# GRASP with Path-Relinking for the SONET Ring Assignment Problem

Lucas de O. Bastos
Inst. de Computação - UFF
Niterói - RJ - Brasil
lbastos@ic.uff.br

Luiz S. Ochi
Inst. de Computação - UFF
Niterói - RJ - Brasil
satoru@ic.uff.br

Elder M. Macambira
Depto. de Estatística - UFPB
João Pessoa - PB - Brasil
elder@de.ufpb.br

## Abstract

*In this paper, we consider a combinatorial optimization problem that arises in telecommunications networks design. It is known as the SONET ring assignment problem (SRAP). In this problem, each client site has to be assigned to exactly one SONET ring and a special ring interconnects the other rings together. The problem is to find a feasible assignment of the client sites minimizing the total number of rings used. We describe a hybrid greedy randomized adaptive search procedure (GRASP), including the path-relinking concept, for finding good-quality solutions of the SRAP. Computational experiments on benchmark instances are reported, comparing the GRASP with path-relinking with previously proposed pure GRASP (without path-relinking) and with other algorithms found in the literature. Experimental results illustrate the effectiveness of the proposed method, over other methods, to obtain solutions that are either optimal or very close to it.*

## 1. Introduction

Consider the following node-partitioning problem. We are given an undirected graph $G = (V,\ E)$ and a positive integer $B$. Associated to each edge $(u,\ v)$ in $E$ there is a non negative integer $d_{uv}$. A feasible solution of the problem is a partition of the vertices in $G$ into disjoint sets $V_1, V_2, \dots, V_k$ such that:

$$\sum_{(u,v)\in G[V_j]} d_{uv} \leq B, \quad \forall j \in \{1, \dots, k\}, \tag{1}$$

$$\sum_{j=1}^{k} \sum_{(u,v)\in\delta(V_j)|u<v} d_{uv} \leq B, \tag{2}$$

where $G[V_j]$ is the graph induced by $V_j$ in $G$ and $\delta(V_j)$ is the set of edges with exactly one extremity in $V_j$. The goal is to find a feasible solution that minimizes the size of the partition, i.e., the value of $k$.

This graph partitioning problem is also known as the SONET ring assignment problem, or SRAP for short. The SRAP arises in the design of fiber-optics telecommunication networks using the SONET (Synchronous Optical NETwork) technology.

In this context, the vertices of graph $G = (V, E)$, with $|V| = n$, are associated to client sites while the edges are associated to the existence of traffic demand between pairs of clients. The edge weights measure the actual demand of communication among clients. Given a feasible solution to the graph partitioning problem stated above, the vertices in each subset of the partition form a local ring or simply a ring.

Due to physical limitations on the equipment used in building the network, the total demand in each ring must not exceed a constant $B$. Besides, the total demand between clients in different clusters (local rings) is handled by an additional ring called the federal ring.

The connection of a local ring to the federal ring is made possible through a device known as a digital cross-connect system (DCS). Since the capacity of the federal ring is also bounded by $B$, the sum of the weights of the edges in the multicut which corresponds to any feasible solution cannot be larger than that amount.

Finally, because the DCS's are by far the most expensive equipment needed to implement a network, a basic problem in the design of low-cost SONET networks is to find a solution that minimizes the number of local rings. One can easily check that the graph partitioning problem discussed at the beginning of this section correctly models the latter problem.

The SRAP is known to be NP-hard [6]. Exact and heuristic algorithms have been proposed for the SONET ring assignment problem [1, 2, 6, 8].

The aim of this paper is to propose a new GRASP heuristic for the SRAP, which makes use of path-relinking as an intensification strategy. We evaluate experimentally this heuristic and compare the solutions obtained by the new heuristic with previously known solutions in the literature.

The remainder of the paper is organized as follows. In

Section 2, we briefly state the implementation of GRASP for the SRAP from [2]. Path-relinking is presented in Section 3. Section 4 gives a description of how GRASP and path-relinking are combined. Experimental results, with benchmark instances, are presented in Section 5. Finally, concluding remarks are made in Section 6.

## 2 GRASP

A greedy randomized adaptive search procedure (GRASP) is a multistart or iterative process, where different points in the search space are probed with local search for high-quality solutions [3, 9]. Each iteration of GRASP consists of the construction of a randomized greedy solution, followed by a local search, starting from the constructed solution. The best solution from all iterations is returned as result.

In the remainder of this section, we describe in detail the phases of the GRASP for the SRAP described in [2] in order to facilitate the discussion of path-relinking that will follow in the next sections.

### 2.1 Construction phase

The construction phase builds, step-by-step, a feasible or infeasible solution for the SRAP. Even when the construction phase discovers that the instance is feasible, infeasible solutions can be visited. So, we allow more flexibility to move along the search space.

The objective function has been modified to turn feasible solutions more attractive than infeasible ones. To explain the changes in the objective function, we define the following value associated to a feasible solution $S$ with vertices set $V_1, \dots, V_r$:

$$BN = \max\left(0, \sum_{j=1}^{r} \sum_{(u,v)\in\delta(V_j)|u<v} d_{uv}\right), \qquad (3)$$

where $BN$ indicate the total traffic through the federal ring. Now, the cost of this solution in the modified objective function is computed by the formula:

$$z = r \times B \quad \text{( feasible solution ) or,} \qquad (4)$$

$$z = 4r \times B + BN \quad \text{( infeasible solution ).} \qquad (5)$$

We now turn our attention to the construction phase of GRASP. The algorithm is an adaptation of the three greedy heuristics (edge-based, cut-based and node-based) presented in [6] to include randomization.

It always starts with a solution containing $n$ rings with only one vertex assigned to each of them. Those rings are feasible since the instance itself is feasible. Therefore, the only constraint that can be violated is the one that imposes a limit on the demand on the federal ring.

At each iteration, the union of two rings into a single one is considered. Such an union is accepted only if the resulting ring is feasible. Clearly, this operation reduces the amount of demand on the federal ring. The greedy choice is guided by the demands on edges which are in the multicut set of the rings. The randomization will not force us to pick the best edge (edge-based heuristic), best cut (cut-based heuristic) or the best node (node-based heuristic). Instead, the selected edge, cut or node is chosen randomly among the best candidates in the restricted candidate list (RCL). We refer to [6] for more details on the edge-based, cut-based and node-based heuristics.

We have also developed an algorithm for the construction phase based on the Relative Neighborhood Heuristic (RNH) introduced in [2].

Let $\{u, v\}$ be pairs of clients and $\delta_{uv}$ be the set of all edges with exactly one extremity in $u$ and those with exactly one extremity in $v$. The RNH, first, generates a neighborhood matrix establishing that two clients $u$ and $v$ are considered relative neighbors if the edge $(u, v) \in \delta_{uv}$ is associated with the biggest value of $d_{uv}$ among all edges in $\delta_{uv}$. After this step, all neighbor clients are gathered together in a ring and the heuristic tries to merge rings the same way as the cut-based heuristic does. We refer to [2] for more details on RNH heuristic.

### 2.2 Local search

After a solution is constructed, a local search phase should be executed as an attempt to improve the initial solution. If an improvement is detected, the solution is updated and a new neighborhood search is initialized. Otherwise, the solution is locally optimal with respect to the neighborhood and the local search ends.

The definition of the neighborhood is crucial for the performance of the local search. The local search procedure here, is based on a neighborhood that tries to empty the ring with the smallest demand, say $r_l$, in the solution.

The current solution is initialized with the solution obtained by the construction phase. Each vertex $u$ is moved from $r_l$ to one of the existing rings $r$ such that $r$ remains feasible and the change in objective function $z$ is the best among all the possible moves. If $r_l$ becomes empty a new ring $r_l$ is chosen and the local search is performed again on the existing solution.

This procedure never turns a local ring infeasible and never increases the traffic on the federal ring, so it never turns a feasible solution into an infeasible one. Besides that, it improves feasible solutions by reducing its number of rings and infeasible ones by making them feasible.

## 3 Path-relinking

Path-relinking is an approach to integrate intensification and diversification in search. It was originally proposed for tabu search and scatter search [4, 5]. It consists in exploring trajectories that connect high-quality (elite) solutions.

Starting from one or more elite solutions, paths in the solution space leading toward other elite solutions are generated and explored in the search for better solutions. The trajectory is generated by introducing into the initial solution, attributes of the guiding solution.

The use of path-relinking within a GRASP, as an intensification strategy applied to each locally optimal solution, was first used by Laguna and Martí in [7]. A recent survey of GRASP with path-relinking is given in [10].

## 4 GRASP with path-relinking

In this section, we describe the aspects of GRASP with path-relinking proposed for the SRAP.

This new heuristic works through a pool of elite solutions $P$ found during the execution. In our implementation, the pool $P$ is formed with solutions found in the previous GRASP iterations. The pool $P$ is originally empty.

The objective of path-relinking is to integrate features of good solutions, found during the iterations of GRASP, into new solutions generated in subsequent iterations.

In pure GRASP, i.e. GRASP without path-relinking, all iterations are independent and therefore most good solutions are simply forgotten. Path-relinking tries to change this, by retaining previous solutions and using them as guides to speed up convergence to a good quality solution.

We define one solution $S$ for the SRAP as a vector of cardinality equal to $n$. Each client site $i$, with $i = 1, \ldots, n$, belongs to the ring indicated in $S(i)$. Each ring $r$ is numbered such that $r = min(i), i \in r$. In this way, solutions having exactly the same rings cannot be differently numbered.

Let $S_1$ and $S_2$ be two solutions, where $S_1$ is the locally optimal solution obtained after local search and $S_2$ is one of a few elite solutions in $P$. The path-relinking procedure starts with one of the solutions (say, $S_1$) and gradually transforms it into the other ($S_2$) by making $S_1(i) = S_2(i)$ where $i$ is chosen among all the localities $l$ where $S_1(l) \neq S_2(l)$.

The choice of the candidate $i$, in each step, is greedy: we realized the most profitable (or least costly) one. The outcome of the process is the best solution found in the path from $S_1$ to $S_2$.

An important aspect of new heuristic is the choice of the solutions $S_1$ and $S_2$. Empirically, we observed that an application of path-relinking to a pair of solutions, with different number of rings, is less probable to be successful than a pair of solutions with same number of rings.

On the other hand, the longer the path between the solutions, the greater the probability that an entirely different local optimum will be found. Therefore, it is reasonable to take into account not only solution quality, but also diversity when dealing with the pool $P$ of elite solutions.

So, we implemented one strategy for selecting $S_1$ and $S_2$ solutions. Let $c(S)$ be the number of rings in $S$, we have the following situations where we apply the path-relinking procedure:

(i) $c(S_1) = c(S_2)$, and $S_1$ and $S_2$ are infeasible;

(ii) $c(S_1) < c(S_2)$ and $S_1$ is infeasible;

(iii) $c(S_1) > c(S_2)$ and $S_2$ is infeasible.

If condition (ii) is satisfied we reduce the number of rings in $S_2$ by emptying its lesser ring $r_l$ moving its nodes to the best profitable rings excluding $r_l$. We repeat this process until $c(S_1) = c(S_2)$. If condition (iii) is satisfied we apply the same strategy to reduce the number of rings in $S_1$. Note that, by doing this, the situations (ii) and (iii) become unsatisfied and the situation (i) is satisfied.

In cases (iv), (v) and (vi) below, we don't apply the path relinking procedure because the existence of a feasible solution prevents it to find a better feasible solution:

(iv) $c(S_1) = c(S_2)$, and either $S_1$ or $S_2$ is feasible;

(v) $c(S_1) < c(S_2)$ and $S_1$ is feasible;

(vi) $c(S_1) > c(S_2)$ and $S_2$ is feasible.

After this selection, we generate two paths, one from $S_1$ to $S_2$ and the other from $S_2$ to $S_1$. This is done because these paths often visit different intermediate solutions.

Another important point of the heuristic is the management of the pool $P$ of elite solutions. In [7], the authors update their pool by maintaining in it three best-quality solutions. We use an approach that obtained the elite solutions within GRASP. The approach is explained below.

Assume that $NR$ indicates the number of rings to guide the path-relinking procedure. The value is defined in the interval $[z_{lb}, c(S_{best})]$ with $z_{lb} = \left(\sum_{u \in V} \sum_{v \in V | u < v} d_{uv}\right) \;/\; B$ and $c(S_{best})$ corresponds to the number of rings of the best solution $S_{best}$ found.

So, one solution $S$ is added to the pool $P$ if it satisfies the condition: $c(S) = NR$. Once accepted for insertion into $P$, $S$ replaces the last elite solution in $P$, which is discarded from $P$. All candidate solutions with more than $NR$ rings have to have its number of rings reduced to be exposed to path-relinking, according to the rules (i) to (iii) explained above.

Now, we show how the procedures described above and in section 2 are combined in our implementation. Figure 1 shows the steps of path-relinking for the SRAP. Let $S$ be the current solution obtained by pure GRASP and $T$ the guiding solution. The procedure will make at most $n$ client site exchanges (lines 3 to 18) until the current solution $S$ becomes equal to the guiding solution $T$. In each step, the best exchange is found (lines 4 to 12) and applied to $S$ (lines 13 and 14). The algorithm also checks if the generated solution is better than the best known solution and, if so, saves it (lines 15 and 16).

```
procedure PATHRL($n, S, T$)
1.  $A \leftarrow S$;
2.  $B \leftarrow (z(T) < z(S)\ ?\ T : S)$;
3.  for $i \leftarrow 1$ to $n$ do
4.      for $j \leftarrow 1$ to $n$ do
5.          if $(S(i) \neq T(i))$ and $(S(j) \neq T(j))$ then
6.              $S(j) \leftarrow T(j)$;
7.              if $(z(S) < z(A))$ then
8.                  $k \leftarrow j$;
9.              end-if
10.             $S(j) \leftarrow A(j)$;
11.         end-if
12.     end-for
13.     $A(k) \leftarrow T(k)$;
14.     $S(k) \leftarrow T(k)$;
15.     if $z(A) < z(B)$ then
16.         $B \leftarrow A$;
17.     end-if
18. end-for
19. return($B$).
end PATHRL.
```

**Figure 1. Path-relinking.**

Figure 2 shows the steps of GRASP with path-relinking. Initially, the pool of elite solutions $P$ is empty (line 1). Each iteration consists of solution construction, local search using the constructed solution as the initial solution (lines 4 and 5), and after nItrPR iterations are done, a path-relinking phase is executed.

Initially, when the pool owns less than two solutions, if the solution generated by GRASP has $NR$ rings and it is infeasible, then the solution is inserted into the pool $P$ (lines 29 to 31). The pool doesn't accept feasible solutions according to the rules (i) to (vi) explained above.

Path-relinking preliminary work starts making the solution produced by local search to have $NR$ rings just like the solution selected at random from $P$ (lines 10 to 12). Then, if $S$ stays infeasible, path-relinking is done from $S$ to $T$ and from $T$ to $S$ and at each step the resulting solution $r$ is checked to enter the pool and to substitute the best solution found $S_{best}$ (lines 13 to 28).

```
procedure GRASP-PR($n$)
1. $P = \emptyset$;
2. $S_{best} \leftarrow$ construct();
3. for $i \leftarrow 1$ to $maxItr$ do
4.     $S \leftarrow$ construct();
5.     $S \leftarrow$ localSearch($S$);
6.     if $(z(S) < z(S_{best}))$ then
7.         $S_{best} \leftarrow S$;
8.     if $i > nItrPR$ then
9.         if $|P| > 1$ then
10.            Select elite solution $T \in P$ at random;
11.            while $c(S) > c(T)$ do
12.                reduceOneRing($S$);
13.            if ($c(S) = c(T)$ and not isFeasible($S$)) then
14.                $r \leftarrow$ PATHRL($n, S, T$);
15.                if not isFeasible($r$) then
16.                    UPDATE_POOL($r, P$);
17.                if $(z(r) < z(S_{best}))$ then
18.                    $S_{best} \leftarrow r$;
19.                $r \leftarrow$ PATHRL($n, T, S$);
20.                if not isFeasible($r$) then
21.                    UPDATE_POOL($r, P$);
22.                if $(z(r) < z(S_{best}))$ then
23.                    $S_{best} \leftarrow r$;
24.            end-if
25.        else if ($c(S) = RN$ and not isFeasible($S$)) then
26.            $P \leftarrow P \cup \{S\}$;
27.        if ($i$ mod $nItrPR) = 0$ then
28.            emptyPool($P$);
29.            $NR \leftarrow (NR = c(S_{best})\ ?\ z_{lb} : NR + 1)$;
30.        end-if
31.    end-if
32. end-for
33. return($S_{best}$).
end GRASP-PR.
```

**Figure 2. GRASP with path-relinking.**

## 5 Computational results

In this section, we illustrate the use of GRASP on randomly generated test problems. All tests were run on a PC desktop equipped with a 1700 MHz Pentium IV processor and 256 Mbytes of RAM under Linux operating system. The GRASP code is written in C ++ language.

### 5.1 Test problems

Before we describe the experimental results, we must comment about benchmark instances used. The set of instances is divided into two categories. Instances in category C1 are taken from [6] and correspond to instances type 1 and 2 in that paper. Category C2 is composed of all instances tested in [1] excluding those already in C1.

In total, there are 111 instances in C1 and 230 in C2.

Within each category, instances are divided into geometric and random ones and, then, further divided into those having low and high ring capacities, respectively, 155 Mbs and 622 Mbs. These characteristics of an instance can be deduced from its name. Instance names always start with two letters. The first letter is either `G` or `R` meaning that the instance belongs to the geometric or the random subdivision, respectively. The second letter is either `L` or `H`, depending if the ring capacity is 155 Mbs or 622 Mbs, respectively. For instances in class C2, the first name is `new`. For more details on how those instances were generated and their properties, we refer the interested reader to the original papers where they were introduced.

## 5.2 Experiments

Our objective with the experimental part of this paper is to evaluate the effectiveness of the path-relinking when used in conjuction with GRASP.

For each instance considered in our experiments, we fix a solution target value equal to the optimal solution. In [8], the author implemented an exact algorithm that provided optimal solutions for the two categories C1 and C2. Moreover, the tests were processed using 1000 iterations for each instance and each one was executed 10 times with different seeds, recording the time taken for each algorithm to find the solution target for each instance.

In Tables 1 and 2, we show the results obtained for the categories C1 and C2, respectively, for each algorithm. Each table lists the instances (type and number of feasible instances), the percent number of optimal solutions found and the average computational time (in seconds) to find a solution having the target value (optimal solution).

**Table 1. Computational results for C1 class instances.**

| Instance | | GRASP | | GRASP-PR | |
|---|---|---|---|---|---|
| type | # FS | # OS(%) | time | # OS(%) | time |
| GL | 23 | 100.0 | 0.047 | 100.0 | 0.010 |
| GH | 27 | 100.0 | 0.054 | 100.0 | 0.014 |
| RL | 28 | 100.0 | 0.072 | 100.0 | 0.024 |
| RH | 33 | 100.0 | 0.046 | 100.0 | 0.047 |

The tables illustrate the effect of path-relinking on GRASP. Even though it is a more expensive procedure in terms of computational time, the path-relinking improved the performance of GRASP for three instance types in C1 class and for one instance type in C2 class. These results show that, in some cases, the GRASP-PR got the convergence to the optimal solution faster than pure GRASP. Furthermore, GRASP with path-relinking found optimal solutions in 340 instances while pure GRASP found optimal solutions in 338 out of 341 instances tested.

**Table 2. Computational results for C2 class instances.**

| Instance | | GRASP | | GRASP-PR | |
|---|---|---|---|---|---|
| type | # FS | # OS(%) | time | # OS(%) | time |
| GL | 70 | 100.0 | 0.314 | 100.0 | 0.407 |
| GH | 70 | 97.1 | 0.384 | 100.0 | 1.630 |
| RL | 70 | 98.6 | 1.044 | 98.6 | 1.094 |
| RH | 20 | 100.0 | 0.747 | 100.0 | 0.204 |

In Table 1, we can see that the proposed GRASP with path-relinking found optimal solutions for all instances. The average execution times for algorithm proposed stayed below 0.024 second whereas for pure GRASP the average times were 0.055 second.

In Table 2, we show the results obtained for second category of instances C2. We make the following observations about this category. Pure GRASP found optimal solutions in 227 out of 230 instances tested and the average execution times were solved in less than 0.623 second. On another hand, GRASP with path-relinking found optimal solutions in 229 out of 230 instances tested and the average execution times stayed below 0.834 second.

Table 3 lists all test problems for which pure GRASP did not find optimal solutions. Besides instance name and value of the optimal solution, the table lists the results found, as well as the optimality gap, i.e. $(100 \times [z_{GRASP} - z_{exact}]/z_{GRASP})$, in the GRASP solution. It should be noted that, except for the new.RL_50_8.5 instance, where pure GRASP did not find a feasible solution, the optimality gap is small, i.e., notwithstanding the GRASP found solutions within 16.67% of the optimal solution, these gaps correspond to only one ring far from the optimal solution. Furthermore, these solutions were obtained at most 1.031 seconds, in average, running 1000 iterations.

**Table 3. Computational results for instances where GRASP did not find optimal solutions.**

| Instance | $z_{exact}$ | GRASP | gap(%) | $T_{1000}$ |
|---|---|---|---|---|
| new.GH_50_3.4 | 5 | 6 | 16.67 | 1.031 |
| new.GH_50_9.4 | 5 | 6 | 16.67 | 1.031 |
| new.RL_50_8.5 | 6 | – | – | 1.141 |

Unfortunately, for one instance of category C2, the

**Table 4. Computational results for instances where GRASP-PR did not find optimal solutions.**

| Instance | $z_{exact}$ | `GRASP-PR` | gap(%) | $T_{1000}$ |
|---|---|---|---|---|
| new.GH_50_3.4 | 5 | 6 | 16.67 | 2.406 |

GRASP with path-relinking did not find optimal solutions. Table 4 presents this instance. The heuristic obtained one solution within 16.67% of the optimal solution. This gap corresponds to only one ring far from the optimal solution. Furthermore, this solution was produced at most 2.406 seconds, on average, running 1000 iterations.

Finally, in table 5, we compare the results obtained by GRASP-PR with the results existing in the literature. The first two columns show instance class and number of feasible solutions, the following columns give the number of optimal solutions found by GRASP-PR, edge-based, cut-based and node-based heuristics [6] and DMN procedure [1]. The results show that GRASP-PR got better results in terms of solution quality. We are not comparing computational times because the experiments where performed on different equipment but we believe that our algorithm is very competitive with those in literature.

**Table 5. Computational results for C1 and C2 class instances.**

| Instance | | `GRASP-PR` | `EB, CB, NB` | `DMN` |
|---|---|---|---|---|
| class | # FS | # OS(%) | # OS(%) | # OS(%) |
| C1 | 111 | 100.00 | 97.46 | 100.00 |
| C2 | 230 | 99.57 | – | 98.26 |

## 6 Concluding remarks

In this paper, we presented a GRASP for the SONET ring assignment problem and showed how the path-relinking technique can be used to improve the performance of the greedy randomized search. We described a new way to implement path-relinking within a GRASP.

Extensive computational experiments were done with benchmark instances for the SRAP in this paper. The GRASP with path-relinking heuristic was shown to improve the performance of a pure GRASP, previously described in [2], both in terms of finding a solution faster and finding a better solution for a fixed number of iterations.

Additionally we compared proposed algorithm with the edge-based, cut-based and node-based heuristics proposed in [6] and the procedure DMN introduced in [1]. The GRASP with path-relinking was superior to the heuristics described in [6] finding optimal solutions for 100.0% of C1 class instances against 97.46%. For C2 class instances GRASP with path-relinking found 99.57% against the 98.26% found by DMN procedure [1].

## References

[1] R. Aringhieri and M. Dell'Amico. Comparing metaheuristic algorithms for the SONET network design problems. *Journal of Heuristics*, 11:35–57, 2005.

[2] L. de O. Bastos, L. S. Ochi, and E. M. Macambira. A relative neighborhood grasp for the sonet ring assignment problem. In *International Network Optimization Conference - INOC*, pages 833–838, 2005.

[3] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[4] F. Glover. Tabu search and adaptive memory programing: advances, applications and challenges. In R. S. Barr, R. V. Helgason, and J. L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, 1996.

[5] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path-relinking. *Control Cybernetics*, 39:653–684, 2000.

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

[7] M. Laguna and R. Martí. GRASP and path-relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

[8] E. M. Macambira. *Modelos e Algoritmos de Programação Inteira no Projeto de Redes de Telecomunicações*. PhD thesis, COPPE, Universidade Federal do Rio de Janeiro, Brazil, 2003. in portuguese.

[9] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2003.

[10] M. G. C. Resende and C. C. Ribeiro. GRASP with path-relinking: recent advances and applications. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: progress as real problem solvers*, pages 29–63. Springer, 2005.