A Simulated Annealing Algorithm for The Forwarding Index of Graphs

Gabor Loerincs
Departamento de Computación
Universidad Simón Bolívar
Aptdo. 89000, Caracas 1080-A, Venezuela
gloerinc@usb.ve

Abstract

For a given undirected graph G, a routing R is a collection of elementary paths connecting every pair of vertices in G. The load $\xi(G, R, v)$ of a vertex v in the routing R is the number of paths of R for which v is an interior vertex. The forwarding index $\xi(G, R)$ of the graph G respect to the routing R, is defined as the maximum of the loads $\xi(G, R, v)$ for all vertex v of G. The forwarding index $\xi(G)$ of the graph G, is the minimum of the forwarding indices $\xi(G, R)$ taken over all possible routings R. In this paper, we develop an algorithm based in simulated annealing to compute the forwarding index of graphs.

Experimental results show that this algorithm behaves well, finding optimal solutions in most cases and performs better than a genetic algorithm suited for the problem.

Keywords: Heuristic Algorithms, Forwarding Index, Simulated Annealing, Genetic Algorithm.

1 Introduction

Let $G(V, E)$ be a simple connected graph on n vertices. $V(G)$ denotes the vertex set of G. A *routing* of G is a set $R = \{R_{xy} : u, v \in V(G), u \neq v\}$ of elementary paths connecting each ordered pair (u, v) (thus, R_{uv} is not necessarily the same as R_{vu}). If all the paths R_{uv} of R are shortest paths from u to v, we say that we have a *routing of shortest paths* and denote it by R_m.

Usually, a communication network is defined as a graph: vertices and edges represent nodes and links of the network. Also, since the purpose of a network is communicating data, it is necessary to provide information about how to interconnect any two nodes. This way, a network can be defined as a pair (G, R), where G is the associated graph and R is a routing that specifies the way to connect each node pair. Chung et al. introduced the notion of forwarding index, as a parameter that measure in some sense, the load or congestion of a network [4].

The *load* $\xi(G, R, v)$ of a vertex v in the routing R is defined as the number of paths of R for which v is an interior vertex. The *forwarding index of the pair* (G, R) is the maximum number of paths of R going through any vertex v in G and is denoted $\xi(G, R)$:

$$\xi(G, R) = \max_{v \in V(G)} \xi(G, R, v)$$

The *forwarding index of* G, denoted by $\xi(G)$, is defined as the minimum of the forwarding indices $\xi(G, R)$ taken over all possible routings:

$$\xi(G) = \min_{R} \xi(G, R)$$

The *forwarding index by routing of shortest paths*, denoted by $\xi_m(G)$, is defined as the minimum of the forwarding indices $\xi(G, R_m)$ taken over all possible routings of shortest paths:

$$\xi(G) = \min_{R_m} \xi(G, R_m)$$

The paper is organized as follows. Section 2 presents some theoretical results about forwarding indices. In section 3, we propose a simulated annealing algorithm to solve the forwarding index and forwarding index by routing of shortest paths. To evaluate the solution quality of the algorithm, we test it in some specific graphs with known forwarding indices. The result of this evaluation is presented in section 4. Section 5 presents an experimental comparision between the simulated annealing algorithm and a genetic algorithm proposed by Barráez and Domínguez [3]. We end in section 6 with a summary of the main results and topics for future work.

2 Some theoretical results

The forwarding index problem is NP-complete; no algorithm is known to solve the problem in polynomial time, except for the particular case for routing of shortest paths for graphs of diameter 2 [2, 5]. However, among many, the following equalities and bounds are well known:

Proposition 2.1 (Chung, Coffman, Reiman, Simon [4]) *Let G be a connected graph of order n. Then*

$\frac{1}{n} \sum_u \sum_{v \neq u} (\delta(u, v) - 1) \leq \xi(G) \leq \xi_m(G) \leq (n-1)(n-2)$

where $\delta(u, v)$ denotes the length of a shortest path between u and v.

Proposition 2.2 (Heydemann, Meyer, Sotteau [5]) *For any cycle C_n of length n, $n \geq 3$,*

$$\xi_m(C_n) = \xi(C_n) = \left\lfloor \frac{(n-2)^2}{4} \right\rfloor$$

Proposition 2.3 (Heydemann, Meyer, Sotteau [5]) *If W_{n-1} is a wheel of order n, we have*

(i) $\xi_m(W_{n-1}) = n^2 - 7n + 6$ *for* $n \geq 7$

(ii) $(n-1)(1-\frac{4}{n}) \leq \xi(W_{n-1}) \leq \frac{(n-3)^2}{4}$

Proposition 2.4 (Heydemann, Meyer, Sotteau [5]) *If Q_p is the p-cube of order $n = 2^p$, then*

$$\xi(Q_p) = \xi_m(Q_p) = \frac{n}{2}\log_2 n - n + 1$$

3 Description of the algorithm

Simulated annealing is an approach that has demonstrated to be useful when used to solve difficult combinatorial problems. A detailed description of this technique and references to applications can be found in the works of E.H.L Aarts [1] and C. Reeves [7].

We develop our algorithm using the parameterized generic annealing algorithm summarized in figure 1.

1. Select an initial solution S and compute $c = cost(S)$. Compute an upper bound c^*.
2. Choose an initial temperature $T > 0$ so that in what follows the *changes/trials* ratio starts out approximately equal to $INITPROB$.
3. Set $freezecount = 0$
4. While $freezecount < FREEZELIM$ do

   4.1 Set $changes$=$trials$=0

   4.2 While $trials < SIZEFACTOR \cdot N$ and $changes < CUTOFF \cdot N$ do

      4.2.1 Set $trials = trials + 1$;

      4.2.2 Generate a random neighbor S' of S and compute $c' = cost(S')$

      4.2.3 Let $\Delta = c' - c$

      4.2.4 If $\Delta \leq 0$

         Set $changes = changes + 1$

         Set $S = S'$ and $c = c'$

         If S' is feasible and $cost(S') < c^*$ then set $S^* = S'$ and $c^* = cost(S')$

      4.2.5 If $\Delta > 0$

         Choose a random number r in [0,1].

         If $r \leq \exp^{-\Delta/T}$ then set $changes = changes + 1$ and set $S = S'$ and $c = c'$

   4.3 Set $T = TEMPFACTOR \cdot T$

      If c^* was changed during 4.2, set $freezecount = 0$:

      If $changes/trials < MINPERCENT$, set $freezecount = freezecount + 1$

5. Output S^* as the solution

Figure 1: The generic simulated annealing algorithm

Let G be a connected graph of order n. A solution is any routing R of G, that is, a set of $n(n-1)$ elementary paths. Two solutions are neighbors if one differs from the other in exactly one path. To generate a random neighbor, we randomly pick an ordered pair $(u, v), (u \neq v)$, generate a random path between u and v and replace it in the routing R for the actual path between u and v[1]. Note that when we apply this procedure on one routing, we obtain another solution, which is also a routing. Using this neighbourhood structure, we assure that every solution is reachable from every other.

At a first glance, we can consider the *cost* function of the algorithm to be the forwarding

[1]For the case of the forwarding index by routing of shortest paths, R is a routing of shortest paths and a random shortest path is generated between u and v

index $\xi(G, R)$ as defined in section 1. But this consideration have a serious drawback.

The function $\xi(G, R)$ returns the maximum of a list of values, i.e the load of each node of G. Let R_x and R_y two routings of the same graph G such that $\xi(G, R_x) = \xi(G, R_y)$.

In one extreme, the load of each vertex v in the routing R_x may be the same that the load of v in R_y. In the other extreme, the load of each vertex v in each routing may be different, except by the vertex with maximum load. In either case, the function $\xi(G, R)$ reports solutions R_x and R_y to "be the same". This lack of differenciability may result troublesome for the annealing process: we have the undesirable situation of large plateau-like areas in which changes will be accepted freely, and the algorithm is likely to wander around aimlessly with no guidance toward improved solutions.

To overcome this situation, we propose a different cost function that takes advantage of this differenciability. The cost is no more a number but a n-tuple where the ith position corresponds to the value $\xi(G, R, v_i)$, that is, the load of vertex v_i respect to the routing R.

To compute Δ, we first sort both costs c and c', from larger to smaller load, and then perform a lexicographic comparision. If the costs are lexicographically equals, Δ is 0. In the other case, let p the position where the first difference ocurred; Δ is the difference between the load in the pth position of c' and the load in the pth position of c.

Note that if a solution R is minimum respect to this cost function, R is also minimum respect to $\xi(G, R)$.

parameter	*value*
INITPROB	0.85
FREEZELIM	100
SIZEFACTOR	5
CUTOFF	5
TEMPFACTOR	0.975
MINPERCENT	0.05
T_0	≈ 5

Table 1: Parameter values used to perform the experiments.

4 Experimental results

We study the behavior of the algorithm for three graph families, whose forwarding index are known: the cycle C_n of order n, the wheel W_n of order $n+1$ and the hypercube Q_p of order 2^p.

Table 1 lists the parameter values used to perform the experiments. The value of N is determined using the formula $20 \cdot n$ where n is the graph order. (See [6] for observations about the values for these parameters and the interactions between them).

Table 2 shows the experimental results obtained from executing the algorithm using differents graphs, computing $\xi(G)$ and $\xi_m(G)$.[2] Table presents the optimal theoretical values and the best values found by the algorithm (denoted by $\tilde{\xi}(G)$ and $\tilde{\xi}_m(G)$).

The solutions found are optimal, or very close to optimal (with a maximum error of 4%). In the cases where no optimal solution is known, we obtained an improvement in the upper bound of aproximately 25%.

[2]The algorithm was implemented in C++ and ran on a PENTIUM 100MHz based computer under LINUX operating system

G	$\xi(G)$	$\tilde{\xi}(G)$	$\xi_m(G)$	$\tilde{\xi}_m(G)$
C_8	9	9	9	9
C_{12}	25	25	25	25
C_{20}	81	81	81	81
C_{40}	361	363	361	362
W_6	[3,4]	3	6	6
W_{10}	[7,16]	12	50	50
W_{12}	[9,25]	19	84	84
W_{20}	[16,81]	61	300	300
Q_3	5	5	5	5
Q_4	17	17	17	17
Q_5	49	51	49	49

Table 2: Experimental results. $\xi(G)$ and $\xi_m(G)$ columns list the theoretical values. $\tilde{\xi}(G)$ and $\tilde{\xi}_m(G)$ columns list the values computed by the algorithm.

5 Experimental comparision with a genetic algorithm

This section compares the performance of our algorithm with a genetic algorithm suited to solve the same problem. For this, we programmed a genetic algorithm according to Barráez and Domínguez [3]. Both programs share the same data structures and low-level routines.

Since both programs are based on heuristic algorithms, our principal concern is to evaluate how the solution evolves over the time. For this, we ran both programs with some graphs and tracked the values of the best solution. The results obtained are showed in figure 2. In each chart, the x-axis represents the execution time of the program; the y-axis value asociated to each x-axis value is the forwarding index of the best solution obtained so far.

At early stages of the execution, the genetic algorithm behaves equal or slightly better than the simulated annealing algorithm; both solutions decreases monotonicaly at the same

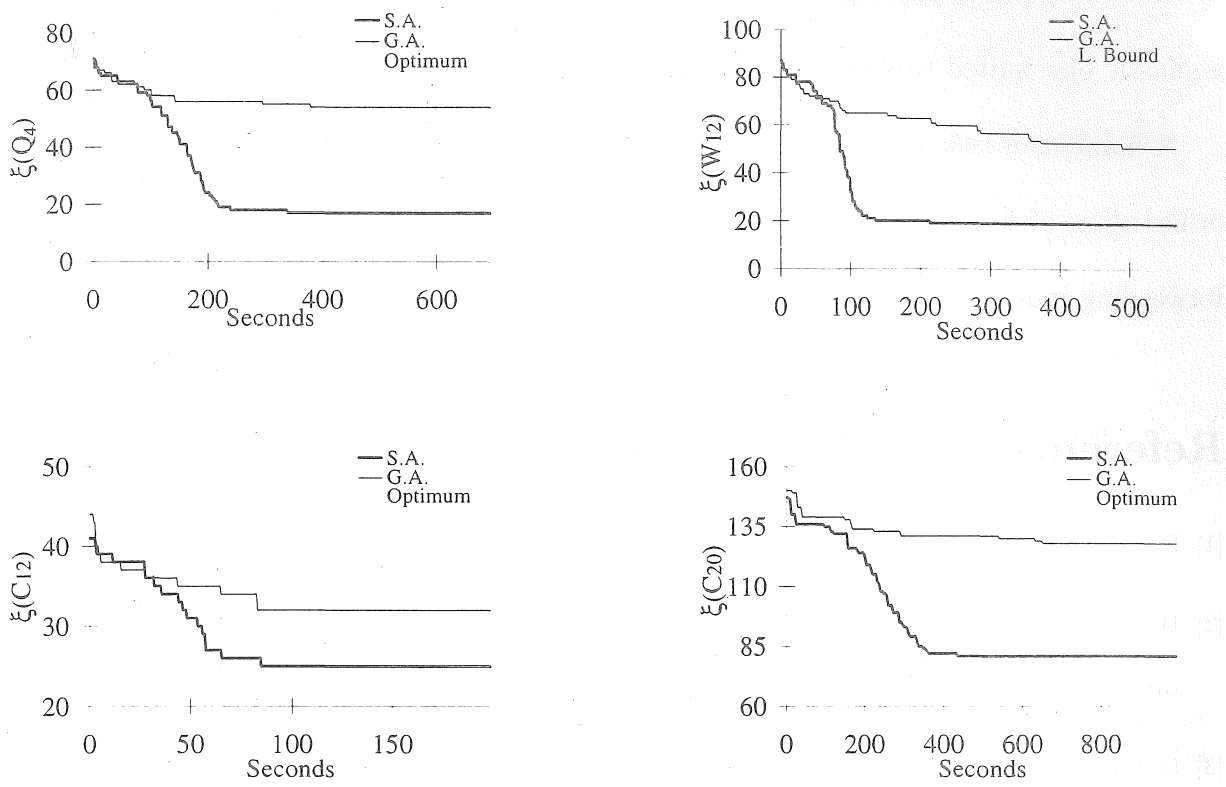

Figure 2: Experimental comparision with a genetic algorithm. For each second, we plot the best solution obtained so far by the simulated annealing algorithm (S.A.) and by the genetic algorithm (G.A.)

speed. But there is a point when the simulated annealing algorithm solution starts to improve significatively faster than genetic algorithm one, whose convergence speed remains the same. This results experimentaly show that our algorithm performs better than the genetic algorithm.

6 Conclusions

We have presented a simulated annealing algorithm for the forwarding index problem. Experiments show the algorithm behaves well, finding optimal or very close to optimal solutions

in some cases and improving upper bounds in the others. The algorithm peforms better than a genetic one, suited to solve the same problem.

A number of issues that we'd like to address in the future are uses of this algorithm in other types of forwarding index problems, like the forwarding diameter [8] and the edge-forwarding index [5].

References

[1] E.H.L Aarts and J.H.M Korst. *Simulated annealing and Boltzmann machines.* Wiley, Chichester, 1989.

[2] D. Barráez. *Diámetro de transmisión, ciclos dominantes y el problema clásico de colocaciones.* PhD thesis, Universidad Central de Venezuela, 1994.

[3] D. Barráez and R. O. Domínguez. A genetic algorithm for the forwarding index of graphs. In *Proceedings CLEI Panel 96*, pages 203–213, 1996.

[4] F. Chung, E. Coffman, M. Reiman, and B. Simon. The forwarding index of communication networks. *IEEE Transactions on Information Theory*, 33:224–232, 1987.

[5] M.C. Heydemann, J.C. Meyer, and D. Sotteau. On forwarding indices of networks. *Discrete Applied Mathematics*, 23:103–123, 1989.

[6] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: Part i. *Operations Research*, 37:865–892, 1989.

[7] C. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Publications, Oxford, 1993.

[8] W. Fernández De La Vega, M. El Haddad, D. Barráez, and O. Ordaz. The forwarding diameter of graphs. Submitted to Discrete Applied Mathematics.