SONET ADMs Minimization with Divisible Paths

Leah Epstein[1,⋆] and Asaf Levin[2]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel. Email: `lea@math.haifa.ac.il`

[2] Department of Statistics, The Hebrew University, Jerusalem, Israel. `levinas@mscc.huji.ac.il.`

Abstract. We consider an optical routing problem. SONET add-drop multiplexers (ADMs) are the dominant cost factor in SONET /WDM rings. The number of SONET ADMs required by a set of traffic streams is determined by the routing and wavelength assignment of the traffic streams. In this paper we consider the version where a traffic stream may be divided into several parts and assigned different wavelengths. A specific division may increase or decrease the number of ADMs needed for a given input. Following previous work, we consider two versions. In the arc version, the route of each traffic stream is given as input, and we need to decide on divisions of streams, and then to assign wavelengths so as to minimize the total number of used SONET ADMs. In the chord version, the route is not prespecified, but is assigned by the algorithm, and only after this step the divisions are done and wavelengths are assigned. The previously best known approximation algorithm for the arc version has a performance guarantee of $\frac{5}{4} = 1.25$ whereas the previously best known approximation algorithm for the chord version has a performance guarantee of $\frac{3}{2} = 1.5$. We improve both these results. We present a $\frac{36}{29} \approx 1.24138$-approximation algorithm for the arc version and a $\frac{7}{5} = 1.4$-approximation algorithm for the chord version.

1 Introduction

WDM (Wavelength Division Multiplexing)/SONET (Synchronous Optical NETworks) rings form a very attractive network architecture that is being deployed by a growing 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 the ring carries some traffic terminating at this vertex. In this paper we study the problem of minimizing the total cost incurred by the SONET ADMs.

⋆ Research supported by Israel Science Foundation (grant no. 250/01).

Earlier studies of problems related to minimizing the total number of SONET ADMs focused on the case of wavelength continuity. I.e., the same wavelength is used on all the links of the path established for a traffic stream. Such earlier studies include [6, 5, 8, 2, 4, 3]. We later describe some of their results. However, Gerstel et al. [5] illustrated that the number of ADMs can be reduced by allowing a traffic stream to be locally transferred from one ADM in a wavelength to another ADM in a different wavelength at any intermediate vertex. Such traffic stream is *divided*. Calinescu and Wan [1] studied this problem. They showed that allowing division of traffic streams may decrease the cost of the optimal solution by a factor of $\frac{1}{3}$, where the route of each traffic stream is prespecified. This major decrease motivated the study of the problem. The paper [1] also proves NP-hardness, and presents approximation algorithms for both versions where the routing along the ring of each traffic stream is either prespecified or need to be decided. The first version is called the *arc version* whereas the second version is called the *chord version*. For the arc version [1] presents a $\frac{5}{4}$-approximation algorithm, and for the chord version it presents a $\frac{3}{2}$-approximation algorithm.

For a pair of vertices a, b we denote by (a, b) the directed edge that connects a to b, and we denote by $\{a, b\}$ the undirected edge between the two vertices.

The ARC VERSION OF MINIMIZING THE NUMBER OF ADMS WITH DIVISION (arc version) is defined as follows. We are given a set E of circular arcs over the vertices $0, 1, \ldots, n-1$, where the vertices are ordered clockwise. 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 S of arcs is non intersecting if each pair of arcs from S is non intersecting. A *division of an arc* (a, b) is defined as a clockwise path (that does not intersect itself) which connects a and b. A division of a subset S of arcs is defined as a (multi) set of arcs resulting from the division of each arc in S. A feasible solution is a partition of a division of E into non intersecting subsets of arcs $E_1, E_2, \ldots, E_p$ (i.e., we find a division of E and also a partition of it into non intersecting subsets). The cost of E_i is the number of different vertices of the ring that are endpoints of the arcs of E_i. The cost of the solution is the sum of the costs of E_i for all i. The goal is to find a minimum cost feasible solution.

The CHORD VERSION OF MINIMIZING THE NUMBER OF ADMS WITH DIVISION (chord version) is defined as follows. We are given a set of undirected edges (chords) E over the vertices $0, 1, \ldots, n-1$, where the vertices are ordered clockwise. First, we need to orient each edge $\{a, b\}$, i.e., to transform it to either (a, b) or (b, a). Last, we obtain an instance of the (divisible) arc version problem that we solve.

The NON-DIVISIBLE ARC VERSION and NON-DIVISIBLE CHORD VERSION of the problem are the variants of the arc version problem and the chord version problem, respectively, where division of arcs or chords is not allowed. These non-divisible problems were studied before in [5, 6, 8, 2, 4, 3, 7]. Calinescu and Wan [2] provided a $\frac{3}{2}$-approximation algorithms for both non-divisible problems. In [4] we considered the non-divisible arc version problem and obtained a $\frac{10}{7}$-

approximation algorithm. A different approximation algorithm of approximation ratio $\frac{10}{7} + \varepsilon$ was given by Shalom and Zaks [7]. In [3] we considered the non-divisible chord version problem and obtained a $\frac{10}{7}$-approximation algorithm as well (using a different approach).

For an arc (i, j), we define its *length* as $\ell(i, j) = j - i \mod n$. For a subset of arcs (a subgraph), the length of the subset (subgraph) is the total length of its arcs. 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 so because 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.

We define *the deficiency of a vertex* v, $def(v)$, in the arc version in the following way. Let $in(v)$ be the number of ingoing arcs of v, and let $out(v)$ be the number of outgoing arcs of V. Then, $def(v) = \frac{1}{2}|in(v) - out(v)|$. In the chord version the deficiency of a vertex v is simply zero if v has an even degree, and $\frac{1}{2}$ otherwise. The deficiency of a set of arcs (chords) S, $def(S)$ is defined as the sum of deficiencies over all vertices of the subgraph with arc (chord) set S. These definitions appeared in [1]. Note that for a set of arcs or chords C, the value $|C| + def(C)$ is a lower bound on the optimal cost for this set, both in the version of the problem where divisions are not allowed, and also in the divisible problem that we study in this paper.

Algorithm Eulerian Rounding (ER) was introduced in [1] as well. For the arc version ER works as follows: Let S be the set of arcs. Add a minimum size set of *fake* arcs F between pairs of vertices with non zero deficiency such that $def(S \cup F) = 0$. This results in an Eulerian (not necessarily connected) graph. Now choose an Eulerian tour, and divide it into $def(S)$ (not necessarily valid) chains and possibly some (not necessarily valid) cycles. In order to get valid chains and cycles, we apply the following process. Consider a chain or cycle which starts at a vertex x. All arcs in that chain or cycle, that contain the ring vertex x, are divided into two arcs. Specifically, an arc (a, b) (such that x is on the clockwise path along the ring from a to b) is divided into (a, x) and (x, b). This converts the Eulerian tour into $def(S)$ valid chains, and some valid cycles.

For the chord version ER works as follows. The set S is a set of edges. In this case F is a matching of odd degree vertices, and therefore $def(S \cup F) = 0$. We choose an Eulerian tour, and we perform the previous algorithm in the two possible directions of the tour. Once the tour is directed, we get a set of arcs, and we apply ER for the arc version. It was shown in [1] that running ER in both directions, and choosing the best solution produces a solution that costs at most $\frac{3}{2}|S| + def(S)$.

Our results. We give improved upper bounds for both versions of the divisible problem. In both cases we combine two algorithms, each of which performs well on different classes of subgraphs.
For the arc version, we combine PIM and GP-ER into Arc-Combination (AC). PIM was already introduced in [2] for the non-divisible problem. It removes

cycles and then creates chains from remaining arcs. It does not divide arcs. GP-ER has a detailed preprocessing phase, where small but dominant subgraphs are greedily removed (very small subgraphs are removed optimally). Then, it runs ER on the remaining instance. We prove that the performance guarantee of AC is at most $\frac{36}{29}$ and at least $\frac{11}{9}$.
For the chord version, we combine D-DAG and P-ER into Chord-Combination (CC). The algorithm D-DAG was already introduced in [2] where it is called Edge Avoidance Routing algorithm, and further discussed and used in [3]. This algorithm orients the edges so that the resulting graph is acyclic, and then solves this instance optimally. D-DAG does not divide any edge, but in the resulting directed instance the optimal solution does not need to divide arcs, as shown in Section 3.2. The algorithm P-ER combines the ideas of PIM and ER. It removes cycles greedily (as PIM does), but runs ER (and not IM as PIM does) on the remaining instance. We prove that the performance guarantee of CC is exactly $\frac{7}{5} = 1.4$.

2 Algorithms for the Arc Version

In this section we introduce algorithm Greedy Preprocessed-ER (GP-ER) for the arc version and afterwards combine it with PIM to produce algorithm AC that is shown to be a $\frac{36}{29}$-approximation algorithm. We first state some properties of optimal solutions.

The following lemma appears as Lemma 3 in [1]. (The proof in [1] is not complete as it fails to consider all cases, in the full version of this paper we provide a complete proof.)

Lemma 1. *For an input arc set E', denote its optimal solution by $OPT_{E'}$. Let a, b, a be a two-arc cycle in E, then $OPT_E = OPT_{E \setminus \{(a,b),(b,a)\}} + 2$.*

By the above lemma, we see that it is possible to assume that the optimal solution has a maximal number of two-arc cycles. These cycles can be found optimally as a preprocessing step of any applied algorithm. We further characterize the optimal solutions we would like to analyze.

A feasible solution SOL induces a partition of the arcs into an Eulerian subgraph and a set of *mega-chains* as follows: We consider the set of cycles and chains used by SOL as a set of arcs in a directed auxiliary graph over $\{0, 1, \ldots, n-1\}$ where cycles are loops and a chain is a directed arc from its starting vertex to its end vertex. In this directed graph we find a maximal subgraph in which the in-degree of each vertex equals its out-degree. The remaining arcs define a minimal set of chains such that each such chain is directed from a vertex whose out-degree is greater than its in-degree, towards a vertex whose in-degree is greater than its out-degree. Each such chain in the auxiliary graph corresponds to a *mega-chain* in the original graph (by replacing each arc in the auxiliary graph by its corresponding chain). Therefore, each mega-chain is composed of chains. The remaining arcs in the original graph are the arcs of the *Eulerian subgraph.* Note that the Eulerian subgraph does not need to be connected. Note that the number of mega-chains in SOL is independent of SOL, and is common to all feasible solutions.

We omit the proof of the following lemma which characterizes the mega-chains in OPT.

Lemma 2. *W.l.o.g. each mega-chain in OPT has length at most* $n-1$.

2.1 Algorithm GP-ER

We are ready to define the first algorithm we use. The algorithm has several fast preprocessing steps, and then it applies Eulerian Rounding. It is not difficult to show that it runs in polynomial time. To be able to analyze the algorithm, we need to know the exact number of mega-chains in OPT which consist of a single arc. Since we do not have this information, we apply the algorithm for every possible such value (between 0 and $|E|$), and choose the best solution we get. Therefore, in the analysis, we can assume that this number, which we denote MC_1, is known. We use two paramaters $0 \le \mu_2 \le 1$ and $0 \le \mu_3 \le 1$ which are optimized later.

begin Algorithm GP-ER

Preprocessing phase:

1. Remove a maximum number of cycles each of them having exactly two arcs.
2. Construct the following bipartite graph $B = (R_B, L_B, E_B)$: The right hand side, R_B, contains the set of vertices whose in-degree in (V, E) is greater than its out-degree, and the left hand side L_B contains the set of vertices whose out-degree in (V, E) is greater than its in-degree. For an arc $(u, v) \in E$, such that both $u \in R_B$ and $v \in L_B$, we add an edge to E_B between the two corresponding vertices. The weight of an edge is simply the length of the corresponding arc. Among all possible b-matchings of cardinality MC_1, we find a maximum weight b-matching in B where the degree bound of a vertex u is twice the deficiency of its corresponding vertex in (V, E). For each edge in the optimal b-matching, we remove the arc between its corresponding vertices.
3. As long as there exists a closed walk of three arcs (i.e., a circuit of three arcs, if its total length is n then it is a cycle called also *triangle*, and otherwise its length is $2n$ and we call it *invalid triangle*), remove such triangle or invalid triangle. If we remove an invalid triangle, we divide one of its arcs and obtain two cycles, each of them with two arcs.
4. As long as there exists a four arc cycle, remove such a cycle.
5. As long as there exists a two-arc chain of length at least $\mu_2 \cdot n$ that connects a vertex whose out-degree is greater than its in-degree, to a vertex whose in-degree is greater than its out-degree, we remove such a chain of two arcs.
6. As long as there exists a three arc chain of length at least $\mu_3 \cdot n$ that connects a vertex whose out-degree is greater than its in-degree, to a vertex whose in-degree is greater than its out-degree, remove such a three arc chain.

Eulerian rounding phase: Apply Algorithm ER on the remaining instance.

end of Algorithm GP-ER

2.2 Algorithm PIM ([2])

In this subsection we introduce two algorithms IM and PIM given in [2], the second builds on the first, and later we will use the second algorithm. Both algorithms do not divide arcs. In the previous analysis of these algorithms [2, 4] the quality of the resulting solution was compared to an optimal solution of the non-divisible arc version problem. This means that this analysis does not hold for the divisible problem.

We first define the algorithm Iterative Matching (IM) (see [2]). The algorithm maintains a set of valid chains of arcs $\mathcal{P}$ that covers E throughout its execution. Initially, $\mathcal{P}$ consists of chains each of which is an arc 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 endpoint, and they can be concatenated to form a valid 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 arcs 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 returned as output. Calinescu and Wan [2] showed that the approximation ratio of Algorithm IM for the non-divisible arc version problem is in the interval $[\frac{3}{2}, \frac{5}{3}]$. These bounds were improved to the interval $[\frac{8}{5}, \frac{5}{3})$ (if two-arc cycles are not removed as a preprocessing step) and the interval $[\frac{14}{9}, \frac{5}{3})$ (if this preprocessing is performed) in [4].

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 arcs until no more cycle can be obtained (the two-arc cycles are removed first).
2. Matching phase: apply Algorithm IM to the arcs remaining after the first phase.

For the non-divisible arc version problem, they showed that Algorithm PIM has an approximation ratio of at most $\frac{3}{2}$, and at least $\frac{4}{3}$. The lower bound was improved to $\frac{3}{2}$ in [4]. We apply PIM exactly as it is defined in [2], and therefore it is a polynomial time algorithm.

2.3 Analysis of PIM and GP-ER

We fix an optimal solution OPT which has a maximal number of two arc cycles (with no divided arcs) and each of its mega-chains has length at most $n - 1$. Note that such a mega-chain is a chain of OPT. Such an optimal solution exists by Lemmas 1 and 2. Consider first cycles of OPT with no divided arcs. For $i = 2, 3, 4$, let CY_i be the number of cycles with i arcs and no divided arcs that OPT has, and let CY be the total number of cycles in OPT with at least five arcs in each and no divided arcs.

Consider next two-arc cycles of OPT such that each has exactly one divided arc. A pair of such cycles, that have the two parts of an arc that is divided into exactly two parts, is called an *invalid triangle* of OPT. Clearly, these two cycles belong to the Eulerian subgraph. We denote by ICY_3 the number of invalid triangles in OPT.

Consider the set of all arcs which belong to the Eulerian subgraph of OPT. From this set remove all arcs that belong to cycles of OPT with no divided arcs and all arcs that belong to invalid triangles of OPT. The total length of the remaining arcs in this subset is denoted by L.

Consider next the sets of all arcs which are mega-chains which consist of a single arc. We denote the total length of all these arcs by L_1. Recall that MC_1 denotes the number of mega-chains with a single arc, and this is the exact number of mega-chains with a single arc removed by Algorithm GP-ER.

In addition, we use the following notations.

- MC - the total number of mega-chains.
- MC_2^ℓ, MC_3^ℓ - the number of mega-chains of exactly two and three arcs, respectively, with length in the interval $[\mu_2 n, n-1]$ and $[\mu_3 n, n-1]$, respectively.
- MC_2^s, MC_3^s - the number of mega-chains with exactly two and three arcs, respectively, and lengths less than $\mu_2 n$ and $\mu_3 n$, respectively.
- MC_4 - the number of mega-chains with at least four arcs.

The ℓ, s superscripts stand for long and short, respectively.

The length of E is at most $UB_L = (CY_2 + CY_3 + CY_4 + CY) \cdot n + ICY_3 \cdot 2n + L_1 + L + (MC_2^\ell + MC_3^\ell + MC_4) \cdot n + \mu_2 n MC_2^s + \mu_3 n MC_3^s$.

Denote by A the number of cycles removed in step 1, $B = MC_1$ - the number of mega-chains removed in step 2, and L_B the total length of these mega-chains. C and D are the number of valid triangles and of invalid triangles removed in step 3, respectively, F - the number of cycles removed in step 4, G - the number of mega-chains removed in step 5, and H - the number of mega-chains removed in step 6.

Then, by the optimality of step 1 using Lemma 1, we get $A = CY_2$. Moreover, step 1 is performed optimally, not only in terms of number of removed cycles, but in the sense that the removed cycles are exactly the same one as OPT has. Therefore this step cannot affect the next steps in any way. Due to the optimality of step 2 in terms of removed total length, we get $L_B \geq L_1$. By definition, $B = MC_1$.

The greedy selection rule at the step 3 implies that $B+3C+3D \geq ICY_3+CY_3$. This holds since step 3 is not over until there are no closed walks of three arcs left. Each removed (valid or invalid) triangle destroys at most three (valid or invalid) triangles of OPT. Each chain removed at step 2 can destroy at most one cycle.

The greedy selection rule at the step 4 implies that $B + 3C + 3D + 4F \geq ICY_3 + CY_3 + CY_4$. As before, each removed chain in Step 2 destroys at most one structure among valid or invalid triangles, and four-arc cycles. Among the structures of OPT that are left after step 2, each structure removed at step 3 destroys at most three structures such that each of these is either a (valid or invalid) triangle of OPT or a four arc cycle. Step 4 is not over until there are no valid four arc cycles. Each such removed cycle can destroy up to four four-arc cycles of OPT.

In the next step, long mega-chains of two arcs are removed. This step, unlike the two previous ones, can stop even if there still exist in the input long mega-chains of OPT with two arcs. The reason is that if the deficiency of at least one endpoint of such a chain becomes zero (as a result of removals in this step or before), we do not remove this mega-chain.

In step 2, the removal of a single mega-chain can destroy at most two mega-chains that belong to MC_2^ℓ. Consider such an arc (a,b). If (a,b) does not participate in any chain of OPT in MC_2^ℓ, then it can still cause a reduction of $\frac{1}{2}$ in the deficiencies of a and b, and thus a reduction in the amount of mega-chains that needs to be removed for each of them by 1 each. Otherwise, assume without loss of generality that (a,b) participates in the mega-chain $a-b-c$ of OPT. In this case the following two things happen. The mega-chain $a-b-c$ is destroyed, and on top of that, there may be a decrease of $\frac{1}{2}$ in the deficiency of b, which can lead to a decrease of 1 in the number of mega-chains to be removed. In the first case, it could still be that (a,b) destroys also a cycle that could be removed in step 3 or step 4. The argument regarding step 5 is similar, each removed mega-chain can prevent from two other mega-chains from being removed. Note that steps 3 and 4 cannot change the deficiency of any vertex, therefore the only way that they can destroy a mega-chain is by removing one of its arcs as an arc in a removed structure. Therefore, each removed structure of step 3 may destroy at most three mega-chains, and of step 4 at most four. We conclude that $3B+3C+3D+4F+4G \geq ICY_3+CY_3+CY_4+MC_2^\ell$.

In the very last step, we remove long mega-chains of three arcs. We can see that each arc removed at step 2 may destroy or prevent from removal up to three mega-chains at the current step. That is since on top of reducing the deficiency for the endpoints of this arc, it can be the middle arc in a mega-chain of OPT. The situation regarding the cycle removal steps is as before, since they do not change deficiency. A removed chain of the previous step can destroy at most two chains by using their arcs, and two additional chains by decreasing deficiency. A removed chain of the current step is has a similar effect, but can destroy up to three chains by using their arcs, and two additional chains by decreasing deficiency. We get $3B+3C+3D+4F+4G+5H \geq ICY_3+CY_3+CY_4+MC_2^\ell+MC_3^\ell$.

We note that the cost of the approximated solution is at most the sum of $|E|+MC$ and the cost of arc divisions, that is one per n units of length. This holds since chains are produced by the algorithm at each time only between pairs of vertices with non zero deficiency. The number of chains is thus no larger than the sum of deficiencies, which is a lower bound on MC. Let UB'_L be an upper bound on the total length of arcs which are not removed in the pre-processing step.

Therefore, the cost of the approximated solution of GP-ER satisfies $GP\text{-}ER \leq |E|+MC+\frac{UB'_L}{n}+D$. To be able to use this bound, we would like to find an upper bound on the value UB'_L+Dn. Since we have an upper bound on UB_L, we can get this from a lower bound on the length of removed structures. This length is at least $An+L_B+Cn+2Dn+Fn+\mu_2 Gn+\mu_3 Hn$, therefore we need a lower bound on $An + L_B + Cn + Dn + Fn + \mu_2 Gn + \mu_3 Hn$. Since we have $A = CY_2$ and $L_B \geq L_1$, we focus on $Cn + Dn + Fn + \mu_2 Gn + \mu_3 Hn$. Let α, β, γ and δ be non-negative values. We further have the requirements $\alpha + \beta + \gamma + \delta \leq \frac{1}{3}$, $\beta + \gamma + \delta \leq \frac{1}{4}$, $\gamma + \delta \leq \frac{\mu_2}{4}$, $\delta \leq \frac{\mu_3}{5}$, $\mu_2 + \gamma + \delta \leq 1$, $\mu_3 + \delta \leq 1$. Using these values as multipliers for the linear inequalities above, we get the following.

$$\begin{aligned}&\alpha(B + 3C + 3D) + \beta(B + 3C + 3D + 4F)\\&+ \gamma(3B + 3C + 3D + 4F + 4G) + \delta(3B + 3C + 3D + 4F + 4G + 5H)\\&\geq (ICY_3 + CY_3)(\alpha + \beta + \gamma + \delta) + CY_4(\beta + \gamma + \delta) + MC_2^{\ell}(\gamma + \delta) + MC_3^{\ell}\delta.\end{aligned}$$

The left hand size of the above expression is

$$\begin{aligned}&\alpha(B + 3C + 3D) + \beta(B + 3C + 3D + 4F)\\&+ \gamma(3B + 3C + 3D + 4F + 4G) + \delta(3B + 3C + 3D + 4F + 4G + 5H)\\&\leq MC_1(\alpha + \beta + 3\gamma + 3\delta) + C + D + F + \mu_2 G + \mu_3 H.\end{aligned}$$

Therefore, we have $An + L_B + Cn + Dn + Fn + \mu_2 Gn + \mu_3 Hn \geq CY_2 n + L_1 + (ICY_3 + CY_3)n(\alpha + \beta + \gamma + \delta) + CY_4 n(\beta + \gamma + \delta) + MC_2^{\ell} n(\gamma + \delta) + MC_3^{\ell} n\delta - MC_1(\alpha + \beta + 3\gamma + 3\delta)$.

Finally, we derive the following corollary.

Corollary 1.

$$\begin{aligned}\text{GP-ER} \leq\ & |E| + MC + CY_3(1 - \alpha - \beta - \gamma - \delta) + ICY_3(2 - \alpha - \beta - \gamma - \delta)\\&+ CY_4(1 - \beta - \gamma - \delta) + CY + MC_2^{\ell}(1 - \gamma - \delta) + MC_3^{\ell}(1 - \delta) + MC_4\\&+ \mu_2 MC_2^s + \mu_3 MC_3^s + MC_1(\alpha + \beta + 3\gamma + 3\delta) + \frac{L}{n}.\end{aligned}$$

We denote by E' the set of arcs in OPT (after dividing some of the arcs in the original set E). Then, $OPT = |E'| + MC$.

We allocate the cost of GP-ER among the subgraphs of OPT (a cycle with or without divided arcs in the Eulerian subgraph, or a mega-chain). To do so, we define a *size* of an arc in E'. We use a parameter $s = \frac{2}{3}$. If the arc was already in E (i.e., if the arc is not a result of division), then its size is one. If it is a result of division of an arc of E into k parts we define its size to be $\frac{1}{k}$. We will change the definition of size for arcs with size $\frac{1}{2}$ in the following way. Consider a pair of *twin arcs* $e, e' \in E'$ that result from dividing a common arc of E (and each has size $\frac{1}{2}$). Consider the subgraphs of OPT that contain e and e'. If at least one of these subgraphs contains also another arc whose size is less than one, then we pick either e or e' that belongs to such a subgraph, and we increase its size to s whereas we reduce the size of its twin arc to $1 - s$. We apply this as long as there exists such pair with at least one of them in a subgraph that contains another divided arc. If there is a pair of twin arcs e, e' such that e is contained in a mega-chain and e' is contained in the Eulerian subgraph of OPT, then we increase the size of e to s and decrease the size of e' to $1 - s$. We apply this procedure for all possible pairs. At the end of this procedure, each arc with size $\frac{1}{2}$ has its *twin arc* with size also $\frac{1}{2}$ and such that the subgraphs of OPT that contain them belong to the Eulerian subgraph and they do not have any other divided arc.

We allocate the cost of GP-ER as follows: for each subgraph of OPT we initialize its allocated cost as the total size of its arc set (this will allocate $|E|$ of the total cost). For each mega-chain of OPT we increase its allocated cost by one (in order to take into account the term MC), and apply the following.

For a three arc cycle of OPT we increase its allocated cost by $1-\alpha-\beta-\gamma-\delta$. For a four arc cycle of OPT we increase its allocated cost by $1-\beta-\gamma-\delta$. For a one-arc mega-chain of OPT we increase its allocated cost by $\alpha+\beta+3\gamma+3\delta$. For a two-arc mega-chain of OPT we increase its allocated cost by $1-\gamma-\delta$. For a three arcs mega-chain of OPT we increase its allocated cost by $1-\delta$. For a mega-chain of OPT with at least four arcs, we increase its allocated cost by 1. The remaining cost corresponds to the part of the Eulerian subgraph that does not consist of cycles. For each such cycle in OPT (that contains divided arc), we check if it is part of a pair that results from division of an arc along an invalid triangle. If so we increase the allocated cost of such cycle by $\frac{2-\alpha-\beta-\gamma-\delta}{2}$ and otherwise we increase the allocated cost of the cycle by 1.

Therefore, by Corollary 1, the total allocated cost is at least the cost of the solution returned by GP-ER. Next, we do a similar allocation for PIM. We do not allocate the real cost of PIM but the cost of the following algorithm that is inferior to PIM (i.e. its cost it at least the cost of PIM). This algorithm removes cycles in the exact same way as PIM does. Next, it considers each cycle and chain of OPT. A valid matching is created on the arcs which are not removed in the preprocessing step. The cost of removed cycles together with the cost of this matching is an upper bound on the cost of PIM. This holds since PIM finds a maximum size matching on the remaining arcs (and continues further to combine chains afterwards) whereas we define some valid (not necessarily maximum) matching on the same arc set. The first step in creating our matching is removal of all divided arcs. We leave these arcs unmatched. After removal of cycles and divided arcs from OPT, we are left with chains. A cycle where i arcs were removed results in at most i chains. A chain where i arcs were removed results in at most $i+1$ chains. Note that according to the definition of PIM, each cycle has at least one removed arc. PIM does not apply IM until the instance does not contain cycles (composed of non-divided arcs). If a cycle has a divided arc, then this arc is removed as well. Clearly, we are left with chains only at this time. Next, each remaining chain is partitioned into pairs of consecutive arcs that form the matching. Each chain may have one unmatched arc (the last one). The cost per divided arc is 2 as it is unmatched. However, since it is divided, we allocate a unit cost for every part of it, and this covers the cost of these arcs. The cost per arc that is removed in the preprocessing is 1 as well. The cost for a remaining chain of j arcs is $\frac{3j+1}{2}$ if j is odd, and $\frac{3j}{2}$ if j is even. The cost of a component of OPT is simply the sum of the allocated cost of its removed arcs and the cost of the matching defined on its remaining arcs. Therefore, the allocated cost of a cycle of OPT with k arcs, where i arcs were removed, is at most $i + 3(k-i)/2 + i/2 = 3k/2$. The allocated cost of a chain of OPT with k arcs, where i arcs were removed, is at most $i+3(k-i)/2+(i+1)/2 = (3k+1)/2$. We need to consider six special cases where the cost allocated to a cycle or chain is actually slightly smaller.

1. A cycle of OPT with two arcs, none of them being a divided arc. These cycles are removed optimally, and therefore the cost allocated to each of them is 2.
2. A chain of OPT which consists of a single divided arc. We allocated a unit cost to this chain.
3. A cycle of OPT with three arcs, none of them being a divided arc (this was considered in [4]). If it has a single removed arc, the remaining chain has an even number of arcs (two), and therefore the allocated cost is 4. If it has two removed arcs, then the remaining chain has cost 2, and the total cost is again 4.
4. A cycle of OPT with five arcs, none of them being a divided arc (this was considered in [4]). If it has a single removed arc, the remaining chain has an even number of arcs (four), and therefore the allocated cost is 7. If it has two removed arcs, three arcs are left, and there is always an adjacent pair of arcs among these three, thus the arcs are split into two chains, of lengths 1 and 2, and the total cost is again 7. If it has three or more removed arcs, then the cost per remaining arc is not larger than 2, and thus the total cost is no larger than 7.
5. A chain of OPT with two arcs, none of them being a divided arc. If no arcs are removed, the allocated cost is 3 (the two arcs are matched exactly as in OPT). If one of them is removed, then the cost for the other one is 2 and in total 3.
6. An invalid cycle of OPT with four arcs. This is a pair of cycles of OPT, each of which contains a single divided arc, and the two divided parts are the only two parts of the same original arc. Considering the resulting cycles of OPT we get two cycles, one of three arcs and the other one of two arcs. The cost of PIM for the two-arc cycle is at most 3. In the longer cycle, if at least one arc is removed in the pre-processing, then the total cost is at most 4. Otherwise, a two-arc chain can be built from the two non-divided arcs, and the cost of the longer cycle is 4 again. This gives a total of at most 7.

2.4 Algorithm Arc-Combination

Algorithm AC is defined as applying both PIM and GP-ER and choosing the better solution. The parameters used for GP-ER are $s = \mu_2 = \mu_3 = \frac{2}{3}$, $\alpha = \frac{1}{12}$, $\beta = \frac{1}{8}$, $\gamma = 0$ and $\delta = \frac{1}{8}$. This algorithm has a performance guarantee that is better than $\frac{5}{4}$ as we establish in the following theorem.

Theorem 1. *Algorithm AC has an approximation ratio of at most $\frac{36}{29} \approx 1.24138$.*

The proof is done by a careful analysis of the costs allocated by each algorithm to each subgraph of OPT. In the full version of the paper, we show an example where the approximation ratio of AC is exactly $\frac{11}{9}$, therefore the approximation ratio of AC is at least $\frac{11}{9} \approx 1.22222$.

3 Algorithms for the Chord Version

In this section we study the chord version of the problem. We develop a $\frac{7}{5}$-approximation algorithm that improves the earlier $\frac{3}{2}$-approximation algorithm of [1]. Our algorithm is composed of a pair of algorithms: a new algorithm named P-ER and algorithm D-DAG that was studied in [2, 3] for the non-divisible chord version problem.

3.1 Algorithm P-ER

In this subsection we introduce algorithm Preprocessed-ER which is a version of algorithm ER for the chord version (see Section 1). The preprocessing phase works as follows. Remove valid cycles one by one until the instance contains no valid cycles. Then, perform algorithm ER.

Therefore, the algorithm is composed of a preprocessing step that was suggested in [2] for PIM, and afterwards applying algorithm ER.

Let S be the input set of edges. Let S_1 be the set of edges that we remove during the preprocessing phase, and let $S_2 = S \setminus S_1$. Since the preprocessing phase removes cycles only, we have $def(S) = def(S_1)$. The cost of the cycles removed in the preprocessing phase is $|S_1|$. The cost of the solution returned by ER when applied to S_2, is $\frac{3}{2}|S_2| + def(S_2)$. Therefore, the total cost of the solution returned by algorithm P-ER satisfies $P\text{-}ER = |S_1| + \frac{3}{2}|S_2| + def(S)$.

Consider a given optimal solution. Let CH_i denote the number of chains in the optimal solution which contain i (original or divided) edges. Let C_i denote the number of cycles in the optimal solution which contain i (original or divided) edges. We are interested in the performance of P-ER as a function of the values CH_i for $i \geq 1$ and C_i for $i \geq 2$.

We charge the cost of P-ER to the components of OPT so that the total charged amount is at least the cost of P-ER.

Each edge of S_1 is charged one unit, and each edge of S_2 is charged $3/2$ units. If an edge was divided into k parts in OPT, each part is charged by an equal share (i.e., $1/k$) so that the total is the amount charged for the edge. On top of that, each chain of OPT is charged by 1. In this way we cover at least the cost of $def(S)$ that we have (the number of chains in OPT is at least this number). We are ready to prove the following lemma.

Lemma 3. *The cost of P-ER is at most* $\sum_{i=1}^{n} \left(\frac{3i}{2} + 1\right) CH_i + \sum_{i=2}^{n} \left(\frac{3i}{2} - \frac{1}{2}\right) C_i$.

Proof. Consider a cycle of OPT with p (original or divided) edges, we show that its charged cost is at most $\frac{3p}{2} - \frac{1}{2}$. We show that for a chain of OPT with p (original or divided) edges, its charged cost is at most $\frac{3p}{2} + 1$.

Consider first a chain of OPT with p edges. The charged cost of each (original or divided) edge is at most $3/2$. Recall that we charge an extra unit for each chain, therefore we charged at most $3p/2 + 1$ in total to the chain.

It is left to consider cycles of OPT. On one hand if the cycle consists of original edges only, then it has at least one edge in S_1, since otherwise it would have been removed before P-ER completes the preprocessing. This edge is charged by 1, and the other edges are charged by at most $\frac{3}{2}$. We conclude that the charged cost is at most $3(p-1)/2+1 = 3p/2-1/2$. On the other hand, if the cycle has at least one divided edge, then its charged cost is at most $3(p-1)/2+3/4 = 3p/2-3/4$, since at least one of its edges is charged a cost of at most $\frac{3}{4}$, and the other edges have charged cost of at most $\frac{3}{2}$ each. The worst case (i.e. with higher charged cost) is $3p/2 - 1/2$.

3.2 Algorithm Directed-DAG

In this subsection we suggest to use algorithm Directed-DAG (D-DAG) that was previously considered by [2, 3] for the non-divisible chord version problem.

Algorithm D-DAG. Choose an arbitrary edge e of the ring e.g. the edge $(n-1, 0)$. Direct all edges into arcs such 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 (for the non-divisible version) 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 0. Then, for every vertex in $i = 2, \ldots, n-1$ (in this order), it merges the existing chains which end at the vertex i with arcs starting at i. A new chain is opened for every arc starting at vertex i that has not been merged.

Note that D-DAG does not divide any chords. We now show that given a directed acyclic graph whose topological order is $0, 1, \ldots, n-1$, there exists an optimal solution for the chord version problem that does not divide any edge. Consider a solution where an edge f is divided into at least two arcs, two of which are (a, b) and (b, c). Consider the two chains in which the two arcs participate (if they belong to a common chain, there is no reason for division). Let x and y be the left endpoints, and z and t the right endpoints, respectively. We create a pair of new chains in the following way. We use the sub-chain from x to b, and concatenate the sub-chain from b to t to it. The other chain that we create, consists of the two remainders, i.e. the sub-chain from y to b and the sub-chain from b to z. In the first created chain, we no longer need an ADM at b, and therefore the cost of the solution is reduced by 1. Therefore, we obtain a contradiction to the optimality of the original solution. Therefore, as we claimed an optimal solution does not divide any arc in the resulting instance.

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

The proof of the following lemma appears in the full version.

Lemma 4. *The cost of D-DAG is at most* $\sum_{i=1}^{n}(i+3)CH_i + \sum_{i=2}^{n}(i+2)C_i$.

3.3 Algorithm Chord-Combination

Algorithm Combination (CC) runs both P-ER and D-DAG, and chooses the cheaper solution.

Theorem 2. *The approximation ratio of CC is exactly* $\frac{7}{5} = 1.4$.

The upper bound proof is based on the analysis in the previous sections. The complete proof can be found in the full version of the paper.

4 Conclusion

We presented improved algorithms for the divisible ADM minimization problem on rings, both for the arc version and the chord version. Obtaining better approximation algorithms is left as future work. An interesting open problem is whether the (divisible) arc version is actually easier than the (divisible) chord version. In the non-divisible problem, the current best known upper bounds are equal (though achieved by different algorithms using different approaches). The same question can be asked there as well; which version is harder to approximate?

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. The chord version for SONET ADMs minimization. *Theoretical Computer Science*. To appear.
4. 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.
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 $10/7 + \varepsilon$ 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.