Wavelength Assignment in WDM Rings with Splitable Lightpaths

Gruia Călinescu *
Peng-Jun Wan
Dept. of Computer Science, Illinois Institute of Technology, Chicago, IL 60616
E-mail:{calinesc,wan}@cs.iit.edu

Abstract

This paper presents a new practical approximation algorithm for wavelength assignment to splitable lightpaths over WDM rings, with the objective of minimizing the number of SONET ADMs. Allowing the splitting of traffic streams can significantly reduce the number of required ADMs.

Moreover, while finding the best assignment is proved to be NP-Hard, the problem seems easier to approximate than the variation when traffic streams cannot be split. In the worst case, the output of the new polynomial-time algorithm is at most 25% more than the optimum solution. This result is significantly better than the best known approximation ratio for non-splitable traffic streams.

Keywords: *wavelength division multiplexing (WDM), optical networks, SONET, add-drop multiplexer (ADM), WADM, grooming, approximation algorithm.*

1 Introduction

WDM self-healing ring (SHR) networks are being deployed by a growing number of telecom carriers to support multiple high-level SONET/SDH rings [7] over a single physical fiber optical ring. One of the most fundamental network design problems for WDM networks is the assignment of wavelengths to a given set of traffic streams. While most of the previous works attempt to minimize the number of wavelengths required or the amount of blocking for the given set of traffic streams [1, 5, 9, 10, 3], it was argued in [4, 6] that unless the wavelength limit is exceeded, the first-order optimization goal should be to minimize the overall network cost which is dominated by the number of required SONET add/drop multiplexers (ADMs) instead of the number of wavelengths. In a WDM SHR, each (logical) SONET ring requires a SONET ADM at each node inside it and no ADM at any other node outside it due to the optical bypass capability of optical add-drop multiplexers. Thus the SONET ADM cost of a SONET ring is equal to the size of this SONET ring, i.e., the number of nodes it contains. The total SONET ADM cost is then the sum of the costs of all SONET rings. Alternatively, the SONET ADM cost of a node is equal to the number of SONET rings that contain this node, and the total ADM cost is the sum of the costs of all nodes. It was shown in [6] that minimizing the number of SONET ADMs is intrinsically different from the minimizing the number of wavelengths, and there exist cases where the two minima cannot be simultaneously achieved.

Recently, many works [4, 8] studied the wavelength assignment to lightpaths over WDM rings to minimize the SONET ADMs. In each fiber ring, a traffic stream can be represented as a (directed) circular arc over the fiber ring, and an instance of ADM-minimization problem is a set A of circular arcs. Depending on the implementation, the circular arcs are or are not be allowed to be split [6]. If splitting is not allowed, a valid wavelength assignment corresponds to a partition of A into groups of non-overlapping circular arcs (a set of circular arcs are said to be *non-overlapping* if the interiors of any pair of arcs have empty intersection). If splitting is allowed, then a valid wavelength assignment consists of a choice of splitting each arc of A, thus obtaining A', and then a partition of A' into groups of non-overlapping circular arcs. In either case, each group of non-overlapping circular arcs can be carried in a wavelength and thus form a logical SONET ring. Finding an optimal solution without splits was shown to be NP-Hard in [8] and a number of polynomial-time approximations algorithms were proposed in both [4] and [8]. The best known worst-case approximation ratio for non-splitable arcs is 1.5 ([2]), or in other words the output of the algorithm presented by [2] is proved to have cost at most 50% more than the optimum solution.

Following the same argument as in [8], this paper shows that finding an optimal solution with splits is also NP-Hard. In [4] it was first argued that splitting lightpaths has the potential to significantly reduce the number of ADMs. Splitting is achieved by placing two ADMs and electrically transferring the data between two wavelengths. A simple example presented in this paper shows that splitting can reduce

*Work performed in part at Georgia Institute of Technology and supported in part by NSF grant CCR-9732746

the number of ADMs by 33%. Wavelength assignment with splitting also tends to use a lower number of wavelengths. Based on the results of this paper, we also believe it is easier to approximate the optimum placement when lightpaths are splitable. The higher port and cross-connect costs associated with splitting will probably be much smaller than the savings achieved by allowing splitting lightpaths.

The two heuristics proposed in [4] (Cut-First and Assign-First) have worst-case examples when their output is 50% bigger than the optimum solution. Three transformations (Merging, Combining and Splitting) proposed by [4] can be used after any algorithm to attempt to further reduce the number of ADMs. These transformations do improve the performance of the Cut-First and Assign-First on the previous worst-case examples. However, it is probably impossible to analyze precisely the effect of the three transformations.

This paper introduces a new polynomial-time approximation algorithm for wavelength assignment to splitable lightpaths over WDM rings, with the objective of minimizing the number of SONET ADMs. The algorithm combines greedy ideas with Eulerian rounding, a technique similar to the Cut-First heuristic of [4]. The worst-case approximation ratio of the algorithm is proved to be at most 1.25. The three transformations of [4] can also be used after our new algorithm, providing practical improvements. An example shows that the new algorithm could produce an output using 11% more ADMs than the optimum solution. The three transformations of [4] do not help in this example.

This paper is organized as follows: in Section 2, we present the formal definition of the problem of assigning wavelengths to splitable lightpaths over WDM rings, with the objective of minimizing the number of SONET ADMs. The potential for improvement over non-splitable lightpaths is quantified. The proof that the problem is NP-Hard is sketched. In Section 3 we present the new approximation algorithm and prove that its approximation ratio is in between $10/9$ and 1.25. We also propose some practical improvements to the algorithm. We conclude with Section 4.

2 Preliminaries

Assume that the WDM self-healing ring consists of n nodes numbered clockwise: $0, 1, \ldots, n-1$. All arithmetic involving nodes is performed implicitly using modulo n operations. The link from the node i to node $i+1$ in the ring is referred to as link i. Each lightpath corresponds to a circular arc (or simply *arc*). A circular arc a is represented by $(o(a), t(a))$, where $o(a)$ is the origin of a and $t(a)$ is the termination of a.

Two arcs *overlap* if their interiors have a nonempty intersection. A sequence of circular arcs is called a *chain* if the termination of each circular arc, except the last one, is the origin of the subsequent circular arc. If no two arcs in a chain overlap, the chain is called *valid*. If the termination of the last circular arc is the also the origin of the first circular arc, the chain is called *closed*, otherwise it is called *open*. The *length* of a chain is the number of arcs in the chain. The length of the chain C is also denoted by $|C|$. The *cost* of a valid chain is the number the nodes in the chain. For a closed chain, the length and cost coincide, while for an open chain, the cost is one more than the length.

Splitting a lightpath corresponds to the following definition for splitting arcs: the arc a is replaced by several arcs $a_1, a_2, \ldots, a_k$ such that $o(a) = o(a_1)$, $t(a) = t(a_k)$, and $t(a_i) = o(a_{i+1})$ for any $1 \leq i < k$. To make notation simple, replacing an arc by itself is also a split, called a *nsplit*. An arc which is the result of an nsplit is called *original*. A split which is not a nsplit is called a *realsplit*. An arc which results from a realsplit is called *fragment*.

Any wavelength assignment, with or without splits, generates naturally a set of disjoint valid chains. The ADM cost of this wavelength assignment is simply the sum of the costs of these valid chains.

The problem of wavelength assignment to splitable lightpaths over WDM rings, with the objective of minimizing the number of SONET ADMs can be formalized as follows: the input is a set of arcs A. Each arc of A must be split, obtaining a set of arcs B. A feasible solution consists of a choice of splitting each arc of A (thus obtaining B) and a partition of B into valid chains. The cost of the solution is the sum of the costs of these valid chains.

This problem is NP-Hard, as sketched in the following. Optimum equals $|A|$ if and only if A can be partitioned into valid closed chains. Indeed, any realsplit or any open chain will result in a solution of cost strictly bigger than $|A|$. The NP-Hardness proof of [8] implies that answering the question if such a "perfect" partition exists is NP-Hard.

As each arc of A can be nsplit (and therefore $B = A$), the optimum with splits is at most the optimum without splits. In [4] an example in which optimum with splits is 25% lower than optimum without splits is presented. In fact, optimum with splits could be 33% lower than optimum without splits, as shown by the following example: $n = 3$, $A = \{a_1, a_2, a_3\}$, with $a_1 = (0, 2)$ (here we write $a = (i, j)$ instead of the more complete $o(a) = i$ and $t(a) = j$), $a_3 = (2, 1)$, and $a_3 = (1, 0)$. The optimum without splits is 6, while with splits a solution of cost 4 can be obtained by splitting a_2 into $a_2' = (2, 0)$ and $a_2'' = (0, 1)$. Then a_1 and a_2' form one closed chain, and a_2'' and a_3 form another.

Actually, the example above is extreme, in the sense that any solution with splits can be converted, by simply putting alone in a chain any arc which is split, into a solution without splits of cost at most 50% bigger. But how the two costs of optimum relate is not as interesting as how the best solutions we can find relate.

More preliminary definitions must be introduced before presenting the algorithm. Given a set of arcs S, the *surplus* of a node i with respect to S is defined as follows:

$$sur_S(i) = |\{s \in S : t(s) = i\}| - |\{s \in S : o(s) = i\}|.$$

Note that the surplus might be negative and in fact $\sum_{i=0}^{n-1} sur_S(i) = 0$. An open chain $P = \langle a_1, a_2, \cdots, a_k \rangle$ in S such that $sur_S(o(a_1)) < 0$ and $sur_S(t(a_k)) > 0$ is called *tight with respect to S*, or simply *tight*, when S is understood as being a current set of arcs. The *deficiency* of a node i is $def_S(i) = \frac{1}{2}|sur_S(i)|$. The deficiency of a set of arcs S is $def(S) = \sum_{i=0}^{n-1} def_S(i)$. Then, as observed in [4], $|S| + def(S)$ is a lower bound on the minimum ADM cost required by S.

For the purpose of the proof, we use arcs which are not in A. Such an auxiliary arc is called a *fake*, and has an origin and a termination like any other arc. We divide the set of arcs in two: *red* arcs and *blue* arcs. An arc is red if it does contain the link $n-1$, and blue otherwise. Given a set of arcs S, the *blue number* of S, denoted by $b(S)$, is the number of blue arcs in S. Sometimes we write $b(a)$ instead of $b(\{a\})$. Note that any valid closed chain has blue number one, while a valid open chain has blue number at most one.

3 The Algorithm

We start by describing the last and most interesting phase of the algorithm, *Eulerian rounding*. Let S be a set of arcs. We consider two cases. In the first case, $def\,(S) > 0$. We first add a set of $def\,(S)$ fake arcs F such that $def(S \cup F) = 0$. This can be easily done by adding one by one fake arcs with the origin being a node of positive surplus and the termination being a node of negative surplus, thus each fake arc decreasing the deficiency by one. Now the directed graph with edges $S \cup F$ is Eulerian. Choosing any Eulerian tour and then removing all fake arcs results in $def(S)$ open chains. For every invalid (open) chain P, break it into valid chains as follows (see Figure 1): for each circular arc a in P that passes through $o\,(P)$, the origin of P, split it into two arcs

$$a' = (o\,(a)\,, o\,(P))\,, a'' = (o\,(P)\,, t\,(a))\,.$$

After these splittings, the invalid chain P is then be decomposed into valid chains by walking along P from $o\,(P)$ and output a valid chain whenever reaching $o\,(P)$.

In the second case, $def\,(S) = 0$, and thus the directed graph with edges S is Eulerian. Choose any Eulerian tour. Let i be any node which is the origin of some arc. For any circular arc a in the oriented Eulerian tour that passes through i, split it into two arcs

$$a' = (o\,(a)\,, i)\,, a'' = (i, t\,(a))\,.$$

After these splittings, the oriented Eulerian tour is then be decomposed into valid (closed) chains by walking along the oriented Eulerian tour from node i and output a valid (closed) chain whenever reaching node i.

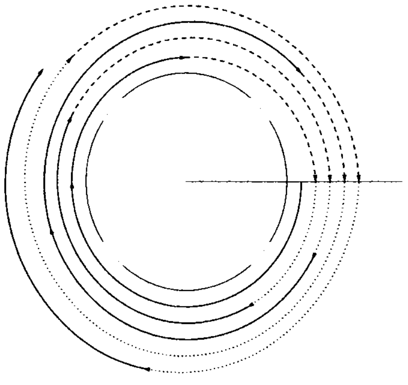

Figure 1. Eulerian rounding. Arcs are split into two fragments when they pass through node $o(P)$: the dashed fragment closes a chain, while the dotted fragment starts another chain.

Lemma 3.1 *The solution produced from Eulerian rounding for S has cost at most $|S| + b(S) + def(S)$.*

Proof. We first consider the case that $def\,(S) > 0$. Let P be any invalid chain. Note that among all valid chains generated by Eulerian rounding from P, exactly one is open and all others are closed. Since the chain P must pass through the link $n-1$ each time before passing through $o(P)$, the number of splits is at most $b(P)$. So the total cost of these valid chains generated from P is at most $|P| + b(P) + 1$. Since there are total $def\,(S)$ open chains, the total cost of all valid chains is at most $|S| + b(S) + def(S)$.

Now we consider the case that $def\,(S) = 0$. In this case, all valid chains are closed, and the number of splits is at most $b(P)$. So the total cost of these valid chains is at most $|S| + b(S)$. ■

Using the machinery developed before, we proposed the following approximation algorithm for Wavelength Assignment in WDM Rings with Splitable Lightpaths:

- The input is a set of arcs A.
- Phase 1: While A contains a valid closed chain P of length two, nsplit the arcs in P, output the valid (closed) chain consisting of the two arcs, and set $A \leftarrow A \setminus P$.

- Phase 2: While A contains a valid closed chain P of length three, nsplit the arcs in P, output the valid (closed) chain consisting of the three arcs, and set $A \leftarrow A \setminus P$.

- Phase 3: While such an arc exists, select $a \in A$ such that a is blue and such the chain $\langle a \rangle$ is tight. Nsplit a, output the valid (open) chain consisting of a, and set $A \leftarrow A \setminus \{a\}$.

- Phase 4: While such a pair of arcs exists, select $a_1, a_2 \in A$ such that $\langle a_1, a_2 \rangle$ is a tight valid chain, and one of a_1, a_2 is blue. Nsplit a_1 and a_2, output the valid (open) chain consisting of the two arcs, and set $A \leftarrow A \setminus \{a_1, a_2\}$.

- Phase 5: Do the Eulerian rounding of A.

The last phase is the most interesting one. If optimum is "perfect", (has only valid closed chains and no splits), Eulerian rounding cannot guarantee a good ratio. This is the reason why we use the first two phases, which achieve good results when there are many short valid closed chains. The third and fourth phase produce open chains, but decrease the deficiency. Since in any solution, the number of open chains is at least the deficiency, it makes sense to reduce deficiency. We also insist on reducing the number of blue arcs while reducing deficiency in phases three and four, to obtain a good Eulerian rounding in the fifth phase according to Lemma 3.1.

Before the proof that the algorithm has approximation guarantee 1.25, we introduce some notation. For a set of arcs A, we use *opt*(A) to denote the value of the optimum solution to Wavelength Assignment in WDM Rings with Splitable Lightpaths which has A as the input. If A is understood, we use only *opt*.

The first phase is very intuitive. Indeed, let B be the set of arcs nsplit and then assigned to valid chains during the first phase. Then there is always an optimum solution which assigns to valid chains the arcs of B exactly the way our algorithm does. This fact follows from the following lemma:

Lemma 3.2 *Let A be a set of arcs and a_1, a_2 be two arcs of A which form a valid closed chain. Then opt$(A) =$ opt$(A \setminus \{a_1, a_2\}) + 2$.*

Proof. We omit the proof for this extended abstract. ■

Based on the lemma above, in the following we assume that no two arcs of the input form a valid closed chain.

Theorem 3.3 *The algorithm above has performance ratio at most 1.25.*

Proof. First we present a general overview of the proof. We fix OPT, an optimum solution, and based on OPT we give credit to arcs of A. Recall that A is the input. Also, each node i gets $def_A(i)$ credit. The exact initial credit allocation scheme is described later.

Before the algorithm starts, as shown in Lemma 3.4, the total credit is at most $1.25opt$. During the execution of phases two, three and four of the algorithm, as the algorithm selects arcs, puts them in valid chains and outputs the chains, (during these phases we only nsplit), we take the credit from these arcs, and in phases three and four, also from the nodes whose deficiency decreases. We use this credit taken to cover the cost of the valid chains the algorithm produces and to give some extra credit to some other arcs, according to a credit transfer scheme to be described later.

Finally, by the time we get to the fifth phase, the total credit remaining covers the cost of the output of Eulerian rounding.

We start the proof somehow backwards, discussing this last phase. This would give the motivation for all the previous steps. Let S be the set of remaining arcs after phase four. Assume each node i has $def_S(i)$ and each arc $a \in S$ has $1 + b(a)$ credit. In total, we have $def(S) + |S| + b(S)$ credit and therefore, by Lemma 3.1, we can cover the cost of the Eulerian rounding. All the remaining discussion is about how to make sure that at the end of phase four, each remaining arc a has $1 + b(a)$ credit and each node i has $def_S(i)$ credit. We now describe the initial credit allocation. For a set of arcs S, we define OPT_S to be the restriction of OPT to the set of arcs S. Each node i gets $def_A(i)$ credit. Every red arc receives 1 credit. Blue arcs receive credit according to their position in OPT, as given by the following scheme.

- Initialize S to be the initial set of arcs, A.

- Phase 1: As long as possible, do the following: if in OPT_S there is a (valid) tight (with respect to S), open chain $P = \langle a \rangle$, where a is blue and nsplit, then a receives 1.5 credit. Remove a from S.

- Phase 2: As long as possible, do the following: if in OPT_S there is a valid tight (with respect to S), open chain $P = \langle a_1, a_2 \rangle$, where either a_1 or a_2 is blue, and both are nsplit, the blue arc receives 1.75 credit. Remove from S the arcs of P.

- Phase 3: As long as possible, do the following: if in OPT_S there is a (valid) closed chain $P = \langle a_1, a_2, a_3 \rangle$, where a_1, a_2 and a_3 are nsplit, the blue arc in P receives 1.75 credit. Remove from S the arcs of P.

- Every remaining blue arc receives 2 credit.

Lemma 3.4 *The total initial credit is at most $1.25opt$.*

Proof. The proof is based on considering many cases. Due to space limitations, we only describe two cases. We consider the chains of OPT one by one. We start with the chains used in the first three phases of the credit allocation scheme above, in the order they are used. Then we continue with the remaining chains of optimum, in some order to be described later.

Take an open chain of OPT as described in the first phase of the scheme. For the arc a, OPT incurs a cost of 2. We give 1.5 credit to a, 0.5 credit to $o(a)$, and 0.5 credit to $t(a)$. Altogether, we give 1.25 times the cost of this chain in OPT. Note that after removing each such arc a from S, both $defs_S(t(a))$ and $defs_S(o(a))$ drop by 0.5.

Take an open chain of OPT as described in the second phase of the scheme. For the chain $P = \langle a_1, a_2\rangle$, OPT incurs a cost of 3. We give 1 credit to the red arc, 1.75 credit to the blue arc, 0.5 credit to $t(a_2)$ and 0.5 credit to $o(a_1)$. Altogether, we give 3.75 credit, which is 1.25 times the cost of this chain in OPT. Note that after removing the arcs from P from S, both $defs_S(t(a_2))$ and $defs_S(o(a_1))$ drop by 0.5.

The remaining cases appear in the journal version of this paper. ■.

We continue with the proof of Theorem 3.3. But first some definitions. If a blue arc has 2 credit, it is *happy*, otherwise it is *unhappy*. As our credit transfer scheme only takes credit from an arc when the arc is put (in phases 2, 3, and 4 of the algorithm) in a valid chain (which is then output), happy arcs stay happy as long as they are in the current set of arcs. By the initial credit allocation scheme, we have three types of unhappy arcs:

1. Blue arcs given 1.5 credit during the first phase of the initial credit allocation scheme. These arcs are called *unhappy singles.*

2. Blue arcs given 1.75 credit in the second phase of the initial credit allocation scheme. The two arcs from the same open chain of OPT are called *partners*, and they form an *unhappy pair.*

3. Blue arcs given 1.75 credit in the third phase of the initial credit allocation scheme. The three arcs from the same closed chain of OPT are called *partners*, and they form an *unhappy triple.*

For a node i, define $dep(i)$ to be the number of unhappy singles a with $o(a) = i$ plus the number of unhappy pairs $\langle a_1, a_2\rangle$ with $o(a_1) = i$. As arcs are made happy or assigned to valid chains (which are then output), $dep(i)$ is decreasing during the execution of the algorithm. Similarly, for a node i, define $arr(i)$ to be the number of unhappy singles a with $t(a) = i$ plus the number of unhappy pairs $\langle a_1, a_2\rangle$ with $t(a_2) = i$. As arcs are made happy or assigned to valid chains (which are then output), $arr(i)$ is decreasing during the execution of the algorithm.

Our credit transfer scheme maintains the following *deparr* invariant during the execution of the algorithm (that is, with respect to the current set of arcs): for any node i, if $arr(i) > 0$, then $arr(i) \le sur(i)$, and if $dep(i) > 0$, then $dep(i) \le -sur(i)$.

To maintain the deparr invariant, our credit transfer scheme ensures that whenever (during phases three or four) we choose an open chain, and we increase the surplus of a node i with $dep(i) > 0$, we also decrease $dep(i)$ by either making a unhappy single happy or by making the blue arc of an unhappy pair happy. Similarly, whenever (during phases three or four) we choose an open chain, and we decrease the surplus of a node i with $arr(i) > 0$, we also decrease $arr(i)$ by either making a unhappy single happy or by making the blue arc of an unhappy pair happy.

Our credit transfer scheme, as we will see, also makes sure that whenever an arc from an unhappy pair or from an unhappy triple is selected and used by the algorithm, any remaining blue arc from the pair or triple receives enough credit to become happy.

Phase two of the algorithm is designed to eliminate the unhappy triples. Please note that no surplus is changed, and therefore there is no need to worry about arrivals or departures.

Our credit transfer scheme is as follows: Let $P = \langle a_1, a_2, a_3\rangle$ be a closed chain selected in phase two of the algorithm, and assume that a_1 is the blue arc. The cost of P in the output of our algorithm is 3. The blue arc a_1 has at least 1.5 credit, and therefore the total credit on these three arcs is at least 3.5. We use 3 credit to cover the output, give to the unhappy partner (if it exists) of a_2 0.25 credit, and give to the unhappy partner (if it exists) of a_3 0.25 credit, thus making happy any remaining unhappy arc which had one of its partners selected. No unhappy triple remains at the end of the second phase.

We defer the description of the credit transfer scheme for the third and fourth phases of the algorithm to the journal version of this paper. ■

Obvious practical improvements are: after Phase 4, but before the Eulerian rounding, insert three other phases:

- Phase 4.1. While A contains a valid closed chain P, find one with minimum number of arcs. Nsplit the arcs in P, output P, and set $A \leftarrow A \setminus P$.

- Phase 4.2 While A contains a tight valid chain $P = \langle a_1, \cdots, a_k\rangle$, nsplit the arcs in P, output P, and set $A \leftarrow A \setminus P$.

- Phase 4.3 While A contains a closed chain P, find one with $b(P)$ minimum. The Floyd-Warshall algorithm on the graph with vertex set N and arc set A finds this minimum. Do an Eulerian rounding on P (producing $b(P)$ closed chains). Then set $A \leftarrow A \setminus P$.

And after Eulerian rounding, the three transformations (Merging, Combining and Splitting) proposed by [4] should also be applied, if possible. There is nothing special about the link $n-1$ deciding which arc is blue and which one is not. If enough computing time is available, one should try running the algorithm using each of the remaining links of the ring to decide which arc is blue and which one is not.

We are not able to prove a better approximation ratio based on these improvements. The worst example (for which the practical improvements also do not help) we have is the following: $n = 6$ and $A = \{a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3\}$, with $a_1 = (0,2)$, $a_2 = (2,5)$, $a_3 = (5,0)$, $b_1 = (2,4)$, $b_2 = (4,1)$, $b_3 = (1,2)$, $c_1 = (4,0)$, $c_2 = (0,3)$, and $c_3 = (3,4)$. If unlucky, the algorithm produces a solution of cost 10 (see Figure 2), while optimum is 9.

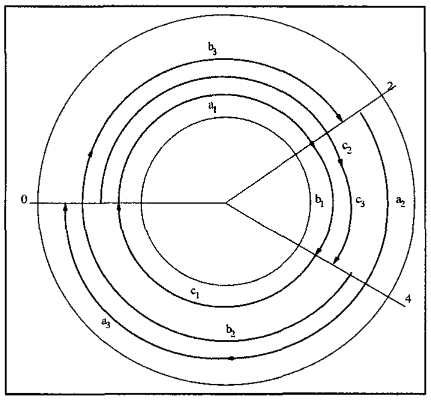

Figure 2. The algorithm selects the closed chain $\langle a_1, b_1, c_1\rangle$. Eulerian rounding then splits one arc in two (a_2, for example) and produces a solution of cost 10. Optimum uses the closed chains $\langle a_1, a_2, a_3\rangle$, $\langle b_1, b_2, b_3\rangle$, and $\langle c_1, c_2, c_3\rangle$, of total cost 9.

4 Conclusions

This paper presents a new approximation algorithm for the NP-Hard problem of wavelength assignment to splitable lightpaths over WDM rings, with the objective of minimizing the number of SONET ADMs. The approximation ratio of the algorithm is proved to be in between $10/9$ and 1.25, and it will be interesting to find the exact worst-case behavior of the algorithm.

We implemented the algorithm in C++ and run it on several randomly generated instances, with at most 160 nodes and 7000 arcs. The code is available upon request. Not being able to compute the optimum, we used as lower bound the number of arcs plus the deficiency. The output of the algorithm was between 7% and 15% more than the lower bound. We also implemented the more natural variation which uses Phases 4.1 and 4.3 before Phase 3. This natural variation, for which we could not prove a good approximation ratio, gave slightly better results on about half the instances.

References

[1] R. Barry and P. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers", IEEE JSAC/IEEE-OSA JLT: Special Issue on Optical Networks, vol. 14, no. 5, pp. 858-867, 1996.

[2] G. Călinescu and P.-J. Wan, "Traffic Partition in WDM/SONET Rings to Minimize SONET ADMs", submitted for publication.

[3] M. Garey, D. Johnson, G. Miller, and C. Papadimitriou, "The complexity of coloring circular arc graphs and chords", *SIAM Journal of Discrete Math*, vol. 1, no. 2, pp. 216-227, 1980.

[4] O. Gerstel, P. Lin, and G. Sasaki, "Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings", Proc. IEEE INFOCOM'98, vol. 1, pp. 94-101.

[5] O. Gerstel, G. Sasaki, and R. Ramaswami, "Dynamic wavelength allocation in WDM ring networks with little or no wavelength con-version", The 34th Allerton Conf. On Communications, Control, and Computing, 1996.

[6] O. Gerstel, G. Sasaki, and R. Ramaswami, "Cost effective traffic grooming in WDM rings", Proc. IEEE INFOCOM'98. vol. 1, pp. 69 -77.

[7] I. Haque, W. Kremer, and K. Raychauduri, "Self-Healing Rings in a synchronous environment", SONET/SDH: a sourcebook of synchronous networking, Eds. C.A. Siller and M. Shafi, IEEE Press, New York, pp. 131-139, 1996.

[8] L.W. Liu , X.-Y. Li, P.-J. Wan, and O. Frieder, "Wavelength Assignment in WDM Rings to Minimize SONET ADMs", INFOCOM 2000.

[9] P. Raghavan and E. Upfal, "Efficient routing in all-optical networks", in Proc. 26th ACM Symp. Theory of Computing, pp. 134-143, 1994.

[10] A. Tucker, "Coloring a family of circular arcs", SIAM Journal of Applied Math, vol. 29, no. 3, pp. 493-502, 1975.