to appear in IEIEC Transactions (2005)
improvement from paper in LACGA 2004,
Santiago, aug. 2004

A GRASP algorithm with RNN based local search for designing a WAN access network

Héctor Cancela [a,1], Franco Robledo [a,b,1,2] and Gerardo Rubino [b,2]

[a] *Instituto de Computación, Fac. de Ingeniería, Universidad de la República*
J. Herrera y Reissig 565, Montevideo, Uruguay

[b] *IRISA/INRIA*
Campus de Beaulieu, Rennes 35042 CEDEX, France

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 the access network topology of a Wide Area Network (WAN). This problem is NP-hard, and can modeled as a variant of the Steiner Problem in Graphs.

The proposed GRASP employs a Random Neural Network (RNN) model in the local search phase, in order to improve the solutions delivered by the construction phase, based on a randomized version of the Takahashi-Matsuyama algorithm. Experimental results were obtained on 155 problem instances of different topological characteristics, generated using the problem classes in the SteinLib repository, and with known lower bounds for their optima. The algorithm obtained good results, with low average gaps with respect to the lower bounds in most of the problem classes, and attaining the optimum in 40 cases (more than 25% of the problem set).

Key words: metaheuristic; topological design; GRASP; RNN

1 Introduction

A wide area network (WAN) can be seen as a set of sites and a set of communication lines that interconnect the sites. A typical WAN is organized as a hierarchical structure integrating two levels: the *backbone network* and the *access network* composed of a certain number of *local access networks* (5).

[1] [cancela,frobledo]@fing.edu.uy
[2] [frobledo,rubino]@irisa.fr

Each local access network usually has a tree-like structure, rooted at a single site of the backbone network, and connects users (terminal sites) either directly to this backbone site or to a hierarchy of intermediate concentrator sites which are connected to the backbone 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.
Let S_C be the set of sites where concentrator equipment can be installed in order to diminish the cost of the access network and S_T the set of terminal sites (the clients). Considering the network of feasible connections on the WAN as a weighted, undirected graph, the Access Network Design Problem (ANDP) consists of finding a subgraph of minimum cost such that $\forall s_t \in S_T$ there exists a path from s_t to the backbone network (which can be represented by a single fixed node z). We introduce the notation used to formalize the problem:

- $S = S_T \cup S_C \cup \{z\}$ is the set of all nodes.
- $C = \{c_{ij}\}_{i,j \in S}$ is the matrix which gives for any pair of sites of S, the cost of laying a line between them. When the direct connection between i and j is not possible, we take $c_{ij} = \infty$.
- $E = \{(i,j); \forall i,j \in S$ such that $c_{ij} < \infty\}$, this is the set of feasible connections between sites of S.
- $G = (S, E)$ is the graph of feasible connections on the Access Network.

We define our Access Network Design Problem ANDP$(G(S,E),C)$ as the problem of finding a subgraph $\mathcal{T} \subset G$ of minimum cost such that $\forall s_t \in S_T$ there exists a unique path from s_t to node z and such that terminal sites can not be used as intermediate nodes (they must have degree 1 in the solution).

This problem belongs to the NP-Hard class (this can be proved by reducing the Steiner Problem in graphs to it). Some references in this area and related problems are (5; 6). This paper proposes a polynomial time heuristic based on the GRASP methodology and using a RNN model in the GRASP local search phase for approximately solving the ANDP. Section 2 presents the GRASP metaheuristic, the RNN model, and their customizations to solve the ANDP. Section 3 includes computational results obtained on a set of 155 problem instances, including topologies with hundreds of nodes.

2 GRASP/RNN descriptions and customization for the ANDP

GRASP is a well known metaheuristic, which has been applied for solving many hard combinatorial optimization problems with very good results. A GRASP is an iterative process, in which each 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. Details of this metaheuristic can be seen in (4).

The Random Neural Network introduced by Gelenbe (1) is a novel model, which has been applied with success to different optimization problems (2; 3). In a RNN signals circulate between a set of neurons, and are either positive or negative. Neurons have a *potential* which is a non-negative integer; it is increased (resp. decreased) by 1 when a positive (resp. negative) signal arrives. It is also decreased by one when the neuron fires. The neuron is *excited* if its potential is strictly positive, and then it fires after i.i.d. exponentially distributed periods; firing means sending a signal (positive or negative) to another neuron, or outside. Signals coming from outside form Poisson processes. This model is parameterized by the following elements: the number n of neurons, the firing rate r_i of neuron i, the probability p^+_{ij} (resp. p^-_{ij}), for a signal sent by i, to go to j as a positive (resp. negative) one, the probability $d_i = 1 - \sum_j (p^+_{ij} + p^-_{ij})$ of the signal to go outside, and the exogenous rates α_i and β_i of the signal flows of positive and negative units arriving at i. E. Gelenbe proved in (1) that the probability q_i that neuron i is excited, in steady state, is given by $q_i = \lambda^+_i / (r_i + \lambda^-_i)$, where λ^+_i and λ^-_i are the mean throughputs of positive and negative units at i. We have $\lambda^+_i = \sum_{j=1}^n q_j \varrho^+_{ji} + \alpha_i$, $\lambda^-_i = \sum_{j=1}^n q_j \varrho^-_{ji} + \beta_i$, where the *weights* are $\varrho^+_{ji} = r_i p^+_{ji}$, $\varrho^-_{ji} = r_i p^-_{ji}$. Gelenbe also gave the stability conditions associated with this system. The computation of the q_i's is thus a fixed point problem. During this computation, if we get a value $q_i > 1$ then we force $q_i = 1$ until convergence (we say that neuron i is *saturated*).

Next, we apply the concepts of GRASP to the approximate solution of the ANDP, using a path-based construction phase and a RNN based local search, which are presented below.

2.1 Construction Phase

Our construction phase can be seen as a customized and randomized version of the Takahashi-Matsuyama algorithm (8), which is a heuristic for computing a (hopefully low cost) Steiner tree. The algorithm (shown in Figure 2) takes as inputs the network G of feasible connections on the access network and the matrix of connection costs C. The current solution $\mathcal{T}_{sol}$ is initialized with the node z. Iteratively the construction phase adds new terminal nodes to the current solution. On each iteration, the algorithm chooses randomly a terminal node $\bar{s}_t$ not yet included in the current solution $\mathcal{T}_{sol}$ and computes the k shortest paths from $\bar{s}_t$ to $\mathcal{T}_{sol}$ (using any standard k shortest paths algorithm). These paths are stored in a restricted candidate list $\mathcal{L}_p$. A path p is randomly (and uniformly) selected from $\mathcal{L}_p$ and added to the current solution. This process is repeated until all the terminal nodes have been added; then the feasible solution $\mathcal{T}_{sol}$ is returned.

Procedure ANDP_Construction_Phase;
Input: $G = (S, E)$, C;

$\forall s_t \in S_T$ **a unique identifier** n_t **is assigned;**
$\mathcal{T}_{sol} \leftarrow \{z\}$; $\mathrm{Y} \leftarrow \emptyset$;
while $(\mathrm{Y} \setminus S_T) \neq \emptyset$ **do**
  $\bar{s}_t \leftarrow \mathrm{ArgMax}\{n_t | s_t \in (S_T \setminus \mathrm{Y})\}$;
  $\mathcal{L}_p \leftarrow$ **the** k **shortest paths from** $\bar{s}_t$ **to** $\mathcal{T}_{sol}$;
  $p \leftarrow \mathrm{Select\_Random}(\mathcal{L}_p)$;
  $\mathcal{T}_{sol} \leftarrow \mathcal{T}_{sol} \cup \{p\}$; $\mathrm{Y} \leftarrow \mathrm{Y} \cup \{\bar{s}_t\}$;
end_while;
return $\mathcal{T}_{sol}$;
end ANDP_Construction_Phase;

Procedure ANDP_Local_Search;
Input: $\mathcal{W}^+ = \{\varrho^+_{ij}\}$, $\mathcal{W}^- = \{\varrho^-_{ij}\}$, $\mathcal{T}_{sol}$, $G = (S, E)$, C;

$I \leftarrow \textsc{concentrators}(\mathcal{T}_{sol})$; $J \leftarrow S_T \cup \{z\}$;
$\mathcal{T}_{best} \leftarrow \mathcal{T}_{sol}$;
while $(I \cup J) \neq S$ **do**
  $q_s \leftarrow 1, \forall s \in J$;
  Compute the solution to the equation:
    $F(q) = q$ **given by the Gelenbe Theorem;**
  $s_c \leftarrow \mathrm{ArgMax}\{q_s | s \in (S_C \setminus J)\}$; $J \leftarrow J \cup \{s_c\}$;
  if $(s_c \notin I)$ **then**
    $\hat{S}_C \leftarrow \textsc{concentrators}(J)$;
    $\mathcal{H} \leftarrow$ **subgraph induced by** $(I \cup \hat{S}_C \cup \{z\})$;
    $\hat{\mathcal{H}} \leftarrow$ **connected comp. of** $\mathcal{H}$ **such that** $z \in \hat{\mathcal{H}}$;
    $\mathcal{T} \leftarrow$ **minimal spanning tree for** $\hat{\mathcal{H}}$;
    Compute $\forall s_t \in S_T$:
      $e \leftarrow$ **the edge of minimum cost from** s_t **to** $\mathcal{T}$;
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{e\}$;
    Iteratively remove all concentrator sites from $\mathcal{T}$ **with degree 1;**
    if $\textsc{cost}(\mathcal{T}) < \textsc{cost}(\mathcal{T}_{best})$ **then** $\mathcal{T}_{best} \leftarrow \mathcal{T}$;
  end_if;
end_while;
return $\mathcal{T}_{best}$;
end ANDP_Local_Search;

Fig. 1. Construction and Local Search pseudo-codes.

2.2 Local Search Phase

We propose a procedure that differs substantially from a classical local search. The algorithm (shown in Figure 2) takes as inputs the solution $\mathcal{T}_{sol}$ (computed in the Construction Phase), the network $G = (S, E)$ and the matrix of connection costs C. The algorithm searches for a better solution using the underlying neural network with the objectives of determining iteratively the order in which to analyze each concentrator node non-present in $\mathcal{T}_{sol}$ and of evaluating the benefit of its inclusion in the current solution. On each iteration, the concentrator node (non belonging to $\mathcal{T}_{sol}$ and that has not been previously analyzed) selected as potential improver will be that one whose associate neuron has greater value of q_j (asymptotically "the most excited one"). When all the concentrators have been evaluated, the best solution found is returned.

The underlying RNN is defined as follows. There exists a neuron for each node of S. The values for the excitatory and inhibitory rates are defined as: $\varrho^+_{ij} = \bar{c}/c_{ij}$ if $(i, j) \in E$, $\varrho^-_{ij} = 1$ if $(i, j) \notin E$ and $r_i = \sum_j (\varrho^+_{ij} + \varrho^-_{ij})$, where $\bar{c}$ is the average edge cost in graph G. The other rates are zero and exogenous signals do not exist. In this way, if a neuron is excited and has low connection costs with their neighboring neurons (i.e. high excitatory rates), it will have a greater excitatory influence on its neighborhood (the adjacent neurons). For the nodes that necessarily will integrate the solution (these are $S_T \cup \{z\}$) the associated neurons are artificially excited by means of the assignment $q_i = 1, \forall i \in S_T \cup \{z\}$.

3 Performance Tests and Conclusions

We present here some experimental results obtained with the GRASP-RNN algorithm. The algorithm was implemented in ANSI C. The experiments were obtained on a Pentium IV with 1.7 GHz, and 1 Gbytes of RAM, running under Windows XP. All instances were solved with identical GRASP 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.

We used a large test set, by modifying the Steiner Problem instances from SteinLib (7). This library contains many problem classes of widely different graph topologies. We considered all problems in all classes; for each problem, we selected the terminal node of the original problem with greatest degree as the z node; the Steiner nodes as concentrator sites, and the terminal nodes as terminal sites. Also, all the edges between terminal sites were deleted (as they are not allowed in feasible ANDP solutions). If the resulting topology was unconnected, the problem instance was discarded. By this process, we obtained 155 ANDP instances. Notice that, since in the ANDP the terminals cannot be used as intermediate nodes the cost of an SPG optimum is a lower bound for the optimum of the corresponding ANDP.

Table 1 shows a summary of computational results. The first column contains the names of the original Steinlib classes and the entries from left to right are: the number of customized instances (NI), the size of the selected instances in terms of number of nodes and edges respectively, the number of instances where the lower bound was obtained reaching therefore the optimum (NOPT), the average of the improvement of the results of the local search phase over the construction phase (Avg. LSI), the average running time per iteration, and the average of the gap of the GRASP solution respect to the lower bound (Avg. LB_GAP). The average improvement is computed as Avg. LSI= $\sum_{p \in Set} \mathrm{LSI}(p)/\mathrm{NI}$, where for problem p, $\mathrm{LSI}(p) = 100 \times [(\sum_{i=1}^{MaxIter} (CC_i - LC_i)/CC_i)]/MaxIter$, CC_i and LC_i being the costs of the solutions delivered in iteration i by the Construction Phase and the Local Search Phase respectively. The average gap is Avg. LB_GAP= $\sum_{p \in Set} \mathrm{LB\_GAP}(p)/\mathrm{NI}$ (where for problem p, $\mathrm{LB\_GAP}(p) = 100 \times (Best\_Cost\_Found - Lower\_Bound)/Lower\_Bound$).

The results show that the algorithm finds in most cases good quality solutions. In 40 instances (out of 155) we reached the lower bound and therefore optimality. As in general only lower bounds and not true optima are known, it is natural that a gap persists in many cases; as shown in the table, with wide variations depending on the problem class. Even then, in most cases this gap is quite small (less than 5% gap average in 7 over 12 problem classes).

Testset	NI	Nodes	Edges	NOPT	Avg. LSI	Avg. secs/itr	Avg. LB_GAP
C	6	500	625-2500	-	19.95%	12.13	0.41%
MC	2	97-120	4656-7140	1	24.89%	2.07	5.90%
X	2	52-58	1326-1653	-	11.00%	0.73	39.56%
PUC	4	64-128	192-750	2	21.04%	1.27	0.14%
I080	57	80	120-3160	13	17.01%	0.85	9.09%
I160	18	160	240-2544	7	21.46%	3.18	3.23%
I320	10	320	480-10208	2	24.93%	9.2	2.28%
I640	10	640	960-4135	2	24.33%	27.15	3.01%
P6E	10	100-200	180-370	2	23.75%	1.83	16.49%
P6Z	5	100-200	180-370	1	22.01%	1.10	23.22%
WRP3	15	84-886	149-1800	7	20.3%	17.00	0.00028%
WRP4	16	110-844	188-1691	3	32.36%	22.56	0.00109%

Table 1: Computational results.

Another point of interest is that the RNN model in the local search phase was used with the aim of capturing global connectivity information about the graph and to determine the order in which the concentrator nodes non-present in the solution delivered by the construction phase are chosen to improve the solution delivered by the greedy construction phase. We observe that for all problem classed, the local search phase improved significantly the solutions built by the construction phase; over 20% average improvement for most problem classes (and always over 10% average improvement).

References

[1] E. Gelenbe, "Stability of the random neural network model", Neural Computation, vol. 5, no. 2, pp. 239-247 (1990).

[2] E. Gelenbe and F. Batty, "Minimum cost graph covering with the random neural network", in Computer Science and Operations Research. New York: Pergamon, pp. 139-147 (1992).

[3] E. Gelenbe, V. Koubi, and F. Pekergin, "Dynamical random neural network approach to the traveling salesman problem", in Proc. Symp. Syst., Man., Cybern., pp. 630-635 (1993).

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

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

[6] C. D. Randazzo, H. P. L. Luna and P. Mahey, "Benders decomposition for local access network design with two technologies", Discrete Math.& Theoretical Comp. Science, vol. 4 no. 2, pp. 235-246 (2001).

[7] http://elib.zib.de/steinlib/testset.php (last access: April 28, 2004).

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