The chord version for SONET ADMs minimization

Leah Epstein*   Asaf Levin †

Abstract

We consider a problem which arises in optical routing. WDM/SONET rings are a network architecture used by telecommunications carriers for traffic streams. The dominant cost factor in such networks is the total number of add-drop multiplexers (ADMs) used. A list of traffic streams to be routed between pairs of nodes is given. In this paper we consider the problem where we need to assign a route and a wavelength to each traffic stream, minimizing the total number of used SONET ADMs. This is called *the chord version* of the SONET ADMs minimization problem, to denote the fact that the routing is not given a priori. The best previously known approximation algorithms for this problem have the performance guarantee of $\frac{3}{2}$. We present an improved algorithm with performance guarantee of exactly $\frac{10}{7} \approx 1.42857$. Moreover, we study some natural heuristics for this problem, and give tight analysis of their approximation ratios.

1 Introduction

WDM (Wavelength Division Multiplexing)/SONET (Synchronous Optical NETworks) rings form a very attractive network architecture. This network is used by a large number of telecom carriers. In this architecture each wavelength channel carries a high-speed SONET ring. The key terminating equipments are optical add-drop multiplexers (OADM) and SONET add-drop multiplexers (ADM). Each vertex is equipped with exactly one OADM. The OADM can selectively drop wavelengths at a vertex. Thus, if a wavelength does not carry any traffic from or to a vertex, its OADM allows that wavelength to optically bypass the vertex. Therefore, in each SONET ring a SONET ADM is required at a vertex if and only if it carries some traffic terminating at this vertex. In this paper we study the problem of minimizing the total cost incurred by the SONET ADMs (see [2, 8] for earlier studies of this problem).

Formally, we are given a set E of chords over the vertices $0, 1, \ldots, n-1$, where the vertices are ordered clockwise. The set E consists of undirected edges. The first goal is to decide on a routing (i.e. to convert each chord into an arc by choosing a single direction for it along the ring). This gives a set of arcs E'. A pair of arcs $(i, j), (k, l)$ is *non-intersecting* if the clockwise path along the cycle $0, 1, \ldots, n-1, 0$ that connects i to j and the clockwise path that connects k to l do not share any arc of the cycle. A set of arcs is non-intersecting if every pair of arcs from this set is non-intersecting. Once a routing has been determined, a feasible solution for the arcs, is a partition of E' into non-intersecting subsets of arcs $E_1, E_2, \ldots, E_p$. The cost of E_i is the number of different vertices of the ring that are end-points of the arcs of E_i. The cost of the solution is the sum of costs of E_i for all i. The goal is to find a minimum cost feasible solution.

*Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il. Research supported by Israel Science Foundation (grant no. 250/01).

†Department of Statistics, The Hebrew University, Jerusalem, Israel. levinas@mscc.huji.ac.il.

For a pair of vertices i, j we denote by $\{i, j\}$ the undirected edge (chord) that connects i and j, and we denote by (i, j) the directed edge (arc) from i to j.

For an arc (i, j), we define its *length* as $\ell(i, j) = j - i \mod n$. For a subset of arcs, the length of the subset is the total length of its arcs. For an edge $\{i, j\}$ we let the length of $\{i, j\}$ to be $\min\{\ell(i, j), \ell(j, i)\}$.

A *chain* is an open directed path of length at most $n - 1$, and a *cycle* is a closed directed path of length exactly n. W.l.o.g. we can assume that the arcs in each E_i form a connected component (either a chain or a cycle). This is true since if the arcs in E_i are disconnected, then we can partition E_i to its connected components without increasing its total cost. Therefore, we ask for a partition of E into cycles and (open-)chains. The cost of a feasible solution equals the sum of $|E|$ and the number of chains in the solution.

Calinsecu and Wan [2] proved that this problem is NP-hard. Further, they proved that any nontrivial heuristic is a $\frac{7}{4}$-approximation algorithm. I.e., any algorithm such that none of its chains can be united to form a larger chain (a local optimum) is a $\frac{7}{4}$-approximation algorithm. They provided two distinct $\frac{3}{2}$-approximation algorithms, and analyzed the worst-case of another heuristic. Below, we describe the results of [2].

The *arc version* of the problem, where the routing of each traffic stream is pre-specified, was studied in several papers [6, 5, 8, 2, 7, 3]. The best approximation ratio that is currently known is $\frac{10}{7}$ [3]. Shalom and Zaks presented a different algorithm with approximation ratio $\frac{10}{7} + \varepsilon$ [7]. Another version of the problem, where traffic streams can be divided into several parts, such that each part is treated as a separate stream was studied in [4, 1], both for the arc version and the chord version. However, the two problems (chord version and arc version) are very different, and different techniques are used to achieve results in the two models. In the chord version a routing decision precedes the partition into chains and cycles, and this changes the structure of the problem as can be seen by the following lemma.

Lemma 1 *The following properties hold in the chord version:*

1. *Given edges $\{a, b\}, \{b, c\}$, then there is a (valid) chain composed of these two edges.*
2. *Given edges $\{a, b\}, \{b, c\}, \{c, a\}$, where a, b and c are distinct, then there is a valid triangle composed of these three edges.*

Proof. To prove the first property, note that we get a cycle if $a = c$, otherwise we get the chain a, b, c if b appears before c, traversing the cycle clockwise starting from a, otherwise we get the chain c, b, a.
To prove the second property, note that one of a, b, c, a and a, c, b, a is a valid triangle (the valid triangle is determined similarly to the valid chain in the previous property). ■

Let OPT be a given optimal solution to our problem. We use OPT to denote its cost as well. For an algorithm A, we also use A to denote the cost of its returned solution. We denote the number of chains in OPT with i edges by CH_i for $i \geq 1$ and by C_i for $i \geq 2$ the number of cycles in OPT with i edges.

We now formalize Algorithm Iterative Matching (IM) (see [2]). The algorithm maintains a set of valid chains of edges $\mathcal{P}$ that covers E throughout its execution. Initially, $\mathcal{P}$ consists of (undirected) chains each of which is an edge in E. The fit graph $\mathcal{F}(\mathcal{P})$ is defined as follows: its vertex set is $\mathcal{P}$, and two of its vertices are connected by an edge if the two corresponding chains have a common end-point, and they can be concatenated to form a valid (directed) chain. The algorithm constructs $\mathcal{F}(\mathcal{P})$, and if its edge set is not empty, then it finds a maximum matching $\mathcal{M}$ in $\mathcal{F}(\mathcal{P})$. Then, it merges each matched pair of chains of edges in $\mathcal{M}$ into a longer chain. When the edge set of $\mathcal{F}(\mathcal{P})$ is empty, $\mathcal{P}$ is the valid chain generation that is given as output. Calinescu and Wan [2] showed that the approximation ratio of Algorithm IM is at most $\frac{5}{3}$, and provided a negative example for the algorithm that shows that its approximation ratio is at least $\frac{3}{2}$. We show an upper bound of $\frac{8}{5}$ on the approximation ratio, and a matching lower bound achieved by a negative example for IM.

Calinescu and Wan considered a variant of Algorithm IM: Algorithm Preprocessed Iterative Matching (PIM) defined as follows:

1. Preprocessing phase: repeatedly remove cycles consisting of remaining edges until no more cycle can be obtained.

2. Matching phase: apply Algorithm IM to the edges remaining after the first phase.

They showed that Algorithm PIM has an approximation ratio of at most $\frac{3}{2}$, and gave a negative example for PIM that shows that its approximation ratio is at least $\frac{4}{3}$.

Our results. In this paper we show that the $\frac{3}{2}$ bound for PIM is tight. We further consider several exponential algorithms, all with preprocessing phases which remove cycles in some order. We show that cycle removal starting from two edge cycles does not perform well, and due to this and the conclusion regarding PIM, we consider a modification of PIM called Modified PIM (MPIM). MPIM removes the cycles in a way that a cycle which consists of two edges is removed only in the case that no longer cycle can be removed. Note that MPIM and PIM both define classes of algorithms such that the class of MPIM is a subclass of the class of PIM. We also provide an analysis of the approximation ratio of MPIM which is better than the analysis of PIM. This improved analysis in Section 2.3 does not improve the worst case performance of the algorithm, but together with Algorithm D-DAG defined in Section 3.1 it provides Algorithm COMB, and the main result of this paper (shown in Section 3.2) is that Algorithm COMB has the approximation ratio of exactly $\frac{10}{7}$. This improves over the previously best known approximation algorithms. See Table 1 for a summary of our results and comparison to previous work.

Algorithm D-DAG basically chooses a direction for each edge such that the resulting instance has no cycles. It was shown by [5, 2] that such an instance of the arc version is polynomially solvable. Algorithm D-DAG may have the approximation ratio 2 in the worst case, but it performs quite well in the case where OPT consists mainly of cycles and chains with many edges in each. MPIM performs quite well in the case where OPT mainly consists of cycles and chains with few edges in each. Intuitively, choosing the better of the two algorithms (as Algorithm COMB does) should perform well on any input. We show that this is indeed the case.

We study the performance of two additional heuristics, IM defined above and OCIM which removes a single cycle before performing IM. We show their approximation ratios are $\frac{8}{5} = 1.6$ and $\frac{3}{2} = 1.5$, respectively.

Note that although we consider the absolute approximation ratio in this paper, all our results are valid for the asymptotic approximation ratio as well. All negative examples can be easily magnified by taking multiple copies of the instance (slightly shifted from each other, with no common vertices), to form arbitrary large negative examples.

2 Algorithms with cycle removal preprocessing

2.1 The performance of PIM

As a first step of designing an algorithm of smaller performance guarantee than $\frac{3}{2}$, we would like to study algorithms which remove cycles as a preprocessing phase. This is a natural step since the cost of cycles in a solution is low (one per edge). These algorithms can perform well on instances where an optimal solution has many cycles with small number of edges.

The first natural algorithm to consider is PIM. The paper [2] showed that the performance guarantee of PIM is in the interval $[\frac{4}{3}, \frac{3}{2}]$. We show that the upper bound of $\frac{3}{2}$ on the performance guarantee of PIM given in [2] is tight, by giving a lower bound of $\frac{3}{2}$ on the performance guarantee of the algorithm PIM. This means that PIM cannot be analyzed in a better way, and alternative algorithms need to be designed. We prove the following theorem.

Theorem 2 *The approximation ratio of PIM is exactly* $\frac{3}{2}$.

Proof. Consider the following instance. Let $n = 4j + 2$ for an integer $j \geq 1$. OPT consists of four cycles. The first cycle consists of the edges $\{a_i, a_{i+1}\}$ for $i = 0, \ldots, 2j$. Similarly, the second cycle consists of the edges $\{b_i, b_{i+1}\}$ for $i = 0, \ldots, 2j$, the third cycles has the edges $\{c_i, c_{i+1}\}$ for $i = 0, \ldots, 2j$, and the fourth cycles has the edges $\{d_i, d_{i+1}\}$ for $i = 0, \ldots, 2j$. We define $a_{2j+1} = a_0,\ b_{2j+1} = b_0,\ c_{2j+1} = c_0,\ d_{2j+1} = d_0$, and

$$a_i = \begin{cases} 2i+1 & \text{for even } i\ (i = 0, 2, \ldots, 2j) \\ 2i & \text{for odd } i\ (i = 1, 3, \ldots, 2j-1) \end{cases}$$

$$b_i = \begin{cases} 1 & \text{for } i = 0 \\ 2i & \text{for } i > 0 \end{cases}$$

$$c_i = \begin{cases} 2i & \text{for even } i\ (i = 0, 2, \ldots, 2j) \\ 2i+1 & \text{for odd } i\ (i = 1, 3, \ldots, 2j-1) \end{cases}$$

$$d_i = \begin{cases} 0 & \text{for } i = 0 \\ 2i+1 & \text{for } i > 0. \end{cases}$$

The set of edges consists of twice the edge $\{0, 3\}$, twice the edge $\{1, 2\}$, and a set of edges such that each edge appears once. For each $2 \leq i \leq 2j - 1$, the vertices $2i,\ 2i + 1$ are both adjacent to the four vertices $2i - 1,\ 2i - 2,\ 2i + 2,\ 2i + 3$. The vertex 2 is adjacent to the vertices $1, 1, 4, 5$, the vertex 3 is adjacent to the vertices $0, 0, 4, 5$, the vertices $n-1, n-2$ are both adjacent to the vertices $n-3, n-4, 0, 1$. Note that all edges have length in the set $\{1, 2, 3\}$. In the preprocessing phase, PIM may remove two cycles of two edges each which consists of the edges $\{1, 2\}, \{2, 1\}$ and $\{0, 3\}, \{3, 0\}$. After this step we claim that no valid cycle can be constructed using the remaining edges. The edges that contain the ring edge $(1, 2)$ in their shorter direction were all removed, therefore a valid cycle must contain one edge directed in its long direction. Since a cycle must consist of at least three edges (we removed all pairs of parallel edges), the long arc cannot have length $n - 1$. It also cannot have length $n - 2$ since edges of length 1 are always from an odd vertex to an even vertex in their shorter direction. However, the edges of length 3 are from an even vertex to an odd vertex in their shorter direction, therefore according to its opposite orientation, it is from an odd vertex to an even vertex, and to complete a cycle, we would need at least one arc of length 1, going from an even vertex to an odd vertex, which is impossible.

Next the iterative matching step is performed. We show a perfect matching of the edges such that the length of each resulting chain is strictly larger than $\frac{n}{2}$, and hence the iterative matching does not continue further and does not perform another iteration. In order to do that, we match pairs of edges where one edge is oriented in its short direction, whereas the other edge is matched in its long direction. This means that each chain has length of at least $(n-3)+1 = n-2 > \frac{n}{2}$ (since $n \geq 6$). The matching is as follows. All edges of length 2, that enter an odd vertex, and all edges of length 3 are oriented in the longer direction. Whereas the other edges are oriented in the shorter direction. We match the pairs of arcs $(2i+3, 2i)$,$(2i, 2i+2)$ and $(2i+3, 2i+1)$,$(2i+1, 2i+2)$ for $1 \leq i \leq 2j-1$. We also match the two pairs $(1, n-2)$,$(n-2, 0)$ and $(1, n-1)$,$(n-1, 0)$. Therefore, $OPT = 4(2j+1)$ whereas $PIM = 12j+4$. The ratio between the costs approaches $\frac{3}{2}$ as j grows. ∎

2.2 Exponential time algorithms

In this section we introduce three algorithms which have a preprocessing phase, and solve the rest of the instance optimally. Clearly, such algorithms take exponential time (unless $P = NP$). However, their performance guarantee is a lower bound on the performance guarantee of any polynomial time algorithm which has the same preprocessing phase. We study these algorithms as a motivation in designing polynomial time algorithms. The first algorithm has the same preprocessing phase as PIM, only it does not need to complete it, i.e. it can stop this preprocessing at any time.

Algorithm Cycles
Remove some cycles in arbitrary order. Solve the remaining instance optimally.

Note that Cycles does not necessarily stop when no more cycles can be removed, but it stops after removing some number of cycles.

Theorem 3 *The approximation ratio of Cycles is exactly* $\frac{11}{8} = 1.375$.

Proof. For the lower bound consider the following example where $n = 5$. The optimal solution is composed of the following cycles. $(0,1),(1,3),(3,0)$, $(1,2),(2,4),(4,1)$ and $(2,0),(0,2)$. Clearly, $OPT = 8$. If Cycles removes the cycle $(0,1),(1,2),(2,0)$, then we are left with five edges that do not form a valid cycle and cannot be covered by two chains, so they are covered by at least three chains. The total cost of the solution is 11. The approximation ratio is thus at least $\frac{11}{8}$. For the upper bound consider OPT. We may assume that it consists of cycles only. Otherwise, we can add a pseudo-edge to each chain of OPT, turning it into a cycle with no extra cost, while the cost of Cycles can only increase (we do not allow removal of pseudo-edges in the cycle removal phase). During the phase of cycle removals, we maintain a list of *pseudo-cycles* of OPT. A pseudo-cycle is either a valid cycle of OPT, or a concatenation of parts of cycles which remain after removal of the edges of one valid cycle from a set of pseudo-cycles. Therefore, a pseudo-cycle is either a valid or an invalid closed chain. A *segment*, is a maximal part of a pseudo-cycle that appeared in an original cycle of OPT, between two edges that were removed (or the two endpoints of one removed edge). Note that the number of (empty or non-empty) segments is exactly the number of edges that were removed during the cycle removal phase. We assign each pseudo-cycle an amount of removed edges that equals to the number of segments in it, and analyze them together. Consider now a pseudo-cycle. We analyze the minimum cost that can still be obtained for this pseudo-cycle together with the edges assigned to it, that were removed in the process of pseudo-cycles creation. These are the edges of at least one removed cycle, and the cost per such edge is 1. We show that this total cost is at most $\frac{11}{8}$ times the number of edges. If the pseudo-cycle consists of two or three edges, then by Lemma 1, it is valid, and therefore the cost per edge is 1. If the pseudo-cycle consists of four edges, then by Lemma 1, two chains of two edges each can be created from it. If $s \geq 2$ is the number segments, then the cost is at most $6 + s \leq \frac{4}{3}(4 + s)$. For a pseudo-cycle of at least five edges, let $u \geq 5$ be the number of edges, and s the number of (empty or non-empty) segments. Then, the number of removed edges is also s. The total number of edges is $s + u$. If the pseudo-cycle is partitioned into chains which are the segments, we get a total cost of $2s + u$. If we apply IM on the pseudo-cycle, then by Theorem 11, we get the cost $s + 3u/2$ if u is even and $s + (3u + 1)/2$ is u is odd. In both cases (odd and even u), the second cost is at most $s + 1.5u + 0.5 \leq s + 1.6u$. To conclude,

$$Cycles \leq \min\left\{2s + u, s + \frac{8u}{5}\right\} \leq \frac{3 \cdot (2s + u) + 5 \cdot \left(s + \frac{8u}{5}\right)}{8} \leq \frac{11}{8} \cdot (s + u) .$$

■

The next algorithm 2Cycles removes two edge cycles as its preprocessing phase.

Algorithm 2Cycles
Remove cycles of two edges each until all such cycles are removed. Solve the remaining instance optimally.

In [2] it was noted that in the arc version, 2Cycles has performance guarantee of 1. This is not the case for the chord version as it was shown in [2] that 2Cycles has performance guarantee of at least $\frac{4}{3}$. We show a tight analysis of the approximation ratio of 2Cycles.

Theorem 4 *The approximation ratio of 2Cycles is exactly* $\frac{15}{11} \approx 1.3636$.

Proof. For the lower bound consider the following example where $n = 7$. The optimal solution is composed of the following cycles. $(0,1),(1,3),(3,5),(5,6),(6,0),\ (2,5),(5,6),(6,2)$ and $(0,1),(1,4),(4,0)$. Therefore, $OPT = 11$. 2Cycles removes $(0,1),(1,0)$ and $(5,6),(6,5)$ as two edge cycles. The remaining seven edges do not form a valid cycle. No subset of three remaining edges can form a valid chain, and therefore the optimal solution of the remaining instance uses four chains with a total cost of 15. We get that the performance guarantee is at least $\frac{15}{11}$. For the upper bound, we follow the proof of algorithm Cycles. We only need to consider again the case $u \geq 5$. Note that since we always remove cycles of two edges, the number of segments in each pseudo-cycle is even (which we prove later). If $s = 2$, then $2s + u \leq \frac{9}{7}(s + u)$. Otherwise, $s \geq 4$. We again have the cost $\min\{2s + u, s + 1.5u + 0.5\}$. Note that $s + 1.5u + 0.5 \leq s + 1.5u + s/8$. Therefore,

$$2Cycles \leq \min\left\{2s + u, \frac{9s}{8} + \frac{3u}{2}\right\} \leq \frac{3 \cdot (2s + u) + 8 \cdot \left(\frac{9s}{8} + \frac{3u}{2}\right)}{11} \leq \frac{15}{11} \cdot (s + u) .$$

It is left to prove that the number of segments in a pseudo-cycle is indeed even. For this purpose we define the number of segments in a cycle of OPT (where none of its edges is removed) to be zero. To be able to prove the claim, we associate a pseudo-cycle with a traversal of it in one direction. This implies an orientation of each segment to be either clockwise or counter-clockwise according to its orientation along the pseudo-cycle. A clockwise orientation means that this segment is a valid chain, whereas a counter-clockwise orientation means that if this segment is traversed in the opposite direction, it becomes a valid chain. We apply the definition also to empty segments (where the orientation is meaningless, but it serves a role in the proof), and to segments which consist of a single edge. In these cases (where both orientations are possible) we initialize the orientation according to the orientation at the last time that this segment was a part of a longer segment, possibly in an original cycle of OPT.

Along a pseudo-cycle we define two segments to be an adjacent pair of segments, if there was a prior step of the algorithm in which we removed a 2-arc cycle whose arcs are adjacent to these two segments along their cycles in OPT. In our cyclic traversal of the pseudo-cycle, these two segments are consecutive. We later allow merging of two segments into a single segment in some cases, where this results in a valid chain. We will show the claim by showing that for each pair of adjacent segments, these two segments have opposite orientations (then since along the pseudo-cycle each segment has a fixed orientation, their number is even).

We will show this last claim via induction on the number of 2-arc cycles that our algorithm removes. After no 2-arc cycle is removed, then the pseudo-cycle contains a collection of cycles of OPT and the number of segments in each of them is zero, so the pseudo-cycle has zero segments. Thus the base case of the induction holds, and we can assume that the claim holds after we remove at most t 2-arc cycles to create a pseudo-cycle.

When we remove an additional 2-arc cycle (a, b), (b, a), consider a few cases. Assume first that the two arcs belong (prior to this removal) to two distinct pseudo-cycles. Orient one pseudo-cycle in a way that the segment which contains (a, b) reaches first a and then b, and orient the other pseudo-cycle in a way that the segment which contains (a, b) reaches first b and then a. If any pseudo cycle does not originally fulfill this property, in order to achieve it, we can traverse the cycle in the opposite direction, and the orientations of all segments are reversed. Let x, y be the endpoints of the segment of (a, b) in the first cycle, and s, t in the second cycles, thus the segments are paths $x \to a \to b \to y$ and $s \to b \to a \to t$. Where each of these paths is a valid chain if its orientation is clockwise, and else the reversed path is a valid chain. If the orientations of the two segments are currently opposite, we get a pseudo-cycle which traverses the first pseudo-cycle from b to a and the second one from a to b. We get that each segment was split into two, and the new pseudo-cycle is composed of segments with alternating orientations. Assume now that both segments have a clockwise orientation (otherwise we can reverse both of them and change the roles of a and b). Since both these segments are oriented clockwise, there are both valid chains and the length of each one is less than n. Thus their total length is less than $2n$. However, the removal of (a, b) from one of them and of (b, a) from the other one results in a reduction of exactly n in their total length. Therefore, we again get a pseudo-cycle which traverses the first pseudo-cycle from b to a and the second one from a to b, however the two segments adjacent to a and the two adjacent to b all have a clockwise orientation. We merge each pair of such segments into one valid segment (as shown above, the total length of all four segments is less than n, thus we get two valid chains). Thus we get a pseudo-cycle which is composed of segments with alternating orientations.

It remains to consider the case where both edges are removed from one pseudo-cycle. The proof is similar though here in one case we get two pseudo-cycles. Given a pseudo-cycle, there are several cases based on the orientations of the segments which contain the two edges (that is either clockwise or counter-clockwise), and the directions in which the two edges appear in their segments (where the direction of an edge is defined according to the orientation of the segment to be either from a to b or from b to a) .

We specify all possible cases.

1. Both segments have the same orientation, and both edges appear in the same direction.

2. Both segments have the same orientation, and the two edges appear in opposite directions (once from a to b and once from b to a).

3. The two segments have opposite orientations, and both edges appear in the same direction.

4. The two segments have opposite orientations, and the two edges appear in opposite directions.

In case 1, we may assume that both segments are oriented clockwise, and that the edges are directed from a to b (otherwise reverse the cycle or exchange the roles of a and b). We get two pseudo-paths (which are concatenations of segments). Both start and end with clockwise oriented segments, and both are from b to a. Reversing one of these pseudo-paths, we get that one of them is from a to b, and starts and ends with counter-clockwise oriented segments. Now we can connect them together into a pseudo-cycle composed of segments with alternating orientations.

In case 2, the situation is similar to the case where the two edges are in one cycle. Again, the sum of lengths of the four segments which are the remainders of the segments of the two removed edges is less than n. However, the removal of the two edges disconnects the pseudo cycle, resulting one pseudo cycle from each connected component which remain after the removal of the two edges. In each such pseudo-cycle, each pair of two segments out of the four mentioned above, which are adjacent to a and b in the two cycles respectively is merged into a single segment.

In case 3, the removal of two edges separates the pseudo-cycle into a pseudo-path from b to b which starts with a segment oriented clockwise, and ends with a segment oriented counter-clockwise, and a path from a to a which starts with a segment oriented counter-clockwise, and ends with a segment oriented clockwise. This gives two pseudo-cycles as required.

In case 4, we consider the segment which appears in a counter-clockwise orientation in its opposite direction. Thus again the sum of lengths of the remainders of the two segments is less than n. Assume that the segment which had the clockwise orientation in the original cycle has the edge (a, b) directed from a to b. The removal of two edges separates the pseudo-cycle into a pseudo-path from b to a which starts with a segment oriented clockwise, and ends with a segment oriented counter-clockwise. The other part contains a pseudo-path from b to a which starts with a segment oriented counter-clockwise, and ends with a segment oriented clockwise. In order to achieve a pseudo cycle, we reverse the second pseudo-path, and get a pseudo-path from a to b which starts with a segment oriented counter-clockwise, and ends with a segment oriented clockwise. However, similarly to case 2, the sum of lengths of all four new segments is at most n, and therefore we can merge the pair of new segments incident to a into one segment, and also merge the two new segments incident to b into one segment. ■

The next algorithm, 2CyclesFirst (2CF), does not remove longer cycles until all two edge cycles are removed.

Algorithm 2CyclesFirst
Remove cycles of two edges each until all such cycles are removed. Remove some other cycles in an arbitrary order. Solve the remaining instance optimally.

Theorem 5 *The approximation ratio of 2CF is exactly* $\frac{11}{8} \approx 1.375$.

Proof. Clearly the upper bound of Cycles holds in this case as well. For the lower bound, consider the following example. where $n = 7$. The optimal solution is composed of the following cycles.

$$(0,4),(4,5),(5,9),(9,0), \quad (0,2),(2,6),(6,0), \quad (2,4),(4,8),(8,2),$$
$$(1,5),(5,7),(7,1) \quad \text{and} \quad (3,7),(7,9),(9,3)\ .$$

Therefore, $OPT = 16$. 2Cycles cannot remove cycles of two edges since there are no such cycles in the input. It removes the six edge cycle $(0,2),(2,4),(4,5),(5,7),(7,9),(9,0)$. The ten edges form two invalid closed chains, each of which consists of five edges. No subset of three of these edges can form a valid chain, and therefore the optimal solution of the remaining instance uses six chains with a total cost of 16. We get that the performance guarantee is at least $\frac{22}{16} = \frac{11}{8}$. ■

Note that an algorithm which removes cycles in the order of the number of edges is a special case of the above algorithm and has the same performance guarantee.

2.3 Algorithm MPIM

As we saw in previous sections, removal of two edge cycles does not seem to be helpful as a preprocessing step. In this section we introduce a version of PIM which is called Modified PIM (MPIM). As in PIM, MPIM first removes all valid cycles and then performs algorithm IM. However, unlike the arc version, where two arc cycles are removed before any other cycle is removed, here we remove such cycles last. In the next section we combine MPIM with another algorithm to obtain an improved approximation algorithm.

Algorithm MPIM
Remove cycles of at least three edges each until no such valid cycles remain in the instance. Next, remove a maximal set of two edge cycles. Finally, perform algorithm IM on the remaining instance.

We are interested in the performance of MPIM as a function of the values CH_i for $i \geq 1$ and C_i for $i \geq 2$.

Lemma 6 *The cost of MPIM is at most* $\sum\limits_{i=1}^{n} \left(\frac{3i}{2} + \frac{7}{6}\right) CH_i + \sum\limits_{i=2}^{n} \left(\frac{3i}{2} - \frac{1}{3}\right) C_i$.

Proof. We assign costs to cycles and chains of OPT, such that the sum of all assigned costs is at least the total cost of MPIM. We assign the costs so that for a cycle of OPT with p edges, we assign a cost of at most $\frac{3p}{2} - \frac{1}{3}$, and for a chain of OPT with p edges, we assign a cost of at most $\frac{3p}{2} + \frac{7}{6}$.

We actually analyze an algorithm that has the following two properties. The algorithm removes cycles in the preprocessing phase until at least one edge of each cycle of OPT is removed. It does not remove a two edge cycle when a longer cycle can still be removed. The algorithm performs a single iteration of IM. The first property is actually more general than the definition of MPIM. The second property will allow us to find a lower bound on the maximum matching and use it. Clearly, the algorithm may perform better by applying the full version of IM.

Consider first a chain of OPT with p edges where no edges were removed. The contribution to the maximum matching is $\frac{p}{2}$ pairs of edges if p is even, and $\frac{p-1}{2}$ if p is odd. In the last case one extra chain which consists of a single edge, is created (in the worst case). Therefore, we assign to the chain a cost of $3(p-1)/2 + 2 = (3p+1)/2 \leq 3p/2 + 7/6$.

For a chain of OPT where at least one edge was removed, we may assume that this is a cycle of OPT (by closing it with a dummy edge). The first and second properties are kept, the cost of OPT does not change, and the cost of IM may only increase. The number of edges becomes $p' = p + 1$. We will assign a cost of $3p'/2 - 1/3$ to this cycle that is $3(p + 1)/2 - 1/3 = 3p/2 + 7/6$.

We now consider cycles of OPT. Similarly to the previous section, we consider pseudo-cycles created by removal of cycles. However, we consider each set of pseudo-cycles which is created from a connected component of OPT together. For a pseudo-cycle of u edges, we get that by matching consecutive pairs of edges we get a cost of at most $\frac{3u+1}{2}$ for IM. Clearly, this cost is reduced further if any dummy edges exist.

Given a set of pseudo-cycles, which are the remainder of a connected component of OPT after the cycle removal pre-processing phase. Let s be the number of removed edges, let k be the number of cycles of OPT that the given connected component of OPT consists of. Let t be the number of pseudo-cycles, i.e. the number of currently connected components. Note that clearly $s \geq k$, since the pre-processing stops only when there are no valid cycles. If $s = k$, exactly one edge was removed from every cycle of OPT. This cannot harm connectivity, and we get a single connected component. Every additional removed edge may create one additional connected component, thus the number of components is at most $s - k + 1$. If $k \geq 3$ we act as follows.

Given a cycle of OPT, every edge of the cycle that is not removed is assigned a cost of $\frac{3}{2}$, the same cost is assigned to a dummy edge, if exists, as these edges are never removed in the pre-processing. As for the removed edges, one of them is assigned cost $\frac{7}{6}$, whereas all other removed edges are assigned a cost of $\frac{3}{2}$. This satisfies out requirement. We explain this cost assignment to removed edges in more detail.

We analyze the first pseudo-cycle together with the k first edges removed from the k cycles of OPT. Each additional pseudo-cycle is analyzed together with one additional removed edge. This clearly covers the additional cost of $\frac{1}{2}$ that may be incurred in each pseudo-cycle on top of a cost of $\frac{3}{2}$ per edge.

For the remaining case $k = 2$ (if $k = 1$ then this single cycle must be removed optimally), consider first the case where a single connected component remains after pre-processing, and only two edges were removed. This means that a two-edge cycle was removed, but that is only allowed after no longer valid cycles exist. Thus each of the two cycles of OPT also contain exactly two edges each, and therefore we are left with a valid cycle of length two. However, this cycle cannot remain after the pre-processing phase, which is a contradiction. We can assume therefore that $s \geq 3$.

As a first step, we prove that every remaining connected component contains edges of both cycles. This holds since each connected component is Eulerian, thus it cannot contain only a proper subset of edges of one cycle. Thus, is order to achieve $t \geq 2$ connected components, we need to have $s \geq 2t$. Therefore, we charge the first edge removed from each cycle of OPT by 1 and each additional edge by $\frac{3}{2}$. For $t \geq 2$, $s - 2 \geq 2t - 2 \geq t$, and thus there are enough removed edges which are assigned a cost of $\frac{3}{2}$ to cover the additional possible cost of $\frac{1}{2}$ per pseudo-cycle. If $t = 1$, we have $s - 2 \geq 1$, and again the possible additional cost of $\frac{1}{2}$ is covered. ■

3 Main result

We present a simple algorithm which in combination with MPIM achieves a good performance guarantee.

3.1 Algorithm Directed-DAG

We now introduce algorithm Directed-DAG (D-DAG).

Algorithm D-DAG

Choose an arbitrary arc e of the ring e.g. the arc $(n-1, 0)$. Direct all edges into arcs so that they do not traverse e. The resulting instance is a directed acyclic graph, whose topological order is $0, 1, \ldots, n-1$. Solve this instance optimally using the Greedy Sweeping algorithm as in [5].

For completeness, we describe the Greedy Sweeping algorithm. This procedure starts with opening a new chain for every arc starting at vertex 0. Then, for every vertex $i = 1, \ldots, n-1$ (in this order), it merges the existing chains which end at vertex i with arcs starting at i. A new chain is opened for every arc starting at vertex i that has not been merged.

Let OPT be the value of an optimal solution for the original instance. Let OPT' be the value of an optimal solution for the directed instance. Note that every cycle and chain of OPT contains at most one edge that traverses e. Therefore, by directing the edges, a cycle of OPT is partitioned into two chains, and a chain of OPT is partitioned into at most three chains.

Lemma 7 *The cost of D-DAG is at most* $\sum_{i=1}^{n}(i+3)CH_i + \sum_{i=2}^{n}(i+2)C_i$.

Proof. The cost of D-DAG is simply OPT'. We have $OPT = \sum_{i=1}^{n}(i+1)CH_i + \sum_{i=2}^{n} iC_i$. The worst case for D-DAG is that indeed two new chains are created from each cycle and chain of OPT. Therefore, the cost per cycle or chain increases by at most two. ■

3.2 Algorithm Combination

Algorithm Combination (COMB) runs both MPIM and D-DAG, and chooses the cheaper solution.

In this section we prove the following theorem.

Theorem 8 *The approximation ratio of COMB is exactly* $\frac{10}{7} \approx 1.42857$.

We prove the theorem using two lemmas.

Lemma 9 *The approximation ratio of COMB is at most* $\frac{10}{7} \approx 1.42857$.

Proof. Since $OPT = \sum_{i=1}^{n}(i+1)CH_i + \sum_{i=2}^{n} iC_i$, we have

$$\begin{aligned}
COMB &= \min\{MPIM, \textit{D-DAG}\} \\
&\leq \frac{1}{7} \cdot (6 \cdot MPIM + \textit{D-DAG}) \\
&= \frac{1}{7} \cdot \left[\sum_{i=1}^{n}\left(6 \cdot \left(\frac{3i}{2} + \frac{7}{6}\right) + i + 3\right) CH_i + \sum_{i=2}^{n}\left(6 \cdot \left(\frac{3i}{2} - \frac{1}{3}\right) + i + 2\right) C_i\right] \\
&= \frac{1}{7} \cdot \left[\sum_{i=1}^{n} (10i + 10)\, CH_i + \sum_{i=2}^{n} 10iC_i\right] = \frac{10}{7} OPT\ .
\end{aligned}$$

■

Lemma 10 *The approximation ratio of COMB is at least* $\frac{10}{7} \approx 1.42857$.

Proof. We show an example for which COMB has an approximation ratio of $\frac{10}{7}$. We have $n = 10$, and the input consist of the fourteen edges $\{0,1\}, \{1,6\}, \{6,9\}, \{9,0\}$, $\{1,2\}, \{2,3\}, \{3,5\}$, $\{5,7\}, \{7,1\}$, $\{0,2\}, \{2,4\}, \{4,5\}, \{5,8\}, \{8,0\}$ that form three cycles in OPT, and therefore $OPT = 14$. If MPIM removes the cycle $(0,1), (1,2), (2,0)$ first, then it is left with eleven edges. We show that these edges cannot form a valid cycle. Vertex 2 cannot participate in a cycle since there is no edge between 3 and 4. Similarly, 1 cannot participate since there is no edge between 6 and 7, and 0 cannot participate since there is no edge between 8 and 9. This means that 6 and 9 cannot belong to the cycle either. We are left with four edges incident to 5, and clearly these edges do not form a cycle.

In the first phase of IM, it creates six chains which can be $1,6,9,\ 4,2,3,\ 4,5,3,\ 8,5,7,$ and $9,0,8$. The last chain consists of one edge $\{1,7\}$ that is still undirected. The first five chains have all lengths 8 or 9, and $\{1,7\}$ has length 4, and therefore the second phase does not match any pairs. Consequently, $MPIM = 20$.

Running D-DAG on the same input, we get a set of arcs where four of them start at vertex 0, and three of them at 1. Only one pair is combined at vertex 1. We may get the six chains $0,1,6,9;\ 0,2,3,5,7;\ 1,2,4,5,8;$ $1,7;\ 0,8;\ 0,9$. We get that *D-DAG* $= 20$. Therefore, $COMB = \min\{MPIM, D-DAG\} = 20$, and the approximation ratio is at least $20/14 = 10/7$. ∎

4 Other algorithms

We study two natural algorithms and find their performance. These algorithms are of practical interest being very simple. Algorithm IM was studied in previous papers. The polynomial time algorithms with cycle removal use IM as a procedure after cycle removal. We also introduce OCIM to show an algorithm which keeps the preprocessing minimal and still achieves the same performance as PIM (i.e., it is strictly better than IM).

4.1 The performance of IM

We now establish the exact approximation ratio of IM which is reviewed in the introduction. It was shown [2] that the performance guarantee of IM is in the interval $[\frac{3}{2}, \frac{5}{3}]$. We show that the following theorem holds.

Theorem 11 *The approximation ratio of IM is exactly* $\frac{8}{5}$.

Proof. The lower bound is similar to the one for IM in the arc version [3]. Let $n = 4$. The edges are $\{0,1\}$,$\{1,3\}$,$\{3,0\}$,$\{0,2\}$,$\{2,0\}$. OPT has the cycles $(0,1), (1,3), (3,0)$ and $(0,2), (2,0)$.

The matching chooses the pairs $(3,0), (0,2)$ and $(2,0), (0,1)$. The edge $\{1,3\}$ is unmatched. The resulting chains cannot be combined, so the next matching step matches nothing, and $APX = 8$ whereas $OPT = 5$.

To prove the upper bound, we consider the multigraph on the vertices of the ring, where each chord is marked with an edge. We prove the performance guarantee of IM separately for each connected component of this multigraph. It is sufficient to prove the upper bound on each connected component, as IM runs independently on each component. Consider first small components of at most three edges.

One edge: For any algorithm, the cost of this component is 2.

Two edges: The multigraph is either a path or a cycle of two edges. In both cases IM matches the two edges, and it costs the same as OPT (i.e., 2 or 3).

Three edges: IM matches at least two edges, and therefore its cost is at most 5. Therefore, if $OPT \geq 4$, then the approximation ratio of IM is at most $\frac{5}{4}$. The only possibility where $OPT = 3$, is when the component is a three edge cycle. But in this case, the first iteration of IM matches two edges of the cycle to a chain, and the second iteration matches the third edge to the chain, resulting the same cycle as OPT, and $IM = 3$.

Next, we consider components of **r edges, where r** $\geq$ **4**. We claim that for such components, already after a single iteration of IM, the performance guarantee is at most $\frac{8}{5}$ (and can only be improved during further iterations).

By Lemma 1, any pair of edges which share an endpoint can be matched. We will show in the sequel that the component can be split into pairs of such edges (possibly except one edge, if r is odd). This means that IM finds a perfect matching in its first iteration. We get the cost $\frac{3r}{2}$ for even r, and $3(r-1)/2+2 = (3r+1)/2$ for odd r. Since $OPT \geq r$, the approximation ratios are at most $\frac{3}{2}$ and $3/2 + 1/(2r)$, respectively. Since $r \geq 4$, the worst case occurs for $r = 5$, where it is $\frac{8}{5}$.

It remains to show that a connected multigraph G with r edges can be partitioned into edge-disjoint paths such that each of them contains a pair of edges, except possibly one path consisting of a single edge (if r is an odd number). We show this property by induction over r. If $r = 1$ or $r = 2$ the claim clearly holds. We assume its correctness for all values of r such that $r \leq r' - 1$, and we will show it for $r = r'$. Consider a DFS tree of the component. The edges are either tree edges or back edges. Removal of back edges does not disconnect the component. Therefore, if there is a vertex with two incident back edges (each of which can be ingoing or outgoing), we remove such a pair of edges and match them. Next, we apply the induction assumption to conclude the claim. Therefore, we can assume that each vertex has at most one incident back edge. If there is a leaf of the DFS tree that has a back edge, then match this back edge to the edge that connects the leaf to its parent, and apply the induction assumption on the remaining instance. Otherwise (leaves have no back edges) if there is a leaf of the DFS tree such that its parent has a back edge, match this back edge to the edge that connects the leaf to its parent, and apply the induction assumption on the remaining instance. Otherwise (no leaf or parent of leaf has a back edge), pick a leaf of maximum depth. If its parent has another child (that must be a leaf as well), match the two edges from the two leaves to their (common) parent, remove the two leaves, and apply the induction assumption on the remaining instance (this can be done as the tree remains connected). If the leaf is an only child, then the parent of the leaf has degree two in the tree (as $r' \geq 3$). Match the two edges incident to the parent, and remove the leaf and its parent from the tree, which remains connected, and therefore we can apply the induction assumption on the remaining instance. ■

4.2 Algorithm OCIM

In the previous sections we saw that the cycle removal preprocessing phase drops the performance ratio of IM from $\frac{8}{5} = 1.6$ to $\frac{3}{2} = 1.5$. A natural question is whether it is necessary to remove all possible cycles before applying IM. The surprising answer is that it is enough to remove one cycle (per connected component of the chord multigraph). We define the algorithm OCIM as follows.

Algorithm One-Cycle-Iterative-Matching (OCIM)
Construct the chord multigraph of the instance. Remove exactly one valid cycle from each connected component (if such a cycle exists). Run IM on the remaining instance.

We prove the following theorem.

Theorem 12 *The approximation ratio of OCIM is exactly $\frac{3}{2}$.*

Proof. The lower bounds follows from the example given in the proof of Theorem 2, since in that example, we can replace one of the two-edge cycles that we removed, by extending the matching so that these two edges are matched. For the upper bound, consider a connected component of r edges. If there are no valid cycles, then an optimal solution has at least one chain, and therefore $OPT \geq r+1$. In the proof of Theorem 11 we show that IM finds a matching of cost at most $\frac{3r+1}{2}$ for all values of r. Since $[(3r+1)/2]/(r+1) \leq 3/2$, the upper bound holds in this case. Next, consider a component where a k edge cycle is removed. This results in possible partition of the component into at most k connected components. This is so since each vertex along the removed cycle belongs to a single connected component, and each connected component contains at least one such vertex. For every $i = 1, \ldots, k$, assume that component i has $r_i \geq 0$ edges, and therefore IM returns a solution whose cost is at most $(3r_i + 1)/2$. Note that $\sum_{i=1}^{k} r_i = r - k$. The total cost of OCIM for this component is at most $k + \sum_{i=1}^{k}(3r_i + 1)/2 = k + k/2 + (3/2)(r - k) = 3r/2$. Since $OPT \geq r$, we get the upper bound $\frac{3}{2}$ for this case as well. ■

5 Conclusion

We introduced a new approximation algorithm COMB for the chord version of the problem of minimizing the number of SONET ADMs. The algorithm is a combination of two algorithms. The first one is based on a previously studied algorithm, with a new preprocessing phase. The second one is based on directing the edges. The first algorithm MPIM works well for instances where in an optimal solution there is a large proportion of cycles and chains with small numbers of arcs. The second algorithm D-DAG works well on the opposite situation, i.e., instances where in an optimal solution there is a large proportion of cycles and chains with large numbers of arcs. Algorithm COMB is the current best approximation algorithm for this problem. We showed that its approximation ratio is exactly $\frac{10}{7}$ improving the earlier $\frac{3}{2}$-approximation algorithm of [2]. Finding a better approximation algorithm (deterministic or randomized) is left for future research.

A summary of the results in the paper can be found in Table 1.

References

[1] G. Călinescu and P.-J. Wan. Splittable traffic partition in WDM/SONET rings to minimize SONET ADMs. *Theoretical Computer Science*, 276(1–2):33–50, 2002.

[2] G. Călinescu and P.-J. Wan. Traffic partition in WDM/SONET rings to minimize SONET ADMs. *Journal of Combinatorial Optimization*, 6(4):425–453, 2002.

[3] L. Epstein and A. Levin. Better bounds for minimizing SONET ADMs. In *Proc. of the 2nd Workshop on Approximation and online Algorithms (WAOA2004)*, pages 281–294, 2004.

[4] L. Epstein and A. Levin. SONET ADMs minimization with divisible paths. In *Proc. of the 3rd Workshop on Approximation and online Algorithms (WAOA2005)*, 2005. To appear.

Heuristic	Polynomial or Exponential	Lower bound on the approximation ratio in [2]	Lower bound on the approximation ratio (this paper)	Upper bound on the approximation ratio (this paper)	Upper bound on the approximation ratio in [2]
COMB	Polynomial	–	$\mathbf{10/7}$	$\mathbf{10/7}$	–
PIM	Polynomial	$4/3$	$3/2$	–	$3/2$
IM	Polynomial	$3/2$	$8/5$	$8/5$	$5/3$
OCIM	Polynomial	$4/3$	$3/2$	$3/2$	–
Cycles	Exponential	$4/3$	$11/8$	$11/8$	$3/2$
2Cycles	Exponential	$4/3$	$15/11$	$15/11$	$3/2$
2CF	Exponential	$4/3$	$11/8$	$11/8$	$3/2$

Table 1: Summary of results

[5] O. Gerstel, P. Lin, and G. Sasaki. Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings. In *INFOCOM1998*, volume 1, pages 94–101, 1998.

[6] L. Liu, X.-Y. Li, P.-J. Wan, and O. Frieder. Wavelength assignment in WDM rings to minimize SONET ADMs. In *INFOCOM2000*, volume 2, pages 1020–1025, 2000.

[7] M. Shalom and S. Zaks. A 10/7+epsilon approximation for minimizing the number of adms in sonet rings. In *Proc. of the First International Conference on Broadband Networks (BROADNETS'04)*, pages 254–262, 2004.

[8] P.-J. Wan, G. Călinescu, L. Liu, and O. Frieder. Grooming of arbitrary traffic in SONET/WDM BLSRs. *IEEE Journal on Selected Areas in Communications*, 18:1995–2003, 2000.