Better Bounds for Minimizing SONET ADMs

Leah Epstein[1,⋆] and Asaf Levin[2]

[1] School of Computer Science, The Interdisciplinary Center, Herzliya, Israel
lea@idc.ac.il.
[2] Department of Statistics, The Hebrew University, Jerusalem, Israel
levinas@mscc.huji.ac.il

Abstract. 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. Following previous work, we consider the problem where the route of each traffic stream is given as input, and we need to assign wavelengths so as to minimize the total number of used SONET ADMs. This problem is known to be NP-hard, and the best known approximation algorithm for this problem has a performance guarantee of $\frac{3}{2}$. We improve this result, and present a $\frac{10}{7}$-approximation algorithm. We also study some of the previously proposed algorithms for this problem, and give either tight or tighter analysis of their approximation ratio.

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 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.

Formally, 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 paths 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

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

G. Persiano and R. Solis-Oba (Eds.): WAOA 2004, LNCS 3351, pp. 281–294, 2005.

cycle. A set of arcs is non-intersecting if each pair of arcs from this set is non-intersecting. A feasible solution 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.

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. Throughout the paper we often use vertex numbers x where $x \geq n$ to denote the vertex $x \mod n$. We omit the mod operation to simplify notations.

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. The cost of a feasible solution equals the sum of $|E|$ and the number of chains in the solution.

Liu, Li, Wan and Frieder [3] proved that this problem is NP-hard. They also considered a set of heuristics, and tested them empirically. Gerstel, Lin and Sasaki [2] also designed some heuristics for this problem. Wan, Calinsecu, Liu and Frieder [4] proved that any nontrivial heuristic is a 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 7/4-approximation algorithm. Calinsecu and Wan [1] provided a 3/2-approximation algorithm, and analyzed the worst-case performance of the previously studied heuristics. Below, we further describe the results of [1].

Let OPT be a given optimal solution to our problem with cost *opt*. Assume that for $i = 2, 3, \ldots$, OPT has CY_i cycles with i arcs, and for $i = 1, 2, \ldots$, OPT has CH_i chains with i arcs. We further assume that CY_2 is maximized among all optimal solutions, and as noted in [1] that no feasible solution can have a higher value of CY_2. For an algorithm A, we also use A to denote the cost of its returned solution. We sometimes use APX to denote the cost of a solution returned by an approximation algorithm.

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 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 Eulerian subgraph. Note that the Eulerian subgraph need not be connected. Note that the number

of mega-chains in SOL is independent of SOL, and is common to all feasible solutions.

We now formalize Algorithm Iterative Matching (IM) (see [1]). 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 end-point, 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 given as output. Calinescu and Wan [1] 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 improve the negative examples of the algorithm by presenting an example where the approximation ratio of the algorithm is at least 1.6. For a variant PPIM of algorithm IM with a preprocessing step that removes all cycles with two arcs each, we present a negative example that shows that the approximation ratio of PPIM is at least $\frac{14}{9}$. We further show that the approximation ratio of IM is strictly less than 5/3.

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.
2. Matching phase: apply Algorithm IM to the arcs 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 $\frac{4}{3}$. We show that the $\frac{3}{2}$ bound is tight. We also provide a better analysis of the approximation ratio of the algorithm. This improved analysis in Section 3 does not improve the worst case performance of the algorithm, but together with our Algorithm GPTS defined in Section 4 it provides Algorithm COMB, and the main result of this paper (shown in Section 5) is that Algorithm COMB is a 10/7-approximation algorithm. We show that the approximation ratio of algorithm COMB is at least $\frac{4}{3}$.

Algorithm Preprocessed Cut and Merge (PCM) is defined as follows:

1. Remove all cycles of two arcs each.
2. Choose a cycle's arc $(i, i+1)$ with minimum load, and let B_i denote the subset of E that pass through $(i, i+1)$. Partition $E \setminus B_i$ into an optimal set of chains using a greedy procedure. Let $\mathcal{P}$ be the obtained chains.
3. Construct a weighted bipartite graph with sides B_i and $\mathcal{P}$ as follows: if $a \in B_i$ can be merged with $P \in \mathcal{P}$, add an edge between a and P with weight equal to the number of their common end-vertices. Find a maximum-weight matching in the resulting graph. Merge each pair of arc and a chain into a larger chain. This step is repeated until no further merging can be obtained.

Calinescu and Wan [1] proved that the approximation ratio of PCM is between 3/2 and 5/3. In this paper we close this gap, and improve the lower bound on the approximation ratio of PCM by showing that the approximation ratio of PCM is exactly 5/3.

Algorithm Preprocessed Eulerian Tour-Trail Splitting (PET-TS) is defined as follows:

1. Remove all cycles of two arcs each.
2. Eulerian tour phase: add a minimum size set of fake arcs E' to make the directed graph with arc set $E \cup E'$ Eulerian (and if it is disconnected, each connected component is Eulerian). Find an Eulerian tour in this graph, and remove all the fake arcs from this tour to obtain a set of trails.
3. Trail decomposing phase: Decompose each trail into simple paths and circuits.
4. Chain split phase: split each (open) path into valid chains by walking along the path from its first arc, and generating a valid chain whenever overlap occurs; split each invalid circuit into valid chains by walking along the circuit from each arc, generating a valid chain whenever overlap occurs, and then choose the one with the smallest number of open chains.
5. Chain merging phase: Repeatedly merge any pair of open chains into a larger valid chain until no more merging can occur.

Calinescu and Wan [1] proved that the approximation ratio of PET-TS is between 3/2 and 7/4. We narrow this gap by showing that the approximation ratio of PET-TS is at least 5/3.

The paper [1] also considered MCC-TS that is a variation of PET-TS in which the Eulerian tour phase is replaced by the following: define a weighted directed graph $H(E)$ with vertex set E as follows. For any pair of non-intersecting arcs $e_1, e_2 \in E$, such that $e_1 = (i, j)$ and $e_2 = (k, l)$, add an arc from e_1 to e_2 and an arc from e_2 to e_1. If e_1, e_2 do not share any end-vertices, then the weights of both arcs are set to two. If $j = k$, then the weight of the arc from e_1 to e_2 is set to one, and the weight of the arc from e_2 to e_1 is set to two. Otherwise, the weight of the arc from e_1 to e_2 is set to two, and the weight of the arc from e_2 to e_1 is set to one. Now, find a minimum weight circuit cover of $H(E)$. Remove from it all the arcs of weight two to obtain a set of paths and circuits. Calinescu and Wan [1] proved that the approximation ratio of MCC-TS is between 3/2 and 8/5. We close this gap by showing that the approximation ratio of MCC-TS is exactly 14/9. If the pre-processing step of two arcs cycles removal is not performed, we show that the bound 8/5 is tight (we call this algorithm NMCC-TS).

Note that although we consider the absolute approximation ratio in this paper, all results are valid for the asymptotic approximation ratio as well. All negative examples can be easily magnified by taking multiple copies of each input arc, to form arbitrary large negative examples.

2 Negative Examples

In this section we give negative examples where the approximation ratio is at least 3/2. This will show that the upper bound of 3/2 on the performance of PIM given in [1] is tight i.e. that the following theorem holds.

Theorem 1. *The approximation ratio of PIM is exactly* $\frac{3}{2}$*. The approximation ratio of any algorithm that removes cycles in an arbitrary order (even if it removes the two arc cycles first) and then solves the remaining instance, is at least* $\frac{3}{2}$*.*

If we are interested in the design of a better approximation algorithm, the negative examples in this section exclude the option that a better analysis of PIM or a design of a similar algorithm that replaces the matching phase may be the answer.

Proof. We start with a very simple example showing that an algorithm which removes cycles in an arbitrary way cannot perform better than 3/2. Let $n = 3$ and the input arcs be $(0,1)$,$(0,2)$,$(1,2)$,$(1,0)$, $(2,0)$,$(2,1)$. Clearly, OPT consists of three two arc cycles which are $(i, i+1), (i+1, i)$ for $i = 0, 1, 2$, and therefore $opt = 6$. However, if the algorithm removes the cycle $(0,1), (1,2), (2,0)$, then it is left with three arcs of length $2 > n/2$ that cannot be combined. Therefore, we have $APX = 9$. This gives approximation ratio of at least 3/2.

The above input consists of two arcs cycles only. As it was already noticed in [1], it is easy to remove such cycles before processing any algorithm, and prevent the situation above. In the next example we show that even if there are no two arc cycles in the input, still an arbitrary removal procedure cannot reach smaller performance ratios. Moreover, we consider the following **exponential-time** algorithm: first, remove all cycles of two arcs, next, remove cycles one after the other until the remaining arcs do not contain a cycle, and finally solve optimally (in exponential time) the remaining arcs. We show that this algorithm has an approximation ratio of at least 3/2. Since this algorithm outperforms PIM, we conclude that it is a 3/2-approximation algorithm (however, not a polynomial-time).

For a given integer parameter $\alpha \geq 2$, consider $n = 2\alpha^2 - 4\alpha + 4$, and the arc set (with the optimal solution) is given by: for every $0 \leq i \leq \alpha - 3$ and every $0 \leq j \leq \alpha - 3$, we have the arcs $(\alpha j + i, \alpha j + i + 1)$, $(\alpha j + i + 1, n - \alpha i - j - 5)$, $(n - \alpha i - j - 5, n - \alpha i - j - 4)$ and $(n - \alpha i - j - 4, \alpha j + i)$. For every $0 \leq i \leq \alpha - 4$, we have the arcs $(n - \alpha(i+1) - 4, n - \alpha(i+1) - 3)$, $(n - \alpha(i+1) - 3, n - \alpha(i+1) - 2)$ and $(n - \alpha(i+1) - 2, n - \alpha(i+1) - 4)$. For every $1 \leq j \leq \alpha - 3$, we have the arcs $(\alpha j - 2, \alpha j - 1)$, $(\alpha j - 1, \alpha j)$ and $(\alpha j, \alpha j - 2)$. Finally, we have the twelve arcs of the following four triangles: $(\alpha^2 - 2\alpha - 2, \alpha^2 - 2\alpha - 1, \alpha^2 - 2\alpha)$, $(\alpha^2 - 2\alpha, \alpha^2 - 2\alpha + 1, \alpha^2 - 2\alpha + 2)$, $(n - 4, n - 3, n - 2)$ and $(n - 2, n - 1, 0)$. Then, OPT has $(\alpha - 2)^2$ cycles of four arcs and $2(\alpha - 2) + 4$ cycles of three arcs, and its total cost is exactly $4(\alpha - 2)^2 + 3[2(\alpha - 2) + 4] = 4\alpha^2 - 10\alpha + 16$.

We now argue that the instance contains the arc $(t, t+1)$ for every t. The arcs $(0,1), \ldots, (\alpha^2 - 2\alpha - 3, \alpha^2 - 2\alpha - 2)$ are given by the arcs $(\alpha j + i, \alpha j + i + 1)$

for $0 \leq i \leq \alpha-3, 0 \leq j \leq \alpha-3$. In this set there is a gap of two arcs every $\alpha-2$ arcs which is filled by the arcs $(\alpha j-2, \alpha j-1), (\alpha j-1, \alpha j)$ for $1 \leq j \leq \alpha-3$. Similarly the arcs $(\alpha^2-2\alpha+2, \alpha^2-2\alpha+3), \ldots, (n-5, n-4)$ are given by the arcs $(n-\alpha i-j-5, n-\alpha i-j-4)$ for $0 \leq i \leq \alpha-3, 0 \leq j \leq \alpha-3$. In this set there is again a gap of two arcs every $\alpha-2$ arcs which is filled by the arcs $(n-\alpha(i+1)-4, n-\alpha(i+1)-3), (n-\alpha(i+1)-3, n-\alpha(i+1)-2)$ for $0 \leq i \leq \alpha-4$. The remaining eight arcs $(\alpha^2-2\alpha-2, \alpha^2-2\alpha-1), \ldots, (\alpha^2-2\alpha+1, \alpha^2-2\alpha+2)$ and $(n-4, n-3), \ldots, (n-1, 0)$ are given by the arcs of the last four triangles (except for the last arc of each triangle). Assume that the cycle which consists of n arcs is exactly the cycle that our algorithm removes.

Note that in the remaining arc set S each arc has length at least 4. We next show that in the optimal solution for the remaining arcs each arc consists of its own chain. To see this it is enough to show that if there is a pair of arcs in S with a common end-vertex v, then their total length is at least $n+1$ (this claim also shows that the original instance does not contain two arcs cycles). First, note that if one of the arcs incident at v occurs in one of the triangles of OPT, then its length is exactly $n-2$, the other arc has length at least 4, and therefore their total length is greater than $n+1$. Therefore, we can assume that the pair of arcs incident at v are from the four arcs cycles of OPT. Let $0 \leq i, j \leq \alpha-3$:

- Assume that $v = \alpha j + i$. Then, the arcs incident at v are $(n-\alpha i-j-4, v)$ and $(v, n-\alpha(i-1)-j-5)$, and their total length is $n+\alpha-1 > n$ for all values of $\alpha \geq 2$.
- Assume that $v = n-\alpha i-j-5$. Then, the arcs incident at v are $(\alpha j+i+1, v)$ and $(v, \alpha(j+1)+i)$, and their total length is $n+\alpha-1 > n$ for all values of $\alpha \geq 2$.

Therefore, our optimal solution for S is a chain for each arc. Since $|S| = 2(\alpha-2)^2 + 1[2(\alpha-2)+4]$ (S contains two arcs from each cycle of four arcs in OPT, and one arc from each triangle of OPT), we conclude that the cost of the approximation algorithm is $|E|+|S| = 6(\alpha-2)^2+4[2(\alpha-2)+4] = 6\alpha^2-16\alpha+24$. Therefore, the approximation ratio of the algorithm approaches $3/2$ as α goes to infinity (also n grows to infinity).

3 A Better Analysis of the Algorithm PIM

In this section we assume that the Preprocessing phase of Algorithm PIM first removes cycles with two arcs, and only if such cycles do not exist, other cycles are removed.

The proof of the next theorem is similar to the proof of Lemma 19 in [1].

Theorem 2. *Algorithm PIM returns a solution whose cost is at most* $1 \cdot 2CY_2 + \frac{4}{3} \cdot 3CY_3 + \frac{7}{5}5CY_5 + 1 \cdot (2CH_1 + 3CH_2) + \frac{5}{4}(4CH_3 + 5CH_4) + \frac{3}{2}4CY_4 + \frac{3}{2}\left(\sum_{i=6}^{n} iCY_i + \sum_{i=5}^{n-1}(i+1)CH_i\right)$.

Proof. To prove the claim we assign the cost of the solution obtained by Algorithm PIM to the arcs, such that the following properties hold:

1. The total cost assigned to the arcs that belong to a cycle in OPT of two arcs is exactly the cost paid by OPT to this cycle, i.e. 2.
2. The total cost assigned to the arcs that belong to a cycle in OPT of three (resp. five) arcs is at most $\frac{4}{3}$ (resp. $\frac{7}{5}$) times the cost paid by OPT to this cycle, i.e. 4 (resp. 7). The total cost assigned to the arcs that belong to a cycle in OPT of four arcs or at least six arcs is at most $\frac{3}{2}$ times the cost paid by OPT to this cycle.
3. The total cost assigned to the arcs that belong to a chain in OPT of at most two arcs is exactly the cost paid by OPT to this chain. The total cost assigned to the arcs that belong to a chain in OPT of three or four arcs is at most $\frac{5}{4}$ times the cost paid by OPT to this chain. The total cost assigned to the arcs that belong to a chain in OPT of at least five arcs is at most $\frac{3}{2}$ times the cost paid by OPT to this chain.

To prove property 1, note that the preprocessing phase take out exactly all cycles of OPT of exactly two arcs, and therefore their cost in the solution obtained by PIM is exactly their cost in OPT.

We prove the other properties by considering not the solution obtained by PIM, but an alternative solution that is no better than PIM in terms of cost. We replace the solution of PIM with the solution obtained by PIM after the first iteration of the matching phase. This is clearly an upper bound on the cost of PIM. We further replace the solution by a solution that does not create an optimal matching, but some feasible matching that we construct. The matching is constructed by uniting matchings that may be created from the remaining arcs (after cycle removal by PIM) from each component of OPT (cycle or chain) separately. Note that a remaining path of s arcs contributes a matching of size $\lfloor s/2 \rfloor$.

We now prove property 2. Each cycle of OPT loses at least one arc in the preprocessing phase. Consider a cycle of OPT which contains k arcs ($k \geq 3$). Let $\ell \geq 1$ be the number of arcs that the cycle loses in the preprocessing phase. There is a matching of the arcs from this cycle of size $\lfloor \frac{k-1}{2} \rfloor - (l-1)$ (after the first arc from this cycle is removed, we have a matching of size $\lfloor \frac{k-1}{2} \rfloor$. Every other arc that is removed destroys at most one arc in the matching). The cost of the cycle depends on the number of chains created from it. This number is at most $\lceil \frac{k-1}{2} \rceil \leq \frac{k}{2}$. I.e., for three arcs cycles we have at most one chain, and five arcs cycles we have at most two chains. This gives the costs 4 and 7 for cycles of three and five arcs (respectively). For a cycle of k arcs we get at most $k/2$ chains, which proves the case of four arc cycles, and cycles of at least six arcs.

Next, we prove property 3. For a chain in OPT of k arcs such that l arcs are taken out in the preprocessing phase, there is a matching of the arcs from this chain of size $\lfloor \frac{k}{2} \rfloor - l$ (before the preprocessing phase we have a matching of size $\lfloor \frac{k}{2} \rfloor$, and every arc that is removed, destroys at most one arc in the matching). Therefore, the number of chains created after the first matching step equals the number of chains OPT has if the number of arcs in the chain is one or two. If the number of arcs in the chain is three or four, then the number of chains after the first matching phase is at most two (whereas OPT has one chain). Therefore, in this case we have a ratio of at most $\frac{5}{4}$ between the cost OPT pays for this

chain and the cost PIM pays for this chain. For longer chains, we get at most $\lceil \frac{k}{2} \rceil - 1$ new chains (compared to OPT), and so PIM pays at most $\frac{3}{2}$ times the cost OPT pays.

Taking the optimal matching instead of the matching we describe above, only decreases the cost of the solution, and therefore the claim holds also for the solution obtained by PIM.

4 Algorithm GPTS

In this section we study a different approximation algorithm for the problem. Given a set of input arcs we apply a certain greedy clean-up preprocessing phase that is composed of five steps. First, we remove all cycles of two arcs (the number of the cycles that we remove in this step is exactly CY_2). Then, we remove greedily certain subgraphs (cycles or chains with a certain number of arcs and certain length), by removing each time a single such subgraph as long as the remaining graph contains a subgraph with the desired property. E.g., in Step 2 we remove cycles of three arcs each one at a time until the remaining arc-set does not contain a cycle with exactly three arcs.

Algorithm Greedy-Preprocessing Trail-Split (GPTS):

1. Remove all cycles of two arcs.
2. Remove greedily cycles of three arcs until there are no such cycles.
3. Remove greedily mega-chains of at most three arcs with length in the interval $[\frac{3n}{4}, n-1]$ until there are no such mega-chains.
4. Remove greedily cycles of four arcs until there are no such cycles.
5. Remove greedily cycles of five arcs until there are no such cycles.
6. Cover the rest of the arcs with chains in the following way:
   (a) Find a set of mega-chains (with arbitrary lengths) that connect the odd-vertices and remove them. For each such mega-chain of length greater than n, decompose it into chains of length at most n.
   (b) Partition the rest of the arcs (these are from the Eulerian subgraph) into chains (or cycles) of length at most n.

Observation 3. *Consider a mega-chain with length in the interval* $[kn, (k+1)n-1]$ *that is created in step 6a. Then, the number of chains that result from it is at most* $2k+1$.

Proof. We perform a greedy partition that chooses a maximum length chain at each step. Therefore, the total length of each pair of consecutive chains in a mega-chain is at least n (otherwise, they can be united).

Observation 4. *The chains obtained in step 6b have an average length of at least* $\frac{n}{2}$.

Proof. The claim follows because the total length of each pair of consecutive chains resulting from the Eulerian subgraph is at least n (otherwise, they can be united).

Notations: consider OPT. Partition it into mega-chains and an Eulerian subgraph. There may be several options to do that, therefore we fix an arbitrary partition. Denote by:

- CY - the number of cycles in OPT that contain at least six arcs.
- CH_E - the number of chains in OPT that are part of the Eulerian subgraph defined with respect to OPT.
- MC - the total number of mega-chains.
- CH_M - the number of chains in OPT that are part of mega-chains defined with respect to OPT.
- MC_3 - the number of mega-chains of at most three arcs with total length in the interval $[\frac{3n}{4}, n-1]$. Note that such a mega-chain is a chain of OPT.
- MC_s - the number of mega-chains with total length less than $\frac{3n}{4}$.
- MC_4 - the number of mega-chains with exactly four arcs and total length in the interval $[\frac{3n}{4}, n-1]$.
- MC_5 - the number of mega-chains with at least five arcs and total length in the interval $[\frac{3n}{4}, n-1]$.
- MC^i - the number of mega-chains with total length in the interval $[in, (i+1)n-1]$. MC^i is defined for all $i \geq 1$.
- CH_E^1 and CH_E^2 - the number of chains in OPT that are part of the Eulerian subgraph, of exactly one arc, and exactly two arcs, respectively.
- CH_M^1 and CH_M^2 - the number of chains in OPT that are part of mega-chains, of exactly one arc, and exactly two arcs, respectively.

Note that a mega-chain in OPT with total length in the interval $[in, (i+1)n-1]$ consists of at least $i+1$ chains (in OPT).

The total length of all arcs is at most $UB_L = (CY_2 + CY_3 + CY_4 + CY_5 + CY)n + CH_E n + MC_3 n + MC_s \frac{3n}{4} + MC_4 n + MC_5 n + \sum_{i=1}^{\infty}[MC^i(i+1)n]$.

Consider Algorithm GPTS, and denote by A the number of cycles removed in step 2, B- the number of chains removed in step 3, C- the number of cycles removed in step 4. D - the number of cycles removed in Step 5. Then, the following inequalities hold: $3A \geq CY_3$, $3A+3B \geq CY_3 + MC_3$, $3A+3B+4C \geq CY_3 + MC_3 + CY_4$, and $3A + 3B + 4C + 5D \geq CY_3 + MC_3 + CY_4 + CY_5$. Subject to these constraints we want to minimize $nA + \frac{3nB}{4} + nC + nD$ (this is a lower bound on the total length that we gain by removing these subgraphs). An optimal solution of this (parametric) mathematical program is $A = \frac{CY_3}{3}$, $B = \frac{MC_3}{3}$, $C = \frac{CY_4}{4}$, and $D = \frac{CY_5}{5}$.

This proves that each feasible solution to the mathematical program has a cost of at least $\frac{n}{3}CY_3 + \frac{n}{4}MC_3 + \frac{n}{4}CY_4 + \frac{n}{5}CY_5$, and this is the cost of the (feasible) solution that we outlined above, and therefore it is optimal.

Therefore, the total length of the arcs that are left in the beginning of step 6 is at most $UB'_L = UB_L - \frac{CY_3}{3}n - \frac{MC_3}{3} \cdot \frac{3n}{4} - \frac{CY_4}{4}n - \frac{CY_5}{5}n$.

By Observations 3 and 4, the total number of chains obtained by our algorithm is at most $\frac{UB'_L}{n/2} + MC$ (note that this include also the chains from step 3).

In the remaining of this section we use the fact that Step 1 of the algorithm correctly identifies all the cycles of OPT that have two arcs. Therefore, we can assume that $CY_2 = 0$.

We use the fact that $MC = MC_3 + MC_s + MC_4 + MC_5 + \sum_{i=1}^{\infty} MC^i$ to obtain that the number of chains resulted by our algorithm is at most: $APX_{CH} = \frac{4}{3}CY_3 + \frac{3}{2}CY_4 + \frac{8}{5}CY_5 + 2CY + 2CH_E + \frac{5}{2}MC_3 + \frac{5}{2}MC_s + 3MC_4 + 3MC_5 + \sum_{i=1}^{\infty} MC^i[2(i+1)+1]$. We denote $OMC = MC_3 + MC_s + \sum_{i=1}^{\infty}[MC^i(i+1)]$. Since $2(i+1)+1 \leq \frac{5}{2}(i+1)$ for all $i \geq 1$, we reduce the term $\sum_{i=1}^{\infty} MC^i[2(i+1)+1]$ into $\frac{5}{2}\sum_{i=1}^{\infty}[MC^i(i+1)]$.

Therefore, the total cost of the solution obtained by our algorithm is $apx = |E| + APX_{CH} \leq |E| + \frac{4}{3}CY_3 + \frac{3}{2}CY_4 + \frac{8}{5}CY_5 + 2CY + 2CH_E + 3MC_4 + 3MC_5 + \frac{5}{2}OMC$, and the cost of OPT is $opt = |E| + CH_E + MC_4 + MC_5 + OMC$.

We consider the partition of E according to the roles of arcs in OPT to the following (disjoint) subsets: $E(CY_3)$: arcs that belong to cycles of three arcs in OPT. $E(CY_4)$: arcs that belong to cycles of four arcs in OPT. $E(CY_5)$: arcs that belong to cycles of five arcs in OPT. $E(CY)$: all the other arcs that belong to cycles in OPT. $E(CH_E)$: arcs that belong to the Eulerian subgraph but not to $E(CY_3) \cup E(CY_4) \cup E(CY_5) \cup E(CY)$. $E(MC_4)$: arcs that belong to mega-chains of exactly four arcs with total length in the interval $[\frac{3n}{4}, n-1]$. $E(MC_5)$: arcs that belong to mega-chains of at least five arcs with total length in the interval $[\frac{3n}{4}, n-1]$. $E(OMC)$: the rest of the arcs that belong to mega-chains.

The following equations and inequalities hold using the numbers of arcs in the subgraphs of OPT.

$$|E(CY_3)| + \frac{4}{3}CY_3 = \frac{13}{9}|E(CY_3)| \tag{1}$$

$$|E(CY_4)| + \frac{3}{2}CY_4 = \frac{11}{8}|E(CY_4)| \tag{2}$$

$$|E(CY_5)| + \frac{8}{5}CY_5 = \frac{33}{25}|E(CY_5)| \tag{3}$$

$$|E(CY)| + 2CY \leq |E(CY)| + \frac{2}{6}|E(CY)| = \frac{4}{3}|E(CY)| \tag{4}$$

$$|E(CH_E)| \geq 3CH_E - 2CH_E^1 - CH_E^2 \tag{5}$$

$$|E(MC_4)| + 3MC_4 \leq \frac{7}{5}(|E(MC_4)| + MC_4) \tag{6}$$

$$|E(MC_5)| + 3MC_5 \leq \frac{4}{3}(|E(MC_5)| + MC_5) \tag{7}$$

$$|E(OMC)| \geq 3OMC - 2CH_M^1 - CH_M^2 \tag{8}$$

Using the inequalities (after re-considering the cycles of two arcs), it is possible to prove the following theorem. We omit the proof due to space constraints.

Theorem 5. *Algorithm GPTS returns a feasible solution whose cost is at most*

$$2CY_2 + \frac{13}{9}|E(CY_3)| + \frac{7}{5}\Big(|E(CY_5)| + |E(MC_4)| + MC_4\Big) + \frac{7}{4}\Big(2CH_E^1 + 3CH_E^2 + 2CH_M^1 + 3CH_M^2\Big) + \frac{11}{8}\Big(opt - 2CY_2 - |E(CY_3)| - |E(CY_5)| - |E(MC_4)| - MC_4 - 2CH_E^1 - 3CH_E^2 - 2CH_M^1 - 3CH_M^2\Big).$$

5 An $\frac{10}{7}$-Approximation Algorithm: Algorithm COMB

In this section we design a new approximation algorithm COMB. Algorithm COMB combines the two algorithms: PIM and GPTS. It simply applies both PIM and GPTS, and picks the better solution.

It is possible to adapt the bound on PIM as follows. $PIM \leq 2CY_2 + \frac{4}{3}|E(CY_3)| + \frac{7}{5}|E(CY_5)| + (2CH_1 + 3CH_2) + \frac{5}{4}(MC_4 + E(MC_4)) + \frac{3}{2}\Big(opt - 2CY - |E(CY_3)| - |E(CY_5)| - 2CH_1 - 3CH_2 - MC_4 - |E(MC_4)|\Big)$.

We omit the proof of the following theorem.

Theorem 6. *The approximation ratio of Algorithm COMB is at most* $\frac{10}{7}$*, and at least* $\frac{4}{3}$*.*

The upper bound is proved by considering a convex combination of the cost of the two algorithms, and showing an upper bound of $10opt/7$ on it. The lower bound is shown by reconsidering example 15 from [1].

6 Other Algorithms

In this section we consider several previously known algorithms, and give tight or tighter bounds on their performance. Due to space restrictions, the analysis of algorithms PCM, PET-TS and IM is omitted.

6.1 MCC-TS

In [1] algorithm MCC-TS has a preprocessing step of two arcs cycles removal. However, the algorithm can be easily adapted to work without this step, and the analysis still works. While building the auxiliary graph the option of two arcs that form a cycle should be taken into account, and the arcs between those arcs both get weight one. It was shown [1] that the performance ratio for this algorithm is in the interval $[1.5, 1.6]$. We show that the upper bound is tight. To distinguish between the two versions we call them MCC-TS (the version with pre-processing) and NMCC-TS (without pre-processing).

The proof of the following theorem is omitted.

Theorem 7. *Algorithm NMCC-TS has approximation ratio of exactly* 1.6*.*

For algorithm MCC-TS (with two arc cycles removal), we can show a tight bound of 14/9. We prove it using the next two lemmas.

Lemma 1. *Algorithm MCC-TS has approximation ratio of at least* 14/9*.*

Proof. Let $n = 24m^4$ for an integer $m > 1$. The input arcs are described in Table 1. The input consists of five families of arcs. Each family has certain amount of parallel copies of arcs (this amount appears in the column Amount). The arc set of each family is parameterized by i or by i, s. For each value of the parameters in the Index range (that appears in the second column) we have the amount of parallel copies of the arcs that appear in the Arcs column.

Table 1. Input arcs

Amount	Index range	Arcs
$12m^3$	$0 \le i < n$	$(i, i+n/2), (i+n/2, i-2m^2), (i-2m^2, i)$
$24m^2$	$0 \le i < n, 1 \le s \le m$	$(i, i+n/3-sm^2)$
$12m^2$	$0 \le i < n, 1 \le s \le m$	$(i, i+n/3+2(s+1)m^2)$
$12m^3$	$0 \le i < n$	$(i, i+2), (i, i+3)$
$6m^3$	$0 \le i < n$	$(i, i-4), (i, i-6)$

We give an upper bound on *opt* by the cost of the following solution. The solution has for every $0 \le i < n$, $12m^3$ cycles which are $(i, i+n/2), (i+n/2, i-2m^2), (i-2m^2, i)$. For $0 \le i < n$ we have $6m^3$ cycles of $(i, i+2), (i+2, i+4), (i+4, i)$ and $6m^3$ of $(i, i+3), (i+3, i+6), (i+6, i)$ for $0 \le i < n$. Finally, for every $0 \le i < n$ and for every $2 \le s \le m$ there are $12m^2$ identical cycles: $(i, i+n/3-sm^2), (i+n/3-sm^2, i+2n/3-2sm^2), (i+2n/3-2sm^2, i)$. The arcs $(i, i+n/3-m^2)$ and $(i, i+n/3+2(m+1)m^2)$, are not combined into cycles but into paths, $12m^2$ copies of $(i, i+n/3-m^2), (i+n/3-m^2, i+2n/3-2m^2)$ for every $0 \le i < n$ and $6m^2$ of $(i, i+n/3+2(m+1)m^2), (i+n/3+2(m+1)m^2, i+2n/3+4(m+1)m^2)$ for every $0 \le i < n$. Since $m > 1$, $4(m+1)m^2 < n/3$.

The MCC solution may consist of the following cycles (it manages to combine all arcs into long cycles). Note that each pair of consecutive arcs in each cycle is indeed valid for MCC as their combined length is less than n. This will hold due to the choice of $n = 24m^4$ which gives $(m+2)m^2 < n/3$ and $2(m+1)m^2 < n/6$. We have $12m^2$ copies of the following cycle $(i, i+n/2), (i+n/2, i+5n/6-sm^2), (i+5n/6-sm^2, i+n/3-(s+2)m^2), (i+n/3-(s+2)m^2, i+2n/3-(2s+2)m^2), (i+2n/3-(2s+2)m^2, i)$, for every $0 \le i < n$ and for every $1 \le s \le m$. These cycles can be decomposed into three chains, no matter which arc is chosen to be first. We have the following cycle $6m^3$ times for every $0 \le i \le 4m^2-1$. The number of arcs in a cycle is $48m^2$, and no vertex is repeated until the cycle is closed. The cycle consists of $6m^2$ phases of eight arcs. For $0 \le q \le 6m^2-1$, we have the eight arcs $(i+4qm^2, i+2+4qm^2), (i+2+4qm^2, i+2+(4q+2)m^2), (i+2+(4q+2)m^2, i+4+(4q+2)m^2), (i+4+(4q+2)m^2, i+(4q+2)m^2), (i+(4q+2)m^2, i+3+(4q+2)m^2), (i+3+(4q+2)m^2, i+3+(4q+4)m^2), (i+3+(4q+4)m^2, i+6+(4q+4)m^2), (i+6+(4q+4)m^2, i+(4q+4)m^2)$. Since $m > 1$ is an integer, $2m^2 \ge 8$, and so vertices with different residues (indices $\mod 2m^2$) cannot coincide. Vertices with the same residue are distinct due to the different coefficients of m^2. The decomposition of each cycle creates $24m^2$ chains. We get that $opt \le n(36m^3+18m^3+18m^3+36m^2(m-1)+54m^2) = n(108m^3+18m^2)$. $APX = (12nm^3) \cdot 8 + 48m^2 \cdot 6m^3 \cdot 4m^2 \cdot 1.5 = nm^3(96+72) = 168nm^3$. This gives a ratio of $168m/(108m+18)$ which tends to $14/9$ for large m.

Lemma 2. *Algorithm MCC-TS has approximation ratio of at most* $14/9$.

Proof. For every arc e, define a weight $w(e)$ in the following way. $w(e) = 1/3 + 2\ell(e)/(3n)$. We show the following properties.

1. The total sum of weights of arcs is at most (5/9)opt.
2. The number of new chains caused by decomposition is at most the total sum of weights.

The total cost for original chains and valid cycles constructed by MCC is bounded by *opt*, so the result of proving the properties would be $APX \leq 14opt/9$.

We start with proving property 1. Consider a cycle C in OPT which consists of k arcs. The total cost paid by OPT for C is k. The total weight of the arcs of C is exactly $k/3 + 2/3$, and $k/3 + 2/3 \leq 5k/9$ for $k \geq 3$. Consider a chain created by OPT which consists of k arcs. The total cost paid by OPT for this chain is $k + 1$. The total weight of the arcs of this chain is at most $k/3 + 2/3$, and $(k/3 + 2/3) \leq 5(k + 1)/9$ for $k \geq 1$. Next, we prove property 2. Consider a (not necessarily valid) cycle of $2k + 1$ arcs constructed by MCC which is of length sn for some integer s. Every such cycle can be split into at most $2s - 1$ chains in the following way. Let i be the starting vertex of an arc of the cycle, then i would be the first end-point of the first chain and the last end-point of the last chain (it can be the case where those two chains are combined into one). The distance to go from the first end-point to the last is sn. The length of two consecutive chains along the cycle is at least $n + 1$ (otherwise, they can be merged). If there are $2s$ chains, this means that the distance between the first and the last is more than sn, and therefore there are at most $2s - 1$ chains. On the other hand any pair of successive arcs can be combined in a chain due to the construction of the MCC graph, so $k + 1$ chains are always possible. We get that the number of new chains is at most $\min(k + 1, 2s - 1)$. The weight for these $2k + 1$ arcs is $(2k + 1)/3 + 2s/3 = (2k + 2)/3 + (2s - 1)/3 \geq (2/3 + 1/3)\min(k + 1, 2s - 1)$. Therefore, the weight of the cycle is at least the amount of additional cost caused by the decomposition.

Consider a cycle of $2k$ arcs which is of length sn for an integer s. We can get that the number of new chains is at most $\min(k, 2s - 1)$. The weight for these $2k$ arcs is $2k/3 + 2s/3 > 2k/3 + (2s - 1)/3 \geq (2/3 + 1/3)\min(k, 2s - 1)$.

Consider a chain of $2k$ or $2k+1$ arcs with length in the interval $[sn, (s+1)n)$. Note that the original connected component built by MCC is already a chain and not a cycle. There are at most $\min(k, 2s)$ new chains. The weight of the chain is at least $2k/3 + 2s/3 \geq \min(k, 2s)$.

Summarizing we proved the following theorem.

Theorem 8. *Algorithm MCC-TS has approximation ratio of exactly* 14/9.

7 Conclusion

We introduced an approximation algorithm COMB for the problem of minimizing the number of SONET ADMs. COMB is a combination of two algorithms, one of them was introduced in this paper and the other was previously studied. Algorithm COMB is the current best approximation algorithm for this problem.

Table 2. Summary of results

Heuristic	Lower bound on the approximation ratio in [1]	Lower bound on the approximation ratio (this paper)	Upper bound on the approximation ratio (this paper)	Upper bound on the approximation ratio in [1]
COMB	–	4/3	**10/7**	–
PIM	4/3	3/2	–	3/2
PCM	3/2	5/3	–	5/3
MCC-TS	3/2	14/9	14/9	8/5
NMCC-TS	3/2	8/5	–	8/5
PET-TS	3/2	5/3	–	7/4
IM	3/2	8/5	$< 5/3$	5/3
PPIM	3/2	14/9	$< 5/3$	5/3

We showed that it is a 10/7 approximation algorithm, and we provided a lower bound on its worst-case performance of 4/3. Closing this gap, and finding a better approximation algorithm is left for future research. We also raise the following question: Is there a good approximation algorithm whose preprocessing step consists of cycle removal solely (without removal of chains)?
A summary of the results in the paper can be found in Table 2.

References

1. G. Calinescu and P.-J. Wan, "Traffic partition in WDM/SONET rings to minimize SONET ADMs," *Journal of Combinatorial Optimization*, **6**, 425-453, 2002.
2. O. Gerstel, P. Lin and G. Sasaki, "Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings," *Proc. INFOCOM 1998*, **1**, 94-101, 1998.
3. L. Liu, X. Li, P.-J. Wan and O. Frieder, "Wavelength assignment in WDM rings to minimize SONET ADMs," *Proc. INFOCOM 2000*, **2**, 1020-1025, 2000.
4. P.-J. Wan, G. Calinescu, L. Liu and O. Frieder, "Grooming of arbitrary traffic in SONET/WDM BLSRs," *IEEE Journal on Selected Areas in Communications*, **18**, 1995-2003, 2000.