A Polynomial Time Approximation Scheme for the SONET Ring Loading Problem

Sanjeev Khanna*

Abstract

A rapidly emerging design paradigm for large-scale optical-fibre networks involves connecting clusters of nodes through a SONET ring and building a network over these rings. While the fiber itself offers virtually unlimited bandwidth, the add-drop multiplexers (ADMs) determine the actual bandwidth available along any edge of the SONET ring. Consequently, there is a cost involved in supporting a high bandwidth along the ring. An important optimization problem arising in this context is as follows: Given a set of nodes connected along a bi-directional SONET ring, determine a routing scheme which minimizes the bandwidth required to satisfy all the pairwise traffic demands. This problem, called the *ring loading problem*, is the focus of this work.

The ring loading problem has been well-studied in the recent years. Cosares and Saniee [2] showed that this problem is NP-hard and thus it is unlikely that there exists a polynomial time algorithm to compute an optimal solution. We, therefore, shift our attention to polynomial time approximation algorithms. Towards this end, Cosares and Sainee [2] gave a polynomial time algorithm that approximates the optimal solution value to within a multiplicative factor of two. More recently, Schrijver, Seymour and Winkler [6] gave an efficient algorithm to compute a solution which exceeds the optimum by at most an additive term of $1.5d_{\max}$ where $d_{\max}$ is the largest traffic demand between any pair of nodes. While this additive term is relatively small, on many input instances, it would represent a significant fraction of the optimum solution value and hence a possibly significant deviation from it. An open problem is if there exists a polynomial time approximation algorithm which allows us to achieve an error which is an arbitrarily small fraction of the optimal solution value. In this paper, we answer this question in the affirmative. More precisely, building on the work of Schrijver *et al*, we show that for any fixed $\varepsilon > 0$, there exists a polynomial time algorithm which computes a solution that requires bandwidth at most $(1+\varepsilon)$ times the optimal bandwidth.

*`sanjeev@research.bell-labs.com`. Fundamental Mathematics Research Department, Bell Labs, 700 Mountain Avenue, NJ 07974.

1 Introduction

The ever-growing demands for high bandwidth services have resulted in a sustained trend towards extensive utilization of fiber-optics technology in information networks. Networks based on this technology are capable of carrying gigabytes of information at lightning speeds. An effective utilization of this new technology requires both new global data transmission standards and new tools for network design. The North American standard for data transmission on optical fibre networks is known as SONET (Synchronous Optical NETwork) (see [1, 5], for example) and is based on the popular ring topology – appealing for both its simplicity and self-healing properties. The emerging paradigm for designing large-scale networks based on the fibre-optics technology involves connecting clusters of nodes through a SONET ring and building a network over these rings.

The nodes on the SONET ring send and receive messages via add-drop multiplexers (ADMs). The actual bandwidth that is available along a SONET ring is determined by the capacity of these multiplexers even though the fiber itself offers virtually unlimited bandwidth. Thus there is an additional hardware cost involved in supporting a desired level of bandwidth along the SONET ring. An important and basic optimization problem arising in this context is as follows: Given a set of nodes connected along a bi-directional SONET ring and a list of traffic demands between every pair of nodes, determine a routing scheme which minimizes the bandwidth required to satisfy all the demands. This problem, called the *ring loading problem*, is the focus of this work.

The ring loading problem has been a subject of some active study in the recent years. Cosares and Saniee [2] showed that this problem is NP-hard by showing a reduction from the *Partitionproblem* [3]. The authors also presented several heuristics to obtain good approximate solutions and in particular, they showed that one can obtain a factor of two approximation in polynomial time. Schrijver, Seymour and Winkler have recently shown that it is possible to obtain a solution in polynomial time such that it exceeds the optimal capacity requirement by at most an additive term of $3/2$ times the maximum pairwise demand on the ring, say $d_{\max}$. Clearly, if the optimum is large, then this additive error term translates into being a relatively small deviation from the optimal solution value. But on the other hand, it is possible that the optimal solution value is not much larger than $d_{\max}$ and thus the additive error term could represent a significant deviation from the optimum. Thus we have the following unresolved question: Given any fixed deviation $\varepsilon > 0$ from the optimum, can one obtain in polynomial time a solution which is at most $(1+\varepsilon)$ times the optimum? We answer this question in the affirmative by designing a *polynomial time approximation scheme* for the ring loading problem.

The rest of this paper is organized as follows. Section 2 covers relevant background concepts and lays a technical foundation for the ideas used in our algorithm. In Section 3, we present our polynomial time approximation scheme. Finally, we conclude with an open problem in Section 4.

2 Preliminaries

2.1 Approximation: Definitions and Terminology

Since NP-hard optimization problems are unlikely to have a polynomial time exact algorithm, it is natural to focus on obtaining good approximate solutions.

Definition 1 (Approximation Algorithm) *A polynomial time algorithm A is an approximation algorithm for an optimization problem Π if given an input instance $\mathcal{I}$, it computes a feasible solution S for the input $\mathcal{I}$.*

Of course, the value of an arbitrary feasible solution can be far from the optimal value. Our interest is in algorithms which return solutions with guaranteed near-optimal values. The following definitions help us characterize the performance of approximation algorithms. We use the notation $\mathrm{OPT}(\mathcal{I})$ to denote the optimum objective function value on instance $\mathcal{I}$ of a problem.

Definition 2 (Performance Ratio) *An approximation algorithm A for an optimization problem Π has* performance ratio $\mathcal{R}(n)$ *if, given any instance $\mathcal{I}$ of Π with $|\mathcal{I}| = n$, it computes a solution whose value, say $A(\mathcal{I})$, satisfies*

$$\max\left\{\frac{A(\mathcal{I})}{\mathrm{OPT}(\mathcal{I})}, \frac{\mathrm{OPT}(\mathcal{I})}{A(\mathcal{I})}\right\} \leq \mathcal{R}(n).$$

A solution of value within a multiplicative factor $\mathcal{R}(n)$ of the optimal value is referred to as an $\mathcal{R}(n)$-approximation. *We say that a problem is approximable to within a factor $\mathcal{R}(n)$ if it has a polynomial time approximation algorithm with performance ratio $\mathcal{R}(n)$.*

Definition 3 (Polynomial Time Approximation Scheme) *An optimization problem Π is said to have a* polynomial time approximation scheme *if for any rational $\varepsilon > 0$, there exists a polynomial time approximation algorithm for Π whose performance ratio is bounded by $(1 + \varepsilon)$.*

2.2 SONET Ring Loading: Problem Statement and Algorithmic Foundations

We assume that a SONET ring consists of n nodes labeled 0 through $(n-1)$ arranged in a clockwise manner. A reference to some node i in the ring is to be interpreted as a reference to the node labeled $(i \bmod n)$. The demand between a pair of nodes, say i and j, is denoted by $d_{i,j}$. The maximum demand is denoted by $d_{\max}$. Given a subset X of nodes in the ring, we denote by $D(X)$ the total pairwise demand between them, that is, $D(X) = \sum_{\{i,j \in X, i<j\}} d_{i,j}$.

2.2.1 The Ring Loading Problem

Given a sequence of non-negative integers $\{d_{i,j}\}_{0 \leq i<j<n}$ with $d_{i,j}$ denoting the pairwise demand between the nodes i and j of the ring of size n, compute a map

$$\phi : \{(i,j) \mid 1 \leq i < j \leq n\} \to \{0,1\}$$

so as to minimize $\max_{0 \leq k<n} c_k$ where

$$c_k = \underbrace{\sum_{0 \leq i<j<n} \{d_{i,j} \mid \phi(i,j) = 1 \wedge (i \leq k < j)\}}_{\text{Load due to demands routed clockwise}} + \underbrace{\sum_{0 \leq i<j<n} \{d_{i,j} \mid \phi(i,j) = 0 \wedge (k < i \text{ or } k \geq j)\}}_{\text{Load due to demands routed counter-clockwise}}.$$

The quantity c_k denotes the total capacity needed at the edge $e_k = (k, k+1)$ to implement the routing specified by ϕ where the demand $d_{i,j}$ is routed clockwise if $\phi(i,j) = 1$ and counter-clockwise otherwise.

While the ring loading problem of interest to us requires the range of ϕ to be $\{0,1\}$, a useful variant of the problem, called the *fractional ring loading problem*, allows ϕ to take any value in the interval $[0,1]$. Such a solution is called a *fractional solution*. Unlike the ring loading problem, this relaxed variant can be solved in polynomial time using linear programming or even a combinatorial approach (see [6]).

2.2.2 Cuts and Demands: Upper and Lower Bounds on the Optimum

Given any two distinct edges, say $e_i = (i, i+1)$ and $e_j = (j, j+1)$, along the ring, they determine a *cut* and their deletion partitions the ring into precisely two connected components: one consisting of the nodes $\mathcal{C}_a(e_i, e_j) = \{i+1, i+2, ..., j\}$ and the other consisting of the nodes $\mathcal{C}_b(e_i, e_j) = \{j+1, j+2, ..., i\}$.

Definition 4 (Demand Across a Cut) *Given a pair of edges e_i and e_j along the ring, the demand across the cut induced by them, denoted $\mathcal{D}(e_i, e_j)$, refers to all the pairwise demands between the nodes of the two components obtained by deleting the edges e_i and e_j. In other words,*

$$\mathcal{D}(e_i, e_j) = \sum_{p \in \mathcal{C}_a(e_i, e_j), q \in \mathcal{C}_b(e_i, e_j)} d_{\min\{p,q\}, \max\{p,q\}}.$$

It is easy to see that if C_i denote the capacity assigned to the edge e_i, then any solution to the fractional (and hence also the integral) ring loading problem requires

$$C_i + C_j \geq \mathcal{D}(e_i, e_j) \qquad 0 \leq i < j < n.$$

The following theorem of Okamura and Seymour [4] shows that for the fractional ring loading problem, the converse also holds.

Theorem 1 [4] *If $C_i + C_j \geq \mathcal{D}(e_i, e_j)$ for $0 \leq i < j < n$, then there exists a solution to the fractional ring loading problem.*

An immediate consequence of the above theorem is that if $\mathcal{D}_{max}$ denotes the largest demand across any cut of the ring, the optimum solution value for the fractional ring loading problem is $\mathcal{D}_{max}/2$. This gives us a lower bound on the optimum for the (integral) ring loading problem.

The following theorem, implicit in the work of Schrijver, Seymour and Winkler [6], establishes an upper bound on the optimum by showing a rather strong relation between the fractional and integral solutions:

Theorem 2 ([6]) *Given a SONET ring and a fractional solution which satisfies all the pairwise node demands, there exists a polynomial time algorithm that maps it to an integral solution such that the maximum capacity required along any edge of the ring does not exceed the capacity required by the fractional solution by more than $3d_{\max}/2$.*

An example in [6] shows that the best performance guarantee possible from the above theorem is a $4/3$-approximation.

3 The Approximation Scheme

We now design a scheme where for any fixed $\varepsilon > 0$, we give a polynomial time algorithm which gives a $(1+\varepsilon)$-approximation. In the following discussion, we slightly abuse notation and denote by OPT both an optimal solution and its value. The basic intuition behind our scheme is simple: we reduce any given instance $\mathcal{I}$ of the ring loading problem to another instance $\mathcal{I}'$ such that the maximum demand between any pair of nodes is at most $2\varepsilon\mathrm{OPT}/3$ where OPT denotes the optimum solution value. Now solving this instance using the approach of [6], we can ensure that the deviation from the optimum is bounded by $\varepsilon \cdot \mathrm{OPT}$.

3.1 Basic Framework

Definition 5 (Long Way Routing) *We say that a demand between two nodes on the ring is routed the* long way *if it is routed along the longer of the two paths connecting them along the ring.*

If the ring has an even number of nodes, then for any pair of nodes that is antipodal, we arbitrarily choose one of the two equal length routings to be the long way routing.

Lemma 1 *Let Γ be the total demand that is routed the long way around the ring. Then some edge in the ring must have capacity at least $\Gamma/2$.*

Proof: A demand that is routed the long way, must traverse at least $\lceil n/2 \rceil$ edges and thus the total capacity needed along the ring is at least $\lceil n/2 \rceil \Gamma$. By pigeonhole principle, some edge must have capacity at least $\Gamma/2$. □

Let S be the set of all pairwise demands and let S_{high} be the set of all pairwise demands which are at least as large as $2\varepsilon \mathrm{OPT}/3$. The set $S \setminus S_{\text{high}}$ is denoted by S_{low}. A useful corollary of Lemma 1 is as follows.

Corollary 1 *In any optimal solution, at most $3/\varepsilon$ demands in S_{high} can be routed the long way.*

3.2 The Algorithm

We are now ready to present our scheme. Assume that we know the optimum solution value, namely OPT. Use this value to partition the set of demands S into two sets S_{low} and S_{high} as above. Now repeat the following steps for every subset S' of size at most $3/\varepsilon$ in S_{high}.

Step A: Route the demands in the set S' the long way. The remaining demands in S_{high} are routed along the shorter of the two paths.

Step B: Let a_i denote the capacity needed at edge e_i to handle the routing determined by Step A. We now solve the following linear program:

Minimize c

Subject to:

$$\begin{array}{rcll} a_i + x_i & \leq & c & 0 \leq i \leq (n-1) \\ x_i + x_j & \geq & \mathcal{D}_{\text{low}}(e_i, e_j) & 0 \leq i < j \leq (n-1) \\ x_i & \geq & 0 & \end{array}$$

Let c^* denote the optimal value for the above program and let $x_0^*, x_1^*, ..., x_{n-1}^*$ be the values taken by the variables corresponding to this solution. Theorem 1 ensures that there exists a fractional solution for routing all the demands in S_{low} on a ring where the edge e_i has capacity x_i^*.

Since solving a linear program is computationally expensive, we use the following combinatorial approach instead. We start with $c = \max_{0 \leq i < n} a_i$ and we set $x_i = c - a_i$ for the edge e_i. If this assignment does not satisfy all the constraints, we pick a violating cut and increase c by the least amount necessary and repeat this process. If we visit the cuts in the increasing order of demands across them, this procedure can be implemented in $O(n^2 \log n)$ as no cut needs to be visited twice.

Step C: Now using the Schrijver *et al* algorithm, we can compute a fractional routing of demands in S_{low} and then convert it into an integral routing by adding at most an additional capacity of $3/2$ times the maximum demand in S_{low}. In other words, the fractional solution can be made integral by adding a capacity of at most $(3/2) \cdot (2\varepsilon \mathrm{OPT}/3) = \varepsilon \cdot \mathrm{OPT}$ on every edge. The running time of this procedure is $O(n^4)$.

3.3 Analysis

We first analyze the algorithm assuming that the optimal solution value is known and then later show how this assumption can be eliminated.

3.3.1 Correctness

Fix an optimal routing scheme OPT. Corollary 1 tells us that OPT routes at most $3/\varepsilon$ demands in S_{high} the long way. Since our algorithm exhaustively tries all possible ways that OPT might route the demands in S_{high}, in some iteration of the algorithm, we must commit ourselves to precisely the same routing scheme. The formulation of the linear program in Step B ensures that for any possible way that OPT may extend the routing scheme chosen for demands in S_{high} to cover the demands in S_{low}, there exists an edge on the ring which must have capacity at least c^*. On the other hand, the solution computed by our algorithm can route all the demands by using a maximum capacity of $c^* + \varepsilon\mathrm{OPT}$. Thus we indeed obtain a $(1+\varepsilon)$-approximation to OPT.

3.3.2 Time Complexity

The total number of times Steps A, B and C are executed is bounded by

$$M = \sum_{i=0}^{\frac{3}{\varepsilon}} \binom{|S_{\mathrm{high}}|}{i}.$$

Using the fact that $\binom{n}{k} \leq (en)^k/k^k$, we obtain

$$M \leq \frac{3}{\varepsilon}\left(\frac{e|S_{\mathrm{high}}|}{\frac{3}{\varepsilon}}\right)^{\frac{3}{\varepsilon}}. \tag{1}$$

A trivial upper bound on $|S_{\mathrm{high}}|$ is $\binom{n}{2}$. A much better bound is given by the following simple lemma.

Lemma 2 *The set S_{high} contains at most $3n/2\varepsilon$ demands.*

Proof: We know that the total demand across any cut of the ring can not exceed $2\mathrm{OPT}$. Since every demand participates in at least $(n-1)$ cuts and has value at least $2\varepsilon\mathrm{OPT}/3$, the following inequality must be satisfied:

$$(n-1)\left(\frac{2\varepsilon\mathrm{OPT}}{3}\right)|S_{\mathrm{high}}| \leq \binom{n}{2}(2\mathrm{OPT}).$$

The left hand side gives a lower bound on the total contribution to the demands across all possible cuts of the ring while the right hand side specifies an upper bound on the aggregated demand across all possible cuts given that there exists a solution of cost OPT. Rearranging the terms, we get that $|S_{\mathrm{high}}| \leq 3n/2\varepsilon$. □

Substituing this bound in Equation 1, we obtain

$$M \leq \frac{3}{\varepsilon}\left(\frac{en}{2}\right)^{\frac{3}{\varepsilon}}.$$

The combined complexity of the steps A through C is $O(n^4)$. Thus the overall running time of this scheme is $O(\frac{1}{\varepsilon}\cdot n^{\frac{3}{\varepsilon}+4})$.

3.3.3 Bypassing the Clairvoyance Assumption

Thus far in the design and analysis of our routing scheme, we assumed a knowledge of the optimum solution value. But since this value is unknown in reality, we must eliminate this assumption. The simple approach to run our scheme for each possible value that the OPT may take, fails, as the OPT may be exponentially large in n. So we proceed as follows instead. We first obtain an upper bound B on the optimum which is at most $2\mathrm{OPT}$. This is easily obtained by running a simple routing scheme such as the one where we route each demand along the shorter of the two paths on the ring (see [2], for instance). We now divide B into $2n$ equal intervals and run our approximation scheme for the following sequence of $2n$ estimators of OPT: the i_{th} estimator, say OPT_i, takes the value $B * i/2n$ where $1 \leq i \leq 2n$. Clearly, this way, in some iteration of our scheme, we have an estimate of OPT, say OPT', which is at least OPT and at most $(1+1/n)\mathrm{OPT}$. All of the preceding analysis now goes through except the bound on the deviation from the optimum is now $\varepsilon\mathrm{OPT}' = (\varepsilon + \varepsilon/n)\mathrm{OPT}$. Thus to achieve a $(1+\varepsilon)$-approximation, we now simply work with $\varepsilon' = \varepsilon/(1+1/n)$. Hence the running time for computing a $(1+\varepsilon)$-approximation is $O(\frac{1}{\varepsilon} \cdot n^{\frac{3(1+1/n)}{\varepsilon}+5})$.

4 Concluding Remarks

We described an algorithm to efficiently route traffic around a SONET ring. Besides the standard input, the algorithm takes an additional parameter ε which signifies the degree of accuracy desired, guaranteeing a solution of cost at most $(1+\varepsilon)$ times the optimal. The running time of the algorithm grows as we choose smaller and smaller values for the parameter ε.

There are at least two distinct phases where a typical SONET network design tool may need to compute a good routing on a ring. The first of these involves computing good routings on a very large number of candidate rings while the second one involves computing (possibly) better routings on a small suitably chosen subset of these candidate rings. Since the algorithm requires a large computational effort when ε is chosen to be small, it is more likely to be of use in the second phase when the basic network design is already in place.

A large number of other NP-hard problems which have such parameterized approximation schemes, have a running time complexity that behaves as $O(2^{f(\varepsilon)}n^{O(1)})$ and thus has a better asymptotic dependence on ε. An interesting question is: does such an approximation scheme exist for the ring loading problem? It is conceivable that results from the theory of fixed parameter tractability may be used to show that such a scheme is unlikely to exist.

Acknowledgements

We would like to thank Peter Winkler for many useful discussions and helpful comments.

References

[1] J. Babcock. SONET: A Practical Perspective. In *Business Communications Review*, Sept 1990: 59–63.

[2] S. Cosares and I. Saniee. An optimization problem related to balancing loads on SONET rings. In *Telecommunications Systems*, 3:165–181, 1994.

[3] Michael R. Garey and David S. Johnson. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. W. H. Freeman (1979).

[4] H. Okamura and P. D. Seymour. Multicommodity Flow in Planar Graphs. In *J. Combin. Theory Ser. B*, 31:75–81, 1981.

[5] G. R. Ritchie. SONET Lays the Roadbed for Broadband Networks. In *Networking Management*, 1990.

[6] A. Schrijver, P. Seymour and P. Winkler. The Ring Loading Problem. In *SIAM J. Discrete Math.* (to appear).