SURVIVABLE COMPOSITE-LINK IP NETWORK DESIGN WITH OSPF ROUTING

DIOGO V. ANDRADE, LUCIANA S. BURIOL, MAURICIO G.C. RESENDE, AND MIKKEL THORUP

ABSTRACT. OSPF, or Open Shortest Path First, is a commonly used interior gateway protocol. Given a network topology, a set of link types to be deployed, each having a different capacity, and predicted traffic demands, the problem considered in this paper is to find a set of OSPF weights that minimizes network cost subject to single arc failures. We propose a genetic algorithm to find near-optimal or optimal solutions for this problem. At each iteration (or generation) of the algorithm, OSPF weights are assigned to the arcs of each member of the population and an external procedure determines which links are to be deployed and the corresponding cost associated with the deployment. Four heuristics used to implement this external procedure are the main topic of this paper. They are designed to minimize the overall cost of the network, but some have additional constraints imposed by specific applications. We show the results of an experiment with the four heuristics on a real network with 54 routers and 278 arcs.

1. INTRODUCTION

OSPF, or Open Shortest Path First, is a commonly used interior gateway protocol. An integer weight is assigned to each arc and the entire network topology and arc weights are known to each router. Each router computes a graph of shortest weight paths from itself to every other router in the network. Traffic is routed on the shortest weight paths. All traffic leaving router s with destination router t is split among all arcs leaving s and on the shortest path graph from s to t.

Given a network topology, a set of link types to be deployed, each having a different capacity, and predicted traffic demands, the problem considered in this paper is to find a set of OSPF weights that minimizes network cost subject to single arc failures. More precisely, we are given a directed network $G = (N,A)$, where N is the set of routers, A is the set of potential arcs where capacity can be installed, and a demand matrix D that, for each pair $(s,t) \in N \times N$, specifies the demand $D_{s,t}$ between s and t. Arc $a \in A$ has length d_a. Link types are numbered $1,2,\ldots,T$, where link type i has capacity c_i and cost per unit of length p_i. We wish to determine integer OSPF weights $w_a \in [1,65535]$ as well as the number of copies of each link type to be deployed at each arc such that when traffic is routed according to the OSPF protocol in a no-failure or any single arc failure situation there is enough installed capacity to move all of the demand and the total cost of the installed capacity is minimized. For routing purposes, we assume that each arc has a single multiplicity, i.e. the installed capacity does not influence routing.

Given an arc weight assignment, OSPF determines, for a given demand matrix D, the loads l_a on each arc $a \in A$. The cost of the network is defined by the number of each type of link deployed on each arc in the network.

Date: January 5, 2006.

Key words and phrases. Internet, OSPF routing, network design, survivability, genetic algorithm.

AT&T Labs Research Technical Report: TD-6KRQN8.

We propose a genetic algorithm similar to the one described in [2] to find near-optimal or optimal solutions for this problem. The algorithm works with a population of solutions. Each solution is represented by an $|A|$-vector of integer weights. At each iteration (or generation) of the algorithm, weights are assigned to the arcs of each member of the population and an external procedure determines which links are to be deployed and the corresponding cost associated with the deployment. This external procedure is the main topic of this paper. The population evolves using the random keys crossover scheme of Bean [1]. The elements are classified according to their cost. The $x\%$ least-cost solutions are placed in category I and the $y\%$ highest-cost solutions are put in category III. The remaining solutions fall into category II. Population dynamics works as follows. All solutions in the initial population have randomly generated weights. All category I elements are promoted to the next generation unchanged. All category III elements are replaced by random weight elements in the next generation. The remaining elements of the population of the next generation are created as follows. A solution (parent) is selected at random from the set of category I solutions. Similarly, a parent is selected at random from the union of the sets of categories II and III. Parents can be selected more than once per generation. A crossover operation is performed on these two parent solutions to produce a child solution. Each weight of the child's arcs is that of the category I parent with probability π ($0.5 < \pi \leq 1$) and that of the other parent with probability $1-\pi$. Each weight is determined independently of the other. The crossovers are repeated until the population of the new generation is of the same size as the current generation.

2. HEURISTICS

Because of the limited space in this abstract, we restrict our discussion to the no-failure case. The OSPF weight vector of an element of the population determines a set of *loads* l_{ij} for each arc (i,j) in the network. To fulfill those loads we have to assign capacities c_{ij} to the arcs, such that $l_{ij} \leq c_{ij}$. To attain a given capacity, one may compose several different *link types* that sum up to the desired capacity ($c_{ij} = \sum_{t \text{ used in } (i,j)} m_t c_t$, where c_t is the capacity of link type t and m_t is the number of copies of this link type).

Naturally, there are different ways to compose link types to satisfy the load. The goal of the heuristic is to satisfy the load while optimizing some objective function. For example, in this paper, we consider as objective function the minimization of the network cost, where the network cost is defined by the sum of the costs of every arc.

Recall that given a set of link types $\{1,2,\ldots,T\}$, we consider two parameters for each link type i: the capacity c_i and the price per unit of length p_i. We make the following assumptions on these parameters:

1. Given capacities $c_1 < c_2 < \ldots < c_T$ and prices $p_1 < p_2 < \ldots < p_T$, we have $(p_T/c_T) < (p_{T-1}/c_{T-1}) < \ldots < (p_1/c_1)$. In other words, the price per unit of capacity is smaller for links with greater capacities.
2. Given capacities $c_1 < c_2 < \ldots < c_T$, we have $c_i = \alpha c_{i-1}$ ($\alpha \in \mathbb{N}, \alpha > 1$). In other words, the capacities are multiples of each other by powers of α.

The first assumption is an economies of scale, i.e. unit cost drops for bigger quantities. The second assumption reflects the reality of link types available in real world applications. For instance, link types could be OC3, OC12, OC28, and OC192, which have capacities 155 Mb/s, 622 Mb/s, 2.5 Gb/s, and 10 Gb/s, respectively.

We have designed four heuristics to solve this subproblem. The heuristics were designed to minimize the overall cost of the network, but some have additional constraints imposed by specific applications.

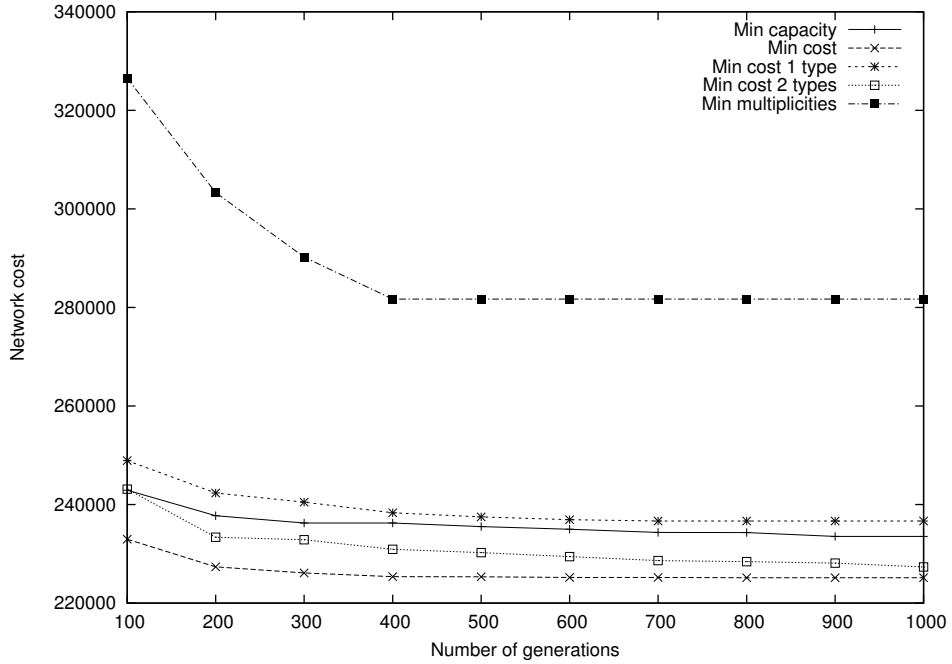

FIGURE 1. Network cost versus GA's number of generations.

1. **Min capacity**: Starting with load l, the heuristic uses as much as possible of the biggest capacity link type without exceeding the load. This means that $\lfloor l/c_T \rfloor$ units of link t are used. Then, it updates the remaining load ($l := l - \lfloor l/c_T \rfloor$), and repeats the operation on the next link type ($T-1$), until the link type (1) with smallest capacity is reached. Then, we satisfy the remaining load with $\lceil l/c_1 \rceil$ units of link type 1.
2. **Min cost**: This heuristic follows the same idea of the *Decreasing capacities* heuristic, but instead of satisfying the load only with the last link type, for each step it also stores the price of satisfying the load with the current link type. At the end, the smallest price configuration is used.
3. **Min cost k types**: This heuristic follows the same idea of the *Min cost* heuristic, but it can use at most k different link types. For practical reasons this additional constraint may be imposed by some applications.
4. **Min multiplicities**: This strategy minimizes the number of copies of a link used to satisfy the load.

Under Assumptions (1) and (2), all of the above heuristics can be implemented to take $O(T)$ time to execute per arc. Moreover, heuristic *Min capacity* gives the optimal solution for the minimum capacity objective function[1], and the heuristic *Min cost* gives the optimal solution for the minimum cost objective function[1]. On the other hand, without Assumptions (1) and (2), to find the minimum capacity solution or the minimum cost solution, we would have to solve a *Knapsack Problem*.

[1]For the given set of loads.

3. Experimental Results

Here we show the results of an experiment on a real network with 54 routers and 278 arcs. Three link types were considered with the following parameters: $c_2 = 4c_1$, $c_3 = 16c_1$; $p_2/c_2 = 0.95p_1/c_1$, $p_3/c_3 = 0.90p_1/c_1$. All 4 heuristics were tested, and heuristic (3) was tested for $k = 1$ and 2. The number of generations of the GA was varied from 100 to 1000 in steps of 100. The GA parameters $x = 25$, $y = 5$, and $\pi = 0.7$ were used and the GA population size was 100.

Figure 1 shows how the four[2] heuristics behave as a function of the number of generations of the GA. In all cases the solution improves as the number of generations grows. Such behavior is expected because if more generations are tried, more solutions are visited, and greater are the chances of finding a better solution.

As for the individual heuristics, *Min cost* achieves the best results of all, as expected. *Min cost 2 types* performs better than *Min cost 1 type*. *Min capacity* and *Min multiplicities* do not perform as well as *Min cost*, or even as *Min cost 2 types*, showing how the results are affected by heuristics that were not designed to optimize the objective function of the problem.

References

[1] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA J. on Comp.*, 6:154–160, 1994.

[2] L.S. Buriol, M.G.C. Resende, and M. Thorup. Survivable IP network design with OSPF routing. *Networks*, 2006. To appear.

(D.V. Andrade) RUTCOR, Rutgers University, Piscataway, NJ, USA
E-mail address, D.V. Andrade: dandrade@rutcor.rutgers.edu

(L.S. Buriol) Dipartimento di Informatica e Sistemistica, University of Roma "La Sapienza", Rome, Italy
E-mail address, L.S. Buriol: buriol@gmail.com

(M.G.C. Resende) AT&T Labs Research, Florham Park, NJ, USA.
E-mail address, M.G.C. Resende: mgcr@research.att.com

(M. Thorup) AT&T Labs Research, Florham Park, NJ, USA.
E-mail address, M. Thorup: mthorup@research.att.com

[2] There are five curves because heuristic (3) was tested with two parameters of k.