Reactive Search for Traffic Grooming in WDM Networks

Roberto Battiti[*][†] Mauro Brunato[*][†][‡]

Abstract

In this paper the *Reactive Local Search* (RLS) heuristic is proposed for the problem of minimizing the number of expensive Add-Drop multiplexers in a SONET or SDH optical network ring, while respecting the constraints given by the overall number of fibers and the number of wavelengths that can carry separate information on a fiber.

RLS complements local search with a history-sensitive feedback loop that tunes the prohibition parameter to the local characteristics of a specific instance.

Simulation results are reported to highlight the improvement in ADM cost when RLS is compared with greedy algorithms used in the recent literature.

1 Introduction

During the last few years, dramatic advances in optical communications have led to the creation of large capacity optical WANs, usually in the form of hierarchies of rings. As shown in Fig. 1, these rings are joined together via hubs in hierarchies at various levels.

In the framework of Wavelength Division Multiplexing (WDM) the overall bandwidth of the fiber is divided into densely packed adjacent frequency bands. In this manner, every fiber can carry a large number of high-capacity channels. Dividing these channels into lower-bandwidth virtual connections between couples of nodes gives rise to technical difficulties: wavelengths are at most a few hundreds, therefore they need to be time-multiplexed to be shared among many couples of communicating nodes. The key components to achieve time multiplexing, the Add-Drop Multiplexers (ADM), are need edeach time a wavelength has to be processed electronically at a node in order to add or drop packets. Therefore they represent a significant fraction of the total cost of the infrastructure.

We consider the problem of dividing the available bandwidth of a single ring into many channels to establish as many node-to-node connections as requested. In the following, the term *virtual connection* shall be used to refer to an elementary node-to-node communication channel.

In Fig. 2 we describe the structure of a node in the ring. Optical bandwidth is shared among communication channels in two ways (both of which are implemented in state-of-the-art optical networks): the incoming signal is initially split into its component wavelengths by means of a Wavelength Demultiplexer (WDEMUX), then some wavelengths are directly relayed to the next node, while some others are further split into packets via a time division multiplexing mechanism operated by Add-Drop Multiplexers (ADM). ADMs are critical components (they need to exploit fast packet conversion, header examinations, and must be tightly synchronised), therefore they are expensive.

In this paper we shall consider unidirectional WDM rings where each wavelength can be time-shared among g virtual connections. In other words, the bandwidth of a wavelength is assumed to be g (also called *grooming factor*), while single virtual connections, carried by a wavelength in a single time slot, have a unit bandwidth.

The paper is organized as follows. Section 2 describes the Reactive Local Search heuristic and summarizes the reasons leading to its choice for the WDM Traffic Grooming problem. Section 3 defines the problem, and summarizes approaches that have already been used for its solution. Section 4 is devoted to the application of RLS to the WDM traffic grooming problem, with a description of the design choices and of the data structures

[*]Università di Trento, Dipartimento di Matematica, via Sommarive 14, I-38050 Pantè di Povo (TN) — Italy
[†]Email: `battiti|brunato@science.unitn.it`
[‡]Corresponding author.

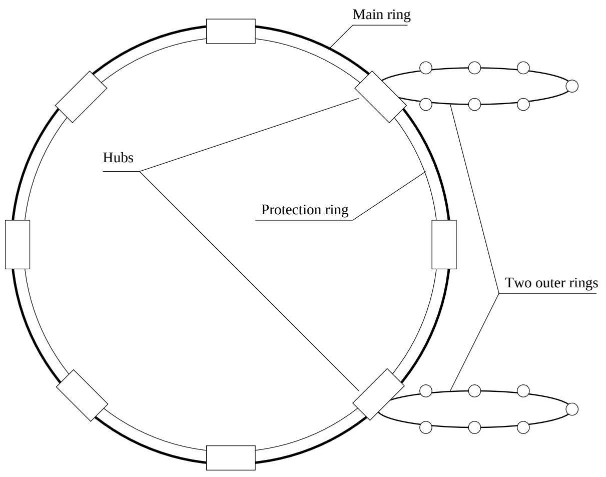

Figure 1: a SONET/SDH hierarchy of rings

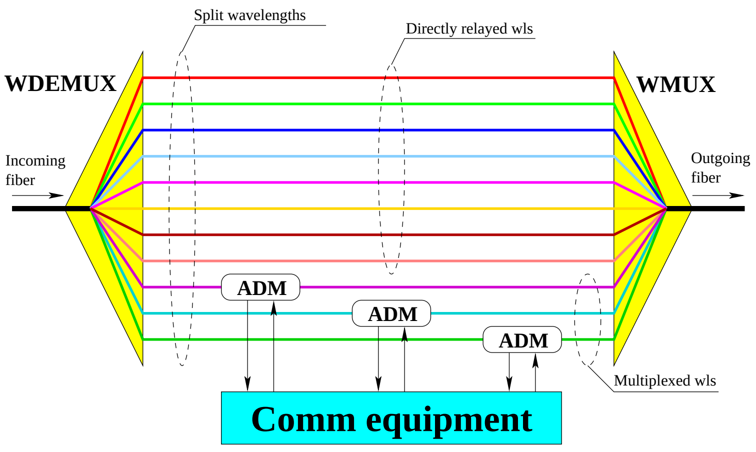

Figure 2: structure of a node in a WDM SONET/SDH ring

```
1.  var x, best_x: binary string; n, cost, best_cost: integer;
2.  best_cost ← +∞
3.  repeat
4.      generate a random solution into x
5.      cost ← f(x)
6.      repeat
7.          let n be one of the bits of x whose flipping most decreases
8.              the value of f(x)
9.          if a decrease is possible then
10.             x[n] = not x[n]
11.             cost ← f(x)
12.     while an improvement is possible
13.     if cost < best_cost then
14.         best_cost ← cost
15.         best_x ← x
16. until some termination condition is satisfied
```

Figure 3: the basic Local Search algorithm on a binary string

involved in the process. Section 5 analyzes simulation results, giving an experimental comparison among previous heuristics and the proposed technique.

2 Reactive Local Search

Let us consider a discrete optimization problem, where the system configuration x is given by parameters varying in a discrete set C, for example a binary string ($C = \{0, 1\}^n$) or an integer vector ($C = \mathbb{N}^n$), and the optimality of a given configuration is evaluated by a cost function $f : C \to \mathbb{R}$.

Many interesting optimization problems are known to be computationally unaffordable, and the WDM minimization problem has been proved NP-hard [10]. Local Search techniques are a family of heuristics aimed at finding near-optimal solutions to hard problems, optimizing the value of the cost function by local modifications of the system configuration.

A simple example is given in Fig. 3, where a repeated *local search* technique is applied to the problem of minimizing a function of a binary string. Here "local" means that the step from a solution to the next is performed by flipping a single bit of the string. In other words, the *neighborhood* of a solution is given by all solutions having Hamming distance 1 from it.

The basic steps of the algorithm are lines 4–12. Here a random configuration is generated (line 4), then local steps are repeatedly performed by evaluating the objective function on all neighbors and finding the best improvement (lines 6–12). If f is bounded from below, this leads to a local minimum of the cost function[1], and by repeating the whole procedure with new random starting points (lines 3–16) the optimum will eventually be found. Once a local minimum is found the algorithm will update the best configuration (lines 13–15) and jump to a completely new starting point.

A substantial improvement to the previous scheme is obtained when one considers that problem instances are not completely unstructured. Usually, local minima tend to cluster, and once one is found it is advisable to continue exploring its vicinities rather than starting immediately from a new random point. The basic scheme cannot do that, because it is forced to explore only towards improving solutions. A possible modifications of this approach is Simulated Annealing, which accepts - with a given probability - solutions that lead to increases of the cost function; notice that Simulated Annealing is a Markovian heuristic, where the next configuration produced is not influenced by past history, but only by the current configuration.

[1]Unless a neighborhood with equal cost is reached (a "plateau"): modifications of the algorithm in Fig. 3 would be necessary in this case

Another branch of local search heuristics consists of non-Markovian techniques, such as prohibition-based heuristics that forbid some moves according to criteria that take into account past moves. Prohibition-based schemes date back to the sixties [8, 9], and have been proposed for a growing number of problems in the eighties with the term *Tabu Search* (TS) [6] or *Steepest Descent-Mildest Ascent* [7]. Fixed TS (see the classification proposed in [1]) implements the Local Search technique with two major modifications:

1. it does not terminate a run when the local minimum is found and
2. once a move (for example a bit flipping) is made, the reverse move (i.e. flipping that bit again) is prohibited for a given number of iterations T (called *prohibition period*).

Although various TS schemes have been shown to be effective for many problems, some of them are complex and contain many possible choices and parameters, whose appropriate setting is a problem shared by many heuristic techniques In some cases the parameters are tuned through a trial-and-error feedback loop that includes the user as a crucial *learning* component: depending on preliminary tests, some values are changed and different options are tested until acceptable results are obtained. The quality of results is not automatically transferred to different instances and the feedback loop can require a lengthy process.

On the other hand, *reactive schemes* aim at obtaining algorithms with an internal feedback (*learning*) loop, so that the tuning is automated. Reactive schemes are therefore based on *memory*: information about past events is collected and used in the future part of the search algorithm. The TS-based Reactive Local Search (RLS) adopts a reactive strategy that is appropriate for the neighborhood structure of the problem: the feedback acts on a single parameter (the *prohibition period*) that regulates the search diversification and an explicit memory-influenced restart is activated periodically. RLS has been successfully applied to a growing number of problems, from Maximum Clique [3] to Graph Partitioning [2].

In this paper, RLS is adapted for the ADM minimization problem in WDM traffic grooming.

3 The WDM Traffic Grooming Problem

Let N be the number of nodes in the ring, and let M be the number of available wavelengths, computed as the number of fibers times the number of wavelengths per fiber. The problem is not affected by the values of the two factors, since the only important number is the overall number of wavelengths.

Every physical link from a node to its neighbor is capable of carrying M wavelengths; at every node, some of these wavelengths will be simply relayed to the outgoing link by an internal fiber without electronic conversion, while some others will be routed through an ADM, which is therefore necessary only if some of the traffic contained in that wavelegth is directed to, or originated from, that node.

Let g be the *grooming factor*, i.e. the number of low-bandwidth channels that can be packed in a single wavelength on a fiber. For example, if the wavelengths are carrying an OC-48 channel each, a grooming factor $g = 16$ means that traffic is quantized into 16 OC-3 time-multiplexed channels, while if $g = 4$ only 4 OC-12 time-multiplexed channels will be carried.

Another fundamental parameter is the *traffic pattern*, an $N \times N$ matrix T whose entry t_{ij} is the number of time-multiplexed low-bandwidth unidirectional virtual connections required from node i to node j. Note that, being the ring unidirectional, there is no reason to consider the matrix as symmetric, and that the diagonal elements must be null: $t_{ii} = 0$.

Let P be the overall number of virtual connections along the ring: $P = \sum_{ij} t_{ij}$. A solution to the problem can be given by an $N \times N$ matrix W whose entry w_{ij} is an integer array of t_{ij} elements (thus empty if $i = j$), one for each virtual connection from node i to node j. Thus, the wavelength assigned to the k-th virtual connection from i to j ($1 \leq i, j \leq N, \quad 1 \leq k \leq t_{ij}$) is w_{ijk} ($1 \leq w_{ijk} \leq M$).

The number of ADMs required at node i ($1 \leq i \leq N$) is the cardinality of the set of wavelengths assigned to virtual connections that originate from, or go to, node i:

$$CH_i(W) = \Big\{ w_{ijk} : 1 \le j \le N \wedge 1 \le k \le t_{ij} \Big\} \bigcup$$
$$\bigcup \Big\{ w_{jik} : 1 \le j \le N \wedge 1 \le k \le t_{ji} \Big\}.$$

Because $CH_i(W)$ is a set, multiple occurrences of the same wavelength are counted once (indeed, only one ADM is required to deal with them). The overall number of ADMs needed for a given wavelength assignment W is

$$f(W) = \sum_{i=1}^{N} \#CH_i(W). \tag{1}$$

Wavelength assignment matrix W is subject to the constraint that no wavelength should be overloaded in any fiber of the ring. The fiber segment exiting from node n (and going to node $(n \mod N) + 1$) is traversed by all virtual connections (i, j) where $i \le n < j$ or $n < j < i$ or $j < i \le n$. The load of wavelength l on the outgoing fiber of node n is given by the cardinality of the set

$$\begin{aligned} WL_{nl} = \Big\{ (i,j,k) : & \\ & (1 \le i \le n < j \le N \vee 1 \le n < j < i \le N \vee 1 \le j < i \le n \le N) \\ & \wedge \quad 1 \le k \le t_{ij} \quad \wedge \quad w_{ijk} = l \Big\}. \end{aligned}$$

The load constraint will assert that

$$\forall n, l \qquad (1 \le n \le N \wedge 1 \le l \le M) \Rightarrow \#WL_{nl} \le g. \tag{2}$$

The WDM traffic grooming problem can be stated as follows.

WDM GROOMING PROBLEM:
Given integers $N > 0$, $M > 0$, $g > 0$ and the $N \times N$ traffic matrix $T = (t_{ij})$
Find a wavelength asignment

$$W = (w_{ijk}) \qquad (1 \le i, j \le N, \quad 1 \le k \le t_{ij}, \quad 1 \le w_{ijk} \le M)$$

that minimizes the objective function (1) subject to the load constraint (2).

Most papers on the traffic grooming problem propose combinatorial greedy algorithms [4, 5, 10]. For example, [5] suggests some techniques for different kinds of traffic matrices, notably the egress-node case where all traffic is directed towards a single hub node, the uniform all-to-all case and a more general distance dependent traffic. Two types of algorithms are presented. Some algorithms attempt to maximize the number of nodes requiring just one ADM, then those requiring two and so forth. Others try to efficiently pack the wavelengths by dividing nodes into groups and assigning to different wavelengths intra-group traffic.

4 Reactive Local Search for the Grooming Problem

To implement RLS in an efficient way, the problem needs to be formulated in terms of an integer array optimization. To transform the wavelength assignment matrix into an integer array we simply associate an array entry to each virtual connection.

First of all, all indices start from 0, so in the following sections nodes indices will vary from 0 to $N-1$ and channel numbers from 0 to $M-1$. All virtual connections between nodes are enumerated consecutively, so that the k-th virtual connection from node i to node j is assigned index

$$p_{ijk} = \sum_{i'=0}^{i-1}\sum_{j'=0}^{N-1} t_{i'j'} + \sum_{j'=0}^{j-1} t_{ij'} + k,$$

so that any node couple (i,j) is assigned a consecutive group of t_{ij} paths. The overall number of paths is, of course,

$$P = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} t_{ij}.$$

The wavelength assignment matrix W is stored into an P-entries integer array $S=(s_i)$, where the wavelength assigned to the k-th virtual connection from node i to node j is stored into $s_{p_{ijk}}$.

The objective function (1) and the load constraint (2) are given in Sect. 3.

To implement a local search heuristic we need to define the neighborhood of a given configuration S. The basic move we chose is equivalent to changing the wavelength assignment of a given virtual connection, i.e. to changing a single entry of the current configuration array S.

Last, we modified the objective function to take into account load constraint violations by adding to it a penalty term given by the number of violations multplied by a very large constant (larger than NM, i.e. the maximum number of ADMs in the system). For this reason we needn't explicitly check for a non-violating string: while minimizing the objective function the number of violations is automatically reduced.

We show in Fig. 4 an outline of the Reactive Local Search algorithm used for the WDM grooming problem. The function accepts three parameters (line 1): the maximum number of moves *max_moves*, the integer array *best_x* which is going to contain the best found solution and the cost function *ADMs*.

In the first section (lines 4–7) variables are declared. The array *x* contains the current problem configuration, *T* contains the current prohibition period (time after which a given move can be repeated), while the array *LastExecuted* is indexed by moves and contains the last iteration a move was performed ($-\infty$ if it was never executed). By *move* we mean the local step from a configuration to the next. In our tests a move is represented by a couple of integers (i,n) meaning that wavelength n is assigned to the i-th virtual connection.

The initialization section (lines 8–15) generates a random configuration, initializes the prohibition period *T* and sets all entries in *LastExecuted* to minus infinity (they have never been performed).

The solution improvement cycle (lines 16–28) is made of two distinct parts. At first a suitable move is found: the move must be the "best" in the sense that, after some nonprohibited moves have been checked (as it may be impractical to check all possible moves, we chose to stop after checking 1000 possible candidates), a move is chosen that most decreases the *ADMs* function or, if it is impossible, it increases *ADMs* as little as possible. After the move has been performed, the configuration string is updated and the new value of the cost function is computed, compared with the best value and eventually stored.

After every move, a *reaction* step is performed (lines 25–28): a configuration dictionary is checked for the current configuration. If it has already been visited, then the prohibition period *T* is increased by some amount (10% in our test), while if no configuration has been repeated for some time the value of *T* is reduced (again, by 10%).

The implementation of the Reactive Local Search algorithm has been done in C++ language. The program operates on the P-entry array, while a 64-bit (`long long int`) hash fingerprint of each visited configuration is used to index a LEDA `dictionary` structure containing relevant data such as the iteration number and the number of times that configuration has been visited.

The initial prohibition period is 1 (in this case a move cannot be undone in the next step). If the number of configurations that have been visited more than three times exceeds 3, the prohibition time is increased by a 10% amount and rounded to the next integer. If prohibition time has not been raised for a certain number of steps, then it is decreased by the same amount (a high prohibition time facilitates escaping from local minima, but prevents a large fraction of neighboring configurations from being explored).

```
1.  function RLS_for_Grooming (max_moves: integer;
2.          by_ref best_x: array of integer;
3.          function ADMs (array of integer): integer);
4.  var
5.      x: array of integer;
6.      n, cost, best_cost, T, mv, step: integer;
7.      LastExecuted: array of moves;
8.  Initialization:
9.      generate a random assignment into x
10.     best_x $\leftarrow$ x;
11.     cost $\leftarrow$ ADMs(x);
12.     best_cost $\leftarrow$ cost;
13.     T $\leftarrow$ 1;
14.     for each mv
15.         LastExecuted[mv] $\leftarrow -\infty$;
16. Improvement:
17.     for step in 1..max_moves do
18.         find the best move mv such that LastExecuted[mv] $<$ step - T
19.         modify x[i] according to mv
20.         cost $\leftarrow$ nADM(x);
21.         LastExecuted[mv] $\leftarrow$ step
22.         if cost $<$ best_cost then
23.             best_cost $\leftarrow$ cost
24.             best_x $\leftarrow$ x
25.         if x has been visited too often then
26.             increase T
27.         else if T hasn't been increased for some time
28.             decrease T
```

Figure 4: the RLS algorithm for the ADM minimization problem. The details about the reactive prohibition scheme determining T (lines 25–28) are given in the text.

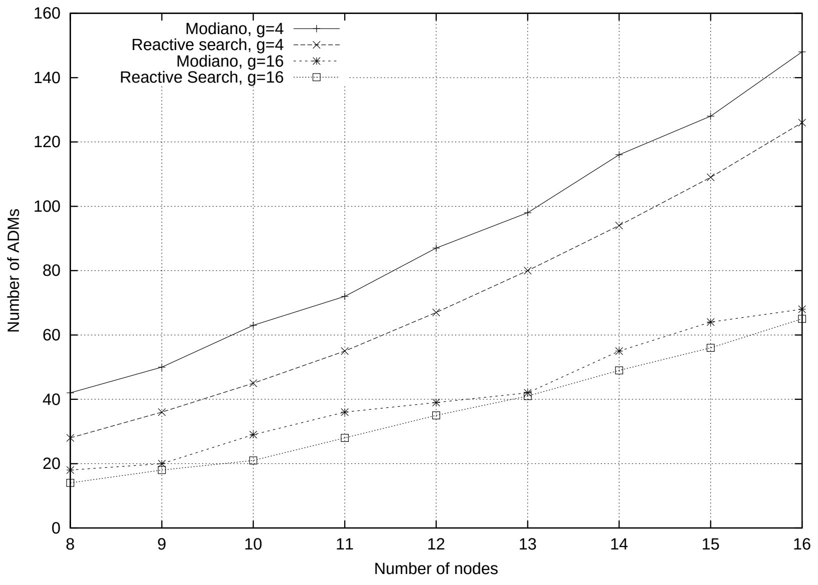

Figure 5: comparison between RLS and Modiano

5 Simulation Results

We tested the algorithm on the all-to-all uniform traffic case, where the traffic requirement is equal to 1 for all couples of nodes:

$$t_{ij} = \begin{cases} 1 & i \neq j \\ 0 & i = j \end{cases} \qquad \Rightarrow \qquad T = \begin{pmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 0 \end{pmatrix}.$$

Figure 5 shows a comparison between the RLS results (best value after 10 10^5-step runs) and Modiano's algorithm for all-to-all uniform traffic with one channel request per couple of nodes. We considered cases $g = 4$ and $g = 16$, as they are analitically studied in [5]. In both cases RLS results in a considerable reduction of the number of ADMs in the ring, up to 28% for $g = 4$ and up to 31% for $g = 16$.

In Fig. 6 we compare the best solution found after 10 RLS runs (already reported in Fig. 5) with the average value for the same set of runs of the algorithm. In order to distinguish the two lines we were forced to restrict the graph to the higher portion of tested values (12 to 16 nodes) and to a grooming factor equal to 16. In fact, most runs return the best found value, and only one or two in the total end with one or two more ADMs.

The RLS algorithm took about 8 minutes of CPU time per run on a 500MHz PentiumIII computer with 64 MB RAM running the Linux operating system. The size of the problem (the number of nodes) did not affect the execution time because we fixed to 1000 the number of neighboring configurations to check.

6 Conclusions

Experimental results obtained by simulations of the all-to-all uniform traffic case show that the proposed RLS technique is competitive with other greedy techniques used in the literature. Of course, local search heuristics need to explore a large set of solutions to find good local minima; this is acceptable, because circuit planning is an off-line operation, where a computation taking a few minutes is perfectly tolerable, in particular when it cuts down hardware costs in a significant way.

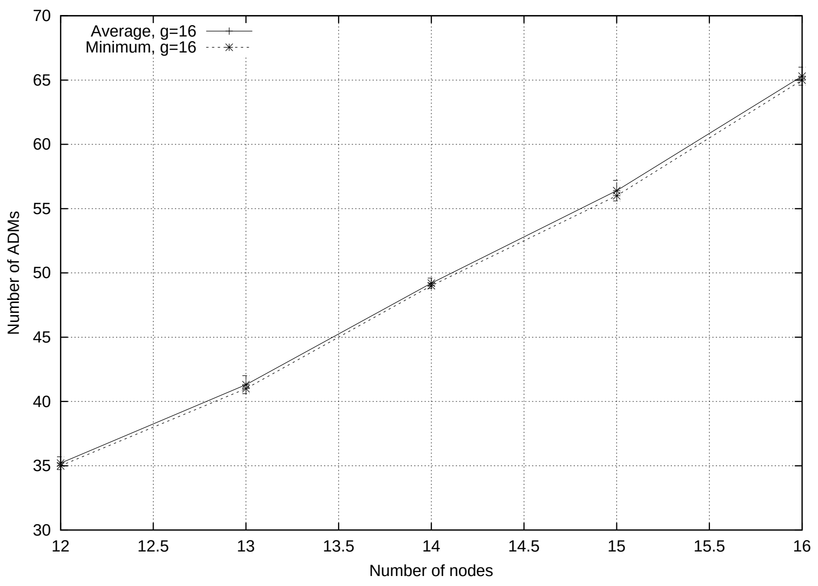

Figure 6: minimum, average and confidence interval for RLS, $g = 16$

References

[1] Roberto Battiti. Reactive search: Toward self-tuning heuristics. In V. J. Rayward-Smith, editor, *Modern Heuristic Search Methods*, chapter 4, pages 61–83. John Wiley and Sons Ltd, 1996.

[2] Roberto Battiti and Alan Albert Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Transactions on Computers*, 48(4):361–385, april 1999.

[3] Roberto Battiti and Marco Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29(4):610–637, April 2001.

[4] Randall Berry and Eytan H. Modiano. Reducing electronic multiplexing costs in SONET/WDM rings with dynamically changing traffic. *IEEE Journal on selected areas in communications*, 18(10):1961–1971, October 2000.

[5] Angela L. Chiu and Eytan H. Modiano. Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. *IEEE Journal of Lightwave Technology*, 18(1):2–12, January 2000.

[6] F. Glover. Tabu search — part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[7] P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44:279–303, 1990.

[8] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, February 1970.

[9] K. Steiglitz and P. Weiner. Some improved algorithms for computer solution of the traveling salesman problem. In *Proceedings of the Sixth Allerton Conf. on Circuit and System Theory*, pages 814–821, Urbana, Illinois, 1968.

[10] Peng-Jun Wan, Gruia Călinescu, Liwu Liu, and Ophir Frieder. Grooming of arbitrary traffic in SONET/WDM BLSRs. *IEEE Journal on Selected Areas in Communications*, 18(10):1995–2003, October 2000.