilfong and Winkler [17], it follows that S is a sufficient set and thus also an SP-sufficient set.

For the other direction we first need to show that any SP-sufficient set needs to contain at least one vertex from each gadget that was used to represent a variable. Let x, y be two adjacent vertices and let a, b be adjacent to x and c, d be adjacent to y. Consider the set of directed paths $\mathcal{P} = \{axb, axyc, dyc, dyx, yxb\}$. These paths have load 2 but require three colors for a valid coloring. Hence, if all paths in $\mathcal{P}$ are shortest, any SP-sufficient set will contain at least one of x or y. Therefore, any SP-sufficient set must contain vertex x_i or $\bar{x}_i$ for every variable x_i. In fact this argument can be generalized to the case where x and y are connected with a bidirected path; in this case any SP-sufficient set must contain a vertex of the bidirected path from x to y. We call this generalized configuration an $\mathcal{H}$*-graph* on x and y.

Now, assume that S is an SP-sufficient set of size $2m + n$. The set S must contain at least one variable vertex from each variable gadget. Also, because of the connecting edges, S should contain at least two of the literal vertices of each clause gadget; otherwise there is an $\mathcal{H}$-graph without a converter. This is easy to see if only u_j^1 or u_j^3 is in S. If only u_j^2 is in S, notice that there is a generalized $\mathcal{H}$-graph on u_j^1 and u_j^3 (with u_j^2 adjacent to both u_j^1 and u_j^3) with nine shortest paths on it with load 2 that require three colors for a valid coloring. Furthermore, we can assume, without loss of generality, that S contains u_j^1 or u_j^3; otherwise we move the converter from the vertex of the bidirected path between u_j^1 and u_j^3 to one of its endpoints, and the $\mathcal{H}$-graph is still hit. Therefore, since $|S| = 2m + n$, S contains exactly one variable vertex from each variable gadget and exactly two literal vertices from each clause gadget. Define a truth assignment for I by setting $x_i = 1$ if $x_i \in S$ and $x_i = 0$ otherwise. Consider a clause C_j. There is exactly one literal vertex in G corresponding to a literal occurrence in C_j that is not in S. The variable of this occurrence, however, must be in S since otherwise there is an $\mathcal{H}$-graph without a converter. Therefore, there is at least one true literal in each clause and T is a satisfying assignment. □

4. Discussion. We have presented a polynomial-time algorithm for deciding whether for a given graph $G = (V, E)$ a subset $S \subseteq V$ is SP-sufficient, and a polynomial-time algorithm for optimally coloring any set of shortest paths in G with respect to an SP-sufficient set. These algorithms can be extended to address sufficient sets for arbitrary routings in both undirected and directed graphs. We also showed that MIN SP-SUFFICIENT SET is at least as hard to approximate as EBIP in undirected graphs, and at least as hard to approximate as VBIP in directed graphs (and hence is MAX SNP-hard in both cases). Finally, we gave a linear-time algorithm that solves MIN SP-SUFFICIENT SET optimally in undirected triangle-free graphs and showed that the problem remains $\mathcal{NP}$-hard even for bidirected triangle-free planar graphs.

A challenging remaining open problem is that of designing good approximation algorithms for MIN SP-SUFFICIENT SET. Although we are able to prove only MAX SNP-hardness of MIN SP-SUFFICIENT SET, our results indicate that it is unlikely or at least highly non-trivial to obtain a constant factor approximation algorithm. Indeed, that would imply a constant approximation for EBIP or VBIP whose approximability status has had a gap of $O(\log n)$ for the last 10 years (an $O(\log n)$ approximation for both problems follows from the multicut algorithm of Garg et al. [9], [10]; see also an older poly-logarithmic approximation given in [11]). In [14] Lund and Yannakakis conjecture that *for every non-trivial, hereditary property π with an infinite number of minimal forbidden subgraphs the node-deletion problem cannot be approximated with constant ratio*. Although the conjecture has been disproved for π = "acyclic" (by a constant approximation for the undirected feedback vertex set problem [2]) and for properties derived from matroids definable on the edge-set of any graph [7], it still remains likely that this is the case for VBIP.

Therefore, the design of a logarithmic approximation or of a reduction that places MIN SP-SUFFICIENT SET in some higher inapproximability class appears to be the most promising direction for future research. From the algorithmic point of view, MIN SP-

SUFFICIENT SET seems similar to EBIP since by placing converters in G we delete certain edges in $T(\mathcal{P}_G(\emptyset))$. Nevertheless, we believe that MIN SP-SUFFICIENT SET is actually harder than EBIP: the reason is that a single converter on some vertex v does not only delete one edge from $T(\mathcal{P}_G(\emptyset))$ but all the edges that correspond to the shortest paths of length 2 that go through v in G. If the edge-sets in $T(\mathcal{P}_G(\emptyset))$ that correspond to single vertices in G are connected subgraphs then a logarithmic approximation, at least for bipartizing $T(\mathcal{P}_G(\emptyset))$, can be obtained by a simple adaptation of the multicut-based algorithm given in [10]. Unfortunately, this is not the case in general, in both undirected and directed graphs. Furthermore, solving MIN SP-SUFFICIENT SET requires not only bipartizing $T(\mathcal{P}_G(\emptyset))$, but also hitting all cycles with a "bad" path of length 3 on it. In all our reductions we explicitly forced the diameter of the constructed graph to be less than 3 in order to focus on the bipartization aspect of the problem and ignored the second condition of Theorem 1. We believe that these additional degrees of complexity that MIN SP-SUFFICIENT SET has, could be exploited in order to place it in some higher inapproximability class. We note finally that an n/k-approximation for any constant k follows trivially from Theorem 1: we search for a valid SP-sufficient set in all subsets up to k vertices; if none is found we place a converter at every vertex.

References

[1] V. Auletta, I. Caragiannis, L. Gargano, C. Kaklamanis, and P. Persiano. Sparse and limited wavelength conversion in all-optical tree networks. *Theoretical Computer Science*, 266(1–2):887–934, 2001.

[2] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[3] R. Cole, K. Ost, and S. Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica*, 21(1):5–12, 2001.

[4] R. Diestel. *Graph Theory*, 2nd edition. Springer-Verlag, New York, 2000.

[5] T. Erlebach and S. Stefanakos. Wavelength conversion in networks of bounded treewidth. TIK-Report 132, ETH Zurich, April 2002.

[6] T. Erlebach and S. Stefanakos. On shortest-path all-optical networks without wavelength conversion requirements. In *Proceedings of the* 20*th International Symposium on Theoretical Aspects of Computer Science* (*STACS* '03), pages 133–144. LNCS 2607. Springer-Verlag, Berlin, 2003.

[7] T. Fujito. Approximating node-deletion problems for matroidal properties. *Journal of Algorithms*, 31:211–227, 1999.

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

[9] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in directed and node weighted graphs. In *Proceedings of the* 21*st International Colloquium on Automata*, *Languages*, *and Programming* (*ICALP* '94), pages 487–498. LNCS 820. Springer-Verlag, Berlin, 1994.

[10] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[11] P. N. Klein, S. Rao, A. Agrawal, and R. Ravi. An approximate max-flow min-cut relation for unidirected multicommodity flow, with applications. *Combinatorica*, 15(2):187–202, 1995.

[12] J. Kleinberg and A. Kumar. Wavelength conversion in optical networks. *Journal of Algorithms*, 38(1):25–50, 2001.

[13] D. König. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen*, 77:453–465, 1916.

[14] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *Proceedings of the* 20*th International Colloquium on Automata*, *Languages*, *and Programming* (*ICALP* '93), pages 40–51. LNCS 700. Springer-Verlag, Berlin, 1993.

[15] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[16] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[17] G. Wilfong and P. Winkler. Ring routing and wavelength translation. In *Proceedings of the* 9*th Annual ACM–SIAM Symposium on Discrete Algorithms* (*SODA* '98), pages 333–341, 1998.

[18] H. Zang, J. P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical Networks Magazine*, 1(1):47–60, 2000.