Optimizing Regenerator Cost in Traffic Grooming

Michele Flammini † Gianpiero Monaco ‡ Luca Moscardelli § Mordechai Shalom ¶ Shmuel Zaks ∥

Abstract
In optical networks regenerators have to be placed on lightpaths every $d$ nodes in order to regenerate the signal. In addition, grooming enables the use of the same regenerator by several lightpaths. Up to $g$ (the grooming factor) lightpaths can use the same regenerator. In this work we consider the problem of minimizing the number of regenerators used in traffic grooming in optical networks. Starting from the 4-approximation algorithm of [10] for $d = 1$ and a path topology, we provide an approximation algorithm with the same approximation ratio for $d = 1$ and the ring and tree topologies. We present also a technique based on matching that leads to the same approximation ratio in tree topology and can be used to obtain approximation algorithms in other topologies. We provide an approximation algorithm for general topology that uses this technique. Finally, all the results are extended to the case of general $d$.

Keywords: Optical Networks, Wavelength Division Multiplexing (WDM), Regenerators, Traffic Grooming, Tree Networks.

1 Introduction
In modern optical networks, high-speed signals are sent through optical fibers using WDM (Wavelength Division Multiplexing) technology. Currently deployed networks carry around 80 wavelengths per fiber, whereas networks with a few hundreds wavelengths per fiber are being used in testbeds. The decrease in the energy of the signal with the traveled distance raises the requirement of optical amplifiers at every (almost) fixed distance. However, optical amplifiers introduce noise into...
the signal, thus after a certain number of amplifications, the optical signal needs to be regenerated. In the current technology, the signal is regenerated by first using a ROADM (Reconfigurable Optical Add-Drop Multiplexer) to extract a set of wavelengths from the optical fiber. Then, for each extracted wavelength, an optical regenerator is needed to regenerate the signal carried by that wavelength. That is, at a given optical node, one needs as many regenerators as wavelengths one wants to regenerate.

Nowadays the cost of a regenerator is considerably higher than the cost of an ROADM. Moreover, as described above, the regenerator cost is per wavelength, as opposed to ROADM cost that is payed once per several wavelengths. Therefore the total number of regenerators is an important cost parameter to be minimized [14]. Another possible criterion is to minimize the number of locations (that is, the number of nodes) in which optical regenerators are placed. This measure is the one assumed in [9], which makes sense when the dominant part of the cost is the set-up of new optical nodes, or when the equipment to be placed at each node is the same for all nodes. In this work we consider the total number of regenerators as the cost function.

A logical path formed by a signal traveling from its source to its destination using a unique wavelength is termed a lightpath. Let $d$ be the maximum number of hops a lightpath can make without meeting a regenerator. Then, for each lightpath $\ell$, we need to place one regenerator every $d$ consecutive vertices in $\ell$, to get an optimal solution. However the problem becomes harder when the traffic grooming comes into the picture.

Traffic grooming: The network usually supports traffic that is at rates which are lower than the full wavelength capacity, and therefore the network operator has to be able to put together (= groom) low-capacity connections into the high capacity lightpaths. In graph-theoretic terms, we associate a path in the graph with each connection, and the problem can viewed as assigning wavelengths to these paths so that at most $g$ of them using the same wavelength ($g$ being the grooming factor) can share one edge. Thus, all paths (i.e. connections) that get the same color (i.e., the same wavelength) and form a connected subgraph correspond to grooming of these connections into one lightpath.

1.1 Related Work

Various variants of regenerator placement problems were studied in [4, 7, 8, 13, 15, 16, 18, 19]. Most of these results concentrate in heuristics and simulations and do not consider traffic grooming.

In [9] theoretical results (upper bounds and lower bounds) are presented for some variants of this problem. This work considers the number of regenerator locations (as opposed to the total number of regenerators) as the cost measure, and does not consider traffic grooming. On the other hand, [14] uses the same cost measure of this paper but still does not consider traffic grooming.

The problem we study is shown to be NP-hard in other contexts such as fiber minimization in [17] and its NP-hardness is also implied by the proof of a similar result in [11] holding even for path topology and $g = 2$.

When the underlying graph is a path the problem is equivalent to a machine
scheduling problem studied in [10]. In particular, the path topology corresponds to a timeline in which every node is an instant; each path corresponds to a job starting at the instant corresponding to its leftmost node and ending to the instant corresponding to its rightmost node; \( g \) is the maximum number of jobs a processor can execute at the same time. Such a scheduling problem aims at minimizing the sum of the computation time of all the machines, provided that an unbounded number of machines is available. Several approximation algorithms are presented in [10] for this scheduling problem and its special cases.

In the literature, two different scenarios have been studied, depending on whether or not it is allowed to split the paths in order, for instance, to reduce the number of used wavelengths or the cost of hardware components. In particular, [1, 3] assume that no splitting is allowed, while [2] allows to split paths and [12] considers both scenarios.

1.2 Our Contribution

In this work we consider the traffic grooming problem to minimize the number of regenerators used. We first consider the case \( d = 1 \), i.e. the case that a regenerator has to be placed at every internal node of every lightpath, and then we extend all the results to general \( d \). Our starting point is a 4-approximation algorithm of [10] that solves a closely related problem for a path topology and \( d = 1 \). We prove that the same algorithm can be used for our problem and show that, always for \( d = 1 \), it has the same approximation ratio not only for path topology, but also for ring topology. We present a greedy 4-approximation algorithm for tree networks. We also show a general technique using matchings that can lead to approximation algorithms in other topologies. We use this technique and show an \( \left\lfloor \frac{L+7}{2} \right\rfloor \)-approximation algorithm for general topology and \( d = 1 \), where \( L \) is the maximum load (i.e. number of paths that share a common edge) in the input.

In Section 2 we present preliminary results and definitions, including the above mentioned algorithm for path networks and extension of its analysis to the case of ring topology. In Section 3 we present an algorithm with the same performance for tree topology. In Section 4 we present the matching technique and its use for general topologies. In all the above mentioned sections we always consider the case \( d = 1 \); in Section 5, apart from summarizing the results and suggesting open research directions, we extend all the results to general \( d \).

2 Preliminaries

2.1 Definitions and Problem Statement

Until Section 5, in which the results will be extended to any \( d \), we deal with the case \( d = 1 \).

We will consider instances \((G, P, g)\) where \( G = (V, E) \) is a graph modeling the optical network, \( P \) is a set of simple paths in \( G \), \( g \in \mathbb{N}^+ \cup \{\infty\} \) is the grooming factor.

A coloring (or wavelength assignment) of \((G, P)\) is a function \( w : P \mapsto \mathbb{N} \). For a coloring \( w \) and color \( \lambda \), \( P_\lambda^w \) is the subset of paths from \( P \) colored \( \lambda \) by \( w \), i.e.
\( \mathcal{P}_\lambda^w \defeq \{ P \in \mathcal{P} | w(P) = \lambda \} \). When there is no ambiguity on the coloring \( w \) under consideration, we omit the superscript \( w \) and use \( \mathcal{P}_\lambda \).

For a node \( v \), \( \mathcal{P}_v \) denotes the subset of paths of \( \mathcal{P} \) having \( v \) as an intermediate node, and similarly for an edge \( e \), \( \mathcal{P}_e \) denotes the subset of paths of \( \mathcal{P} \) using the edge \( e \). For every \( e \in E \) we define \( \text{load}(\mathcal{P}, e) \defeq |\mathcal{P}_e| \) and \( \text{load}(\mathcal{P}) \defeq \max_{e \in E} \text{load}(\mathcal{P}, e) \). A valid coloring (or wavelength assignment) \( w \) of \((G, \mathcal{P}, g, d)\) is a coloring of \( \mathcal{P} \) in which for any edge \( e \) at most \( g \) paths using \( e \) are colored with the same color, i.e. for every color \( \lambda \) we have \( \text{load}(\mathcal{P}_\lambda^w) \leq g \).

We denote by \( \text{INT}(P) \) the set of intermediate nodes, i.e. of all the nodes not being endpoints, of a path \( P \) in \( G \), and \( \text{int}(P) \defeq |\text{INT}(P)| \). For a set \( \mathcal{P} \) of paths we define

\[
\text{SPAN}(\mathcal{P}) \defeq \bigcup_{P \in \mathcal{P}} \text{INT}(P),
\]
\[
\text{span}(\mathcal{P}) \defeq |\text{SPAN}(\mathcal{P})|,
\]
\[
\text{len}(\mathcal{P}) \defeq \sum_{P \in \mathcal{P}} \text{int}(P).
\]

A set of paths is called a no-split instance or shortly an NSI if the union of its paths (as sets of edges) induces a graph of maximum degree at most 2. Since in this paper we assume (as \([1, 3, 12]\) do) that splitting paths is not allowed, paths using the same wavelength and going through the same edge of the network can be routed only to another unique edge, and therefore every set of paths with the same color has to be an NSI.

The number of regenerators operating at wavelength \( \lambda \) is \( \text{span}(\mathcal{P}_\lambda^w) \); in fact, at each node being an intermediate node of some path in \( \mathcal{P}_\lambda^w \) a regenerator operating at this wavelength is needed.

We are now ready to give a formal definition of our problem.

**Total Regenerators with Grooming (Trg)**

**Input:** A triple \((G, \mathcal{P}, g)\), where \( G = (V, E) \) is a graph, \( \mathcal{P} = \{P_1, P_2, ..., P_n\} \) is a set of simple paths in \( G \), and \( g \) is an integer, namely the grooming factor.

**Output:** A valid coloring \( w : \mathcal{P} \mapsto \mathbb{N} \) of the paths such that, for every \( \lambda \), \( \mathcal{P}_\lambda \) is an NSI (we will refer to the latter condition as the no splitting condition throughout this work).

**Objective:** The cost of a solution is given by the total number of regenerators \( \text{REG}_w \defeq \sum_\lambda \text{span}(\mathcal{P}_\lambda^w) \). The goal is to minimize the total number of regenerators \( \text{REG}_w \).

\( \text{OPT}(G, \mathcal{P}, g) \) denotes the cost of any optimal coloring and \( \text{ALG}(G, \mathcal{P}, g) \) denotes the cost of the coloring returned by some algorithm \( \text{ALG} \) on instance \((G, \mathcal{P}, g)\). As the cost function depends only on the partition of the paths induced by the coloring, with some abuse of notation, a coloring \( w \) denotes also the equivalence class of colorings that induce the same partition as \( w \).
2.2 Lower Bounds

We have the following trivial lower bounds for the cost of any coloring \( w \), in particular for an optimal coloring.

- The grooming bound:
  \[
  REG^w \geq \frac{\text{len}(P)}{g}.
  \]

- The span bound:
  \[
  REG^w \geq \text{span}(P).
  \]

The grooming bound holds because a regenerator can be used by a maximum of \( g \) intermediate nodes of paths. The span bound holds because at least one regenerator is needed on any node that is an intermediate node of some path.

2.3 Path and Ring Networks

In this subsection we focus on ring and path networks. We adapt Theorem 2.1 in [10] to our problem and generalize it to the case of ring networks. Specifically, we show that the FirstFit algorithm presented in [10] is a 4-approximation algorithm for our problem. The proof goes along the same lines, and we bring it here for sake of completeness; the main difference is in Lemma 2, whose proof required modifications of the proof of the corresponding claim in [10] in order to assure correctness for the case of ring topology.

Notice that when \( G \) is a ring or a path, all subsets of \( \mathcal{P} \) constitute an NSI.

Algorithm FirstFit colors the paths greedily by considering them one after the other, from longest to shortest. Each path is assigned the lowest possible color for it.

**Algorithm 1 FirstFit** \((G, \mathcal{P}, g)\) with \( G \) being a path or a ring

1. Sort the paths in non-increasing order of length, i.e., \( \text{int}(P_1) \geq \text{int}(P_2) \geq \ldots \geq \text{int}(P_n) \).
2. Consider the paths by the above order and, for any path \( P_j, j \in \{1, \ldots, n\} \), assign to \( P_j \) the first possible color \( \lambda \) that will not violate the load condition. Namely, find the minimum value \( \lambda \geq 1 \) such that, for every edge \( e \) of \( P_j \),
   \[\text{load}(P^{\lambda}_j, e) \leq g - 1\]
   and set \( w(P_j) \leftarrow \lambda \).

The upper bound proof is based on the observation stated in the following lemma, and depicted in Figure 1.

**Lemma 1** Let \( w \) be the coloring returned by FirstFit. Let \( P \) be a path colored \( \lambda \), i.e., \( P \in \mathcal{P}_\lambda^w \), for some \( \lambda \geq 2 \). Then for any \( \lambda' < \lambda \), (a) there is an edge \( e \in P \) such that \( \text{load}(P^{\lambda'}_w, e) = g \), (b) each path \( P' \in \mathcal{P}_\lambda \cap \mathcal{P}_e \) is no shorter than \( P \).

**Proof:** In order to prove property (a), assume by contradiction that for some \( \lambda' < \lambda \) and every edge \( e \in P \), \( \text{load}(P^{\lambda'}_w, e) \leq g - 1 \). Since the algorithm assigns colors to paths incrementally (i.e. it never un-colors paths), this condition should have held at the point FirstFit considered \( P \) for coloring. In this case, \( P \) should
have been colored at most $\lambda'$ by FirstFit, a contradiction to the fact that $P$ is colored $\lambda$.

Property (b) follows from property (a) and the fact that the paths are considered by the algorithm in a non-increasing order of their lengths. □

We use the property stated in Lemma 1 in order to show the following claim, which will be crucial in order to prove the desired result.

**Lemma 2** Let $W \geq 1$ be the number of colors used by $w$; for any $1 < \lambda \leq W$, $\text{len}(P_{\lambda-1}) \geq \frac{g}{\text{span}(P_\lambda)}$.

**Proof:** For every path $P \in P_\lambda$, we fix $\lambda' = \lambda - 1$ and choose arbitrarily an edge $e$ of $P$ among those whose existence is guaranteed by Lemma 1. Let $b(P) \overset{def}{=} P_{\lambda-1} \cap P_e$ be the blocking paths of $P$. By Lemma 1, $|b(P)| = g$ and $\text{len}(b(P)) \geq g \cdot \text{int}(P)$. Let $\mathcal{B}$ be the set of all blocking paths defined as above, i.e. $\mathcal{B} \overset{def}{=} \bigcup_{P \in P_\lambda} b(P)$. Clearly $\mathcal{B} \subseteq P_{\lambda-1}$.

Now, we consider a blocking path $P' \in \mathcal{B}$ (see Figure 2). Consider the set of all paths in $P_\lambda$ blocked by $P'$. With a little abuse of notation we denote them by $b^{-1}(P')$. Consider a node $v \in \text{SPAN}(b^{-1}(P'))$. It is in some path $P'' \in P_\lambda$ which is no longer than $P'$ and intersects with $P'$, therefore there exists an intermediate node of $P'$ which is at distance to $v$ at most $\text{int}(P'') \leq \text{int}(P')$. As $G$ is a path
or a cycle the number of such nodes \( v \) is at most \( 3 \cdot \text{int}(P') \). We conclude that 
\[
\sum_{P' \in \mathcal{P}} \text{span}(b^{-1}(P')) \leq 3 \cdot \sum_{P' \in \mathcal{P}} \text{int}(P') = 3 \cdot \text{len}(\mathcal{P}).
\]
Consider a node \( v \in \text{SPAN}(\mathcal{P}_\lambda) \). It is an intermediate node of at least one path \( P \in \mathcal{P}_\lambda \), which in turn is blocked by at least \( g \) paths of \( \mathcal{P} \). Therefore \( v \in \text{SPAN}(b^{-1}(P')) \) for at least \( g \) paths \( P' \) of \( \mathcal{P} \), in other words \( v \) contributes at least \( g \) to the sum in the left hand side above. Thus we have 
\[
\sum_{P' \in \mathcal{P}} \text{span}(b^{-1}(P')) \geq g \cdot \text{span}(\mathcal{P}_\lambda).
\]

Therefore, 
\[
3 \cdot \text{len}(\mathcal{P}_{\lambda-1}) \geq 3 \cdot \text{len}(\mathcal{P}) \geq \sum_{P' \in \mathcal{P}} \text{span}(b^{-1}(P')) \geq g \cdot \text{span}(\mathcal{P}_\lambda).
\]

The following theorem, providing an upper bound to the approximation ratio of the FirstFit algorithm, and Lemma 3 follow from similar claims in [10].

**Theorem 1** If \( G \) is a path or a ring, then for any instance \((G, \mathcal{P}, g)\), 
\[
\text{FirstFit}(G, \mathcal{P}, g) \leq 4 \cdot \text{OPT}(G, \mathcal{P}, g).
\]

**Proof:** Combining the span bound and Lemma 2 we can now complete the analysis of the algorithm.

Let \( W \geq 1 \) be the number of colors used by \( w \), then \( \text{FirstFit}(G, \mathcal{P}, g) = \sum_{\lambda=1}^{W} \text{span}(\mathcal{P}_\lambda) \). Moreover 
\[
\sum_{\lambda=2}^{W} \text{span}(\mathcal{P}_\lambda) = \sum_{\lambda=1}^{W-1} \text{span}(\mathcal{P}_{\lambda+1}) \leq \frac{3}{g} \sum_{\lambda=1}^{W-1} \text{len}(\mathcal{P}_\lambda)
\]
\[
\leq \frac{3}{g} \sum_{\lambda=1}^{W} \text{len}(\mathcal{P}_\lambda) = \frac{3}{g} \text{len}(\mathcal{P}) \leq 3 \cdot \text{OPT}(G, \mathcal{P}, g),
\]
where the last inequality follows from the grooming bound.

Using the span bound, we have \( \text{span}(\mathcal{P}_1) \leq \text{span}(\mathcal{P}) \leq \text{OPT}(G, \mathcal{P}, g) \). Therefore, 
\( \text{FirstFit}(G, \mathcal{P}, g) \leq 4 \cdot \text{OPT}(G, \mathcal{P}, g) \).

**Lemma 3** For any \( \epsilon > 0 \), there are infinitely many instances \((G, \mathcal{P}, g)\) having arbitrarily large input sizes, such that 
\[
\text{FirstFit}(G, \mathcal{P}, g) > (3 - \epsilon) \cdot \text{OPT}(G, \mathcal{P}, g).
\]

**Proof:** Consider the instance \((G, \mathcal{P}, g)\) depicted in Figure 3 where \( m \) is a non-zero integer. For this instance an optimal solution uses one regenerator in each one of the nodes 1, ..., \( m \), one regenerator in each node 2\( m + 1 \), ..., 3\( m \), and \( g - 1 \) regenerators in the nodes \( m + 1 \), ..., 2\( m \), for a total cost of \( mg \). To see that this is an optimal solution, observe that this solution achieves the grooming bound tightly. In contrast as all the paths have the same length, they might be sorted in any order by the Step 1 of Algorithm FirstFit. If they happen to be sorted in the order shown in the figure from top to bottom, then FirstFit will use \( g \) regenerators in 1, ..., 3\( m \) for a total cost of \( 3mg \). Choosing \( g \) sufficiently large we get 
\[
\text{FirstFit}(G, \mathcal{P}, g) > (3 - \epsilon) \cdot \text{OPT}(G, \mathcal{P}, g).
\]

Combining Theorem 1 and Lemma 3, we finally get the following theorem.
Figure 3: Instance for the proof of the lower bound

**Theorem 2** The approximation ratio of FirstFit is between 3 and 4 in ring and path networks.

### 3 Tree Networks

In this section we present an optimal algorithm *GreedyMatch* for the case where the graph $G$ is a tree and $g = \infty$. Combining this algorithm and algorithm *FirstFit* described in the previous section we obtain a 4-approximation algorithm for tree networks and any value of $g$.

#### 3.1 $G, \mathcal{P}, \infty$ instances

We first consider the special case of $g = \infty$, that will be useful in order to provide an approximation algorithm for general $g$. When $g = \infty$, any solution is a valid coloring. It remains to satisfy the no splitting condition. Therefore the problem becomes to partition $\mathcal{P}$ into no-split instances $\mathcal{N}_1, \mathcal{N}_2, \ldots$ such that $\sum \lambda \text{span}(\mathcal{N}_\lambda)$ is minimized.

Note that the span (lower) bound holds in this special case, i.e. $\text{OPT}(G, \mathcal{P}, \infty) \geq \text{span}(\mathcal{P})$.

Since $g = \infty$, we can assume that there is no path $P \in \mathcal{P}$ completely included in another path $P' \in \mathcal{P}$, because in this case we could remove $P$ from the input. In any solution of the remaining instance $P$ can be added to the NSI containing $P'$ without increasing the cost.

We introduce some additional notation.
Two NSIs \( \mathcal{N} \) and \( \mathcal{N}' \) are said to be \textit{compatible} if their union is also an NSI. We denote this fact as \( \mathcal{N} \sim \mathcal{N}' \). Otherwise they are said to be \textit{incompatible} and denoted as \( \mathcal{N} \not\sim \mathcal{N}' \).

The overlap of two NSIs \( \mathcal{N} \) and \( \mathcal{N}' \) is \( \text{OV}(\mathcal{N}, \mathcal{N}') \overset{\text{def}}{=} \text{SPAN}(\mathcal{N}) \cap \text{SPAN}(\mathcal{N}') \) and \( \text{ov}(\mathcal{N}, \mathcal{N}') \overset{\text{def}}{=} |\text{OV}(\mathcal{N}, \mathcal{N}')| \).

Two NSIs \( \mathcal{N} \) and \( \mathcal{N}' \) are \textit{overlapping} if \( \text{ov}(\mathcal{N}, \mathcal{N}') > 0 \).

An NSI is said to be \textit{connected} if the union of its paths (as sets of edges) induces a connected graph.

We say that \( \mathcal{N} \subseteq \mathcal{N}' \) if \( \cup_{P \in \mathcal{N}} P \subseteq \cup_{P \in \mathcal{N}'} P \).

Consider algorithm \textit{GreedyMatch}; the following lemmata are needed for proving Theorem 3, in which it is shown that such an algorithm is optimal.

**Lemma 4** Every two NSIs in an optimal solution of \((G, \mathcal{P}, \infty)\) are either non-overlapping or incompatible.

**Proof:** Assume, by contradiction that there are two NSIs that are both compatible and overlapping. Then they can be joined to form one NSI, and decrease the cost of the solution by the size of their overlap. \(\square\)

**Algorithm 2** \textit{GreedyMatch}(G, \mathcal{P}, \infty), G being a tree

1. \( \forall P_i \in \mathcal{P}, \mathcal{N}_i \leftarrow \{P_i\} \quad \triangleright \) Every path constitutes a connected NSI.
2. \textbf{while} there exist \( \mathcal{N}_i, \mathcal{N}_j \) such that \( \mathcal{N}_i \subseteq \mathcal{N}_j \) \textbf{do} \quad \triangleright \) Eliminate inclusions
3. \( \mathcal{N}_j \leftarrow \mathcal{N}_j \cup \mathcal{N}_i \)
4. \( \mathcal{N}_i \leftarrow \emptyset \)
5. \textbf{end while}
6. \textbf{while} there exist two compatible and overlapping NSIs \( \mathcal{N}_i, \mathcal{N}_j \) \textbf{do}
7. \quad Find two compatible NSIs \( \mathcal{N}_i, \mathcal{N}_j \) maximizing \( \text{ov}(\mathcal{N}_i, \mathcal{N}_j) \)
8. \quad \( \mathcal{N}_j \leftarrow \mathcal{N}_j \cup \mathcal{N}_i \)
9. \textbf{end while}

We note that at the beginning of the algorithm each sets \( \mathcal{N}_i \) consists of a single path, therefore connected. Moreover, since a new NSI is constructed by unifying two compatible and overlapping NSIs, this condition continues to hold at any point of the execution of Algorithm. Therefore, the following observation holds:

**Observation 1** At any given point of the execution of Algorithm \textit{GreedyMatch} the sets \( \mathcal{N}_i \) are connected.

**Lemma 5** When the Algorithm \textit{GreedyMatch} reaches Step 7, there are no inclusions.

**Proof:** Inclusions are eliminated at the beginning of the algorithm. Therefore the claim is correct for the first time the algorithm reaches Step 7. Consider the first time that the claim is falsified, i.e. there is an NSI included in another. This can only happen at Step 8 of the previous iteration. In this step the algorithm
unifies two NSIs $\mathcal{N}_i, \mathcal{N}_j$. Assume that there is some $\mathcal{N}_k$ such that $\mathcal{N}_k \subseteq (\mathcal{N}_i \cup \mathcal{N}_j)$. As there were no inclusions before this step $\mathcal{N}_k \not\subseteq \mathcal{N}_i$ and $\mathcal{N}_k \not\subseteq \mathcal{N}_j$. Then $OV(\mathcal{N}_i, \mathcal{N}_j)$ is a proper subset of $SPAN(\mathcal{N}_k)$. In other words $OV(\mathcal{N}_i, \mathcal{N}_j)$ is a proper subset of $OV(\mathcal{N}_i, \mathcal{N}_k)$. A contradiction to the maximality of $ov(\mathcal{N}_i, \mathcal{N}_j)$.

**Lemma 6** At any given point of the execution of Algorithm GreedyMatch, after Step 1, consider the partition \{\mathcal{N}_1, \mathcal{N}_2, \ldots\}. There is an optimal solution \{\mathcal{N}_1^*, \mathcal{N}_2^*, \ldots\} such that every $\mathcal{N}_i$ is a subset of some $\mathcal{N}_i^*$, or in other words the partition given by the algorithm is a refinement of the partition given by some optimal solution.

**Proof:** Without loss of generality we can assume that all the NSIs in an optimal solution of $(\mathcal{G}, \mathcal{P}, \infty)$ are connected, because if we have a disconnected NSI $\mathcal{N}$ we can replace $\mathcal{N}$ with a connected NSI for each connected component of $\mathcal{N}$. The claim is obviously true immediately after Step 1 of the algorithm. Assume by contradiction that the claim is false and consider the first time during the execution of the algorithm that it becomes false. This can happen only after execution of Step 8. $\mathcal{N}_i$ and $\mathcal{N}_j$ are overlapping and compatible, because they are chosen by the algorithm in Step 7. They are also connected by Observation 1. As the condition was true prior to the execution of Step 8, there is some optimal solution $S^* = \{\mathcal{N}_1^*, \mathcal{N}_2^*, \ldots\}$ such that $\mathcal{N}_i \subseteq \mathcal{N}_i^*$ and $\mathcal{N}_j \subseteq \mathcal{N}_j^*$. Therefore $\mathcal{N}_i^* \supset \mathcal{N}_i$ and $\mathcal{N}_j^* \supset \mathcal{N}_j$ are overlapping. Also, by Lemma 4, $\mathcal{N}_i^* \approx \mathcal{N}_j^*$.

As, by Lemma 5 there are no inclusions, the $OV(\mathcal{N}_i, \mathcal{N}_j)$ is a proper subset of both $SPAN(\mathcal{N}_i)$ and $SPAN(\mathcal{N}_j)$ (see Figure 4). Let $a, b, c, d \in V$ be four distinct nodes of the tree such that $SPAN(\mathcal{N}_i)$ (resp. $SPAN(\mathcal{N}_j)$) is the path between $b$ and $d$ (resp. $a$ and $c$). Then $OV(\mathcal{N}_i, \mathcal{N}_j)$ is the path between $b$ and $c$. The partition \{\mathcal{N}_1, \ldots\} is a refinement of the partition \{\mathcal{N}_1^*, \ldots\}. Let $\mathcal{N}_i^* = \mathcal{N}_i \cup \mathcal{N}_ii \cup \mathcal{N}_ij \cup \ldots$. We observe that for none of these sets $SPAN(\mathcal{N}_i^*)$ can intersect with both $a \sim b$ and $c \sim d$, because this would imply that $OV(\mathcal{N}_i, \mathcal{N}_j) \subsetneq OV(\mathcal{N}_i, \mathcal{N}_i^*)$, a contradiction to the way $\mathcal{N}_i$ and $\mathcal{N}_j$ are chosen by the algorithm. Given this observation we partition the set $\mathcal{N}_i^*$ into three sets $\mathcal{N}_i, \mathcal{N}_ii$ and $\mathcal{N}_ij$ such that the sets $\mathcal{N}_i^*$ spanning at least one edge of $c \sim d$ (resp. $a \sim b$) are in $\mathcal{N}_ii$ (resp. $\mathcal{N}_ij$), the rest are divided arbitrarily. We do the same for $\mathcal{N}_j^*$.

![Figure 4: Proof of Lemma 6](image)

$\mathcal{N}_i, \mathcal{N}_ii, \mathcal{N}_ij$ are pairwise compatible, because they make part of $\mathcal{N}_i^*$, and so are
\( N_j, N_{jj}, \) and \( N_{ji} \). Moreover \( N_{ij} \sim N_{ji} \) and \( N_{ii} \sim N_{jj} \), because the underlying graph is a tree and thus they can overlap only in the path \( b - c \) in which there cannot exist nodes with induced degree 3 or more.

We conclude the proof by case analysis. For each case we show how an optimal solution \( S^* \) can be built from \( S^* \) such that \( N_i \) and \( N_j \) are contained in the same set of \( S^* \), a contradiction to the assumption that the condition became false.

Assume \( N_{ji} \sim N_i \):

- \( N_{ji} \sim N_{ii} \): In this case we can move \( N_{ii} \) and \( N_i \) into \( N_j^* \) without increasing the cost of the solution.

- \( N_{ji} \sim N_{ij} \): In this case there exists a node belonging to \( \text{SPAN}(N_{ji}) \cap \text{SPAN}(N_{ij}) \) with induced degree more than 2, and such a node is necessarily beyond (in Figure 4, at the right of) the node \( d \), proving that \( \text{SPAN}(N_{ji}) \) contains the path \( c - d \). Therefore we can move \( N_i \) into \( N_j^* \) without increasing the cost of the solution.

After handling the case \( N_{ij} \sim N_j \) symmetrically, it remains to handle the case \( N_{ji} \sim N_j \) and \( N_{ij} \sim N_j \). If \( N_{ji} \) and \( N_{ij} \) are overlapping then we can repartition these six sets into two sets \( N_{ij} \cup N_{ji} \) and \( N_i \cup N_j \cup N_{ii} \cup N_{jj} \) without increasing the cost. Otherwise we build three sets \( N_{ij}, N_{ji} \) and \( N_i \cup N_j \cup N_{ii} \cup N_{jj} \) without increasing the cost.

We are now able to prove that Algorithm GreedyMatch is optimal.

**Theorem 3** When Algorithm GreedyMatch ends, the solution \( \{N_1, N_2, \ldots \} \) is optimal.

**Proof:** By Lemma 6, the solution \( \{N_1, \ldots \} \) is a refinement of some optimal solution \( \{N_1^*, N_2^*, \ldots \} \). We claim that these partitions are equal. Assume, by contradiction that there is some set \( N_i^* \) containing at least two of the sets \( N_i \). These sets are pairwise non-overlapping because they are compatible and the algorithm stopped. Therefore \( N_i^* \) is not connected, a contradiction.

### 3.2 An approximation algorithm scheme for any graph \( G \) and any value of \( g \)

We propose the algorithm scheme Combined\((A, (G, P, g))\) for general graphs and any value of \( g \), depending on a generic Algorithm \( A \) working for the specific case in which \( g = \infty \).

**Algorithm 3** Combined\((A, (G, P, g))\)

1. Partition \( P \) into NSIs \( N_1, N_2, \ldots \) using algorithm \( A \) computed on the corresponding \( (G, P, \infty) \) instance.
2. For each \( i \), let \( G(N_i) \) be the graph induced by the paths of \( N_i \). Split \( N_i \) into sets \( P_{i,1}, P_{i,2}, \ldots \) by solving the instance FirstFit\((G(N_i), N_i, g)\).
3. Assign each one of the sets \( P_{i,j} \) a distinct color \( \lambda_{i,j} \).

Technion - Computer Science Department - Technical Report CS-2011-07 - 2011
Lemma 7 Given any \(g \geq 1\), if Algorithm \(A\) is a \(\rho\)-approximation algorithm for instance \((G, \mathcal{P}, \infty)\), then Algorithm Combined\((A, (G, \mathcal{P}, g))\) is a \((\rho + 3)\)-approximation algorithm for instance \((G, \mathcal{P}, g)\).

**Proof:** In order to prove the correctness of the algorithm, it is sufficient to notice that every \(N_i\) is a no-split instance, thus satisfies the no splitting condition. Therefore any subset \(P_{i,j}\) of it also satisfies the no splitting condition. Moreover, by the correctness of FirstFit the output is a valid coloring.

By Lemma 2, for any instance \((\text{SPAN}(N_i), N_i, g)\) and any color \(\lambda_{i,j}\) we have \(\text{span}(P_{i,j} + 1) \leq 3g\ \text{len}(P_{i,j})\).

Therefore

\[
\sum_{i,j \geq 1} \text{span}(P_{i,j}) \leq \frac{3}{g} \sum_{i,j \geq 1} \text{len}(P_{i,j}) = \frac{3}{g} \sum_{i \geq 1} \text{len}(N_i) = \frac{3}{g} \sum_{i \geq 1} \text{len}(P) \leq 3 \cdot \text{OPT}(G, \mathcal{P}, g).
\]

On the other hand

\[
\sum_{i \geq 1} \text{span}(P_{i,1}) \leq \sum_{i \geq 1} \text{span}(N_i) \leq \rho \cdot \text{OPT}(G, \mathcal{P}, \infty) \leq \rho \cdot \text{OPT}(G, \mathcal{P}, g).
\]

Combining we get

\[
\text{Combined}(A, (G, \mathcal{P}, g)) = \sum_{i,j \geq 1} \text{span}(P_{i,j}) \leq (\rho + 3) \cdot \text{OPT}(G, \mathcal{P}, g).
\]

\(\square\)

3.3 The approximation algorithm for tree networks

By combining Theorem 3 with Lemma 7, we get the following theorem.

**Theorem 4** Given any \(g \geq 1\), Algorithm Combined\((\text{GreedyMatch}, (G, \mathcal{P}, g))\) is a 4-approximation algorithm when \(G\) is a tree network.

The following lemma and its proof exploit arguments similar to the ones used in Lemma 3.

Lemma 8 For any \(\epsilon > 0\), there are infinitely many instances \((G, \mathcal{P}, g)\) having arbitrarily large input sizes, such that Combined\((\text{GreedyMatch}, (G, \mathcal{P}, g)) > (3 - \epsilon) \cdot \text{OPT}(G, \mathcal{P}, g)\), where \(G\) is a tree network.

**Proof:** Starting from the instance \((G, \mathcal{P}, g)\) depicted in Figure 3, we consider a slightly different scenario in which the leftmost paths start from node 0 and end at node \(m + 2\) (instead of \(m + 1\)); the central paths start from node \(m\) and end at node \(2m + 2\) (instead of \(2m + 1\)) and finally the rightmost paths start from node \(2m - 1\) (instead of \(2m\)) and end at node \(3m + 1\). Note that all the paths have the same length. It is easy to see that in this case GreedyMatchAlgorithm produces as input one NSI containing all the paths. Following the same arguments of Lemma

---

Technion - Computer Science Department - Technical Report CS-2011-07 - 2011
3, we argue that the optimal solution uses one regenerator in each one of the nodes 1, ..., m + 1, one regenerator in each node 2m, ..., 3m, and g − 1 regenerators in the nodes m + 1, ..., 2m + 1, for a total cost of m(g + 1) + g + 1. A solution of the FirstFit Algorithm (if it sorts paths in the order shown in Figure 3 from top to bottom) will use g regenerators in the nodes 1, ..., 3m for a total cost of 3mg = 3\left(\frac{g+1}{g}\right)^2 \text{OPT}(G, \mathcal{P}, g)$. Choosing g sufficiently large and m much greater than g we get $\text{Combined}(\text{GreedyMatch}, (G, \mathcal{P}, g)) > (3 - \epsilon)\text{OPT}(G, \mathcal{P}, g)$. □

By combining Theorem 4 with Lemma 8 we get the following theorem.

**Theorem 5** The approximation ratio of $\text{Combined}(\text{GreedyMatch})$ is between 3 and 4 in tree networks.

## 4 Beyond Tree Networks: a Matching Technique

In this section we present a new technique to approximate $(G, \mathcal{P}, \infty)$ instances in any topology. In particular, we show a general technique able to reduce an instance of the general network to instances of ring and path networks. Using such a technique, and exploiting a reduction of the problem to an instance of the Maximum Weighted Matching on an auxiliary graph, we present an approximation algorithm for the general topology.

### 4.1 The Endpoint Intersection Graph

In order to describe the matching technique, we need to define the **edge-weighted endpoint intersection graph** $EIG(G, \mathcal{P}) = (V', E')$ of $G$ and $\mathcal{P}$. $V'$ contains $2|\mathcal{P}|$ nodes $v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2}, ..., v_{i,1}, v_{i,2}, ...$, one for each endpoint of a path $P_i \in \mathcal{P}$. There is an edge between two nodes $v_{i,k}, v_{j,k'}$ if $P_i \cap P_j$ is either a path or a ring and $P_i \cap P_j$ contains a path in $G$ with endpoints $v_{i,k}$ and $v_{j,k'}$.

The weight function $f : E' \rightarrow \mathbb{N}$ is defined as follows: $f(v_{i,k}, v_{j,k'})$ is the length of the path between $v_{i,k}$ and $v_{j,k'}$ in the intersection, minus one. As usual, the weight of a set of edges is defined as the sum of the weights of its edges.

**Definition 1** Given a solution $w$ of $(G, \mathcal{P}, \infty)$, the endpoint matching $EM(w)$ of $w$ is defined as a matching of $EIG(G, \mathcal{P})$ that is constructed as follows in polynomial time.

Let $w$ be a solution of $(G, \mathcal{P}, \infty)$ and $N$ an NSI of $w$. Assume without loss of generality that $N$ is connected. By definition, $\cup N$ is a connected subgraph $H$ of $G$ with maximum degree 2. Thus $H$ is either a path or a cycle. Orient the edges of $H$ arbitrarily in some common direction. Let the paths of $N$ be numbered as $P_1, P_2, ..., P_l$ according to the order, in the chosen direction, of their starting nodes. Let without loss of generality these nodes be $v_{1,1}, v_{2,1}, ...$. As the paths are inclusion-free, the order of the ending nodes in this direction is the same, namely $v_{1,2}, v_{2,2}, ...$

We distinguish between two cases:

- $H$ is a path (upper part of Figure 5): For every two consecutive paths $P_i, P_{i+1}$, their intersection is the segment of the path between $v_{i,2}$ and
Therefore \((v_i, 2, v_{i+1, 1})\) is an edge of \(EIG(G, P)\), and the edges 
\((v_{1, 2}, v_{2, 1}), (v_{2, 2}, v_{3, 1}), \ldots, (v_{l-1, 2}, v_{1, 1})\) constitute a matching of \(EIG(G, P)\) 
with \(l - 1\) edges.

- \(H\) is a cycle (lower part of Figure 5): This case is similar to the case of a path, except that in this case \((v_{1, 2}, v_{1, 1})\) is also an edge of \(EIG(G, P)\), and \((v_{1, 2}, v_{2, 1}), (v_{2, 2}, v_{3, 1}), \ldots, (v_{l-1, 2}, v_{1, 1})\) constitutes a matching of \(EIG(G, P)\) with \(l\) edges.

\[
\begin{align*}
\text{A path } Q \text{ induced by an NSI} & \quad \text{A cycle } C \text{ induced by an NSI}
\end{align*}
\]

Figure 5: Correspondence between feasible solutions of \((G, P, \infty)\) and the matchings of \(EIG(G, P)\)

**Lemma 9** For every solution \(w\), \(REG^w = \text{len}(P) - f(EM(w))\).

**Proof:** Consider a connected NSI \(\mathcal{N}\) of \(w\) that induces a path of \(G\). Let without loss of generality \(\mathcal{N} = \{P_1, P_2, \ldots, P_l\}\) in the chosen direction of the path, and assume that the endpoints of each path are indexed by 1 and 2 in this direction. Letting \(END_{\mathcal{N}}\) be the set of all the endpoints of the paths in \(\mathcal{N}\), we obtain

\[
\begin{align*}
\text{span}(\mathcal{N}) &= \text{int}(P_1) + (\text{int}(P_2) - f((-v_{1, 2}, v_{2, 1}))) + \ldots \\
&\quad + (\text{int}(P_l) - f((-v_{l-1, 2}, v_{1, 1})))) \\
&= \text{len}(\mathcal{N}) - f \left( \{\{u, v\} \in EM(w) | u, v \in END_{\mathcal{N}} \} \right).
\end{align*}
\]

The same result holds, similarly, when \(\mathcal{N}\) induces a cycle. Summing up over all the NSIs of \(w\), we get

\[
\begin{align*}
REG^w &= \sum_{\lambda} \text{span}(\mathcal{N}_\lambda) = \sum_{\lambda} \text{len}(\mathcal{N}_\lambda) - \sum_{\lambda} f(M(w) \cap \mathcal{N}_\lambda) \\
&= \text{len}(P) - f(EM(w)).
\end{align*}
\]
Lemma 10 \textbf{EM} is a one-to-one function from the set of solutions of an \((G, \mathcal{P}, \infty)\) instance of \textsc{Trg} to the set of matchings of \textsc{Eig}(G, \mathcal{P}). Moreover if \(G\) is a ring or tree, then \(\text{EM}\) is onto. The inverse (partial) function \(\text{EM}^{-1}\) can be computed in polynomial time.

\textbf{Proof:} Consider a matching \(M\) of \textsc{Eig}(G, \mathcal{P}). Consider also the auxiliary edges (not belonging to the edge set of \textsc{Eig}(G, \mathcal{P})) \(M' = \{\{v_{i, 1}, v_{i, 2}\} | P_i \in \mathcal{P}\}\). Every node has degree at most 1 with respect to the edges in \(M\), and degree exactly 1 with respect to the edges in \(M'\), thus degree at most 2 with respect to the edges in \(M \cup M'\). Therefore the connected components of the graph \(G'\) induced by the edge set \(M \cup M'\) are (alternating) paths and cycles. Note that a path in \(G'\) ends with an auxiliary edge in \(M'\) (because a node with degree 1 has its only incident edge in \(M'\)), thus has odd length; the cycles have even length. For each connected component \(C\) of \(G'\) we pick a distinct color \(\lambda(C)\). First assume that \(C\) is a path. Then \(C\) is of the form \(v_{i_1, k_1} - v_{i_1, k'_1} - v_{i_2, k_2} - v_{i_2, k'_2} - \ldots - v_{i_l, k_l}, v_{i_l, k'_l}\), where \(k_i \neq k'_i\) for all \(i = 1, \ldots, l\). This corresponds to the sequence of paths \(\mathcal{N}(C) = P_{i_1}, \ldots, P_{i_l}\) of \(\mathcal{P}\). We color these paths so that \(w(P_{i_1}) = \cdots = w(P_{i_l}) = \lambda(C)\). The case of the cycle is treated similarly.

The connectedness of a component \(C\) implies that \(\mathcal{N}(C)\) is connected. If for every connected component \(C\), \(\mathcal{N}(C)\) constitutes an NSI then \(w\) is a solution of \((G, \mathcal{P}, \infty)\). Note that \(\text{EM}(w) = M\). Moreover \(w\) is the only solution having this property. Thus \(\text{EM}\) is one-to-one and \(\text{EM}^{-1}(M) = w\).

If \(G\) is a ring, then every subset of \(\mathcal{P}\) and in particular every \(\mathcal{N}(C)\) is an NSI, therefore \(w\) is a solution of \((G, \mathcal{P}, \infty)\), thus \(\text{EM}\) is onto. If \(G\) is a tree then for every \(\mathcal{N}(C) = P_{i_1}, \ldots, P_{i_l}\), using the fact that \(\mathcal{N}(C)\) is inclusion free, one can show by induction on \(l\) that \(\mathcal{N}(C)\) is an NSI. We conclude in the same way, that \(\text{EM}\) is onto. \(\square\)

Since the problem of finding a maximum matching is polynomial time solvable [6], it is worth noticing that, as an immediate consequence of Lemma 9 and Lemma 10, the following lemma provides another polynomial time algorithm finding an optimal solution for instances where \(G\) is a tree and \(g = \infty\).

\textbf{Algorithm 4 MaxMatch}(G, \mathcal{P}, \infty)
1: Construct the weighted endpoint intersection graph \textsc{Eig}(G, \mathcal{P}) of \(G\) and \(\mathcal{P}\) with the weight function \(f\).
2: Calculate a maximum weighted matching \(MM\) of \textsc{Eig}(G, \mathcal{P}) with weights \(f\).
3: Return \(\text{EM}^{-1}(MM)\).

\textbf{Lemma 11} If \(G\) is a tree, then algorithm MaxMatch runs in polynomial time and provides an optimal solution for any instance (G, \mathcal{P}, \infty).

4.2 Algorithm for General Networks

Unfortunately, as shown in the following theorem, the problem for \((G, \mathcal{P}, \infty)\) instances, with \(G\) being a general network, is \(NP\)-hard. Therefore, an approximation algorithm for solving it has to be provided.
Theorem 6 The problem TRG for \((G, \mathcal{P}, \infty)\) instances, with \(G\) being a general network, is NP-hard.

Proof: In order to prove the NP-hardness, we provide a polynomial reduction from the TRIPART problem, known to be NP-complete (see [5]).

An instance of the TRIPART problem is a simple graph \(G' = (V'_G, E'_G)\). The question is whether or not there is a partition of \(E'_G\) into triangles. Let \(V'_G = \{v'^0_G, v'^1_G, \ldots, v'^n_G\}\) and \(E'_G = \{e'^0_G, e'^1_G, \ldots, e'^3q_G\}\) (note that if \(|E'_G|\) is not a multiple of 3, a partition does not exist and the answer is obviously NO).

From the above instance \(G' = (V'_G, E'_G)\) of TRIPART we build the following instance \((G, \mathcal{P}, \infty)\) of the TRG problem. \(G = (V_1 \cup V_2, E)\), where \(V_1 = \{a_1, b_1, c_1 | i = 1, \ldots, n'\}\), \(V_2 = \{d_{j,k}, e_{j,k}, f_{j,k} | j = 1, \ldots, 3q, k = 1, \ldots, 3q + 1\}\) and the edge set \(E\) is the minimal one containing all the paths of the instance, that are defined as follows.

For each edge \(e'^j \in E'_G\), connecting nodes \(v'^i_G\) and \(v'^i_G\) \((i < i')\), we add the following pair of paths (see Figure 6): the top path \([a_1, b_1, c_1, d_{j,1}, e_{g(j,1),h(j,1),i}, f_{g(j,1),h(j,1),i}, e_{g(j,2),h(j,2),i}, f_{g(j,2),h(j,2),i} \ldots, e_{g(j,3q),h(j,3q),i}, f_{g(j,3q),h(j,3q),i}]\) and the bottom path \([c_1, b_1, a_1, d_{j,1}, e_{g(j,1),h(j,1),i}, f_{g(j,1),h(j,1),i}, d_{j,2}, e_{g(j,2),h(j,2),i}, f_{g(j,2),h(j,2),i} \ldots, e_{g(j,3q),h(j,3q),i}, f_{g(j,3q),h(j,3q),i}]\), where \(g(j, k)\) is \(j\) if edges \(e'^j\) and \(e'^k\) are consecutive in \(G\), and is the minimum between \(j\) and \(k\) otherwise, and \(h(j, k)\) is \(k\) if edges \(e'^j\) and \(e'^k\) are consecutive in \(G\), and is the maximum between \(j\) and \(k\) otherwise.

The following properties hold:

Property 1. The top path and the bottom path relative to each edge \(e'^j \in E'_G\) cannot be put in the same NSI, since otherwise nodes \(d_{j,1}\) and \(f_{j,3q+1}\) would have degree 3.

Property 2. Any two (bottom or top) paths relative to non-consecutive edges \(e'^j\) and \(e'^k\) \((j < k)\) of \(G'\) (overlapping on edge \(\{e_{j,k}, f_{j,k}\}\)) cannot be put in the same NSI, since otherwise nodes \(e_{j,k}\) and \(f_{j,k}\) would have degree 3.

Figure 6: The top (in dashed line) and the bottom (in solid line) paths corresponding to edge \(e'^j \in E'_G\), connecting nodes \(v'^i_G\) and \(v'^i_G\) \((i < i')\).
Property 3. As it can be easily verified, the only nodes in which it is possible to save regenerators are the $b$ nodes of $V_1$.

In order to prove the theorem, it is sufficient to prove that (i) if the answer to the TRIPART problem is YES, then there exists a solution of the constructed $(G, P, \infty)$ instance in which it is possible to save 6$q$ regenerators and, conversely, (ii) if it is possible to save 6$q$ regenerators in the constructed $(G, P, \infty)$ instance, then the answer to the TRIPART problem is YES.

Figure 7: The two NSIs corresponding to triangle in $G'$ with vertices $v_{G'}^i$, $v_{G'}^{i'}$ and $v_{G'}^{i''}$ ($i < i' < i''$).

In order to prove (i), it is sufficient to notice that a triangle in $G'$ with vertices $v_{G'}^i$, $v_{G'}^{i'}$, and $v_{G'}^{i''}$ ($i < i' < i''$) induces 6 paths in $P$ that can be rearranged in 2 NSIs as follows (see Figure 7): the top paths corresponding to edges $\{v_{G'}^i, v_{G'}^{i'}\}$ and $\{v_{G'}^{i'}, v_{G'}^{i''}\}$ and the bottom path corresponding to edge $\{v_{G'}^i, v_{G'}^{i''}\}$ belong to an NSI, while the bottom paths corresponding to edges $\{v_{G'}^i, v_{G'}^{i'}\}$ and $\{v_{G'}^{i'}, v_{G'}^{i''}\}$ and the top path corresponding to edge $\{v_{G'}^i, v_{G'}^{i''}\}$ belong to the other NSI. Therefore, in such paths 6 regenerators (one per path, at nodes $b_i$, $b_{i'}$, $b_{i''}$) are saved. Since when the TRIPART problem is YES $E'$ can be partitioned in $q$ triangles, 6$q$ regenerators are saved in total.

It remains to prove (ii). First of all, Property 3 ensures that regenerators can be only saved at $b$ nodes. By Property 2, only paths corresponding to edges of $G'$
sharing a node can be put in the same NSI, and moreover, by Property 1, the two paths corresponding to the same edge cannot be put in the same NSI. Therefore, regenerators can be saved only by putting in the same NSI either $x$ paths corresponding to edges of $G'$ sharing a same node, or the 3 paths corresponding to a triangle in $G'$. In the first case, $x - 1$ regenerators are saved ($\frac{x}{x}$ regenerators per path), whereas in the second case 3 regenerators are saved (1 regenerator per path). Since in $P$ there are $6q$ paths, if it is possible to save $6q$ regenerators, then all the savings have to be due to $2q$ NSIs each containing 3 paths and in which 1 regenerator per path is saved; therefore, since at most 2 different NSIs correspond to the same triangle of $G'$, $q$ triangles have to be in $G'$ and the claim follows.

The following lemma provides an approximation algorithm for the $(G, P, \infty)$ problem in general networks.

**Lemma 12** For every matching $M$ of $EIG(G, P)$ we can find in polynomial time a matching $\overline{M} \subseteq M$ such that $f(\overline{M}) \geq f(M)/2$ and $EM^{-1}(\overline{M})$ is defined.

**Proof:** We start as in the proof of Lemma 10. For any path and any cycle of $M \cup M'$, let $\{e_1, e_2, \ldots\}$ be the edges of $M$ in the considered path (resp. cycle). We obtain a matching $M_O \subseteq M$ (resp. $M_E \subseteq M$) by removing the edges with odd (resp. even) indices in $\{e_1, e_2, \ldots\}$. This breaks such a path (resp. cycle) into sub-paths of length at most three, in other words into paths containing exactly one edge of $M_O$ (resp. $M_E$), which in turn corresponds to a sequence of at most two paths of $P$. Assume without loss of generality $f(M_O) \geq f(M_E)$. We claim that $\overline{M} = M_O$ satisfies the claim. Clearly $f(M_O) \geq f(M)/2$. It remains to show that each connected component of $M \cup M_O$ corresponds to an NSI. If such a component consists of one path, than it is an NSI, otherwise it consists of two paths with at least one edge of $EIG(G, P)$ between their endpoints, therefore these two paths are compatible thus constitute an NSI. □

**Algorithm 5** MatchAndCut$(G, P, \infty)$

1: Construct the weighted endpoint intersection graph $EIG(G, P)$ of $G$ and $P$ with the weight function $f$.
2: Calculate a maximum weighted matching $M^*$ of $EIG(G, P)$ with weights $f$.
3: Calculate the matching $\overline{M}$ of $EIG(G, P)$ as described in proof of Lemma 12.
4: Return $EM^{-1}(\overline{M})$.

The following lemma relates the approximation ratio of a solution $w$ for a $(G, P, \infty)$ instance of Trg to the approximation ratio of $EM(w)$ with respect to the maximum weighted matching problem.

**Lemma 13** Let $M$ be a $\rho$-approximation to the maximum weighted matching of $EIG(G, P)$ for some $\rho \geq 1$. If $EM^{-1}(M)$ is defined, then $w = EM^{-1}(M)$ is a $(1/\rho + (1 - 1/\rho) \text{load}(P))$-approximation for the $(G, P, \infty)$ instance.

**Proof:** Let $M^*$ be a maximum weighted matching of $EIG(G, P)$, and let $w^*$ be an optimal solution of $(G, P, \infty)$. Let $\rho' = 1/\rho$, and let $M$ be a matching
satisfying our assumption, i.e. $M = EM(w)$ and $f(M) \geq \rho' \cdot f(M^*)$. We have

$$REG^w - REG^* = f(EM(w^*)) - f(EM(w)) = f(EM(w^*)) - f(M) \leq f(EM(w^*)) - \rho' \cdot f(M^*) \leq (1 - \rho')f(EM(w^*)) = (1 - \rho')len(P) + \rho' \cdot REG^*.$$  

Therefore,

$$REG^w \leq (1 - \rho')len(P) + \rho' \cdot REG^* \leq (1 - \rho')load(P) \cdot REG^* + \rho' \cdot REG^* = (\rho' + (1 - \rho')load(P)) \cdot REG^*$$

where the second inequality follows because a regenerator can be exploited by at most $load(P)$ paths and therefore $REG^* \geq len(P)$.

**Theorem 7** Combined($\text{MatchAndCut}, (G, P, g)$) is a $(7 + load(P)^2)$-approximation algorithm for ($G, P, g$) instance of Trg.

**Proof:** By Lemma 12, $\overline{M}$ is a 2-approximation to the maximum matching of $EIG(G, P)$. Substituting $\rho = 2$ in Lemma 13, we get $\rho'' = (1 + load(P)) / 2$ as the approximation ratio of Algorithm MatchAndCut. Finally, by Lemma 7, we obtain the claim.

**5 Extensions and Conclusion**

In this paper we have studied an optimization problem in optical networks, that minimizes the use of regenerators when traffic grooming is exploited. Up to here, we have considered the case in which a regenerator has to be placed at every internal node of every lightpath ($d = 1$). Now we focus on the more general problem ($G, P, g, d$) that is defined as the problem ($G, P, g$) with the difference that $d > 1$ is the maximum number of hops a path can make without meeting a regenerator.

The following theorem allows to extend the results of the previous sections to the general case of any $d > 1$.

**Theorem 8** Given a polynomial time $\rho$-approximation algorithm $\mathcal{A}$ for ($G, P, g$), for any $d > 1$, it is possible to obtain a polynomial time algorithm $\mathcal{A}'$ guaranteeing a $4 \cdot \rho$-approximation for ($G, P, g, d$).

**Proof:** Given a coloring $w$, we denote by $N_1^w, N_2^w, \ldots$ the NSIs induced by $w$, and by $N_1^*, N_2^*, \ldots$ the NSIs induced by and optimal coloring $w^*$ of ($G, P, g$).

Recall that for the case $d = 1$, the number of regenerators associated with a coloring $w$ is $REG^w = \sum_i \text{span}(N_i)$. As $w^*$ is optimal, then $\sum_i \text{span}(N_i^*) \leq \sum_i \text{span}(N_i^w)$ for any feasible coloring $w$.

---

1It is worth noticing that we cannot use a maximum matching $M^*$ of $EIG(G, P)$ because it could be the case that $EM^{-1}(M^*)$ does not exist.
Let \( \bar{w} \) the coloring returned by algorithm \( \mathcal{A} \); then \( \sum_i \text{span}(N^w_i) \leq \rho \sum_i \text{span}(N^w_i) \) for any feasible coloring \( w \).

We now turn our attention to an instance of the \((G, \mathcal{P}, g, d)\) problem in which we can discard all the paths of length at most \( d \), because they do not need any regenerator. Notice that a solution for \((G, \mathcal{P}, g, d)\) is not completely characterized by a coloring \( w \) (as in the \( d = 1 \) case), because it also has to specify where to put regenerators.

Let \( G(N) = (V(N), E(N)) \) be the graph corresponding to the NSI \( N \), i.e. such that \( V(N) = \bigcup_{P \in \mathcal{P}} \bigcup_{(u,v) \in P} \{u,v\} \) and \( E(N) = \bigcup_{P \in \mathcal{P}} \bigcup_{(u,v) \in P} \{(u,v)\} \). Without loss of generality, we consider NSIs \( N \) verifying the following property: the subgraph of \( G(N) \) induced by the nodes in \( \text{SPAN}(\mathcal{N}) \) is connected. Notice that given an NSI \( N \) not satisfying such a property, we can easily split \( N \) into two or more NSIs \( N_1, N_2, \ldots \) satisfying it such that \( \text{span}(N) = \sum_i \text{span}(N_i) \). Note that \( |E(N)| \geq \text{span}(N) \).

In order to guarantee that \( d > 1 \) is the maximum number of hops a path traverses without meeting a regenerator, it must be that any edge in \( E(N) \) is at distance at most \( d \) from a regenerator (we consider the edges incident to a node at distance 1 from it). Since a regenerator serves at most 2 edges (i.e. at most \( d \) edges on each direction) it follows that at least \( \left\lfloor \frac{|E(N)|}{2d} \right\rfloor \) regenerators are needed for \( N \). Consider an optimal solution for \((G, \mathcal{P}, g, d)\) associated with a coloring \( w^{**} \) with NSIs \( N_1^{**}, N_2^{**}, \ldots \). By the above argument, the number \( \text{REG}^{**} \) of regenerators needed by \( w^{**} \) is \( \text{REG}^{**} = \sum_i \left\lfloor \frac{|E(N_i^{**})|}{2d} \right\rfloor \geq \sum_i \left\lfloor \frac{\text{span}(N_i^{**})}{2d} \right\rfloor \geq \frac{1}{2d} \sum_i \text{span}(N_i^{**}) \).

On the other hand, starting from \( \bar{w} \), \( \mathcal{A}' \) builds a solution for \((G, \mathcal{P}, g, d)\) in the following way: for any NSI \( N_i^\bar{w} \) it puts a regenerator every \( d \) nodes in \( \text{SPAN}(N_i^\bar{w}) \), clearly satisfying the regenerator constraint. Since we have discarded all the paths with length at most \( d \), for any NSI \( N_i^\bar{w} \) we have that \( \text{span}(N_i^\bar{w}) \geq d \). Therefore, the number \( \text{REG}^\bar{w} \) used by \( \mathcal{A}' \) is \( \text{REG}^\bar{w} = \sum_i \left\lfloor \frac{\text{span}(N_i^\bar{w})}{d} \right\rfloor \leq \sum_i \left( \frac{\text{span}(N_i^\bar{w})}{d} + 1 \right) \leq 2d \sum_i \text{span}(N_i^\bar{w}) \leq 2d \sum_i \text{span}(N_i^{**}) \leq 4 \rho \cdot \text{REG}^{**} \). \( \square \)

By exploiting Theorem 8 we can extend the results of Algorithms 1 and 3 to the case of \( d > 1 \) and thus obtain, for problem \((G, \mathcal{P}, g, d)\), 16-approximation algorithms for ring and tree topologies.

The main open problem is that of finding a better approximation algorithm for general topologies.

Another natural open problem is to discover the exact approximability of the problem. The problem is NP-complete already for \( g = 2 \) and networks with path topology. In this paper we have shown that the problem is in \( \text{APX} \) in tree networks. Determining whether the problem is in \( \text{PTAS} \) for these topologies and for particular cases is an open problem.

It would be also interesting to extend our result by considering more involved cost functions taking into account other switching parameters (e.g., the ADMs - Add-Drop-Multiplexers - used at the endpoints of the lightpath) or the possibility of splitting paths. Finally, studying the on-line version of the problem is an intriguing future research direction.
References


