

A Genetic Algorithm with Evolutionary Path-relinking for the SONET Ring Assignment Problem

Lucas de Oliveira Bastos, Luiz Satoru Ochi

Universidade Federal Fluminense
Instituto de Computação
Niterói, RJ, Brazil
{lbastos,satoru}@ic.uff.br

Abstract

The SONET ring assignment problem (SRAP) arises in telecommunications network design, specifically on the infrastructure planning phase of a network project. In a SONET network, each client site is associated or assigned to exactly one SONET ring, called local ring. The communication among clients in different rings is done through a special ring, called federal ring, that interconnects the local rings. A capacity constraint on each ring, including the federal ring, is imposed. The objective is to find an assignment of the clients that attends to all constraints and minimizes the total cost of the network. As the majority of this cost comes from the equipment used to connect local rings to the federal ring, the problem is simplified to find a feasible assignment that reduces the number of local rings in the network. We propose a genetic algorithm with evolutionary path-relinking for finding good-quality solutions. The heuristics included in the algorithm, as well as the evolutionary path-relinking, are detailed. Experimental results illustrate the effectiveness of the proposed method to obtain very good solutions in a short period of time.

Keywords: Scheduling, Planning, Genetic Algorithms, Path-relinking.

1 Introduction

In recent years, the evolution of the services offered to users by telecommunications industry has brought new challenges in the planning of its networks. Consequently, the need to solve optimization problems during this phase of the project appeared [10].

This work will be focused on a problem related to the planning of telecommunications networks, more specifically, to the definition of the topology and the technology involved. This planning consists in the determination of the connections among a set of client sites, at a low cost, satisfying a set of constraints. Amongst these constraints, the capacity of survival of the network is fundamental nowadays.

The capacity of survival of a network is its capacity to remain active even if a physical error occur. The ring topology working together with the SONET (Synchronous Optical Network) technology takes care of this requirement [8, 11].

The planning problem we are going to discuss can be defined as follows:

Given a set of client sites L , belonging to a telecommunications network, a matrix of traffic demands d_{uv} between two client sites u and v , with $u, v \in L$, an integer B representing the traffic capacity in a ring and a function of cost c , is desired to determine an attribution of client sites to subsets, that represent rings, such that the capacity of these subsets is satisfied according to the parameter B and all client sites traffic demands are satisfied at the lowest possible cost.

This problem is part of the physical project of a network and is called SONET Ring Assignment Problem or SRAP. The SRAP arises in the design of fiber-optics telecommunication networks using the SONET or Synchronous Digital Hierarchy (SDH) technology.

Each client site needs an Add Drop Multiplexer (ADM) to be connected to the network through a local ring. The traffic among local rings is done through a special ring called Federal Ring (FR). Each local ring requires a Digital Cross-connect System (DCS) to be connected to the federal ring.

The aim of this paper is to propose a Genetic Algorithm heuristic for the SRAP, which makes use of evolutionary path-relinking as an intensification strategy.

The remainder of the paper is organized as follows: in Section 2 we will present a definition for the SRAP Problem. Related work found in literature will be commented in Section 3. In Section 4 we detail the genetic algorithm and evolutionary path-relinking. Experimental results, with benchmark instances, are presented in Section 5. Finally, concluding remarks are made in Section 6.

2 Problem definition

Consider the following node-partitioning problem. 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 for the problem is a partition of the vertexes in G into disjoint sets V_1, V_2, \dots, V_k such that:

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

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

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 value of k .

This graph partitioning problem models the SRAP problem introduced in Section 1. In this context, the vertexes of graph $G = (V, E)$, with $|V| = n$, are associated to client sites whereas 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 vertexes in each subset of the partition form a local ring.

Due to physical limitations on the equipment used in building the network, the total demand in each local ring (or simply ring) must not exceed a constant B . Besides that, 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 the DCS's. 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 SONET network, a basic problem in the design of low-cost SONET networks is to find a solution that minimizes the number of local rings.

3 Results found in literature

Goldschmidt, Laugier and Olinick studied the SRAP by the first time in [6]. The authors demonstrated that the SRAP is NP-Hard. They also proposed techniques of integer programming and developed three greedy heuristics to solve the problem: edge-based, cut-based and node-based.

In 2005, Aringhieri e Dell'Amico [1] have studied several algorithms to solve the SRAP. The authors have modeled the problem and proposed construction procedures, search neighborhoods and tabu lists.

Macambira [7] proposed a branch-and-price exact algorithm to solve SRAP. His work has provided the optimal solutions to instances found in [6] and [1]. Besides that, his results presented competitive computational times to the investigated instances in terms of exact algorithms. More details about these instances and about the branch-and-price algorithm can be found in the original papers where they were introduced.

4 Genetic Algorithm

Genetic algorithms (GAs) are adaptive methods, which may be used to solve search and optimization problems [3]. They are based on the genetic process of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection, i.e. survival of the fittest, first clearly stated by Charles Darwin in *The Origin of Species by Natural Selection*. GAs simulate the natural population processes that are essential to evolution. This way, its possible to "evolve" solutions for real world problems if they have been suitably encoded.

A potential solution to a problem may be represented as a set of parameters. These parameters (known as genes) are joined together to form a string of values (chromosome). In genetic terminology,

the set of parameters represented by a particular chromosome is referred to as an individual. The fitness of an individual depends on its chromosome and is evaluated by the fitness function. During the reproductive phase, the individuals are selected from the population and recombined, producing offspring, which comprise the next generation. Parents are randomly selected from the population using a scheme, which favors fitter individuals. Having selected two parents, their chromosomes are recombined, typically using mechanisms of crossover and mutation. Mutation is usually applied to some individuals, to guarantee population diversity.

```

procedure GENETIC-ALGORITHM( $n, S, T$ )
1.  Generate initial population  $P_0$ ;
2.  Evaluate population  $P_0$ ;
3.  Initialize generation counter  $g_0$ ;
4.  while stopping criteria not satisfied repeat
5.      Select some elements from  $P_g$  to copy into  $P_{g+1}$ ;
6.      Crossover some elements of  $P_g$  and put into  $P_{g+1}$ ;
7.      Mutate some elements of  $P_g$  and put into  $P_{g+1}$ ;
8.      Evaluate new population  $P_{g+1}$ ;
9.      Increment generation counter:  $g \leftarrow g + 1$ ;
10. end-while
end GENETIC-ALGORITHM.

```

Figure 1: Genetic algorithm pseudocode.

Figure 1 shows a pseudocode for a basic genetic algorithm. Steps 1 and 2 generates and evaluates the initial population, respectively. In step 5, the adaptation of individuals by competing is made and the worst individuals are eliminated. Steps 6 represents reproduction by crossover and step 7 allows population diversification by mutation.

In this work, we propose a genetic algorithm where constructive heuristics generate initial population and a crossover operator generates offspring for the next generations. Additionally, we apply a post optimization mechanism known as Evolutionary Path-Relinking (EvPR) to intensify the search for better solutions.

4.1 Chromosome Representation

We represent a SRAP solution as an array of positive integers with size n . This array represents a chromosome. Each array position is an individual gene. The position of the gene identifies a specific client in the real problem and the gene content is the information about which ring that client is assigned to. So, the gene in the position i indicates the ring which the i -th client is assigned to.

4.2 Fitness Function

The fitness function has been defined such that it turns feasible solutions more attractive than infeasible ones. To explain the ideas behind the fitness function, we define the following values associated to a feasible solution S with clients sets V_1, \dots, V_r :

$$D_{FR} = \sum_{j=1}^r \sum_{(u,v) \in \delta(V_j) | u < v} d_{uv}, \quad (1)$$

$$D_i = \sum_{(u,v) | u \in V_i, v \in V_i} d_{uv}, \forall i, 1 \leq i \leq r \quad (2)$$

where D_{FR} indicates the total traffic through the federal ring and D_i indicates the traffic through the local ring i . Now, the cost of this solution in the objective function is computed by the equation:

$$z = r \times B \text{ (feasible solution)}$$

$$z = \sum_{i=1}^r D_i + D_{FR} + r \times B \text{ (infeasible solution).}$$

Note, that objective function hardly penalizes infeasible solutions. The idea here is to avoid an infeasible solution to get better evaluation than a feasible one.

4.3 Evolutionary strategy

The generation of an initial population is the genetic algorithm evolution starting point. Our approach generates initial solutions by the use of two constructive heuristics. The first constructive heuristic is called Random Node-Based (RNB) and the second is called Relative Neighborhood (RN). The RNB heuristic is an adaptation of the Node-based heuristic, proposed in [6], to include randomness in the nodes selection. The RN heuristic was proposed in [2]. By running one of these heuristics n times, we get a population of size n . To help increase diversification, each heuristic was used to generate half of the total population.

After the generation of the initial population, it becomes the current population and the GA starts the evolutionary process. Given the current population, we perform the following operators in order to obtain the next generation:

Crossover: The crossover operator selects two individuals randomly to act as parents of an offspring. One of the parents is chosen amongst the best individuals in the population, while the other is randomly chosen from the whole current population. We then, randomly choose some genes from the parents chromosomes to form the offspring chromosome. This way, at this point, the offspring chromosome remains incomplete. In order to make it complete, we run a local search to find the best missing genes.

A important aspect to note is that the missing genes allows some diversification because they can be freely chosen from all the search space and not only from the two parents correspondent possibilities. Furthermore, a lot of missing genes may generate a very different solution and allow large jumps into the search space.

Survival: We use an elitist [5] operator for survival of the individuals between two generations. The best individuals in the current population survive and become part of the next. Elitist strategy provides a means of ensuring that the best individuals are identified and allowed to pass their traits to latter generations.

4.4 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]. 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.

Path-relinking heuristic works through a pool of elite solutions P found during the execution. In our implementation, the pool P is composed by the elite solutions in the current GA population.

We define one solution S for the SRAP as an array of cardinality equals to $n = |V|$. Each client site i , with $i = 1, \dots, n$, belongs to the ring indicated in $S(i)$. Each ring r in S is numbered such that $label(r) = \min\{i : i \in r\}$. 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 client sites l with $S_1(l) \neq S_2(l)$.

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 this 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.

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. We use a special approach to feed the pool of elite solutions. This approach is explained below.

Let $I = [z_{lb}, z_{ub}]$ be the range of desired numbers of rings that one solution should have to be a candidate to path-relinking, where z_{ub} corresponds to the number of rings of the best infeasible solution found or the number of rings of the best feasible solution found minus one and $z_{lb} = (\sum_{u \in V} \sum_{v \in V | u < v} d_{uv}) / B$. Moreover, let $C_{z_{lb}}, C_{z_{lb} + 1}, \dots, C_{z_{ub}}$ be sets of solutions such that C_i is composed by all solutions having number of rings equals to the i th element of I . We can define C_I as the set of all candidate solutions to path-relinking given by $C_I = C_{z_{lb}} \cup C_{z_{lb} + 1} \cup \dots \cup C_{z_{ub}}$.

Once defined the set of candidate solutions, a good pool structure P is mandatory to the performance of the algorithm. So, for each set of solutions C_i we have provided a corresponding structure P_i within P . As we cannot have an unlimited pool P , we have reserved a number k_s of solutions to each structure P_i . One solution S is added to P_i if it belongs to C_i and once accepted for insertion, it replaces the worst elite solution if P_i is full. If a feasible solution with r rings, belonging to C_r , is found, the range I changes to $[z_{lb}, r - 1]$. In this way, P is updated and all solutions in P_i , with $i \geq r$, are discarded. If there is not a feasible solution, one solution with more than z_{ub} rings may become a candidate one by reducing its number of rings.

Figure 2 shows the steps of path-relinking for the SRAP. Let S be the initial solution and T the guiding solution. The procedure will make at most n client site exchanges (steps 3 to 15) until the initial solution S becomes equal to the guiding solution T . In each step, the best exchange is found (steps 4 to 12) and applied to S (step 13). The algorithm also checks if the generated solution is better than the best known solution and, if so, saves it (step 14).

4.5 Evolutionary Path-Relinking

The evolutionary path-relinking (EvPR) was proposed in [9] as post optimization step in a hybrid heuristic for the p-median problem. The main idea is to combine different local optima found in the previous steps and possibly find a better solution. We use the same idea here.

As the genetic algorithm generates new populations, the current population becomes rich in local optima due to the survival operator. When enough populations have been generated, the EvPR algorithm starts over the elite solutions of the last population.

Every solution in the elite pool is combined with each other by path-relinking. The solutions generated by this process are added to a new pool of elite solutions, representing a new path-relinking generation. The algorithm proceeds until it creates a generation that does not improve upon previous generations.

5 Computational results

In this section, we illustrate the use of proposed algorithms on literature instance problems. All tests were run on a PC desktop equipped with an AMD Athlon 64 X2 5600+ processor and 2048 MB of RAM under Ubuntu Linux operating system. The code was written in C++ language and compiled with GCC compiler.

```

procedure DoPathRelinking( $n, S, T$ )
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.                  $S(j) \leftarrow A(j)$ ;
10.            end-if
11.        end-for
12.    end-for
13.     $A(k) \leftarrow T(k)$ ;  $S(k) \leftarrow T(k)$ ;
14.    if  $z(A) < z(B)$  then  $B \leftarrow A$ ;
15. end-for
16. return( $B$ ).
end DoPathRelinking.

```

Figure 2: Path-relinking pseudocode.

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 classes. Instances in class C1 are taken from [6] and correspond to instances type 1 and 2 in that paper. Class C2 is composed of all instances tested in [1] excluding those already in C1.

In total, there are 118 feasible instances in C1 and 230 in C2. Within C1 and C2 classes, instances have between 15 and 50 client sites. They are divided into geometric and random ones and, then, further divided into those having low and high ring capacities, respectively, 155 Mbps and 622 Mbps. 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 use of adaptive memory and path-relinking procedure together with GA.

In [7], the authors implemented an exact algorithm that provided optimal solutions for the two classes C1 and C2. So, for these instances, we fixed a solution target value equals to the value of the optimal solution. Moreover, the tests were processed within a maximum of 5 seconds. The EvPR was started after 15 generations.

We randomly chose 10 different random seeds for the experiment. The algorithms were run over each instance one time for each seed. The average computation times was then calculated.

In Table 1, we show the average results obtained. The table lists the instances class, the number of feasible instances per class (# FI), the percent number of optimal solutions found (% OS) and the average computational times in seconds.

Table 1: Average results for C1 and C2 classes instances.

instances		GA		GA+EvPR	
class	# FI	%OS	time	%OS	time
C1	118	98.3	0.21	100.0	0.11
C2	230	85.7	1.93	98.3	0.78

The table illustrates the effect of EvPR on GA over instances in C1 and C2 classes. The GA+EvPR got better results, in terms of solution quality, and also smaller average computation times.

For 2 instances in C2 class, GA+EvPR did not find the optimal solution but found feasible solutions with a gap of only one ring. For other 2 instances GA+EvPR could not find feasible solutions.

6 Concluding remarks

In this paper, we presented a GA for the SONET ring assignment problem and showed how the use of evolutionary path-relinking technique can be used to improve the performance of the genetic algorithm.

Computational experiments were done with benchmark instances for the SRAP in this paper. The GA+EvPR heuristic was shown to improve the performance of a traditional GA, both in terms of finding a solution faster and in terms of finding a better solution in a given computation time.

As future work, we propose the introduction of new genetic operators and the use of path-relinking in the phase of population generation.

References

- [1] ARINGHERI, R., DELL'AMICO, M. Comparing metaheuristic algorithms for sonet network design problems. *Journal of Heuristics*, 2005, Vol. 11(1), 35–57.
- [2] BASTOS, L. O., OCHI, L. S., MACAMBIRA, E. M. A relative neighbourhood GRASP for the SONET ring assignment problem. *Proc. of Proceedings of the International Network Optimization Conference*, 2005, p. 833–838.
- [3] BEASLEY, D., BULL, D. R., MARTIN, R. R. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 1993, Vol. 15(4), 170–181.
- [4] GLOVER, F. Tabu search and adaptive memory programming – advances, applications and challenges. In *Interfaces in Computer Science and Operations Research*, R. S. Barr, R. V. Helgason, and J. L. Kennington, Eds. Kluwer Academic Publishers, 1996, p. 1–75.
- [5] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
- [6] GOLDSCHMIDT, O., LAUGIER, A., OLINICK, E. V. SONET/SDH ring assignment with capacity constraints. *Discrete Applied Mathematics*, 2003, Vol. 129(1), 99–128.
- [7] MACAMBIRA, E. M., MACULAN, N., DE SOUZA, C. C. A column generation approach for SONET ring assignment. *Networks*, 2006, Vol. 47(3), 157–171.
- [8] OMIDYAR, C. G., ALDRIDGE, A. Introduction to SDH/SONET. *IEEE Communications Magazine*, 1993, Vol. 31, 30–33.
- [9] RESENDE, M., WERNECK, R. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, Vol. 10(1), 59–88.
- [10] RIBEIRO, C. C., MARTINS, S. L., ROSSETI, I. Metaheuristics for optimization problems in computer communications. *Computer Communications*, 2007, Vol. 30(4), 656–669.
- [11] WASEM, O. J., WU, T.-H., CARDWELL, R. H. Survivable SONET networks-design methodology. *IEEE Journal on Selected Areas in Communications*, 1994, Vol. 12(1), 205–212.