Network design for OSPF routing

Luciana S. Buriol[1], Paulo M. França,[1]
Mauricio G.C. Resende[2], and Mikkel Thorup[2]

[1] Faculdade de Engenharia Elétrica e de Computação, UNICAMP, SP, Brazil.
{buriol,franca}@densis.fee.unicamp.br,
[2] Internet and Network Systems Research, AT&T Labs Research, Florham Park, NJ, USA.
{mgcr,mthorup}@research.att.com

Abstract. Internet protocol (IP) traffic follows rules established by routing protocols, such as Open Shortest Path First (OSPF). Each router computes shortest paths using weights assigned by the network operator, and creates destination tables used to direct each IP packet to the next router on the path to its final destination. Furthermore, the routing protocol is used to establish procedures to be taken in case of a failure in the network. In this extended abstract, we describe a new genetic algorithm for designing a network with minimal total link capacity necessary to route demand without overload in case of any single edge or node failure.

The Internet is made up of many routing domains, called autonomous systems (ASes). Internet protocol (IP) traffic flows follows rules established by routing protocols. Shortest path first protocols, such as Open Shortest Path First (OSPF), are the most commonly used Interior Gateway Protocols (IGPs). These routing protocols direct traffic based on link weights assigned by the network operator. Each router in the AS computes shortest paths and creates destination tables used to direct each IP packet to the next router on the path to its final destination.

It is also the role of the routing protocol to specify how the network should quickly react to changes in the topology of the AS, such as when transmission lines go out of service or come back on, routers crash, or network policies change [6]. In such situations, IP traffic is re-routed through the shortest paths not traversing the affected part of the network. If such links have insufficient bandwidth capacity, overload occurs.

To satisfy requirements for Quality of Service (QoS) in IP routing, it is desirable to design a network to easily handle a single link or router failure without causing overload. One possible solution is to maintain part of the link bandwidth free in the eventuality of failures.

This extended abstract addresses the issue of designing a OSPF-routed network with minimum total link capacity needed to route the required demand and handle any single (link or router) failure. We assume the topology is given but link capacities must be determined. The capacity is limited to a discrete set of values $0, m, 2m, 3m, \ldots$, i.e. multiples of a unit value m. For simplicity we consider $m = 1$. Our aim is to design an efficient network and routing scheme, i.e. determine OSPF link weights and link capacities such that the total capacity is minimized.

We propose a genetic algorithm for this problem, and apply it to two real-world problem instances. To the best of our knowledge, this problem has not yet been addressed in the literature.

1 A genetic algorithm for weight and capacity assignment

Let us represent a data communication network by a directed network graph $G = (V, E)$, where V and E denote, respectively, the sets of routers (nodes) and transmission links (arcs). Let us assume that each link $a \in E$ has capacity c_a and that we are given a demand matrix D that, for each pair $(s, t) \in V \times V$, specifies the demand d_{st} in traffic flow from node s to node t.

Given a set of traffic demands between origin-destination pairs [3], the *OSPF weight setting problem* consists in assigning positive integer weights $w_a \in [1, w_{\max}]$ to each arc $a \in E$, such that a cost function is optimized when the demands are routed according to the rules of the OSPF protocol.

In this section, we propose a genetic algorithm for assigning weights and capacities to links. Given a weight assignment $w_1, w_2, \ldots, w_{|E|}$, the values $m_1, m_2, \ldots, m_{|E|}$ should be defined such that a maximum link utilization restriction is satisfied. The link utilization rate is defined as the ratio between load on the link and the link capacity. The aim is to find a weight assignment which minimizes the total link capacity.

Let T be the set of destination nodes and let $g^t = (V, E^t)$ denote the shortest path graph associated with each destination node $t \in T$. The distance from each node to the destination t is stored in the $|V|$-vector d^t, while $|V|$-vector ρ^t stores the sum of the capacities of the outgoing links $a \in g^t$. For each pair (s, t) and each arc a, let l_a be the total load on arc a for all pairs (i, j) with $i, j \in V$, i.e., the sum of the flows going over a.

The genetic algorithm uses the same population structure proposed in [2] and later used in [1]. We next present how the capacities are computed. The maximum utilization for a network in normal operation (*MU*) and with failure (*MUf*) are given. Furthermore, *lf* and *rf* indicate, with values *1* (up) or *0* (down), the state of link and router failure, respectively.

Figure 1 presents a pseudo-code for determining the capacities. In the pseudo-code, procedure `defineMulti()` receives as input the values of *w*, *MU*, *MUf*, *lf* and *rf*. As output, the procedure returns the vector of link capacities. Initially, all links receive unit capacity and the solution data structures are populated (line 2). Lines 4–5 update the capacities to satisfy the maximum utilization constraint under normal operation, while in the remaining part of the loop in lines 3–8 the capacities are increased to satisfy the maximum utilization constraint under single-failure mode. Procedure `simulateFail()` returns *1* when at least one capacity has changed and *0* otherwise. Finally, in line 11, the capacities are summed up to determine the objective function value.

The function `updateMulti()` increases the capacities to satisfy the maximum utilization MU required. For each link $a = (\overrightarrow{u, v})$, its minimum capacity value m_* is set to $\lceil sol.l_a / (U * c_a) \rceil$. In case m_a is less than the minimum, it is set to m_*, and the vector ρ updated. The tail nodes of the arcs whose capacities were increased are inserted into the set L.

```
procedure defineMulti(w, MU, MUf, lf, rf)
1      for a ∈ E do m_a ← 1;
2      sol ← fillSolMemory(w, m);
3      do
4              L ← updateMulti(sol, m, MU);
5              if |L| > 0 then updateSol(w, sol, m, MU, L);
6              if rf = 1 then ac_1 ← simulateFail (sol, m, w, R, |V|, MUf);
7              if lf = 1 then ac_2 ← simulateFail (sol, m, w, E, |E|, MUf);
8      while ac_1 = 1 or ac_2 = 1;
11     fit ← ∑_{a∈E} m_a;
end defineMulti.
```

Fig. 1. Capacity determination for a weight vector w and maximum loads MU and MUf.

If the capacities are adjusted, the flow is reloaded according to the new capacity values. The new loads can require different capacity values. This process is repeated until no capacities or loads are adjusted, i.e. each link has at least the minimum capacity value required to not violate any restriction. This procedure is executed by function updateSol().

Procedure simulateFail() is called to simulate any link or router failure. Set R is computed once and contains $|V|$ vectors, each set R_i contains the set of disabled links due to the failure of router $i \in V$. If there is any single link or router failure, g^t is updated [1], the load recomputed and the capacities adjusted. This procedure stops when all failures are tested consecutively without causing the increase of any capacity.

2 Computational Results

This section presents preliminary experimental results using the genetic algorithm.

The experiments were performed on a 1.7 GHz Intel Pentium IV computer with 256 MB of RAM, running RedHat Linux 8.0. The codes were written in C, and compiled with the gcc compiler version 3.2, using the -O3 optimization option. CPU times were measured with the system function getrusage.

The GA parameters were set exactly as in Buriol et al. [1]. The population size was set to 50 and maximum link weight value $w_{\max} = 20$. For the experiments presented in this extended abstract, the stopping criterion is 500 generations and the maximum utilization for the network without failure is set to $MU = 0.80$.

The experiments were conducted on two real-world networks: att and attAS. Instance att, first used by Fortz and Thorup [4], corresponds to old data from the *AT&T Worldnet backbone network*, while instance attAS corresponds to a recent snapshot of the domestic U.S. AT&T backbone. The networks are summarized in Table 1.

Table 2 presents results for the GA applied to instance attAS using the basic objective function in the top half of the table, and another objective function in the bottom half. This second objective favors solutions that minimize total link length, i.e. it is a

Table 1. Network characteristics.

Class	Name	$|V|$	$|E|$	$|T|$	o-d pairs	$\sum d_{uv}$
AT&T WorlNet backbone	att	90	274	17	272	18465
AT&T AS/USA backbone	attAS	54	278	18	9900	153470

Table 2. Results for GA using instance attAS and two objective functions.

		$\sum_{a \in E} multi_a$								
		$MUf = 0.85$			$MUf = 0.90$			$MUf = 1.00$		
lf	rf	best	time	impr	best	time	impr	best	time	impr
0	0	343	50	23	343	50	23	343	50	23
0	1	614	599	23	595	605	21	545	610	23
1	0	619	1735	21	593	1736	21	543	1699	24
1	1	624	1891	24	590	1824	25	548	1892	26

		$\sum_{a \in E} multi_a * length_a$								
		$MUf = 0.85$			$MUf = 0.90$			$MUf = 1.00$		
lf	rf	best	time	impr	best	time	impr	best	time	impr
0	0	112565	50	59	112565	50	59	112565	50	59
0	1	218179	631	64	206329	634	65	193031	619	65
1	0	203845	1866	67	186463	1774	77	175367	1523	75
1	1	215127	1866	70	207103	1927	70	186434	1830	73

weighted version of the basic capacity minimization problem. For each objective function, three values of *MUf* were tested. The rows give the combination of failures, with the first row not considering any failure and last row considering both failures. Column *impr* shows the improvement achieved after 500 generations compared to the initial solution using random weights.

We observe that failures raise considerably the total required capacity. The required capacity is similar for both link or router failure, while the running times are longer for link failure. This increase in running time is understandable, since there are 278 links to test for failure, against only 54 routers. Looking at the last row, it is possible to verify that when both failures are considered, neither the total capacity nor the running time is affected much as compared to the results for single link failure.

Table 3 presents results for instance att considering the objective function $m_a * length_a$ and 12 distinct demand matrices [4]. In this test the time and solution increases with the increase of the traffic flow. The parameters are fixed at *MUf* = 0.90, *lf=1* and *rf=0*.

Analyzing these results, we conclude that both the solution quality and running time increase with the increase of the traffic flow.

Table 3. Results for GA using instance `att` and objective function $multi_a * length_a$.

inst	best	time	impr	inst	best	time	impr	inst	best	time	impr
att_1	89511	575	6	att_5	95089	1521	13	att_9	128556	1565	26
att_2	89948	672	7	att_6	97202	1507	22	att_10	144881	1581	23
att_3	90926	784	7	att_7	108824	1513	20	att_11	157154	1605	22
att_4	91672	1258	10	att_8	111820	1563	33	att_12	168583	1583	20

3   Concluding Remarks

This extended abstract presented a new network design problem and a genetic algorithm for network design that supports single link or router failures. The computational results were conducted using data from two real world network design problems, using different objective functions, maximum utilization, demand matrices, and link and router failure combinations. From the analysis of the results, we conclude that the time and solution values increase with traffic flow. Besides, they are greater for networks with failures. The solution values are similar when considering only link or router failure, while the times are longer for link failure. Considering both failures does not change much the solution value or time when compared with single link failure.

References

1. L.S. Buriol, C.C. Ribeiro M.G.C. Resende, and M. Thorup. A hibrid genetic algorithm for the weight setting problem in ospf/is-is routing. Technical Report TD-5NTN5G, AT&T Labs Research, 2003.
2. M. Ericsson, M. G. C. Resende, and P. M. Pardalos. A genetic algorithm for the weight setting problem in ospf routing. *J. of Comb. Opt.*, 6:299–333, 2002.
3. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, 9:265–279, 2001.
4. B. Fortz and M. Thorup. Increasing internet capacity using local search. Technical report, AT&T Labs Research, 2000. Preliminary short version of this paper published as "Internet Traffic Engineering by Optimizing OSPF weights," in Proc. 19th IEEE Conf. on Computer Communications (INFOCOM).
5. M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.
6. J.T. Moy. *OSPF, Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.