A GRASP algorithm for designing a Wide Area Network backbone

Héctor Cancela [1], Franco Robledo [1,2], Gerardo Rubino[2]
[1] Universidad de la República
Address : Facultad de Ingeniería, J.Herrera y Reissig 565, Montevideo, Uruguay
Phone:+598-2-7114244 ext. 112, fax: +598-2-7110469
Email: [cancela,frobledo]@fing.edu.uy
[2] IRISA
Address : Campus de Beaulieu, Rennes 35042 CEDEX, France
Phone: +33-2-99847296, fax: +33-2-99847171
Email:Gerardo.Rubino@irisa.fr

Abstract

The greedy randomized adaptive search procedure (GRASP) is a well-known metaheuristic for combinatorial optimization. In this work, we introduce a GRASP for designing a survivable Wide Area Network backbone. The algorithm builds topologies which comply with heterogeneous node-connectivity requirements. This work is motivated by the need of designing a low-cost backbone network which can survive to failures in switch sites as well as in the connection lines. The method is applied to a set of problem instances with different connectivity requirements, obtaining results which appear promising.

keywords metaheuristics; topological design; survivability.

1 Introduction

A wide area network (WAN) can be seen as a set of the sites and a set of communication lines that interconnect the sites. A typical WAN is organized as a hierarchical structure integrating two levels: one *backbone network* and a number of *local access networks* [9]. Each local access network usually has a tree-like structure, rooted at a single switch site of the backbone network, and connects users (terminal sites) either directly to this switch site or to a hierarchy of intermediate concentrator sites (local servers) which are connected to the switch site. The backbone network has usually a meshed topology, and its purpose is to allow efficient and reliable communication between the switch sites of the network that act as connection points for the local access networks (eventually incorporating other switch sites for efficiency purposes).

The topological design of a Backbone network basically consists of finding a minimum cost topology which satisfies some additional requirements, generally chosen to improve the survability of the network (that is, its capacity to resist failures of some of its components). One way to do this is to specify a connectivity level, and search for topologies which have at least this number of disjoint paths (either edge disjoint or node disjoint) between pairs of switch sites. In the most general case, the connectivity level can be fixed independently for each pair of switch sites (heterogeneous connectivity requirements). This problem can be modelled as a Generalized Steiner Problem with Node-Connectivity [13, 18]; finding a minimum cost topology with these constraints is a NP-hard problem. Some references in this area are [7, 9, 10, 11, 12, 14, 17]; these works are either focused on the edge-disjoint flavour of the problem, or explore particular cases (for example, where the connection requirement is set to two disjoint paths for all switch sites [3, 4, 8]).

Topologies verifying edge-disjoint path connectivity constraints assure that the network can survive to failures in the connection lines; while node-disjoint path constraints assure that the network can survive to failures both in in switch sites as well as in the connection lines. This paper proposes an algorithm based on the GRASP methodology for finding a Backbone network topology that satisfies connection requirements on the number of node-disjoint paths, working upon the Generalized Steiner Problem with Node-Connectivity model.

Section 2 formalizes our backbone network design problem (BNDP). Section 3 presents the GRASP metaheuristic, and Section 4 introduces its adaptation for solving the BNDP. Experimental results are presented in Section 5, showing that the algorithm works very well for a number of test cases corresponding to highly structured topologies of more than a hundred nodes. Finally, conclusions and future work are presented in Section 6.

2 Notation and Problem Formalization

We introduce the notation used to formalize the problem:

- S_D is the set of sites where the switch equipments can be installed; these sites also will be called potential switch sites or backbone sites. The number of the switch sites is given by $n_D = |S_D|$.
- $S_D^{(I)} \subseteq S_D$ is a distinguished subset of switch sites, which will always be included in the backbone network topology (usually because they are the access points for some local access subnetworks). We usually call these the *fixed sites* of the backbone. The nodes in $S_D - S_D^{(I)}$ are called Steiner nodes.
- $R = \{r_{ij}\}_{i,j \in S_D^{(I)}}$ is a matrix of requirements of connection between pairs of sites of $S_D^{(I)}$. We will require r_{ij} node disjoint paths between fixed sites i and j, where r_{ij} usually is strictly greater than 1.
- $C = \{c_{ij}\}_{i,j \in S_D}$ is the matrix which gives for any pair of sites of S_D, the cost of laying a line between them. When the direct connection among both places is not possible, we take $c_{ij} = \infty$.
- $E = \{(i,j); \forall i \in S_D, \forall j \in S_D / c_{ij} < \infty\}$, this is the set of feasible connections between two switch sites.
- $G_B = (S_D, E)$ is the graph of feasible connections on the Backbone Network.

We define our Backbone Network Design Problem $BNDP(S_D, E, C, R)$ as the problem of finding a subgraph $\mathcal{H}_B$ of G_B of minimum cost such that $\mathcal{H}_B$ satisfies the connection requirements specified in R.

3 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is a well known metaheuristic, which has been applied for solving many hard combinatorial optimization problems with very good results [5, 6]. A GRASP is an iterative process, where each GRASP iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is explored by local search. The best solution over all GRASP iterations is returned as the result.

Figure 1 illustrates a generic GRASP implementation pseudo-code. The GRASP takes as input parameters the candidate list size, the maximum number of GRASP iterations and the seed for the random number generator. After reading the instance data (line 1), the GRASP iterations are carried out in lines 2-6. Each GRASP iteration consists of the construction phase (line 3), the local search phase (line 4) and, if necessary, the incumbent solution update (lines 5 and 6).

```
Procedure GRASP(ListSize,MaxIter,RandomSeed);

1   InputInstance();
2   for k = 1 to MaxIter do
3      InitialSolution = ConstructGreedyRandomizedSolution(ListSize, RandomSize);
4      LocalSearchSolution = LocalSearch(InitialSolution);
5      if cost(LocalSearchSolution) < cost(BestSolutionFound) then
6           UpdateSolution(BestSolutionFound, LocalSearchSolution);
7   end_for;
8   return BestSolutionFound;
```

Figure 1: GRASP pseudo-code

In the construction phase, a feasible solution is built, one element at a time. At each step of the construction phase, a candidate list is determined by ordering all non already selected elements with respect to a greedy function that measures the (myopic) benefit of including them in the solution. The heuristic is adaptive because the benefits associated with every element are updated at each step to reflect the changes brought on by the selection of the previous elements. Then one element is randomly chosen from the best candidates list and added into the solution. This is the probabilistic component of GRASP, which allows for different solutions to be obtained at each GRASP iteration, but does not necessarily jeopardize the power of the adaptive greedy component.

The solutions generated by the construction phase are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Hence, it is beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution from its neighborhood. It terminates when there is no better solution found in the neighborhood. The local search algorithm depends on the suitable choice of a neighborhood structure, efficient neighborhood search techniques, and the starting solution. The construction phase plays an important role with respect to this last point, since it produces good starting solutions for local search. Normally, a local optimization procedure, such as a two-exchange, is employed. While such procedures can require exponential time from an arbitrary starting point, empirically their efficiency significantly improves as the initial solutions improve. Through the use

of customized data structures and careful implementation, an efficient construction phase that produces good initial solutions for efficient local search can be created. The result is that often many GRASP solutions are generated in the same amount of time required for the local optimization procedure to converge from a single random start. Furthermore, the best of these GRASP solutions is generally significantly better than the solution obtained from a random starting point.

4 GRASP for the BNDP

At this point, we will introduce the customization of GRASP methodology for its application on the BNDP. We apply the concepts of GRASP to the approximate solution of the BNDP, using a path-based construction phase and a tree-based local search, which are presented below.

4.1 Solution Construction Phase

The algorithm takes as input the network G_B of feasible connections on the backbone network, the matrix R of connection requirements between fixed switch sites of $S_D^{(I)}$, and the matrix of connection costs C. The current solution is initialized with the fixed sites without any connection among them. An auxiliary matrix M is initialized with the values of R. This is used with the purpose of maintaining on each step the connection requirements not yet satisfied between sites of $S_D^{(I)}$. The paths found on each iteration are stored in a data structure $\mathcal{P}$.

Iteratively the construction phase searches for node-disjoint paths between fixed sites of $S_D^{(I)}$ that have not yet satisfied their connection requirements. The algorithm chooses on each iteration such a pair of fixed switch sites $s_w^i, s_w^j \in S_D^{(I)}$ The current solution is updated by adding a new node-disjoint path between the chosen sites. For this, we employ an extension for the Takahashi-Matsuyama algorithm [15] in order to compute efficiently the k shortest node-disjoint paths from s_w^i to s_w^j. These paths are stored in a restricted candidate list $\mathcal{L}_p$. A path is randomly selected from $\mathcal{L}_p$. This process is repeated until all the connection requirements have been satisfied; then the feasible solution $\mathcal{G}_{sol}$ and the set of node-disjoint paths $\mathcal{P}$ are returned. The pseudo-code of BNDP_Construction_Phase is given in Figure 2.

```
Procedure BNDP_Construction_Phase(G_B, R, C);

1  G_sol ← (S_D^(I), ∅); M ← R; P_ij ← ∅ ∀ s_w^i, s_w^j ∈ S_D^(I);
2  while ∃ m_ij > 0 do
3     Let s_w^i, s_w^j ∈ S_D^(I) be two sites that satisfy m_ij > 0.
4     L_p ← the k shortest node-disjoint paths from s_w^i to s_w^j computed by GTMA
            (our Generalization of Takahashi-Matsuyama algorithm);
5     p ← Select_Random(L_p);
6     G_sol ← G_sol ∪ {p}; P_ij ← P_ij ∪ {p}; M ← Update_Matrix(G_sol, M, R, p);
7  end_while;
8  return G_sol, P;
```

Figure 2: Construction Phase pseudo-code

4.2 Local Search

The local search strategy proposed is a generalization of the key-path based local search, proposed by Verhoeven, Severens and Aarts [16]. A key-path is a path such that its end nodes are either non-fixed switch sites of degree at least 3, or fixed switch sites; and all its intermediate nodes are non-fixed switch sites of degree 2.

The algorithm takes as inputs the network G_B, the matrix R, the matrix C, the set $\mathcal{P}$ of found paths, and the decomposition in key-paths $K_p(\mathcal{G}_{sol})$. The algorithm searches the best neighbor solution, according to our definition of Neighborhood Structure, which we have called *Tree Neighborhood*. Iteratively the local search analyzes each key-path of the decomposition $K_p(\mathcal{G}_{sol})$, replacing (if it is possible) a key-path by a tree of smaller cost, maintaining the feasibility of the current solution. When there are no more improvements to be obtained by substituting key-paths by trees, the best feasible solution and the set of found paths are returned. The pseudo-code of BNDP_Local_Search is given in Figure 3.

5 Implementation and Performance Tests

We present here some experimental results obtained with the algorithm presented in the previous section.

```
Procedure BNDP_Local_Search(G_B,R,C,P,K_p(G_sol));

1   improve ← TRUE;
2   while improve do
3      improve ← FALSE;
4      for k = 1, ..., |K_p(G_sol)| do
5         Let p_k be the k-th key-path;
6         T ← the minimal key-tree computed by MKTA
                (our Generalization of Verhoeven-Severens-Aarts algorithm);
7         if (cost(T) < cost(p_k)) then
8            G_sol ← (G_sol \ p_k) ∪ T; P ← Update_Paths(P, p_k, T);
9            K_p(G_sol) ← Update_KeyPaths(K_p(G_sol), p_k, T);
10           improve ← TRUE;
11        end_if;
12     end_for;
13  end_while;
14  return G_sol,P;
```

Figure 3: Local Search Phase pseudo-code

The algorithm was implemented in ANSI C language. Adjacency matrices were used as data structures for the representation of graphs. The experiments were obtained on a Pentium III computer with 800 MHz, and 256 MB of RAM memory, running under Windows NT 4 operating system.

All instances were solved with identical parameter settings. The candidate list size was $ListSize = 10$, and the maximum number of iterations $MaxIter = 100$. These values were chosen from GRASP reference literature [5, 6].

We scanned previous literature and selected four problem instances with relatively important size (more than 100 nodes in three cases). In the four cases an optimal solution has been published [10, 11, 14]; this fact allows us to study the effectiveness of GRASP. In addition, we introduce another instance which was designed constructively applying certain topological properties related to graphs with triangular inequality. Figure 4 shows the topologies associated to the test cases 1, 2, 3 and 5. The red (or light grey) nodes represent the fixed switch sites, while the black nodes represent other potential switch sites, which may or may not be included in the solution (Steiner nodes). In instances 1 and 2 a link between non-fixed sites has cost 1, a link between a non-fixed site and a fixed site has cost 2 and a link between two fixed sites has cost 4. In instance 3 the links have costs randomly selected in the interval $[1, 200]$, satisfying the triangular inequality.

- Network 1 has a double grid structure, with 33 fixed sites, 87 Steiner sites and 286 feasible connections. The objective is to find a 2-node-survivable network with minimal cost (all connection requirement between fixed sites are equal to 2).
- Network 2 represents a simplified version of a HSODTN (High Speed Optical Data Transmission Network) connecting different parts of a war ship, allowing for communication between weapon systems. The objective is to find a 3-node-survivable network with minimal cost. This model and other variants can be found in [7, 14].
- Network 3 has 22 fixed sites of $S_D^{(I)}$, 61 Steiner sites and 262 feasible connections. The objective is to find a 4-node-survivable network spanning $S_D^{(I)}$ (all connection requirement between fixed sites are equal to 4).
- Networks 4 and 5 are test cases based on real networks from Bell Communications Research (Bellcore) [2, 14]. Particularly, we have selected the problems called LATAD5S and LATADL respectively, in order to test the performance of our algorithm. The objective is to find a 2-node-survivable network with minimal cost. The LATA5S-problem has 38 fixed sites and 71 feasible connections, and the LATADL-problem has 116 fixed sites and 173 feasible connections.

Optimal solutions for Network 1 and Network 2 have been published in [12]; the respective minimum costs are 140 and 74 respectively. These optimal topologies were found by an exact parallel-distributed backtracking algorithm. For Network 4 and Network 5, optimal solutions have been published in [2, 14], with costs 4739 and 7400. For Network 3, we know an optimal solution of cost 680, obtained from the constructive procedure used to build this test case.

The runs performed on these instances proved to be successful, the optimal solution being obtained by GRASP for the first four cases, and a sub-optimal solution with percent relative error of 0.60 for the last case. The following table shows some data about the performance of our GRASP algorithm for each instance. The entries correspond to the test case names, the iteration number where the best feasible solution was found, the optimum cost and the cost of the best feasible solution found by our algorithm, and the percentual gap between these costs.

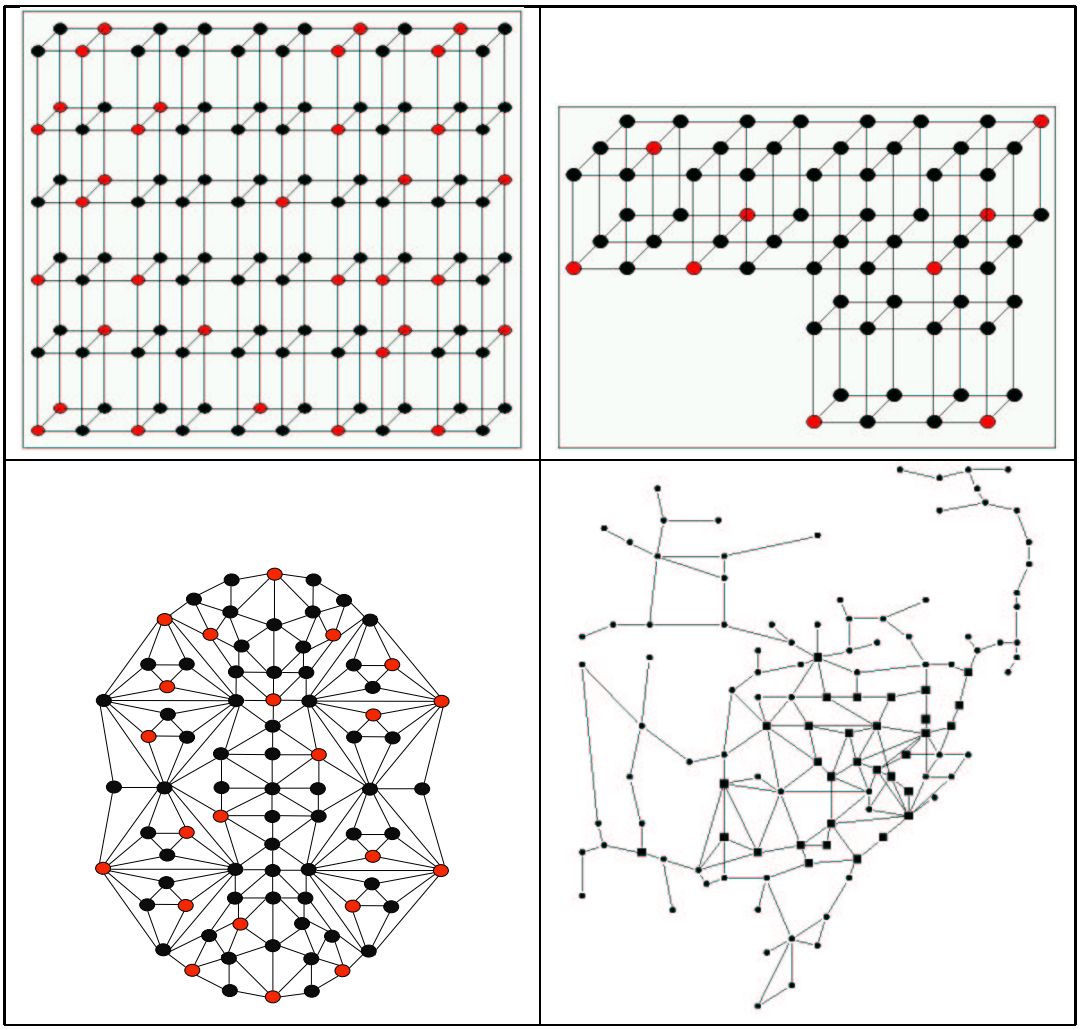

Figure 4: Topologies associated to the instances 1, 2, 3 and 5.

Topology	Best Iteration	Optimal cost	Best found	Gap
Network 1	5	145	145	0
Network 2	5	74	74	0
Network 3	7	680	680	0
Network 4	6	4739	4739	0
Network 5	14	7400	7445	0.60

6 Conclusions

We described a greedy randomized search adaptive procedure for the Backbone network design problem with heterogenous connectivity requirements. This problem can be modeled as a Generalized Steiner Problem with node-connectivity requirements.

The implementation of our algorithm was tested on different problems taken from literature related to the problem, and was shown to find good quality solutions within few iterations (in most cases an optimal solution was found; the only exception was one case where the cost of the best solution found was within 0.61% of the optimum). These are interesting results considering that to compute the best solution is a NP-Hard problem [14, 18]. We noted that, as expected, the running times increase with the number the sites, links, and the connection requirements.

This is (up to our knowledge) the first results published about the application of methauristics for solving the Backbone Network Design Problem in the general case with node connectivity constraints.

Further work can explore different local search strategies, which may improve solution quality. At present we

are experimenting with an alternative local search strategy, based on paths, which is simpler than the one presented in this paper; preliminary results are encouraging.

7 Acknowledgments

This work is a result of the PAIR project, funded by the INRIA, France. It has also been partially funded by the PDT program (MEC, Uruguay).

References

[1] A. Agrawal, P. Klein, and R. Ravi, "When trees collide: An approximation algorithm for the Generalized Steiner Problem in Networks" report CS-90-32 (1994), Department of Computer Science, Brown University.

[2] Bellcore, "FIBER OPTIONS" http:\\www.bellcore.com (1988).

[3] M.Baïou and A.R. Mahjoub, "Steiner 2-edge-connected subgraph polytopes on series-parallel graphs", (1993) SIAM Journal on Discrete Mathematics.

[4] M. Baïou, "Le problème du sous-graphe Steiner 2-aréte-connexe: approche polyédrale", Phd. Thesis, University of Rennes 1 (1996).

[5] T.A. Feo and M.G.C Resende, "A probabilistic heuristic for a computionally difficult set covering problem", Operations Research Letters, 8:67-71, (1989).

[6] T.A. Feo and M.G.C Resende, "Greedy randomized adaptive search procedures", Jounal of Global Optimization, 6:109-133, (1995).

[7] M. Grötschel, C.L. Monma, and M. Stoer, "Polyhedral and computational investigations for designing communication networks with high survivability requirements", Operations Research 43 (1995).

[8] A.R. Mahjoub, "Two-edge-connected spanning subgraphs and polyhedra", Mathematical Programming 64 (1994), pages 199-208.

[9] M. Priem and F. Priem, "Ingénierie des WAN", ISBN 2-10-004510-5, Dunod InterEditions (1999).

[10] F. Robledo, "Diseño topológico de redes : casos de estudio *The generalized Steiner problem* y *The Steiner 2-Edge-Connected subgraph problem*". Master Thesis. InCo, PEDECIBA Informática, (2000). 147 p. technical report RT 00-08, Facultad de Ingeniería, Universidad de la República, J. Herrera y Reissig 565, Montevideo, URUGUAY.

[11] F. Robledo and O. Viera, "An Ant-System algorithm for the Generalized Steiner Problem with edge-connectivity", Internal Report 1503, IRISA, (2002).

[12] F. Robledo and O. Viera, "A parallel algorithm for the Steiner 2-edge-survivable network problem", Internal Report 1504, IRISA, (2002).

[13] K. Stiglitz, P. Weiner, D. J. Kleitman, "The design of minimum-cost survivable networks", IEEE Trans. on Circuit Theory, CT-16, 4 (1969) pages 455-460.

[14] M. Stoer, "Design of Survivable Networks", Lecture Notes in Mathematics, ISBN 3-540-56271-0, ISBN 0-387-56271-0, Springer-Verlag, (1996).

[15] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs", Math. Jpn., 24:537-577, (1980).

[16] M.G.A. Verhoeven, M.E.M. Severens, and E.H.L. Aarts, "Local search for Steiner trees in graphs", In *Modern Heuristics Search Methods*, V.J. Rayward-Smith et al. (Eds.), John Wiley, 117-129, (1996).

[17] P. Winter, "Generalized Steiner problem in series-parallel networks", Journal of Algorithms 7 (1986), pages 549-566.

[18] P. Winter, "Steiner problem in networks: A survey", Networks 17 (1987), pages 129-167.