Spare Capacity Planning for Survivable Mesh Networks

Adel Al-Rumaih[1], David Tipper[1] *, Yu Liu[1] *, and Bryan A. Norman[2]

[1] University of Pittsburgh, Department of Information Science and Telecommunications, Pittsburgh, PA 15260, USA
alrumaih,tipper,yuliu@tele.pitt.edu

[2] University of Pittsburgh, Department of Industrial Engineering, Pittsburgh, PA 15261, USA
banorman@engrng.pitt.edu

Abstract. The design of survivable mesh based STM networks has received considerable attention in recent years and is a complex multi-constraint optimization problem. In this paper, a new spare capacity planning methodology is proposed utilizing genetic algorithms. The method is based on forcing flows/traffic which are on paths that are disjoint to share backup spare capacity. The major advantages of the new approach are a polynomial time complexity and the capability of incorporating nonlinear variables such as nonlinear cost functions into the solution algorithm. Numerical results illustrating the form of the genetic algorithm solution and comparing the proposed methodology to existing techniques from the literature are presented.

1 Introduction

Due to the widespread use of telecommunications networks and society's growing dependence upon the exchange of information, the need for reliable communication service has become increasingly important. Several highly publicized outages have illustrated that disruption of communication services is very costly to businesses, governments and the general public. This has led to growing interest in the design of networks which are survivable [1–11].

One component in constructing a survivable network is designing the physical network with enough spare capacity to enable fault recovery via rerouting of traffic in the event of a failure. The problem of spare capacity placement for survivable mesh networks has recently been studied for STM, WDM, and ATM networks. In this paper, we consider the problem of given a STM mesh type network topology, the normal traffic demand, and the capacity allocation to meet the normal traffic demand, how much spare capacity should be provisioned and where should it be located in order for the network to tolerate a specified set of failure scenarios (e.g., loss of any single link). The term "mesh" doesn't

* Supported in part by DARPA under No. F30602-97-1-0257 and NSF grant NCR-9506652

imply that the network topology is a full mesh, but rather that the network nodes are at least two connected. Operational backbone networks typically have an average nodal degree in the range of 2.5 to 4.5 [5].

Previous research on spare capacity assignment in STM mesh networks adopts the problem context above and uses either mathematical programming techniques [1–7, 11] or heuristics[8–11] to determine the spare capacity allocation. In both cases, algorithms have been developed for link restoration and path restoration. In link restoration, the nodes adjacent to a failed link are responsible for rerouting the affected traffic flows around the failed link. Thus, one merely patches the hole in the original route. In contrast, for path restoration, the source destination node pairs whose traffic traversed the failed device are responsible for restoration and reroute over the entire path set between each affected source destination pair. In general, path restoration is known to require less spare capacity than link restoration [1, 3–6]. However, path restoration is more complex to implement as many more nodes are involved in the restoration process. Also, in the general case path restoration requires the release of stub capacity (i.e., the capacity on the unaffected portion of a failed working path) both upstream and downstream of the failed device. A variation on path restoration which speeds up restoration and does not require stub release is *path restoration with link disjoint routes* [11]. In path restoration with link disjoint routes, no portion of the failed working path is used for backup routes, thus the restoration process can begin quickly without additional signaling overhead specifying exactly which component failed in the working path and confirming the release of stub capacity. This is an important advantage in current STM and WDM networks where fault identification signaling capabilities are limited.

In both link and path restoration one is interested in determining a set of backup routes for the working traffic and the amount of spare capacity required on the backup routes to achieve a desired level of traffic restoration for a specific failure scenario. A typical goal is to determine the spare capacity placement such that all normal traffic can be restored for any single link failure in the network. Mathematical programming methods have been used to formulate the spare capacity planning problem for link and path restoration approaches as Integer Programming (IP) models [2–6, 11]. The objective function adopted is to minimize the spare capacity required for achieving restoration from a specific failure condition. Unfortunately, the resulting IP formulation is NP-hard and in order to solve the model sub-optimal heuristic approaches have been tried. One approach is to approximate the IP model by a Linear Programming (LP) model which has a polynomial time bound solution and then round the solution to integer values. An alternate approach is to solve the IP model exactly over a reduced search space. Typically, the search space is reduced by limiting the set of paths over which the backup routes are determined. For example, placing a hop count limit on the path set equal to the diameter of the network or a limit on the number of hops in addition to the shortest path hop count for each source-destination pair. How to pick the path set within a general setting in order to achieve good results while maintaining a computational feasible problem that

scales is still an open problem. An additional limitation of the IP approach is the inability to incorporate nonlinear variables such as a nonlinear capacity cost function or quality of service (QoS) variables. In contrast to the mathematical programming approach, heuristics in the literature [5, 8–11] essentially seek to quickly find a reasonable spare capacity assignment that meets the fault tolerance requirements.

Here we present a new spare capacity planning technique based on path restoration with link disjoint routes that uses genetic algorithms in the solution process. Genetic algorithms (GA) have received considerable attention in recent years for use in solving various optimization problems [12], including the solution of integer programming problems [13] and data network topology design [14, 15]. One advantage of the approach is the ability to incorporate nonlinear variables in the solution process, such as realistic nonlinear capacity cost values. Additionally, the computational time is easily controlled allowing the approach to scale to large networks. The description of the proposed GA is given in Section 2. Section 3 presents a study of numerical results illustrating the application of the proposed GA technique, guidelines for parameter selection and computational complexity. Additionally, for the sake of comparison, numerical results are given for the standard IP approaches and a popular heuristic. Section 4 gives our conclusions.

2 A New Approach to Spare Capacity Design

Our methodology consists of using a GA to implement the concept that traffic flows which travel over disjoint routes may be able to share spare capacity on a backup path, since it is unlikely that more than one failure will occur simultaneously. Thus, our approach tries to reduce the cost of spare capacity needed for a specific fault tolerance requirement (e.g., full recovery from any single link failure) by finding a set of backup paths that enables the sharing of spare capacity. This can be achieved by forcing the backup paths to use a subset of the links in the network, thus concentrating the backup paths on certain routes of the network, which results in reducing the total cost due to the nonlinear economy of scale of capacity cost [5, 16].

GAs are stochastic search techniques that mimic the survival of the fittest (or best) paradigm observed in nature [17]. A GA optimizes a fitness function (i.e., objective function) by evolving successive generations of a population utilizing breeding and mutation transformations to move from one generation to the next in a manner such that only the fittest (best solutions) survive from generation to generation. A GA first creates an initial population of solutions (i.e., a generation) by either using random solutions or solutions constructed using some heuristic procedure and these solutions represent the basis for the evolutionary improvement process. In our GA, selection of a new population from the current population is accomplished, as shown in Fig. 1, by performing three operations: (1) elitist reproduction, (2) crossover, and (3) mutation. In elitist reproduction a portion of the current population is copied directly into the next generation to maintain progress of the search through successive generations. In crossover

two members of the current generation are selected for breeding and are bred via a crossover operation creating a pair of offspring to add to the population. In mutation a portion of the population is randomly selected and stochastically altered in order to maintain diversity of the genetic search. The new population is evaluated (sorted according to the fitness function) and the new generation is selected from the best members of the population with the remainder being eliminated. This process is repeated until a stopping criterion is met, usually either a finite number of generations of the population or when the improvement in the fitness function between successive generations is less than a specified value.

There are many variations of GAs based on the choice of reproduction, crossover and mutation operators. Here we develop a GA for our problem domain based in part on the approach given in [13] for the solution of nonlinear IP problems. Before applying the GA to determine spare capacity, the paths to be used and the required capacity under normal conditions are determined based on the traffic demands, topology and shortest path routing [15, 18]. We use a two-stage algorithm to implement the GA method for spare capacity planning. Stage I determines an initial population for the GA, in effect finding a set of feasible solutions to the spare capacity assignment problem. Stage II then tries to improve the spare capacity assignment resulting from Stage I via a GA search as shown in Fig. 1. The two stages of the GA are briefly described below; detailed information is given in [19].

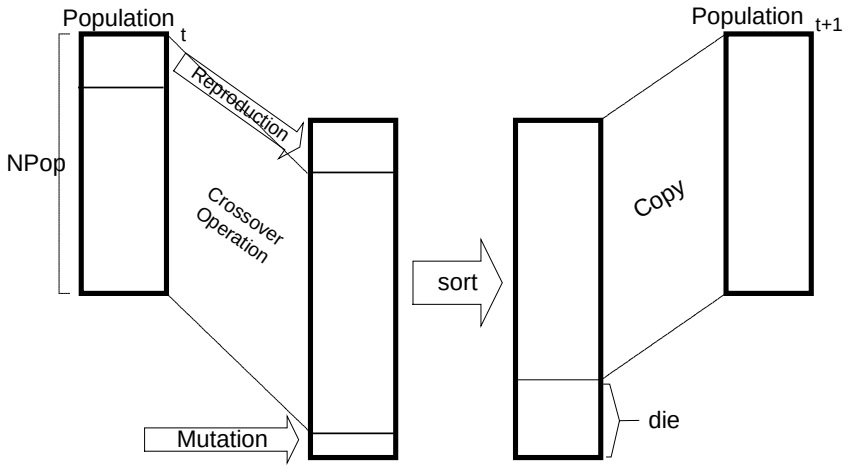

Fig. 1. Transition between two successive generations in a GA

Consider a network of L links and N nodes. We represent the network topology by an adjacency matrix Top where element Top_{ij} is 1 if a link exists between node i and node j. The routes used to carry traffic under normal conditions are denoted by WP (working paths), where WP_{ij} is the path used between node i

and node j. Similarly, we define $BTop$ as the adjacency matrix that represents a virtual network topology used to calculate a set of backup paths BP. Let SA denote a set of spare capacity assignments that meet a required restoration level for a specific failure scenario. We define $Npop$ as the size of the initial population and subsequent generations. Let SA^k denote the kth spare capacity assignment in the population and CSA^k the cost of SA^k, where $1 \leq k \leq Npop$. Then, $SA^k_{(lm)}$ is the spare capacity needed on the link between node l and node m in the kth spare capacity assignment. Fig. 2 shows the basic Stage I algorithm.

2.1 Stage I: Initial population generation

Step 1 involves randomly choosing a link (jk) in the backup topology matrix $BTop$ then temporarily deleting it and checking if the topology of the virtual backup network is still 2-connected. The term 2-connected means that there are at least 2 disjoint paths between every node pair in the topology. If the backup topology matrix $BTop$ is no longer 2-connected then the deleted link is returned and Step 1 is repeated.

Step 2 enacts the successful repetition of Step 1, until the number of deleted links equals a specified input parameter *delete*.

Note, that Step 1 and 2 in effect randomly reduce the path set to be considered for routing backup connections, in contrast many IP formulations reduce the path set by imposing hop count limits.

Step 3 determines the backup paths BP^k and spare capacity requirements SA^k for a specific failure scenario and restoration level requirement. Here we discuss the case of 100% restoration of all traffic affected by the failure of any single link in the network; other scenarios are given in [19] and require slight modifications. The BP^k and SA^k are found by applying the following steps, with $f = 1$ initially:

Step 3-1. Fail link f in the network by removing it from the topology matrix Top and determining which working paths in WP are affected by the failure. Also remove the failed link from the current backup topology matrix $BTop$.

Step 3-2. For a source destination pair ij affected by the failure, remove the working path WP_{ij} between node i and node j, by deleting all the links of WP_{ij} from the backup topology matrix $BTop$.

Step 3-3. Find backup path BP_{ij} between node i and node j using the current backup topology matrix $BTop$ to determine the possible path set and shortest path routing to find the actual path. Determine the spare capacity requirements for the links on the backup path.

Step 3-4. Return the links of the working path WP_{ij} between node i and node j to the current backup topology matrix $BTop$.

Step 3-5. Repeat Steps 3-2 to 3-4 for every source destination pair affected by the failure of link f, and adjust the spare capacity requirements for each link. That is, if a link (lm) is used on multiple backup paths for the same link failure case then the spare capacity requirement at that link

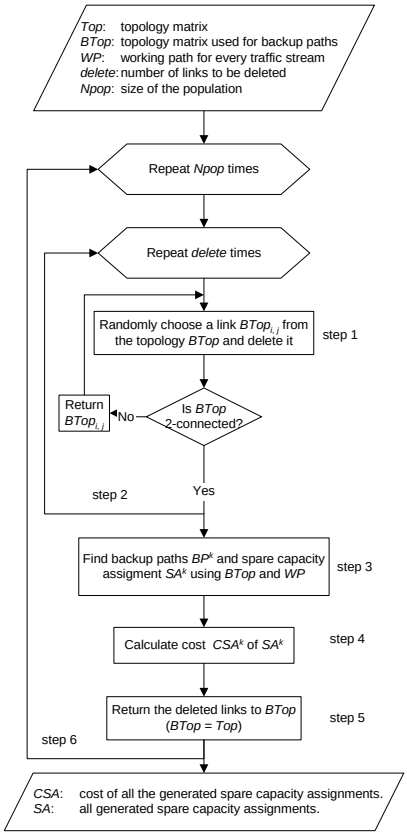

Fig. 2. Stage I create initial population

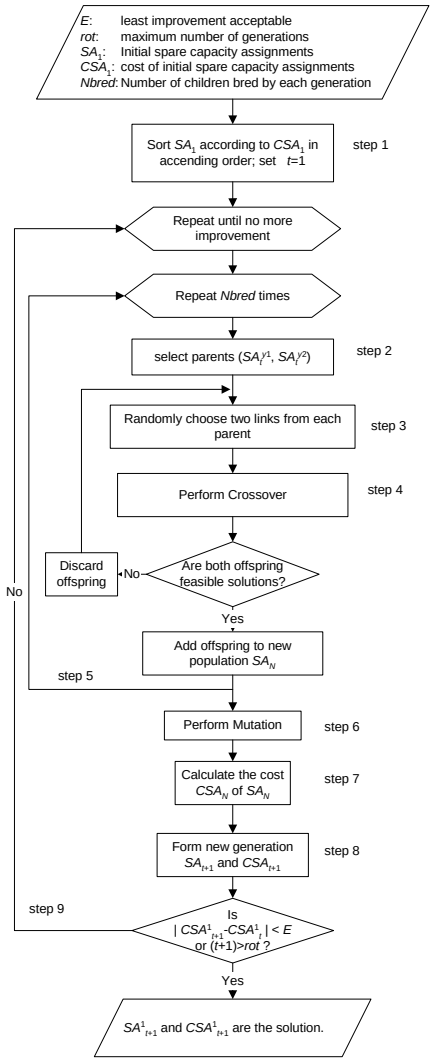

Fig. 3. Stage II genetic algorithm

is the sum of the spare capacities needed by each backup path passing through the link.

Step 3-6. Repeat Steps 3-1 to 3-5 for every link in the network Top (i.e., increment f) and adjust the spare capacity requirements. The final spare capacity requirement on a link (lm) is the maximum spare capacity required by the backup paths using that link for any single link failure f.

The steps above result in each backup path BP_{ij} between a pair of nodes i and j being link disjoint with its corresponding working path WP_{ij} thus allowing implementation of path restoration with link disjoint routes.

Step 4 calculates the cost CSA^k of the current spare capacity assignment SA^k.

Step 5 returns the links deleted in Step 1 to $BTop$ resulting in $BTop = Top$.

Step 6 repeats Steps 1-5, until the number of spare capacity assignments generated equals the specified population size $Npop$ - an input parameter.

2.2 Stage II Genetic Search

Stage II performs a GA search on the population generated from Stage I, until a stopping criterion is met. Fig. 3 shows the basic algorithm.

Step 1 sorts the $Npop$ spare capacity assignments SA in ascending order according to their costs CSA to form the initial generation SA_t with cost CSA_t for $t = 1$.

Step 2 selects pairs of the $Npop$ spare capacity assignments for breeding. The pairs (SA_t^{y1}, SA_t^{y2}) were selected according to a quadratic procedure by applying the following equation twice to get $y1$ and $y2$.
$y = \lceil x^2 \rceil$ where x is a real number uniformly distributed between $[0, \sqrt{Npop}]$. This approach is biased towards choosing the better solutions in the current generation for breeding. Steps 3 and 4 perform breeding and are called the crossover steps in GA terminology. These steps combine elements of two existing spare capacity assignments (parents) to produce two new spare capacity assignments (offspring) that are added to the population for possible continuation into the next generation. Several types of crossover operators were tested to find the operator best suited for our problem domain [19]. The crossover operator that was selected was two-position random crossover as it produced the highest rate of offspring which were feasible solutions.

Step 3 randomly selects two links from each parent, that is links (jk and lm) for SA_t^{y1} and links (pq and rs) for SA_t^{y2}.

Step 4 performs two-position crossover by switching the spare capacity assignment on the chosen links of the parents to create a pair of offspring (SA_t^{y1o}, SA_t^{y2o}). The child SA_t^{y1o} is equivalent to the parent SA_t^{y1} except the spare capacity value for link jk is taken from link pq of the other parent (i.e., SA_t^{y2}). Similarly, child SA_t^{y2o} is equivalent to the parent SA_t^{y2} except the spare capacity value for link rs is taken from link lm of the other parent (i.e., SA_t^{y1}). If either of the two offspring has a spare capacity assignment which is not a feasible solution for the required restoration level, then both

offspring are discarded and Steps 3 and 4 are repeated. Otherwise the two offspring are added to the population of new solutions SA_N.

Step 5 repeats Steps 2 to Step 4 $Nbred$ times, where $Nbred$ is a input parameter determining the number of children bred by each generation.

Step 6 performs mutation by first selecting a specific percentage of the current population, set by $mrate$ (the mutation rate in GA terminology). Then the spare capacity values for the selected solutions are altered randomly with mutation probability $mprob$. The resulting alternated solutions, if feasible, are added to the population of new solutions SA_N.

Step 7 calculates the cost CSA_N for all the new spare capacity assignments SA_N.

Step 8 forms a new generation of spare capacity assignments SA_{t+1}. First, by copying the $Nreprod$ best assignments from the current generation to the new generation along with their cost. Note $Nreprod$ is a user input parameter. Second, the new solutions from SA_N are added to the new generation along with their cost CSA_N. The new generation is then sorted in ascending order according to the cost and the first $Npop$ spare capacity assignments are retained in the next generation with the rest being discarded.

Step 9 repeats Steps 2-8, until the improvement in the spare capacity cost between successive generations is less than E or the number of the generations created equals rot. Both E and rot are input parameters.

3 Numerical Results

Extensive numerical experimentation was conducted with the GA approach presented above for a variety of network topologies, loading conditions, failure scenarios and restoration requirements with details in [19]. The experimentation had two purposes, namely: (1) selection of parameter values and sensitivity analysis for the GA method and (2) comparison with existing spare capacity planning methods.

In the first set of experiments, the effects of different population sizes, number of links deleted in determining the initial population, reproduction rates, crossover algorithms, mutation rates, mutation probabilities and stopping criteria were studied. The parameters governing the performance of the Stage I algorithm outlined above are the population size, $Npop$, and $delete$ the number of links deleted randomly from the virtual backup topology before determining the backup paths. Consider a network topology with N nodes and L links. We assume that the network nodes are at least two connected so the average network node degree deg ranges from 2 for a ring to $(N-1)$ for a full mesh. The corresponding number of links L ranges from $N \leq L \leq (N(N-1)/2)$. Obviously, as the number of links and the average nodal degree increases, the greater the spare capacity assignment search space and therefore $Npop$ and $delete$ should increase. Numerical experimentation showed that a population size in the range $Npop \in [(\lceil deg \rceil - 1)L, (\lceil deg \rceil)L]$ worked well. For the number of links to delete, it was found that the formula $delete = L-(N-1)-\lceil (L-N)/(deg-1) \rceil$ yielded good results.

In considering the computational complexity of Stage I, we note that Steps 1 and 2 are repeated a random number of times until a feasible backup topology is found with *delete* links removed. The worst case number of repetitions is L. The most complex part of Step 1 is the test for a 2-connected network which is $O(L)$ complexity [18]. Thus, Steps 1 and 2 are $O(L^2)$. Step 3 involves finding the backup paths and spare capacity assignments. This involves the repeated application of a shortest path algorithm which has complexity $O(L+N\times\log N)$ for all $(N\times(N-1))$ source destination pairs and all L possible link failures. Thus, Step 3 is $O((L\times N^2\times(L+N\times\log N))$. Steps 4 and 5 involve the calculation of the spare capacity cost for the link assignments and reinitializing the backup topology, the complexity is $O(L)$. Step 6 repeats Steps 1-5 $Npop$ times. Thus for Stage I, the complexity is $O(Npop\times L\times N^2\times(L+N\times\log N))$.

For Stage II, the input parameters are $Nbred$, $Nreprod$, $mrate$, $mprob$, E and rot. $Nbred$ determines the number of children created for consideration in each new generation and was selected to keep the population size constant $(Nbred=(Npop-Nreprod)/2)$. Since the GA above uses an elitist reproduction approach the value of $Nreprod$ was set to a constant value in each experiment. The numerical results showed that the quality of the solution was best when $Nreprod$ was in the range $Nreprod\in[10\%,50\%]$. The values for the mutation rate, $mrate$, and the mutation probability, $mprob$, were varied over a wide range ($mrate\in[.05\%-10\%]$, $mprob\in[.005,.05]$) with no significant effects on the quality of the GA. This may be due to the random nature of the crossover algorithm used or the small number of generations considered. Since the GA was relatively insensitive to the choice of mutation parameters, values of $mrate=1\%$, $mprob=.05$ were used. Note, E and rot specify the stopping criteria of the GA search and are largely determined by computer runtime and optimality considerations. One can see that increasing E and decreasing rot results in longer computer runtimes with possibly better final results. Here, $E=10^{-5}$ and $rot=10$, which results in the stopping criteria being rot for the results shown.

Considering the complexity of Stage II, Step 1 involves a simple sort of $Npop$ values, for which the quick sort algorithm has $O(Npop\times\log Npop)$. In Step 2, pairs of the existing population are randomly selected for breeding which is $O(1)$. The most computationally intensive steps in Stage II are Steps 3 and 4. The pairs of links in the parents are randomly selected and switched to create offspring. The worst case for this algorithm is $O(L^2)$. The two offspring are then tested for feasibility. This involves considering all the failure scenarios and checking whether the required restoration level is met by examining the spare capacity assignment on backup paths for all the source destination pairs. Here, we are considering 100% restoration for any single link failure, so the test requires considering every link and in the worst case every source destination pair for each link failure. The resulting Step 3 and 4 algorithms have $O(L^2\times L\times N^2\times(L+N\times\log N))$ complexity. Note that Step 5 invokes the repetition of Steps 3 and 4 until $Nbred$ feasible solutions are found. In general the number of solutions that need to be created and tested to result in $Nbred$ feasible solution is a random quantity with worst case $Npop$. Thus Steps 3-5 have complexity $O(Npop\times L^3\times N^2\times$

$(L + N \times \log N))$. Step 6 involves performing mutation and a test for feasibility of the mutated solution resulting in an $O(Npop \times L \times N^2 \times (L + N \times \log N))$ complexity. Steps 7 and 8 consist of calculating the cost of the various spare capacity assignments and forming a new generation and the complexity is $O(L + Npop)$. Step 9 involves testing if a stopping criterion is meet. If we assume the stopping criterion is the maximum number of generations rot, then the overall complexity of Stage II is $O(rot \times Npop \times (\log Npop + L^3 \times N^2 \times (L + N \times \log N))$. Note that both stages of the genetic algorithm method have a polynomial time complexity.

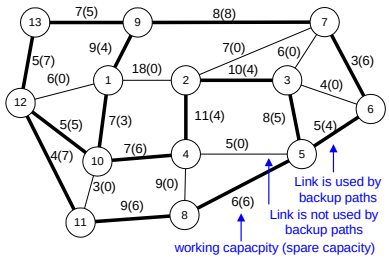

Fig. 4. Network 1 (13 nodes, 23 links)

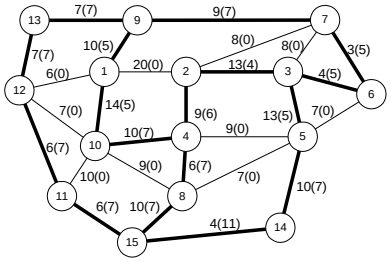

Fig. 5. Network 2 (15 nodes, 27 links)

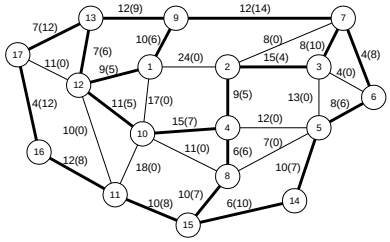

Fig. 6. Network 3 (17 nodes, 31 links)

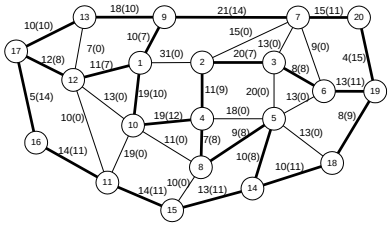

Fig. 7. Network 4 (20 nodes, 37 links)

Four networks with topologies shown in Fig. 4–7 was used to evaluate the above GA method. One traffic demand (e.g. STS-1) between each directed node pair in the network was assumed. Table 1 summarizes the parameters of the four networks. The allocated working capacity for each network was determined using shortest path routing. The working capacity and the spare capacity required to provide fault tolerance for any single link failure are marked beside each link as the first and second number respectively in Fig. 4–7. The bold lines indicate links with non-zero spare capacity. Notice that the spare capacity is concentrated on a subset of the links in the network with some links having zero spare capacity. In determining the results shown in Fig. 4–7, the GA tried to minimize the total

Table 1. Summary of the four networks studied

Network	Number of nodes	Number of links	Average node degree	Number of S-D pairs	Total network load
1	13	23	3.54	78	156
2	15	27	3.60	105	210
3	17	31	3.65	136	272
4	20	37	3.70	190	380

network cost when a nonlinear capacity cost function was employed. The total link cost is the cost of the total capacity as determined by the summation of both the working and the spare capacity in the link. The parameters for the piece-wise cost function are selected from an actual network service provider charges for bandwidth namely: 3.7, 5.8, 19.6, 64.2 for link units of OC-1, OC-3, OC-12, OC-48 respectively [16]. In order to permit a fair comparison with other approaches, the path set used for backup topologies in the GA was hop count limited with a limit of 7.

In the second set of experiments the GA approach was compared with three methods from the literature: (1) the Spare Link Placement Algorithm (SLPA) heuristic [5]; (2) Link restoration using Integer Programming (Link IP) [2]; and (3) Path restoration with link disjoint routes using Integer Programming (Path IP) [3, 5]. The Spare Link Placement Algorithm (SLPA) is a heuristic approach with polynomial time complexity, and has been used by several telecom network operators for spare capacity planning. The Link IP approach reroutes all the affected traffic demands locally around the failure. As noted earlier, the Path IP approach is expected to provide the best results since it reroutes failed traffic over the remaining paths between each source destination pair after a failure. Both the Link IP and the Path IP techniques were solved using the commercial optimization package CPLEX. Our implementations of the three methods above were validated by reproducing published results from the literature. Note that a lower bound on the spare capacity requirements can be found by taking the Path IP formulation and relaxing the integer requirement, i.e., solving a Linear Programming version of the problem.

Table 2. Total spare capacity

Net-work	Working Capacity	Spare Capacity Bound	Spare Capacity, % Redundancy							
			SLPA		Link IP		GA		Path IP	
1	324	129.7	230	70.99	199	61.42	158	48.77	135	41.67
2	464	175.0	322	69.40	264	56.90	218	46.98	176	37.93
3	640	236.0	444	69.38	353	55.16	310	48.44	236	36.88
4	966	349.8	684	70.81	535	55.38	460	47.62	350	36.23

In Table 2, the total spare network capacity for 100% restoration for any single link failure as determined by the four algorithms using a hop count limit at 7 is given. The lower bound on the spare capacity requirements is found by solving the LP relaxation of the Path IP model is also shown. The % redundancy as measured by the ratio of spare capacity to the working capacity is also given in Table 2. For the Link IP and Path IP approaches the objective is to minimize the total spare capacity in the network. Compared to the Path IP results, the genetic algorithm has about 7-12% higher redundancy in the four networks. However, it results in 7-23% less redundancy than the spare capacity assignments given by the Link IP and SLPA approaches. Note, that the GA approach was designed to minimize the cost not the amount of spare capacity.

Neither the Link IP nor Path IP methods can deal directly with a nonlinear cost function. Adapting IP algorithms to nonlinear cost functions usually leads to an explosion of the search space size. On the contrary, whether the cost function is linear or not makes no difference to the GA approach. In order to compare the Link IP, Path IP and SLPA approaches cost, we solve the problem in two steps. The first step utilizes the algorithms to minimize the spare capacity. The second step minimizes the total network cost by fixing the link capacity at the level found in the first step and then finding the combination of capacity assignments that minimizes the cost. For example if a link requires 9 units of capacity this could be achieved be either 9 OC-1 or 3 OC-3 links, here the minimum cost option of 3 OC-3 links would be selected. Table 3 shows the total network costs from the above algorithms. The least cost solution is obtained by the Path IP method. The Link IP and SLPA yield solutions with a higher cost than the GA method or the Path IP method. Notice, that the GA approach can produce a cost close to that given by the Path IP method (within 3-7%).

Table 3. Total spare capacity cost

Network	Total cost			
	SLPA	Link IP	GA	Path IP
1	1033	970	883	859
2	1467	1387	1262	1183
3	2024	1829	1698	1591
4	2960	2687	2516	2347

The primary benefit of the GA approach is its computational efficiency and scalability. This can be seen by examining the computer run times of the four methods for the networks studied as shown in Table 4. Both the SLPA and GA methods were implemented in PASCAL and executed on a Pentium II 400 MHz PC with 128 MB RAM running Windows NT 4.0. The Link IP and Path IP methods and LP relaxation of the Path IP model were implemented using the commercial package CPLEX 6.0 running on a SUN Enterprise 4000 server (10 parallel UltraSPARC-II 250 MHz CPUs with 2.6GB RAM). Thus the execution

Table 4. Execution times

Network	LP	Execution Time (minutes)			
	Bound	SLPA	Link IP	GA	Path IP
1	0.2	1.02	1.0	0.48	91.7
2	0.5	4.33	3.01	0.65	367
3	2.5	7.07	8.17	0.73	833
4	14.67	16.67	20.17	1.38	5833

times shown below are not directly comparable as they are on different platforms but the scalability of each scheme can be inferred. From the results one can see that the execution time of the GA approach is the least affected by increases in the network size. Additional, numerical results for other network topologies including networks of more than 100 nodes (with average node degree 3.2) are given in [19].

4 Conclusions

In this paper, we proposed a genetic algorithm for spare capacity planning in STM networks based on link disjoint path restoration. The primary advantages of the GA method were it's scalability and the capability of incorporating non-linear cost functions. Parameter selection and the computational complexity of the GA approach were discussed. Numerical results illustrating the GA method solution were presented and compared to existing approaches from the literature. It was shown that the GA method provides a reasonably good spare capacity assignment with substantial computational savings.

References

1. R. Doverspike and B. Wilson. Comparison of capacity efficiency of DCS network restoration routing techniques. *Journal of Network and System Management*, 2(2):88–99, 1994.
2. M. Herzberg, S. Bye, and U. Utano. The hop-limit approach for spare-capacity assignment in survivable networks. *IEEE/ACM Transactions on Networking*, pages 775–784, December 1995.
3. R. Iraschko, M. MacGregor, and W. Grover. Optimal capacity placement for path restoration in STM or ATM mesh survivable networks. *IEEE/ACM Transactions on Networking*, pages 325–336, June 1998.
4. M. Herzberg, D. Wells, and A. Herschtal. Optimal resource allocation for path restoration in mesh-type self-healing networks. In *Proceedings of 15th International Teletraffic Congress*, volume 15, Washington, D.C., June 1997.
5. W. D. Grover, R. R.Iraschko, and Y. Zheng. Comparative methods and issues in design of mesh-restorable STM and ATM networks. In P. Soriano B.Sanso, editor, *Telecommunication Network Planning*, pages 169–200. Kluwer Academic Publishers, 1999.

6. Yijun Xiong and Lorne G. Mason. Restoration strategies and spare capacity requirements in self-healing ATM networks. *IEEE/ACM Transactions on Networking*, 7(1):98–110, February 1999.
7. H. Sakauchi, Y. Nishimura, and S. Hasegawa. A self-healing network with an economical spare-channel assignment. In *Proceeding of IEEE Global Conference on Communications*, pages 438–442, November 1990.
8. B. Venables, W. Grover, and W. MacGregor. Two strategies for spare capacity placement in mesh restorable networks. In *Proceeding of IEEE Global Conference on Communications*, pages 267–271, November 1993.
9. J. Yamada. A spare capacity design method for restorable networks. In *Proceeding of IEEE Global Conference on Communications*, pages 931–935, November 1995.
10. B. Van Caenegem, N. Wauters, and P. Demeester. Spare capacity assignment for different restoration strategies in mesh survivable networks. *Proceeding of IEEE International Conference on Communications*, pages 288–292, June 1997.
11. B. Van Caenegem, W. Van Parys, F. De Turck, and P. M. Demeester. Dimensioning of survivable WDM networks. *IEEE Journal on Selected Areas of Communications*, 16(7):1146–1157, September 1998.
12. B. Norman and J. Bean. Random keys genetic algorithm for complex scheduling problems. *Naval Research Logistics*, 46:199–211, 1999.
13. B. Hadj-Alouane and J. Bean. A genetic algorithm for the multiple choice integer program. *Operations Research*, 46:92–101, Jan.-Feb. 1997.
14. K.-T. Ko, K.-S. Tang, C.-Y. Chan, K.-F. Man, and S. Kwong. Using genetic algorithms to design mesh networks. *IEEE Computer*, pages 56–60, August 1997.
15. D. Medhi and D. Tipper. Some approaches to solving a multi-hour broadband network capacity design problem. to appear Telecommunication Systems.
16. E. McDysan and D. L. Spahn. *Hands-on ATM*. McGraw-Hill, 1998.
17. D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing, 1989.
18. A. Kershenbaum. *Telecommunication Network Design Algorithms*. McGraw-Hill, 1993.
19. A. Al-Rumaih. *A Spare Capacity Planning Methodology for Wide Area Survivable Networks*. Ph.D. dissertation, Department of Information Science and Telecommunications. University of Pittsburgh, May 1999.