Solving the SONET Ring Assignment Problem using Integer Programming Models

Elder M. Macambira*[1] Nelson Maculan†[1] Cid C. de Souza‡[2]

[1] Universidade Federal do Rio de Janeiro
Programa de Engenharia de Sistemas e Computação, COPPE
Caixa Postal 68511, 21941-972
e-mail: {elder,maculan}@cos.ufrj.br

[2] Universidade Estadual de Campinas
Instituto de Computação
Caixa Postal 6176, 13084-971
e-mail: cid@ic.unicamp.br

Abstract

In this paper, we study the problem of assigning a set of customer sites using SONET rings of equal capacity. Each customer has to be assigned to exactly one ring and these rings have to be connected through a single federal ring. A capacity constraint on each ring is considered. The objective is to minimize the number of rings in the network subject to a ring capacity limit. This problem is called SONET Ring Assignment Problem (SRAP). The SRAP is known to be $\mathcal{NP}$-hard and it can be described formally as a node-partitioning problem.

We propose two new integer linear programming formulations for the SRAP. Comparisons with other formulations presented earlier in the literature are made. Computational results are discussed and empirical evidence indicates that our formulations perform significantly better than the existing ones. While previous formulations cannot be solved to optimality in a reasonable amount of maximum CPU time, our formulations are computed much faster and optimal solutions are obtained in minutes in a typical desktop machine.

Keywords: SONET Ring Assignment, Integer Programming Models, Telecommunications.

*The author is supported by CNPq (grant 146245/1999-7) and FAPERJ (grant E-26/152.603/2001).
†correspondig author.
‡The author is supported by CNPq (grant 300883/94-3).

1 Introduction

Due to the international increasing use of fiber-optic technology in telecommunications, mainly in the transmission and multiplexing standard for high speed signal, the telecommunications provider has adopted a new network design concept. One of the most popular designs is the Synchronous Optical Network (SONET) which is more widely used in North America.

The typical topology of a SONET network corresponds to a collection of local rings, or simply *rings*, directly connecting a subset of customer sites (see [2, 9, 10]). Each customer site sends, receives and relays messages through a device called Add-Drop Multiplexer (ADM). This requires that the ring must have a *capacity*. The capacity is limited and corresponds to the volume of traffic between all pairs of customer sites connected by the ring. Finally, the inter-ring relations are then possible. All rings are connected together by a special ring, called *federal ring*, through a device - the Digital Cross-connect System (DCS) - which joins each ring to the federal ring. The number of DCS's required for the design network corresponds to the cost-effective of the network.

Problem Definition In the design of real optical networks, a set of rings has been taken into account: maximum length of a ring, maximum number of customer sites in a ring and maximum traffic on a ring. In this paper, we will exclusively deal with the problem of forming such rings. This problem requires three conditions. The first one assumes that each customer site has to be assigned to exactly one ring; the second condition assumes that the maximum capacity of each ring is bounded by a common value; and finally, the third condition establishes that the number of rings, i.e., the cost of DCS's installed, has to be minimized. The problem is called SONET Ring Assignment Problem (SRAP) with equal capacity constraints.

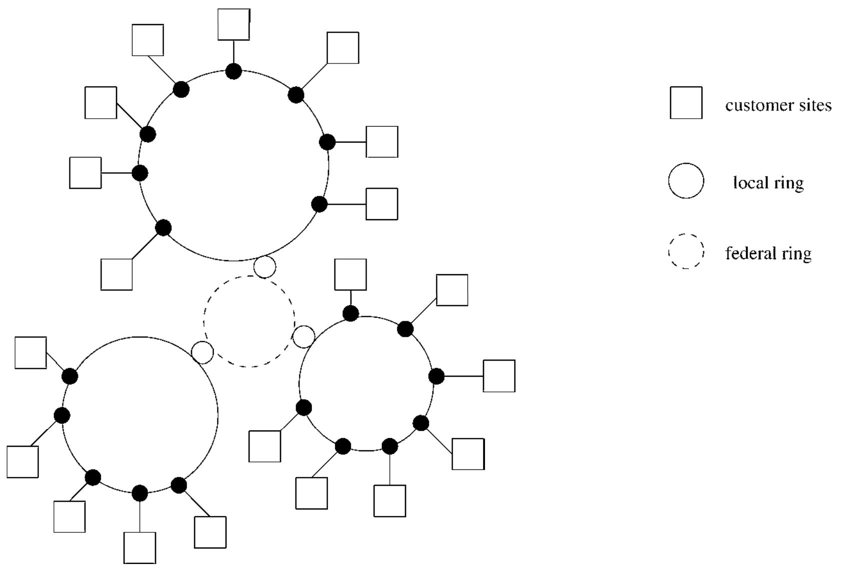

Figure 1: A SONET network.

The SONET Ring Assignment Problem can be described formally as a node-partitioning problem for a given graph G. The nodes of G represent the customer sites to be linked, and the edge weights correspond to the traffic demands between cutomer sites. The clusters of a solution to our problem are clusters of customer sites that will be placed on the same SONET ring.

Example An example of SONET network is given in Figure 1. In this example, we have one network with 20 nodes and 3 rings.

Previous Work We refer to [4] for a more detailed discussion on the architecture of SONET networks and a review of the literature on the SRAP. In their work the authors proved that SRAP is $\mathcal{NP}$-hard. They also introduce an integer programming formulation for the problem and propose heuristics and exact methods to solve it. Another recent work on heuristics for SRAP can be found in [1] where tabu and scatter search meta-heuristics are proposed.

Contributions In [4], Goldschmidt *et al.* pointed out to the inadequacy of solving SRAP instances using their integer programming formulation and commercial linear programming solvers. In this paper we are interested to overcome these drawbacks. With this objective, we propose some variants of IP formulations for SRAP and tested the performance of standard branch-and-bound algorithms implemented in a commercial code when computing them. The results are significantly better than those reported earlier in the literature.

Outline The paper is organized as follows. In Section 2, we present the original model developed for the SRAP. In Section 3, we present two integer programming formulations and explore properties of these formulations, for example these formulations are based in packing problem. In Section 4, we present our computational results for randomly-generated instances and discuss empirical results to support the claim that these models may be adequate in solving SRAP instances of moderate size to optimality. Finally, concluding remarks are given in Section 5.

2 Mathematical Models Review

It is by now well-known that the choice of a "good" model may crucially affect our ability to solve large instances of integer programming problems. For a compreensive discussion of this point, see Nemhauser and Wolsey [8], and Wolsey [11]. In this section, we present an integer linear programming formulation for the SRAP. This formulation was proposed by Goldschmidt *et al.* [4].

In the following discussion, we describe SRAP as a graph optimization. We describe it in terms of the network design application presented in Section 1. Next, we describe the SONET Ring Assignment Problem as an integer linear programming problem.

The SRAP can be described formally as the following graph optimization problem: given an undirected graph $G = (V, E)$ with nonnegative edge weight d_{uv} associated for each $e = (u, v) \in E$, an integer B, and an integer $l \leq n = |V|$, partition the set of nodes of G at most l disjoint clusters, e.g., $V_1, V_2, \ldots, V_l$, so that l is minimized and the constraints about capacity of the clusters

$$\sum_{\substack{u,v \in V_i \\ u<v}} d_{uv} + \sum_{\substack{u \in V_i \\ v \notin V_i}} d_{uv} \leq B, \qquad i = 1, \ldots, l, \tag{1}$$

$$\sum_{i=1}^{l-1} \sum_{j=i+1}^{l} \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B, \tag{2}$$

may be satisfied. Note that each cluster V_k, with $k \in \{1, \ldots, l\}$, corresponds to a ring.

Now we present an integer linear programming for the SRAP. It has for groups of binary variables, namely x, y, p and z. In the first group, we have n^2 variables x_{ui} one for each pair $u \in V$ and $i \in V$. The value of x_{ui} is one if and only of vertex u is assigned to ring i. In the second group, there are n variables each representing a possible local ring to be opened in a feasible solution. We say that a ring is open if there is at least one vertex assigned to it. Therefore, the variable y_i is one if and only if the i-th ring is opened. For every edge (u, v) in E, with $u < v$, and $i \in V$, p_{uvi} is one if and only if both vertices u and v are assigned to ring i. The third group of variables has mn elements. Finally, in the fourth group we have $2mn$ z variables. Each z variable is indexed by three elements, the first two taken from $V \times V \setminus \{(u, v) | u \in V\}$ and the last one from V. The variable z_{uvi} is then defined to be one if and only if u is in ring i while v is not. A possible formulation for SRAP with these variables is given below.

(P1)

$$\text{minimize} \sum_{i=1}^{n} y_i, \tag{3}$$

subject to:

$$\sum_{u=1}^{n-1} \sum_{v=u+1}^{n} d_{uv} p_{uvi} + \sum_{u=1}^{n} \sum_{v=1}^{n} d_{uv} z_{uvi} \leq B, \qquad \forall i \in V, \tag{4}$$

$$\sum_{u=1}^{n-1} \sum_{v=u+1}^{n} \sum_{i=1}^{n} d_{uv} z_{uvi} \leq B, \tag{5}$$

$$\sum_{i=1}^{n} x_{ui} = 1, \qquad \forall u \in V, \tag{6}$$

$$x_{ui} - y_i \leq 0, \qquad \forall u, i \in V, \tag{7}$$

$$x_{ui} + x_{vi} - p_{uvi} \leq 1, \qquad \forall u, v \in V, u < v,\ \forall i \in V, \tag{8}$$

$$x_{ui} - x_{vi} - z_{uvi} \leq 0, \qquad \forall u, v \in V,\ \forall i \in V, \tag{9}$$

$$x_{ui} \in \{0, 1\}, \qquad \forall u \in V,\ \forall i \in V, \tag{10}$$

$$y_i \in \{0, 1\}, \qquad \forall i \in V, \tag{11}$$

$$p_{uvi} \in \{0, 1\}, \qquad \forall u, v \in V, u < v,\ \forall i \in V, \tag{12}$$

$$z_{uvi} \in \{0, 1\}, \qquad \forall u, v \in V,\ \forall i \in V. \tag{13}$$

Let us briefly examine the constraints in formulation (P1). The objective function (3) minimizes the number of rings. Constraints (4) limit the total demand of each ring to the bandwidth B. Constraint (5) assures the capacity of the federal ring to be less than or equal to B. Remember that the federal ring needs to carry all the traffic among customer sites on different rings. Constraints (6) assure that each customer site is assigned to exactly one ring. Constraints (7) impose that one ring is active if a customer site is placed on it. For a proper triple (u, v, i), constraint (8) forces p_{uvi} to one if both vertex u and vertex v are assigned to ring i while constraint (9) forces z_{uvi} to one if vertex u is assigned to ring i and v is not assigned to that ring. Constraints (10)-(13) observe the integrality conditions on the variables x_{ui}, y_i, p_{uvi} and z_{uvi}, respectively.

The mindful reader has certainly noticed that equations (8)-(9) are not enough to express the inteded relationship between x and p (z) variables. However, as observed by Goldschmidt *et al.* [4], given a proper triple (u, v, i) and a feasible solution to (P1) if $p_{uvi} = 1$ and either $x_{ui} = 0$ or $x_{vi} = 0$, another feasible solution with the same cost is obtained by simply setting p_{uvi} to zero. An analogous reasoning applies if $z_{uvi} = 1$ and either $x_{ui} = 0$ or $x_{vi} = 1$. Therefore, we can drop from the SRAP formulation the constraints that force the p or z variables to zero.

The efficiency of most models is dependent on their ability to solve instances within an average computing time. We are now going to evaluate the formulation (P1) described on the basis of this criterion. Goldschmidt *et al.* [4] made computational experiments using some instances of moderate size. The results indicated that CPLEX took within a maximum computing of 46,000 CPU seconds to solve the instances. It is worth noting that this formulation is not useful for solving the SRAP with a commercial and general purpose code (e.g., CPLEX), due to the enormous computing times.

3 New Integer Programming Formulations

In this section, we propose two integer programming formulations for the SRAP (see Macambira [6]). Furthermore, we show that these formulations correctly describe the SRAP.

3.1 First IP Formulation

Goldschmidt *et al.* [4] observed in their paper that formulation (P1) is not suitable to identify SRAP instances which are infeasible. A possible way to overcome this disadvantage is to relax the previous model by replacing constraints (6) by constraints of the form

$$\sum_{i=1}^{n} x_{ui} \leq 1, \quad \forall u \in V. \tag{14}$$

However by doing that we also have to change the objective function so that it favors the feasible solutions that are actual partitions of the vertex set if they exist. This technique is not a novelty in the solution of hard combinatorial problems. Examples of its application can be found for the generalized assignment problem (see Martello and Toth [7]) or traveling salesman problem (see Grötschel and Padberg [5]). Let us briefly describe this change.

This replacing is possible by adding one artificial variable for each constraint (6). Let $t_u = 1 - \sum_{i=1}^{n} x_{ui}$ be this artificial variable and $\Theta_u > 0$ the cost associated to every variable t_u with $u \in V$. Hence, the objective function (3) can be written as:

$$\overline{z} = \min \sum_{i=1}^{n} y_i + \sum_{u=1}^{n} \Theta_u t_u. \tag{15}$$

Note that the cost associated to the variable x_{ui} is equal to zero in the objective function (3). Thus, the new objective function, namely (15), is penalizing the vectors x. So, the constraint (14) is satisfied as one inequality strict, e.g., $\sum_{i=1}^{n} x_{ui} = 1$ for each $u \in V$.

By eliminating the artificial variables t_u in (15), we obtain the following objective function:

$$\begin{aligned} \overline{z} &= \min \sum_{i=1}^{n} y_i + \sum_{u=1}^{n} \Theta_u t_u = \\ &= \min \sum_{i=1}^{n} y_i + \sum_{u=1}^{n} \Theta_u (1 - \sum_{i=1}^{n} x_{ui}) = \\ &= \min \sum_{i=1}^{n} y_i - \sum_{u=1}^{n} \sum_{i=1}^{n} \Theta_u x_{ui} + \sum_{u=1}^{n} \Theta_u, \end{aligned} \tag{16}$$

defined just in terms of variables x_{ui} and y_i.

Now, we will transform the objective function (16) into the maximization problem:

$$\overline{z} = \max \sum_{u=1}^{n} \sum_{i=1}^{n} \Theta_u x_{ui} - \sum_{i=1}^{n} y_i - \sum_{u=1}^{n} \Theta_u. \tag{17}$$

In this way, an equivalent objective function of the problem is obtained.

If the constant Θ_u represeting the vertex assignment costs are suitably chosen, any solution corresponding to a vertex partition has a larger objective value than a pure packing solution where one of the vertex remains unassigned. It is a easy test to check that $\Theta_u = n{+}1$ fulfills this property. This is the value used for Θ_u in our computations.

In the sequel, we shall present the first IP formulation proposed in this work. As before, we consider the variables x_{ui}, y_i, p_{uvi} and z_{uvi} defined in the formulation (P1). Thus, the SONET Ring Assignment Problem can be stated as the 0-1 integer linear programming problem:

(P2)

$$\begin{aligned}
&\text{maximize } \sum_{u=1}^{n}\sum_{i=1}^{n}\Theta_u x_{ui} - \sum_{i=1}^{n} y_i - \sum_{u=1}^{n}\Theta_u,\\
&\text{subject to:}\\
&\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv}p_{uvi} + \sum_{u=1}^{n}\sum_{v=1}^{n} d_{uv}z_{uvi} \leq B, && \forall i \in V,\\
&\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sum_{i=1}^{n} d_{uv}z_{uvi} \leq B,\\
&\sum_{i=1}^{n} x_{ui} \leq 1, && \forall u \in V,\\
&x_{ui} - y_i \leq 0, && \forall u, i \in V,\\
&x_{ui} + x_{vi} - p_{uvi} \leq 1, && \forall u, v \in V, u < v,\ \forall i \in V,\\
&x_{ui} - x_{vi} - z_{uvi} \leq 0, && \forall u, v \in V,\ \forall i \in V\\
&x_{ui} \in \{0,1\}, && \forall u \in V,\ \forall i \in V,\\
&y_i \in \{0,1\}, && \forall i \in V,\\
&p_{uvi} \in \{0,1\}, && \forall u, v \in V, u < v,\ \forall i \in V,\\
&z_{uvi} \in \{0,1\}, && \forall u, v \in V,\ \forall i \in V.
\end{aligned}$$

Inspecting this formulation, we observe that it requires the same number of binary variables and constraints of the formulation (P1), e.g., $n^2 + 3mn + n$ binary variables and $n^2 + 3mn + 2n + 1$ constraints, where n is the number of customer sites and m is the number of the pair of customer sites which communicate.

Note that if $\Theta_u = 1$ the integer solution obtained by formulation (P2) corresponds to a feasible partition of V, e.g., the rings $V_1, \ldots, V_l$, that satisfies the capacity constraints. So, in this case, we can deduce that any optimal solution of the formulation (P2) is optimal solution of the formulation (P1).

In the sequel, we shall present the second IP formulation for SRAP.

3.2 Second IP Formulation

Next we derive a new model for SRAP based on a slightly different set of variables. The purpose is to reduce the number of variables with respect to the previous formulations while keeping the same number of constraints. We hope by doing so, we end up with a model which is easier to compute. The second IP formulation also features a packing problem.

In order to describe this alternative formulation, consider a feasible solution for SRAP where the vertices of V are partitioned into sets $V_1, \ldots, V_l$. For every $j \in [1, \ldots, l]$, the internal and external demands of ring j are respectively D_j and W_j, where

$$D_j = \sum_{u \in V_j} \sum_{\substack{v \in V_j \\ u<v}} d_{uv}, \text{ for each } j = 1, \ldots, l, \tag{18}$$

$$W_j = \sum_{u \in V_j} \sum_{\substack{v \notin V_j \\ u<v}} d_{uv}. \tag{19}$$

Let us denote he total demand by D and the total demand of ring j by d_j, i.e.,

$$D = \sum_{(u,v) \in E} d_{uv}, \tag{20}$$

$$d_j = D_j + W_j. \tag{21}$$

According to those definitions, the demand flowing throug the federal ring is computed as

$$\overline{D} = D - \sum_{j=1}^{l} D_j, \tag{22}$$

and we have that

$$2 \times \overline{D} = \sum_{j=1}^{k} W_j. \tag{23}$$

Finally, the capacity constraint on the federal ring implies that

$$\begin{aligned} &\overline{D} \leq B \Longrightarrow D + \overline{D} \leq D + B \Longrightarrow \\ &\overset{(22)}{\Longrightarrow} \overline{D} + \overline{D} + \sum_{j=1}^{l} D_j \leq D + B \Longrightarrow \\ &\overset{(23)}{\Longrightarrow} \sum_{j=1}^{l} W_j + \sum_{j=1}^{l} D_j \leq D + B. \end{aligned} \tag{24}$$

Thus, equation (24) gives us another way to express the capacity constraint for the federalring. It also suggests that we can define new binary variables for the edges in G. For all (u, v) in E with $u < v$ and $i \in V$, f_{uvi} is set zero if and only if both vertices u and v are not assigned to ring i. In other words, for a fixed ring i, the edges with $f_{uvi} = 1$ are those which are internal to ring i or link vertices of ring i to vertices at some other ring.

To ensure the correct relationship among x and f, the following constraints are added to the new IP formulation for SRAP:

$$\begin{aligned} f_{uvi} - x_{ui} &\geq 0, \quad \forall u, v \in V, u < v, \ \forall i \in V, & (25)\\ f_{uvi} - x_{vi} &\geq 0, \quad \forall u, v \in V, u < v, \ \forall i \in V, & (26)\\ f_{uvi} - x_{ui} - x_{vi} &\leq 0, \quad \forall u, v \in V, u < v, \ \forall i \in V. & (27) \end{aligned}$$

The constraints (25) enforce that if the customer site (or node) u belongs to the ring i, e.g., $x_{ui} = 1$, so the variable f_{uvi} is equal to 1. The same sense can be applied with relation the constraints (26); however, it is took the customer site v. Constraints (27) assure that if both customer sites u and v do not belong to the ring i, e.g., $x_{ui} = x_{vi} = 0$, then f_{uvi} is equal to 0.

With the variables x and y defined as before, the new IP formulation for SRAP reads as follows:

(P3)

$$\text{maximize} \sum_{u=1}^{n}\sum_{i=1}^{n} \Theta_u x_{ui} - \sum_{i=1}^{n} y_i - \sum_{u=1}^{n} \Theta_u,$$

subject to:

$$\begin{aligned} &\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv} f_{uvi} \leq B, && \forall i \in V, & (28)\\ &\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sum_{i=1}^{n} d_{uv} f_{uvi} \leq D + B, && & (29)\\ &\sum_{i=1}^{n} x_{ui} \leq 1, && \forall u \in V, &\\ &x_{ui} - y_i \leq 0, && \forall u \in V, i \in V, &\\ &f_{uvi} - x_{ui} \geq 0, && \forall u, v \in V, u < v, \ \forall i \in V, &\\ &f_{uvi} - x_{vi} \geq 0, && \forall u, v \in V, u < v, \ \forall i \in V, &\\ &f_{uvi} - x_{ui} - x_{vi} \leq 0, && \forall u, v \in V, u < v, \ \forall i \in V, &\\ &x_{ui} \in \{0,1\}, && \forall u \in V, \ \forall i \in V, &\\ &y_i \in \{0,1\}, && \forall i \in V, &\\ &f_{uvi} \in \{0,1\}, && \forall u, v \in V, u < v, \ \forall i \in V. & (30) \end{aligned}$$

Let us briefly discuss the model. The objective function is the same as that for the (P2) model, and as discussed earlier, if Θ_u taken to be $n + 1$, every partition, if there is one, has a more attractive cost than any packing of the vertex set. The first set of inequalities, constraints (28), assures that the capacity of each ring is equal to B. The second set of inequalities, constraint (29), refers to the capacity limitation of federal ring. Finally, the constraints (30) observe the integrality conditions on the variables f_{uvi}. Thus, an optimal solution to this 0-1 integer linear programming corresponds to the optimal feasible partition of V.

Note that the second IP formulation requires $n^2 + mn + n$ binary variables and $n^2 + 3mn + 2n + 1$ constraints. So, this formulation is more compact than first IP formulation that requires $n^2 + 3mn + n$ binary variables.

4 Computational Results

In this section, we show some empirical results obtained from computational experiments. Our goal is to confirm that the formulations proposed here are useful when optimizing the instances of the SRAP, e.g., suppose one is given some instance of the problem and a linear and integer program solver, what would be the most promising formulation to choose: (P1), (P2) or (P3)? In order to answer this question, we established two criteria: number of branch nodes in the enumeration tree and CPU time.

Our computational experiments were performed on a Linux PC AMD K6/450 MHz with 256 Mbytes of main memory. Initially, we describe two classes of instances used in the tests. Next, we present the results obtained by formulations.

4.1 Test Instances

We used the following classes of instances, namely, classes C1 and C2, for our experiments.

Class C1

The class of instances C1 was generated by Goldschmidt *et al.* [4]. Their instances were divided into two types of random test data:

- geometric instances representing *natural cluster*, i.e., customer sites try to communicate more with close neighbors than with distant ones;
- random instances were generated from complete graphs and retaining edge (u, v) with probability p, e.g., each edge of the complete graph exists in the random instance if $p \in (0, 1)$.

Each type of random test data contains both high and low demand graphs, namely the *high-demand* cases, where $622Mbs$ ADM are being considered, and the *low-demand* cases, where the ring capacity is $155Mbs$.

We have tested 23 high-demand and 24 low-demand instances. Finally, the dimension of the instances is 15 and 25 customer sites (or nodes). The distribution of these instances among geometric and random, high and low-demands, and values of n, are reported in Table 1.

Type	Capacity (B)	Cardinality ($\|V\|$)	
		15	25
geometric	$155Mbs$	4	5
geometric	$622Mbs$	4	7
random	$155Mbs$	7	8
random	$622Mbs$	7	5

Table 1: Distribution of the instances tested in class C1.

Class C2

This class of instances was generated by Aringhieri and Dell'Amico [1]. The authors generated *harder* instances with respect to those belonging to class C1. Aringhieri and Dell'Amico defined one instance as *hard* if the greedy algorithms, proposed by Goldschmidt *et al.* [4], are not able to find the optimal solution, within a reasonable computing time.

We tried to solve 10 high-demand and 20 low-demand instances. The dimension of the instances is equal to the instances belonging to class C1, e.g., 15 and 25 customer sites. Table 2 presents the distribution of these instances.

Type	Capacity (B)	Cardinality ($\|V\|$)	
		15	25
geometric	$155Mbs$	5	5
geometric	$622Mbs$	0	5
random	$155Mbs$	6	4
random	$622Mbs$	5	0

Table 2: Distribution of the instances tested in class C2.

4.2 Comparing Formulations

In this subsection, we present and discuss the results obtained from formulations (P1), (P2) and (P3) for the classes of instances presented above.

For comparison, the instances have been solved by XPRESS version 12.05 (see [3]) within a maximum computing time equal to 18,000 seconds. We have also established that the maximum number of nodes for the branch-and-bound procedure has been set to 300,000 nodes. Furthermore, we have defined Θ_u as being equal to $n+1$ for each customer site $u \in V$.

4.2.1 Class C1

Tables 3-8 below contain the results for these instances obtained with formulation (P1) and the formulations proposed in this work, namely (P2) and (P3). A total of 47 instances were used in the study. The results are divided with respect to the type, e.g., geometric and random instances.

Those tables have the following columns: **name**: the instance tested; z^*: the optimal value of the instance; **#nodes**: total number of nodes explored in the enumeration tree; z: value of lower bound reached by formulation; **gap (%)**: relative gap between the optimal value and the lower bound, that is

$$gap\ (\%) = \left(\frac{z^* - z}{z^*}\right) \times 100; \tag{31}$$

time (sec.): time in CPU seconds required by XPRESS to find the optimal (or lower bound) solution for the instance tested using one formulation.

Furthermore, whenever "–" appears for gap entries, the formulation has reached the optimal solution, for the corresponding instance, but the LP optimality was not proved, due to the maximum CPU time.

Geometric instances

As one should notice from Tables 3-5, optimality was proved for 16 out of the 20 instances solved by formulation (P3). It should also be noticed that the formulation (P2) has proven optimality in 16 out of 20 instances, whereas the formulation (P1) has found optimal solution for only 9 instances.

Furthermore, in relation to the results presented in Tables 3-5, for the remaining 4 instances which are left unsolved (to proven optimality), gaps between the best lower bound and optimal value have been closed by at most 50% for the formulation (P2), in just 4 instances, whereas for the formulation (P3) optimal solution has been reached, but LP optimality was not proved. Finally, the gap has been closed by at least 33.33% and at most 50% for the formulation (P1). As it can be appreciated, the gap is very weak when we used the formulation (P1). This shows that our formulations may provide a tight lower bound on the optimum integer solution.

It should also be noticed that the number of branch nodes is typically fairly small, in most instances tested, when using formulations (P2) and (P3). Finally, we also observe that the CPU time for the formulations (P2) and (P3) run at least 1.22 and 11.82 times, respectively, small than the CPU time obtained by formulation (P1).

Random instances

Results were reported in Tables 6-8. Across 27 instances tested, the formulation (P3) found the optimal solution in 21 instances. It is interesting to observe that the formulation (P2) proved the optimality for 20 out of 27 instances. Furthermore, the formulation (P1) found the optimal solution in 15 instances tested. Once again, as before, this confirms the effectiveness of the formulations proposed here.

It should also be noticed that for the remaining instances that were left unsolved, the formulation (P3) reached the optimal solution. However, optimality was not proved, due to maximum CPU time. The relative gap obtained by formulation (P2) have been closed by at least 33.33% (in just one instance) and by at most 50% (in 6 instances). Finally, the gap for the formulation (P1) have been closed by at least 33.33% (in 5 instances) and by at most 50% (in 7 instances).

As can be appreciated from the results, the number of branch nodes in the enumeration tree was small when the formulation (P3) was used to solve those instances. Moreover, one should also notice that the CPU times required to prove optimality have been modest for the formulation (P1), while the solution times for the formulations (P2) and (P3) are significantly better. In particular, for most of the instances tested, the overall CPU time spent by formulation (P2) dominated at least 1.01 the CPU time required by formulation (P1), whereas the CPU time required by formulation (P3) was better than the CPU time required by formulation (P1) at least 2 times.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gl_15_1	3	7208	3	0	851
gl_15_4	3	7317	3	0	823
gl_15_7	3	7450	3	0	800
gl_15_9	3	300000	2	33.33	7414
gl_25_1	4	16100	2	50	18000
gl_25_3	3	32600	2	33.33	18000
gl_25_4	4	15300	2	50	18000
gl_25_7	3	16600	2	33.33	18000
gl_25_8	4	13200	2	50	18000
$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gh_15_1	3	46806	3	0	2847
gh_15_2	3	27695	3	0	8888
gh_15_8	3	7047	3	0	3053
gh_15_9	3	20324	3	0	3096
gh_25_1	3	8600	2	33.33	18000
gh_25_2	2	3261	2	0	1808
gh_25_3	2	2162	2	0	962
gh_25_4	3	7000	2	33.33	18000
gh_25_7	4	5200	2	50	18000
gh_25_8	3	7700	2	33.33	18000
gh_25_9	3	7000	2	33.33	18000

Table 3: Results obtained by formulation P1 for class C1: geometric instances.

4.2.2 Class C2

Tables 9-14 reported the results obtained by some instances belonging to class C2. A total of 30 test instances were used in the study. As before, we compared the performance of the

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gl_15_1	3	4873	3	0	654
gl_15_4	3	3994	3	0	665
gl_15_7	3	9838	3	0	1880
gl_15_9	3	16030	3	0	1875
gl_25_1	4	16500	2	50	18000
gl_25_3	3	12568	3	0	11289
gl_25_4	4	10200	2	50	18000
gl_25_7	3	12612	3	0	14755
gl_25_8	4	12400	2	50	18000

$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gh_15_1	3	11954	3	0	2222
gh_15_2	3	7424	3	0	2296
gh_15_8	3	3645	3	0	1731
gh_15_9	3	12357	3	0	2599
gh_25_1	3	3202	3	0	2481
gh_25_2	2	923	2	0	451
gh_25_3	2	1233	2	0	689
gh_25_4	3	8466	3	0	2942
gh_25_7	4	7100	2	50	18000
gh_25_8	3	2781	3	0	4325
gh_25_9	3	7420	3	0	10261

Table 4: Results obtained by formulation P2 for class C1: geometric instances.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gl_15_1	3	1870	3	0	72
gl_15_4	3	1544	3	0	32
gl_15_7	3	1405	3	0	46
gl_15_9	3	5499	3	0	69
gl_25_1	4	34300	4	–	18000
gl_25_3	3	3772	3	0	1014
gl_25_4	4	29500	4	–	18000
gl_25_7	3	2265	3	0	576
gl_25_8	4	42100	4	–	18000

$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gh_15_1	3	8476	3	0	156
gh_15_2	3	2259	3	0	106
gh_15_8	3	2004	3	0	171
gh_15_9	3	1758	3	0	68
gh_25_1	3	2283	3	0	723
gh_25_2	2	376	2	0	52
gh_25_3	2	288	2	0	39
gh_25_4	3	2583	3	0	989
gh_25_7	4	34300	4	–	18000
gh_25_8	3	3639	3	0	1327
gh_25_9	3	1836	3	0	780

Table 5: Results obtained by formulation P3 for class C1: geometric instances.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rl_15_1	3	11671	3	0	1136
rl_15_3	2	170	2	0	2
rl_15_4	3	5449	3	0	992
rl_15_6	3	14729	3	0	1245
rl_15_8	3	19342	3	0	2343
rl_15_9	3	24793	3	0	1750
rl_15_10	3	11663	3	0	2306
rl_25_2	3	19100	2	33.33	18000
rl_25_4	4	15100	2	50	18000
rl_25_5	4	20700	2	50	18000
rl_25_6	4	14800	2	50	18000
rl_25_7	4	15900	2	50	18000
rl_25_8	3	16900	2	33.33	18000
rl_25_9	4	12800	2	50	18000
rl_25_10	4	19100	2	50	18000
$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rh_15_1	3	20188	3	0	4123
rh_15_4	2	108	2	0	3
rh_15_5	3	29568	3	0	4349
rh_15_6	3	37361	3	0	4335
rh_15_7	2	134	2	0	4
rh_15_8	2	1166	2	0	26
rh_15_9	3	32864	3	0	4140
rh_25_2	3	8100	2	33.33	18000
rh_25_3	3	9300	2	33.33	18000
rh_25_7	3	9600	2	33.33	18000
rh_25_9	2	374	2	0	96
rh_25_10	3	10100	2	33.33	18000

Table 6: Results obtained by formulation P1 for class C1: random instances.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rl_15_1	3	9118	3	0	1066
rl_15_3	2	168	2	0	3
rl_15_4	3	5752	3	0	1201
rl_15_6	3	12548	3	0	1816
rl_15_8	3	9591	3	0	1775
rl_15_9	3	15270	3	0	1730
rl_15_10	3	8853	3	0	1825
rl_25_2	3	4869	3	0	4037
rl_25_4	4	9200	2	50	18000
rl_25_5	4	9100	2	50	18000
rl_25_6	4	8200	2	50	18000
rl_25_7	4	7200	2	50	18000
rl_25_8	3	4044	3	0	2136
rl_25_9	4	23100	2	50	18000
rl_25_10	4	12600	2	50	18000

$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rh_15_1	3	4361	3	0	1450
rh_15_4	2	85	2	0	6
rh_15_5	3	9039	3	0	2378
rh_15_6	3	16800	3	0	3827
rh_15_7	2	187	2	0	10
rh_15_8	2	546	2	0	34
rh_15_9	3	18319	3	0	3629
rh_25_2	3	15600	2	33.33	18000
rh_25_3	3	2354	3	0	4006
rh_25_7	3	3103	3	0	4588
rh_25_9	2	353	2	0	77
rh_25_10	3	14037	3	0	13870

Table 7: Results obtained by formulation P2 for class C1: random instances.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rl_15_1	3	1917	3	0	24
rl_15_3	2	53	2	0	1
rl_15_4	3	1990	3	0	53
rl_15_6	3	4503	3	0	59
rl_15_8	3	4246	3	0	63
rl_15_9	3	3084	3	0	37
rl_15_10	3	1965	3	0	64
rl_25_2	3	4776	3	0	848
rl_25_4	4	50100	4	–	18000
rl_25_5	4	66600	4	–	18000
rl_25_6	4	52400	4	–	18000
rl_25_7	4	52600	4	–	18000
rl_25_8	3	14363	3	0	1450
rl_25_9	4	49400	4	–	18000
rl_25_10	4	55500	4	–	18000

$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rh_15_1	3	4417	3	0	114
rh_15_4	2	52	2	0	1
rh_15_5	3	9093	3	0	203
rh_15_6	3	6648	3	0	119
rh_15_7	2	143	2	0	1
rh_15_8	2	660	2	0	8
rh_15_9	3	7724	3	0	140
rh_25_2	3	178513	3	0	12796
rh_25_3	3	214700	3	–	18000
rh_25_7	3	33740	3	0	3867
rh_25_9	2	93	2	0	9
rh_25_10	3	160709	3	0	15650

Table 8: Results obtained by formulation P3 for class C1: random instances.

formulations (P1), (P2) and (P3) with relation to two criteria: the number of branch nodes and CPU time.

Geometric instances

For these instances, the results are shown in Tables 9-11. A total of 15 instances were tested. As one should notice from Tables 9-10, optimality was not proved for all the instances tested here, whenever we used the formulation (P1) and (P2). The instances were not solved to optimality within maximum CPU time, e.g .18,000 seconds. However, as one should also notice from Table 11, optimality was proved for 5 out of the 15 instances solved by formulation (P3). Furthermore, in relation to the results in this table, for the remaining 10 instances that were left unsolved (to proven optimality), formulation (P3) reached the optimal solution, but the LP optimality is not proved due to maximum CPU time.

A direct comparison of the results in Table 9 with those in Tables 10-11 appear very conclusive with relation to our formulations. Results show that the number of nodes in the enumeration tree is small when using formulations (P2) and (P3). At this point, the results indicate that our formulations seem to be more attractive than formulation (P1).

Random instances

Results were presented in Tables 12-14. For the 15 instances, note that formulation (P1) obtained the optimal solution in 8 instances, whereas formulations (P2) and (P3) solved to optimality 7 and 10 instances, respectively. On the other hand, in most instances, where the optimality was proved, the number of branch nodes in the enumeration tree required by our formulations, (P2) and (P3), was significantly smaller than the number of nodes required by formulation (P1).

Although formulation (P1) found the optimal solution value in half of the instances tested, we noticed that formulations (P2) and (P3) obtained a small relative gap in most instances. Among those instances, just one has been closed by at most 50% for formulation (P2), and just five instances the formulation (P3) reached the optimal solution, but the LP optimality is not proved. So, these tighter formulations allow us to substantially reduce the CPU times required by XPRESS to prove optimality.

5 Conclusion

In this work, we have briefly surveyed some of the previous attempts to solve a problem that comes up in network design. It is called SONET Ring Assignment Problem. We have argued that these attempts usually give rise to IP formulations which provide weak bounds on the optimal solution value of the SRAP.

We have proposed two new alternative IP formulations, namely, (P2) and (P3). These IP formulations have been tested through extensive computational experiments, using benchmark instances from the literature. Regarding computational results, some examples have

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gl_15_3.1	4	65800	3	25	18000
gl_15_3.2	4	59400	3	25	18000
gl_15_3.3	4	69800	3	25	18000
gl_15_3.4	4	58100	3	25	18000
gl_15_3.5	4	53400	3	25	18000
gl_25_9.1	4	14300	2	50	18000
gl_25_9.2	4	18400	2	50	18000
gl_25_9.4	4	10800	2	50	18000
gl_25_10.4	5	4200	2	60	18000
gl_25_10.5	5	3900	2	60	18000
$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gh_25_5.2	4	6100	2	50	18000
gh_25_5.3	4	2600	2	50	18000
gh_25_5.4	4	1800	2	50	18000
gh_25_5.5	4	2600	2	50	18000
gh_25_5.6	4	8000	2	50	18000

Table 9: Results obtained by formulation P1 for class C2: geometric instances.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gl_15_3.1	4	40000	3	25	18000
gl_15_3.2	4	37500	3	25	18000
gl_15_3.3	4	37200	3	25	18000
gl_15_3.4	4	33500	3	25	18000
gl_15_3.5	4	37500	3	25	18000
gl_25_9.1	4	16600	2	50	18000
gl_25_9.2	4	14800	2	50	18000
gl_25_9.4	4	10500	2	50	18000
gl_25_10.4	5	20300	2	60	18000
gl_25_10.5	5	26300	2	60	18000
$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gh_25_5.2	4	4800	2	50	18000
gh_25_5.3	4	3200	2	50	18000
gh_25_5.4	4	4400	2	50	18000
gh_25_5.5	4	4200	2	50	18000
gh_25_5.6	4	2900	2	50	18000

Table 10: Results obtained by formulation P2 for class C2: geometric instances.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gl_15_3.1	4	221627	4	0	5882
gl_15_3.2	4	218671	4	0	5924
gl_15_3.3	4	273976	4	0	7355
gl_15_3.4	4	232905	4	0	6309
gl_15_3.5	4	216932	4	0	6023
gl_25_9.1	4	49100	4	–	18000
gl_25_9.2	4	60700	4	–	18000
gl_25_9.4	4	60600	4	–	18000
gl_25_10.4	5	29400	5	–	18000
gl_25_10.5	5	29500	5	–	18000
$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
gh_25_5.2	4	27000	4	–	18000
gh_25_5.3	4	29600	4	–	18000
gh_25_5.4	4	38400	4	–	18000
gh_25_5.5	4	31000	4	–	18000
gh_25_5.6	4	34800	4	–	18000

Table 11: Results obtained by formulation P3 for class C2: geometric instances.

$B = 155Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rl_15_2.1	3	10286	3	0	2143
rl_15_2.2	4	56300	3	25	18000
rl_15_2.3	3	28293	3	0	4513
rl_15_5.1	3	12619	3	0	2351
rl_15_5.2	3	7329	3	0	2128
rl_15_5.3	3	5782	3	0	1822
rl_25_3.1	4	17700	2	50	18000
rl_25_3.2	4	15500	2	50	18000
rl_25_3.3	4	14300	2	50	18000
rl_25_3.5	4	17900	2	50	18000
$B = 622Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rh_15_10.1	3	11190	3	0	6188
rh_15_10.2	3	17964	3	0	11246
rh_15_10.4	3	30100	2	33.33	18000
rh_15_10.6	3	10064	3	0	5826
rh_15_10.8	3	31200	2	33.33	18000

Table 12: Results obtained by formulation P1 for class C2: random instances.

$B = 155 Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rl_15_2.1	3	5783	3	0	858
rl_15_2.2	4	114900	3	25	18000
rl_15_2.3	3	9514	3	0	1221
rl_15_5.1	3	17745	3	0	6673
rl_15_5.2	3	15193	3	0	4803
rl_15_5.3	3	3009	3	0	981
rl_25_3.1	4	10700	2	50	18000
rl_25_3.2	4	5400	2	50	18000
rl_25_3.3	4	17000	2	50	18000
rl_25_3.5	4	12200	2	50	18000
$B = 622 Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rh_15_10.1	3	11200	2	33.33	18000
rh_15_10.2	3	3421	3	0	4113
rh_15_10.4	3	11100	2	33.33	18000
rh_15_10.6	3	12533	3	0	15542
rh_15_10.8	3	11200	2	33.33	18000

Table 13: Results obtained by formulation P2 for class C2: random instances.

$B = 155 Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rl_15_2.1	3	10744	3	0	181
rl_15_2.2	4	300000	4	–	9303
rl_15_2.3	3	8281	3	0	191
rl_15_5.1	3	1113	3	0	28
rl_15_5.2	3	2552	3	0	68
rl_15_5.3	3	2774	3	0	62
rl_25_3.1	4	54100	4	–	18000
rl_25_3.2	4	54500	4	–	18000
rl_25_3.3	4	59700	4	–	18000
rl_25_3.5	4	48400	4	–	18000
$B = 622 Mbs$					
name	z^*	**#nodes**	z	**gap (%)**	**time (sec.)**
rh_15_10.1	3	9943	3	0	643
rh_15_10.2	3	7740	3	0	716
rh_15_10.4	3	14690	3	0	817
rh_15_10.6	3	13631	3	0	1287
rh_15_10.8	3	14690	3	0	819

Table 14: Results obtained by formulation P3 for class C2: random instances.

shown that these IP formulations may be adequate to solve instances of the SRAP to optimality, as long as these formulations provide stronger bounds on the optimal value of the problem.

An interesting question for the future research is to strengthen the IP formulations by deriving additional classes of valid inequalities. More precisely, we would like to give a partial description of the polyhedron, whose extreme points are the 0-1 solutions of the system (7), (10), (11), (14), (25)-(27), (28), (29) and (30).

References

[1] R. Aringhieri and M. Dell'Amico. Solution of the sonet ring assignment problem with capacity constraints. Technical Report DISMI TR-12, Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, 2001.

[2] S. Cosares, D. Deutsch, I. Saniee, and O. Wasem. SONET toolkit: a decision support system for designing robust and cost-effective fiber-optic networks. *Interfaces*, 25(1):30–40, 1995.

[3] Dash Optimization, Inc. *XPRESS-MP: user manuals*, 1999. Version 12.05.

[4] O. Goldschmidt, A. Laugier, and E. V. Olinick. SONET/SDH ring assignment with capacity constraints. *Discrete Applied Mathematics*, 129:99–128, 2003.

[5] M. Grötschel and M. Padberg. Polyhedral theory. In R. Kan e D. B. Shmoys (eds.) L. Lawler, J. K. Lenstra, editor, *The Traveling Salesman Problem: a guided tour of combinatorial optimization*, pages 251–305. John Wiley & Sons, 1985.

[6] E. M. Macambira. *Modelos e Algoritmos de Programação Inteira no Projeto de Redes de Telecomunicações.* PhD thesis, Universidade Federal do Rio de Janeiro, Programa de Engenharia de Sistemas e Computação, May 2003. In portuguese.

[7] S. Martello and P. Toth. *An algorithm for the generalized assignment problem*, pages 589–603. Operational Research, volume 81. North Holland, Amsterdam, 1981.

[8] G. L. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* Wiley & Sons, 1988.

[9] J. Sosnosky and Tsong ho Wu. SONET ring applications for survivable fiber loop networks. *IEEE Communications Magazine*, 29(6):51–58, June 1991.

[10] O. J. Wasem, Tsong-Ho Wu, and R. H. Cardwell. Survivable SONET networks - design methodology. *IEEE Journal on Selected Areas in Communications*, 12(1):205–212, January 1994.

[11] L. A. Wolsey. *Integer Programming.* Wiley & Sons, 1998.