An Effective Evolutionary Computational Approach for Routing Optimization in Networks with Shared Risk Link Groups

Xubin Luo and Qing Li

Southwestern University of Finance and Economics,
Chengdu, Sichuan, 610051, China
{xbluo,liqt}@swufe.edu.cn

Abstract. In this work, we study a routing optimization problem in networks with shared risk link groups (SRLGs). Specifically, a path between a source and a destination is determined such that the combined path cost and the weight of SRLGs to which the links of the path belong is minimized. We develop evolutionary computation based algorithms to solve the problem. The performance of the proposed algorithms is evaluated via extensive simulation and is compared with the solutions obtained by integer linear programming and the heuristic algorithm in [1].

Keywords: routing, shared risk link group (SRLG), evolutionary algorithm, combined cost.

1 Introduction

A shared risk link group (SRLG) is used to represent a set of links that are affected by a single failure (e.g., a failure in the physical layer such as a cable cut). The failure of any SRLG that a connection traverses will disrupt the service provided to the connection. A cost is also incurred on a link when it is used to provide the service to the connection. An link in a path introduces two component costs, one is the risk of an SRLG failure and the other is the link cost. The failure of an SRLG can be measured by a risk factor, e.g., the probability of failure. If the probability of failure of an SRLG i is p_i, the probability of no SRLG failure along a path can be calculated as (with standard independence assumptions) $\Pi_i(1-p_i)$ where i is any SRLG that the path traverses, which can be expressed as a summation, $\sum_i \ln(1-p_i)$, on the logarithmic scale. When provisioning a connection service for a client, it is important for the network to minimize both the risk and cost of a route (where the cost can be monetary cost, delay, etc), i.e., the accumulative risk and cost on the route of the connection should be minimized. However, it may not be possible to minimize the two potentially conflicting objectives simultaneously. The problem is then to determine a path that minimizes a combined cost in a network with SRLGs and finds a tradeoff between the two objectives.

P. Sénac, M. Ott, and A. Seneviratne (Eds.): ICWCA 2011, LNICST 72, pp. 207–215, 2012.

The solution to this problem can be used to calculate a risk bounded and delay bounded path for a connection, or to design algorithms that find a pair of low-cost SRLG-diverse paths between a source and a destination for survivable service provisioning. In SRLG diverse routing, a demand requires two paths in the network, one working path and one protection path, so that the service to the demand can be honored in case of a single network failure, such as a fiber cut. The diverse routing problem in networks is to find a pair of paths between a source and a destination such that no single failure in the network may cause both paths to fail. The diverse routing problem in networks with generally defined SRLG failures has been proved to be *NP* -complete [2] where an SRLG may include an arbitrary group of links. In addition, finding a pair of least cost SRLG-diverse routes is also *NP* -complete [2]. This problem can be solved in polynomial time under some special definition of SRLGs [3, 4]. In many cases, it is desirable for service providers to make sure that both the working path and backup path are risk-bounded and cost bounded to provide certain degree of quality of service and protection.

In work [1], an ILP formulation and a heuristic algorithm for the minimum combined cost routing problem were presented. The heuristic is a Bellman-Ford style algorithm. Instead of keeping routing information of the optimal path alone, the algorithm maintains at each node the routing information for the best k routes from the source to the node, where k is a tunable parameter. The reason to keep k routing entries at each node is that, unlike the single-metric minimum cost routing, given an optimal path from the source node *src* to the destination node *dest* , a subpath on the end-to-end optimal path may not be optimal for the two end nodes of the subpath when the combined path cost is considered. By maintaining multiple routing entries at a node, the algorithm has a better chance of obtaining the optimal path from the source.

In this work, we develop evolutionary computation based algorithms to solve the minimum combined cost routing problem that minimizes the combined link and SRLG cost. The performance is evaluated via extensive simulation and compared with existing solutions. A genetic algorithm (GA) is a type of evolutionary computation based search technique that finds true or sub-optimal solutions for optimization problems [5]. An individual solution is usually represented by a string, called a chromosome. The chromosome can be decoded to obtain a solution. A typical genetic algorithm imitates the evolutionary process of genetic population over generations. The objective of a genetic algorithm can be expressed as finding a string such that a function of the string is maximized or minimized. A fitness function is defined to evaluate how good the solution of each individual chromosome is with regard to our optimization objective. A GA starts by initializing a set of chromosomes which is called the initial population. Then the current population produces the next generation of population by mating, mutation, and survivor selection. It borrows from Darwin's Theory of Evolution, i.e., natural selection acts to preserve and accumulate minor advantageous genetic mutations. By preserving those individuals with higher fitness, a GA is expected to converge at optimal solutions or sub-optimal solutions. The flow of a typical genetic algorithm is shown in Figure 1.

Algorithm 1. *Flow of a Typical Genetic Algorithm*

```
1: Initialize population: create an initial population and evaluate each
   individual's fitness w.r.t. the problem.
2: while the stopping condition is not met do
3:   while the number of offsprings is not enough do
4:     Select a pair of parent chromosomes;
5:     Do mating between the parents (crossover) and produce offspring(s);
6:     Randomly select N chromosomes, mutate them, and add them into the
       new population;
7:   end while
8:   Evaluate all the individuals in the new population;
9:   Select the survivors based on their fitness and put them into the next
     generation. Those individuals that are not selected are discarded.
10: end while
11: return the solution represented by the best individual in the last generation.
```

Genetic algorithms are popular in solving graph theory problems such as travel salesman's problem (TSP) [6, 7, 8]. At the same time, GAs have also been applied to solve service provisioning problems in WDM optical networks. Reference [9] studies the traffic grooming problem in a ring network with a static traffic model and [10, 11, 12] extends the work to deal with dynamic traffic models. A genetic algorithm is also developed to solve the traffic grooming problem in WDM optical mesh network in [13]. In [14, 15, 16], a genetic algorithm is applied to solve the optimal logical topology design problem in WDM optical networks. The routing and wavelength assignment problem (RWA) in WDM optical networks is studied in [17, 18, 19, 20]. The authors of [21, 22] study the problem of optimal placement of wavelength converters in WDM ring networks using genetic algorithms.

The rest of this paper is organized as follows. The formal definition of the problem under study is given in Section 2. Section 3 presents the details of our algorithm. Performance evaluation and results are reported in Section 4. Section 5 concludes the paper.

2 Problem Definition

We formally define the routing optimization problem as follows. Consider a network $\mathrm{E} = (v, \varsigma)$ where v is the set of vertices and ς is the set of links in the network, each of which is associated with a link cost $c(e)$. A set of SRLGs are given as $R = \{r_i \mid 1 < i < |R|\}$, where r_i is the i-th shared risk link group. Each SRLG has a set of links that are included in this risk group, which can be represented as $r_i = \{l_1, l_2, ..., l_m\}$. A cost $c(r_i)$ that captures the group's risk factor is associated with each SRLG. A path P is represented as $\mathrm{P} = \{v_1, v_2, v_3, ..., v_i, v_{i+1}..., v_{n-1}, v_n\}$ where (v_i, v_{i+1}) is a link on P (which is denoted as $(v_i, v_{i+1}) \in \mathrm{P}$ or $l_i \in \mathrm{P}$), and v_1 and v_n are the source and destination, respectively. We say a path P travels through an SRLG r_i (represented as $\mathrm{P} \to r_i$) when there exists at least one link such that

$l \in r_i$ and $l \in \mathrm{P}$. Given a source node *src* and a destination node *dest*, the objective of this route optimization problem is to find a path P from *src* to *dest*, such that the combined link cost and SRLG cost are minimized. More specifically, the objective is to find a path $\mathrm{P} = (src, v_2, v_3, ..., v_i, v_{i+1}, v_{n-1}, dest)$ such that $C(\mathrm{P}) = w \cdot \sum_{r_i:\mathrm{P}\to r_i} c(r_i) + (1-w) \cdot \sum_{l\in \mathrm{P}} c(l)$ is minimized where $0 \le w \le 1$.

3 Proposed Algorithm

One important aspect of genetic algorithms is to define the encoding and decoding rules for the chromosomes. Another important aspect is to define the fitness function or evaluation function. For our routing optimization problem, a chromosome is defined as a string of node identities. We choose to have each individual in the population representing a valid path for a connection request. That is, each chromosome should represent a connected path and this path should start from the source node (src) of the request and end at the destination node $(dest)$. Therefore a chromosome is defined as follows:

$$C = src, n_1, n_2, ..., n_k, dest$$

and the fitness function is defined as:

$$f(C) = \alpha \cdot S_l + \beta \cdot S_{srlg} \tag{1}$$

where S_l is the total cost of the links on the path, S_{srlg} is the total weight of SRLGs that the path is associated with, and α and beta determine how much weight the link cost and SRLG cost, respectively, take in the combined cost.

Using this fitness function, we penalize the path that has high total link cost and/or high total SRLG weight. In the survivor selection, we try to preserve the chromosomes with small fitness values. Thus in the evolutionary process the algorithm is more likely to converge to a solution that is optimal or is close to the optimal.

Yet another important issue in our genetic algorithm is the generation of offsprings. Since we require each chromosome to represent a valid solution, we can not simply do random mating and mutation as many other genetic algorithms do. Our approach is detailed as follows:

For mating we pick two chromosomes as the parents if the corresponding two paths share at least one common node (not including the source and destination nodes). For example

$$C^a = src, n_1^a, n_2^a, ..., n, ..., n_{k_a}^a, dest,$$

$$C^b = src, n_1^a, n_2^a, ***, n, ***, n_{k_b}^b, dest.$$

In addition to nodes *src* and *dest*, C^a and C^b share a common node n. Through mating we generate two offspring chromosomes, C_c^1 and C_c^2, by swapping the segments of the two chromosomes at the common node n:

$$C_c^1 = src, n_1^a, n_2^a, ..., n, ***, n_{k_b}^b, dest,$$

$$C_c^2 = src, n_1^b, n_2^b, ***, n, ..., n_{k_a}^a, dest.$$

If there is more than one common node between the two paths, the algorithm randomly picks one and the segment swapping is performed.

It is worth noting that it is possible to have loops in the paths corresponding to the generated chromosomes due to the crossover. As a result, some chromosomes may grow bigger and bigger in the long run. We can add an additional step to remove the loops in the path. We choose not to spend time on checking and removing those loops, but to let the selection scheme deal with the problem. Note that those big chromosomes are inferior to those short ones in surviving the selection process.

For the mutation of a chromosome, the algorithm randomly picks two nodes (suppose they are n_i and n_j) on the path and replaces the sub-path between n_i and n_j with a detour route between these two nodes. A random path between these two nodes is generated as the detour route. To avoid producing loops in this process, this detour should exclude any node on the original path except the two end nodes (n_i and n_j in this case). The detailed algorithm for the mutation process is described in Algorithm 2, which calls Algorithm 3 to generate the random path.

Algorithm 2. *Mutation with Detour Routing* (C)

Require: $C = src, n_1, ..., n_{i-1}, n_i, ..., n_j, n_{j+1}, ..., n_k, dest.$

1: Pick two nodes n_i and n_j randomly, such that the path is divided into three parts: $C' = P_{src,n_i}, P_{n_i,n_j}, P_{n_j,dest}$

2: Call subroutine $RandomPath(n_i, n_j, FN)$, to route a random path P'_{n_i,n_j} between nodes n_i and n_j. This random path, should exclude the nodes in P_{src,n_i} and $P_{n_j,dest}$ (except n_i and n_j)

3 Mutate C to $C' = P_{src,n_i}, P_{n_i,n_j}, P_{n_j,dest}$

4 **return** C'

Subroutine $RandomPath(n_i, n_j, FN)$ (Algorithm 3) is designed to find a random path between a pair of nodes. Certain nodes in the graph can be excluded from this random path, by specifying those nodes in a *forbidden set* FN. The basic idea of this algorithm is, starting from the source node, to randomly select a next node to append to the path, until it reaches the destination node at a certain point. In this algorithm, the last node in the path is the current node (represented as $CurNode$ in Algorithm 3), and we randomly select a *next* node (represented as *next* in Algorithm 3) to append to the path. We select from all the nodes that are reachable from the current node through a single link, but excluding the nodes in the forbidden set FN. In some cases, it is possible that there is no available node to select from, i.e., the path reach a deadend at

the current node. In this case, we remove the last node from the path and set the immediate upstream node as the current node. Meanwhile, to avoid loops in the path, all the nodes that have already been visited are inserted into the forbidden set F N , so that no node will be visited more than once by the path. By *visited*, we mean that the node has (ever) been selected as a part of the path, although it is possible that the node is removed later from the path.

Algorithm 3. *RandomPath*(n_i, n_j, FN)

Require: *src* is the source node, *dest* is the destination node, and *FN* is a forbidden set that constains the nodes that should be excluded from the path.

1: $CurNode \leftarrow src, path \leftarrow src$.
2: **while** the path has not reached the destination node *dest* **do**
3: $FN \leftarrow FN \bigcup CurNode$;
4: Set $next\_node \leftarrow$ all the nodes that can be reached from *CurNode* through a single link;
5: Remove from next nodes any node that belongs to *FN* ;
6: **if** $next\_node \neq NULL$ **then**
7: Randomly pick a node *next* from $next\_node$;
8: Append path by link $(CurNode, next)$;
9: **else**
10: Remove the last node from *path* ;
11: If there is no more node in *path* ,terminate with failure;
12: $CurNode \leftarrow$ the last node of path;
13: **end if**
14: **end while**
15: **if** *CurNode* is the destination node , return *path* ;
16: **else** return failure

Another key point in the genetic algorithm is the generation of the initial population. We can generate the k-shortest paths(KSP) or k random paths (KRP) from the source node to the destination node as the initial population.

Finally, for the stopping condition that determines when to stop the iterations of the evolutionary process in the main algorithm 1, one way is to set a fixed number for the iterations. But in the simulations we find that it is hard to set a proper value for this number. Instead, we deem the chance of further improvement is very slow and stop the iteration, if the best solution in the population is not improved over the last 3 generations. In this way the number of iterations can be automatically adapted with the progress of the evolutionary and the complexity of the problem. Still, we set a upper limit on the number of iterations. Once the number of iterations reaches that limit, we terminate the iteration no matter whether there is improvement on the solution or not.

4 Performance Evaluation

In this section, we evaluate the proposed algorithms using a topology with 100 nodes and 455 bidirectional links, which is generated by Waxman's random network topology generator [23]. Different options in this genetic algorithm are implemented and compared through extensive simulations. Specifically, we simulated the k-shortest paths as the initial population (KSPIP) and the k random paths as the initial population (KRPIP). The performance is compared with the ILP formulation (ILP) and the heuristic algorithm (k-Heuristic) with k set to be 3 in [1]. For the comparison, we relax the constraints of Eqns. (9-10) in the ILP formulation in [1] to consider the same minimum combined cost routing problem as the problem considered in this work. In the simulation, both α and β in the combined cost are set to be 0.5 for all these four approaches considered in the simulation.

Two types of SRLGs (localized and non-localized SRLGs) are generated and studied. The first method randomly picks a node, and then randomly picks the links that are within N hops from the selected node to be included in an SRLG. If we want to generate an SRLG with certain number of link but there are not enough links within within N hops from the selected node, N will be automatically increased by 1 so that we get enough links. This method produces *N-hop* localized SRLGs. We set the N to be 5 in our simulation. The second method randomly picks links in the topology to be included in an SRLG. This method produces non-localized SRLGs. In both cases, the weight of an SRLG is proportional to its size, i.e., the number of links in the group multiplied by a constant (50 in our simulation). The size of an SRLG (number of links in a SRLG) is uniformly distributed in [8, 12]. For each case simulated, 10 SRLG sets of fixed size are generated. For each SRLG set,100 source-destination pairs are randomly generated. So the total number of simulation instances for a case is 1000. The proposed algorithms and ILP are then run to find the paths. The performance figures are based on the average performance of all the instances with the same settings.

For each case simulated, 10 SRLG sets of fixed size are generated. For each SRLG set, 100 source-destination pairs are randomly generated. Therefore the total number of simulation instances for a case is 1000. The performance figures are based on the average performance of all the instances with the same settings.

As we can see from Figure 1 for the localized SRLG case and 2 for the non-localized SRLG case, firstly, KSPIP and KRPIP are performing very closely, which means the initial population does not really have much influence on the final solution of the genetic approach in both cases. We also observe that, in the non-localized SRLG case (Figure 2), the solutions obtained by KSPIP and KRPIP are close to, but not as good as the solutions obtained by ILP and the k-heuristic. However, in the localized SRLG case the genetic approaches perform as well as the k-heuristic algorithm. Obviously the genetic algorithm works better in the localized SRLG case, which can be reasoned that the segment mutation and crossover on the chromosomes works better for the localized SRLGs. Meanwhile, Figure 3 shows that the time needed to obtain a solution with genetic algorithm, in localized SRLG case, does not grow as fast as ILP and k-heuristic. Both KSPIP and KRPIP take less time than ILP and k-heuristic when the number of SRLGs in the topology is 160, although the genetic algorithm consumes more time when the number of

SRLGs is 20. The reason behind this is, for the ILP approach the computation time usually grows exponentially as the complexity of the network is growing. As for the k-heuristic, one major component of the algorithm is to compare the SRLGs associated with the routing entries on a node, and the complexity compare two sets of SRLGs is $O(|R|^2)$ where n is the number of SRLGs in the topology. For the genetic algorithm the complexity to evaluate a chromosome is $O(|R|^2)$. Therefore the time consumed by both the ILP and k-heuristic grows faster than the genetic algorithm in this work as the number of SRLGs in the network increases. Thus, the genetic algorithm in our work can provide a tradeoff between the cost and the time when the complexity of the network is high. In the meantime, we can see that with k-shortest paths as the initial population, KSPIP consumes less time than KRPIP which uses the k random paths as the initial population. This is reasonable since generally the k-shortest paths are more close to the optimal solution region than the pure random paths, thus it take less time for KSPIP to converge.

5 Conclusions

In this work, we have studied a routing optimization problem in networks with shared risk link groups (SRLGs) to deter- mine a path between a source and a destination such that the combined path cost and the weight of SRLGs to which the links of the path belong is minimized. The path cost is measured as the sum of cost of links on the path while the weight of SRLGs is calculated as the sum of the weight of individual SRLGs along the path. We have developed evolutionary computation based algorithms to solve the problem and compared the performance with previous heuristic algorithm proposed in [1]. Our simulation studies showed that, when the complexity of the problem is high, the proposed algorithm consumes less time to obtain the solution than the heuristic algorithm. In the localized SRLG cases, the genetic algorithm can obtain solutions as good as the ones obtained by the integer linear programming and the heuristic algorithm in [1] but consumes less time when the number of SRLGs in the topology is large.

References

1. Luo, X., Wang, B.: Multi-constrained routing in networks with shared risk link groups. In: IEEE Broadnets, 3rd International Conference on Broadband Communication, Networks and Systems, San Jose CA (October 2006)
2. Hu, J.Q.: Diverse Routing in Optical Mesh Networks. IEEE Transactions on Communications 51(3), 489–494 (2003)
3. Datta, P., Somani, A.K.: Diverse Routing for Shared Risk Resource Groups (SRRG) Failures in WDM OpticalNetworks. In: BROADNETS 2004, San Jose CA, October 2004, pp. 120–129 (2004)
4. Luo, X., Wang, B.: Diverse Routing in WDM Optical Networks with Shared Risk Link Group (SRLG) Failures. In: Proceedings of the 5th IEEE International Workshop on Design of Reliable Communication Networks (DRCN), Island of Ischia (Naples), Italy (October 2005)
5. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer, New York (2003)

6. Nagata, Y., Kobayashi, S.: Analyses of genetic algorithms for traveling salesman problems for effective searches. In: 32nd ISCIE International Symposium on Stochastic Systems Theory and Its Applications, pp. 44–45 (2000)
7. Nagata, Y., Kobayashi, S.: An analysis of edge assembly crossover for the traveling salesman problem. In: Proc. 1999 International Conference on Systems, Man and Cybernetics, pp. III-628–III-633 (1999)
8. Nagata, Y., Kobayashi, S.: Edge assembly crossover: High-power genetic algorithm for the traveling salesman problem. In: Proc. 7th International Conference on Genetic Algorithms, pp. 450–457 (1997)
9. Xu, S.C., Wu, B.X.: Traffic grooming in unidirectional WDM ring networks using genetic algorithms. Computer Communications 25(13), 1185–1194 (2002)
10. Xu, Y., Xu, S.C., Wu, B.X.: Strictly nonblocking grooming of dynamic traffic in unidirectional SONET/WDM rings using genetic algorithms. Computer Networks 41(2), 227–245 (February)
11. Liu, K.H., Xu, Y.: A new approach to improving the grooming performance with dynamic traffic in SONET rings. Computer Networks 46(2), 181–195 (2004)
12. Jiao, Y.G., Zhou, B.K., Zhang, H.Z., Guo, Y.L.: Heuristic Algorithms for Grooming of Arbitrary Traffic in WDM Ring Networks. Photonic Network Communications 8(3), 309–318 (2004)
13. Jiao, Y.G., Zhou, B.K., Zhang, H.Z., Guo, Y.L.: Grooming of Arbitrary Traffic in Optical WDM Mesh Networks Using a Genetic Algorithm. Photonic Network Communications 10(2), 193–198 (2005)
14. Gazen, C., Ersoy, C.: Genetic algorithms for designing multihop lightwave network topologies. Artificial Intelligence in Engineering 13(3), 211–221 (1999)
15. Borella, A., Cancellieri, G., Chiaraluce, F.: Design techniques of two-layer architectures for WDM optical networks. International Journal of Communication Systems 14(2), 171–188 (2001)
16. Zheng, J., Zhou, B., Mouftah, H.T.: Virtual Topology Design and Reconfiguration of Virtual Private Networks (VPNs) over All-Optical WDM Network. Photonic Network Communication 7(3), 255–266 (2004)
17. Banerjee, N., Mehta, V., Pandey, S.: A genetic algorithm approach for solving the routing and wavelength assignment problem in WDM networks. In: International Conference on Networks (ICN 2004), French Caribbean (2004)
18. Bisbal, D.: Dynamic routing and wavelength assignment in optical networks by means of genetic algorithms. Photonic Network Communications 7(1), 43–58 (2004)
19. Le, V.T., Ngo, S.H., Jiang, X., Horiguchi, S., Guo, M.: A genetic algorithm for dynamic routing and wavelength assignment in WDM networks. In: Proc. Inter. Symp. Parallel and Distributed Processing and Applications, HongKong (December 2004)
20. Ali, M., Ramamurthy, B., Deogun, J.S.: Routing and wavelength assignment with power considerations in optical networks. Computer Networks 32(5), 539–555 (2000)
21. Chan, T.M., Kwong, S., Man, K.F.: Solving the Converter Placement Problem in WDM Ring Networks using Genetic Algorithms. Computer Journal 46(4), 427–448 (2003)
22. Vijayanand, C., Kumar, M.S., Venugopal, K.R., Kumar, P.S.: Converter placement in all-optical networks using genetic algorithms. Computer Communications 23(13), 1223–1234 (2000)
23. Waxman, B.M.: Routing of multipoint connections. IEEE Journal on Selected Areas in Communications 6(9), 1617–1622 (1988)