Optimal Virtual Path Routing
Using a Parallel Annealed Genetic Algorithm

Murat Eren  Cem Ersoy

Computer Networks Research Laboratory
Computer Engineering Department
Bogazici University, 80815 Bebek, Istanbul, Turkey
ersoy@boun.edu.tr

ABSTRACT
The coexistence of a wide range of services with different quality of service (QoS) requirements in today’s networks makes the efficient use of resources a major issue. One way of managing these resources is employing the virtual path concept. In virtual circuit oriented networks such as ATM, the overall performance of the network is strongly affected by the optimization of virtual paths. Here, the objective of this optimization is to minimize the maximum link utilization in the network, where the network topology, link capacities, and traffic requirements are given. For this optimization, a parallel annealed genetic algorithm (PAGA) is used. Since the performances of genetic algorithms depend a lot on their parameters, we run each genetic algorithm with different parameters on each computer in parallel. We also perform some information exchange between the computers during the run. In order to measure the quality of the solutions; extensive computational experiments are performed using different network topologies, and different traffic requirements. The performance of PAGA is compared to several competitors for these different problems.

I. INTRODUCTION
Modern networks are accommodating a wide range of services with different traffic characteristics. Use of virtual paths is an efficient way of controlling the resources on these networks such as ATM networks [1]. In ITU-T terminology, a VP is defined as a labeled path (bundled multiplexed virtual channels or connections) between its end-nodes. The VP concept and implementation issues are addressed in detail in [2]. Assuming that all connections are made using VPs, this logical VP network is a very flexible tool that can be employed in the management of network resources. The configuration of the VP network has a major impact on the processing/control costs and transmission costs of a network.

This work deals with the efficient usage of VPs in networks in which not only conventional processing time constraints exist, but also link capacities are limited. These constraints are expected to occur more frequently as wireless technology for multimedia is starting to consider the use of VP concept. In today’s communication community there is a rapidly growing interest in wireless networks to provide a medium for the transport of voice and data with the advantages of mobility. For supporting multimedia applications in wireless systems, ATM is being considered as a major potential contender [3]. In these networks, the available link bandwidth (radio bandwidth) can be severely constrained in the whole network or in segments of it. Any practical solution to ATM based VP communication must therefore consider this limitation. Also in wired telecommunication networks, network bandwidth may again become a limited resource. Furthermore, even in the hypothetical case of a practically unlimited bandwidth, it is important to distribute traffic in a way that reduces the maximum link load in order to increase network robustness. Clearly, the higher the maximum load on any specific link in the network, the more catastrophic may be the effect of the failure of a link carrying a potentially very larger number of connections. Also in a network with different capacity links, which is the usual case, there is the probability of connection blocking, which may be explained as not being able to find enough resources and therefore rejecting a connection request. Therefore, the load has to be distributed in a way that the connection-blocking probability is minimized.

Extensive research on optimizing the system of VPs has been performed. Connection blocking probability is used frequently, sometimes in the cost functions, sometimes as a constraint. Detailed review of previous work on optimizing the system of VPs can be found in [12].

Since the VP optimization is a difficult combinatorial problem, we choose the genetic algorithms as our solution technique. With the improvement of the original algorithm, we developed a parallel annealed genetic algorithm (PAGA).

The rest of the paper is organized as follows. In the following section, we define our problem. Section 3 describes the problem solving techniques that we used. Section 4 presents computational experiments, results, and comments on the performance of the algorithm. Finally, a summary, conclusions, and subjects for further work are given in Section 5.
II. PROBLEM DEFINITION AND FORMULATION

We have chosen our objective function to be the maximum of the link utilizations. If the utilization of all links is minimized, then there will be more available bandwidth on each link. We did not distribute the traffic equally on all links, since the link capacities may be different and not utilizing high capacity links is a waste of resources.

We model the network topology by a directed graph $G=(V,E)$, where $V$ is the set of nodes (vertices) and $E$ the set of links (directed edges). It is assumed that $G$ is strongly connected, i.e., there is a directed path from any node $i$ to any other node $j$. A virtual path (VP) is defined by a directed route $V_{ij}$ from vertex $i$ to vertex $j$, and a capacity $C_{ij}$. The capacity, also referred to as demand, expresses the fact that a given portion of transmission capacity is reserved for the VP to serve the traffic demand brought about the virtual path. For simplicity, we assumed that the resources are assigned deterministically. In reality, there are some statistical parameters and the actual traffic rate may change within these limits. We assume that the bandwidth assigned to a VP is determined by a mechanism called equivalent bandwidth [4]. The idea is to find the required deterministic bandwidth from the statistical QoS requirements.

The objective function should reflect all parameters that are taken into account when considering the network cost. In this stage, we opted to use a very simple objective function given in Equation (1) because of the need to calculate its value repetitively.

$$\text{minimize } \{ \max_j U_{ij} \}, \quad \text{where} \quad U_{ij} = \frac{L_{ij}}{C_{ij}} \quad (1)$$

Our objective function is minimizing the maximum utilization $U$ among all links. $L_{ij}$ is the load of physical link from node $i$ to node $j$ and $C_{ij}$ is capacity of physical link from node $i$ to node $j$. This objective function is very simple. Only summing the flows from node $i$ to node $j$ and a division over the capacity are needed. Although the objective function is very simple, it is also very powerful. We are distributing the traffic uniformly (weighted with the link capacities) on the whole network. This provides a robust network. In the case of a link failure, minimum traffic will be affected. Since the traffic is distributed, the slack capacity is also distributed, which results in a smaller call-blocking rate.

The constraints are simply capacity limitations of the network. Additionally, there should be a path in the VP Network to carry each defined traffic stream, and no traffic stream is left out in the VP Network.

Since the solution space grows exponentially in the increasing number of nodes and links, this is a difficult combinatorial optimization. In such cases, heuristic algorithms are developed for reaching a suboptimal solution. Details about the solution techniques we used and the implementation can be found in the following section.

III. PARALLEL ANNEALED GENETIC ALGORITHM

Genetic algorithms [5] (also called evolutionary computing) are “search procedures” based on the mechanics of natural selection and natural genetics, which can be summarized as the survival of the fittest. Genetic algorithms are applying this rule of the nature to computationally hard problems. The algorithm operates on a set of offered solutions called “populations.” Each solution, called an “individual,” can be any solution in the solution space represented by a string called “chromosome.” Different solutions will be coded into different chromosome values. The solution space is searched in order to find the optimum individual according to a “fitness function.” Each individual is evaluated by the fitness function. The resulting value is referred as the “fitness value” of the solution.

The current population is evolved creating a new generation with higher fitness. There are three operators:

- Reproduction: Individuals from the current population are selected for the next generation.
- Crossover: Two individuals are mated in order to exchange genetic information.
- Mutation: Mutation is a random change in the chromosome. This operator gives the genetic algorithms an opportunity to search in new corners of the solution space.

In our case, each gene will represent a path $p_{ij}$ from node $i$ to node $j$. These paths are kept using one byte codes of the nodes in a null-terminating string. The algorithm first finds the shortest paths from each node to every node according to hop distance. These paths are assigned to the first individual. Then some random weights are assigned and again shortest paths will be found according to these weights. These paths are assigned to the rest of the individuals. In this way, first generation is created. Additionally, the fitness values are calculated using the fitness function and recorded to prevent unnecessary calculation of reproduced individuals.

Next comes the iterative part. In each iteration, first crossovers and mutations are performed on the individuals according to their respective probabilities. This will be done (maximum population size - population size) times or until they reach the maximum population size. Each time random numbers are generated, and they are checked against the probabilities of crossover and mutation, to see whether this trial will result in a new individual. Right after these operations the new individual’s fitness value is found and recorded. If its fitness value is below a threshold, the individual will be killed immediately (in analogy with the dead births in the nature). This method will speed up the process and increase the convergence to the solution. At the end of the iteration, reproduction is performed. In this step, some individuals are selected
for the next generation and others will die. The whole process can be followed in Figure 1.

![Figure 1: Iteration of genetic algorithms.](image)

The algorithm stops when no more improvement statistically can be achieved or an upper limit value of the fitness value (a satisfactory solution for the user) is reached. Note that there are many parameters besides the ones related to topology and traffic streams. First allocation size, crossover rate, mutation rate, population size (maximum and normal), living threshold according to fitness value, stopping statistics will greatly affect the performance of the algorithm both in the speed and quality of the result. The variety of parameters, many possible choices, and the complexity of the interactions between various parameters make the perfectly tuned genetic algorithm for a given problem very difficult, if not impossible. This is the nature of the genetic algorithms and many researches have been performed on finding the optimum set of parameters for genetic algorithms. There is a very good survey of various forms of control, which have been studied in evolutionary computation community in recent years, with 144 references in [7]. Their classification covers the major forms of parameter control in evolutionary computation and suggests some directions for further research.

We used a Message Passing Interface (MPI) based cluster computer [ASMA] for running our parallel annealed genetic algorithm. Message passing is a powerful and very general method of expressing parallelism. Message-passing programs can be used to create extremely efficient parallel programs, and message passing is currently the most widely used method of programming many types of parallel computers [8]. Different genetic algorithms can be run on a different processor and interchange data using message passing. In fact, we are not running totally different algorithms on each computer. However, we are running the same algorithm with different parameters. In this way, we can eliminate the disadvantage of the parameter sensitivity of genetic algorithms. Usually some tests are performed to tune the genetic algorithm for a specific problem. However, it is almost impossible in our case, since very different network topologies, and very different traffic distributions can occur. Having some fixed values for the important parameters (namely crossover and mutation rates) will not be proper for our problem. Some research is done on adaptively changing these parameters during the run of the program. Unfortunately, how to adapt the parameters to the current problem is another unsolved question and it is an important research area [7]. However, if we run our genetic algorithm with different crossover and mutation rates on each computer, we overcome the hardest problem and disadvantage of genetic algorithms. At least one of the parameter sets will be relatively suitable for the current problem.

![Figure 2: Better parallelism with some exchanges.](image)

The most basic idea would be running the program with different parameters on each computer, and then collect the results, and present the best one as the solution. On the other hand, genetic algorithms by themselves are parallel search algorithms. They perform their search on a number of individuals. Unfortunately, the selection step makes it infeasible for the implementation of a completely parallel genetic algorithm, because we have to select the best ones among the whole individuals. In the case of a completely parallel genetic algorithm, we should redistribute selected individuals to the computers equally at every iteration, which results in too many transactions between the computers.

Nevertheless, we can select a way between the two extreme solutions (see Figure 2). That is the algorithm will run separately with different parameters on different computers. If we do not exchange any information during the separate runs, then our gain from the parallelism would be less. However, if we make some information (individual) exchange at some particular places during the run, then we will benefit more from the parallelism.

How frequent these exchanges should be is another question. However, the answer is the result of a simple tradeoff between the CPU speed and transfer speed. If our CPUs are much faster than our connections (transfer speed), then we should make less frequent exchanges. However, if our connections are very fast, we can easily make exchanges that are more frequent. The synchronization required is provided with a very simple function called MPI_Barrier. Each separate program will continue only after all the programs come to the point of that function call.
Let us give a real life example to better visualize the situation. We can think of each computer as a separate island. The population of each island will mate among each other to form new generations. From time to time, some of the individuals will travel to one of the other islands over the sea. Of course, only the individuals, who are stronger (or richer), dare this journey. How frequent they can travel is dependent on the distance between the islands (transfer speed). Another point is that the visitors will have a higher change in the new island, since a newcomer will be more interesting to mate. In this way, the genetic information of the visitor will be distributed more in the new island. At the end, a competition will be performed, to which the best individual of each island will be send. The winner is our solution.

**IV. COMPUTATIONAL EXPERIMENTS**

The parallel annealed genetic algorithm (PAGA) proposed in this study tries to optimize the network performance by minimizing the maximum link utilization. As far as we know, there are no computational results or numerical examples provided in the literature achieved by using exactly the same objective function and constraints to test the quality of the proposed algorithm directly. A lower bound for the objective function is also hard to find since the link utilization is not an absolute value but the ratio of the used capacity over the total capacity of the link. Moreover, a network large enough to yield a non-trivial VP network is too large for an exhaustive search unless the number of VPs to be routed is limited, which is not a realistic approach since a traffic matrix is normally not sparse. In a fully connected network (i.e., there is a direct physical link between each network) with four nodes, there are three possibilities for each VP. If all nodes will communicate with the other three nodes, then there are 12 VPs to be routed. Since there are three possible ways of routing each VP, there are $3^{12}$ (which equals 531,441) different solutions for the VP network topology. The solution space grows exponentially with the increasing number of nodes and links. Therefore, test algorithms have to be developed explicitly to see the quality of the results. The first competitor is our algorithm itself, but running on a single computer. As we will see in the results, this one will be our strongest competitor. The second competitor is a solution technique based on pure simulated annealing (SA). The implementation details of the SA can be seen in [12]. We also applied a statistical goodness measure. The first and commonly applied method for complex problems is the statistical goodness measure described in [9]. In this measure, a large number of solutions are generated for each examined problem. A histogram corresponding to objective values of these solutions are created and compared with the solutions of the PAGA.

PAGA is first tested regarding the parameters of itself. It is important to see the effects of the parameters to understand the different methods used in PAGA. These parameters are

(a) $\alpha$ and $\text{astep}$, which are the parameters related to the annealing process;
(b) the minimum improvement rate $\text{minimp}$, an important stopping criteria;
(c) the population size, which is an important parameter in the genetic algorithms;
(d) exchange rate, how frequent data (individual) exchange is performed between different computers;
(e) and of course the number of computers on which the PAGA is running.

After the parameter tests are finished, PAGA is tested against its competitors described above regarding two important criteria in the evaluation of a network design methodology. These are the network size and the traffic type. In fact, also the traffic load is an important criterion in the evaluation. However, since we are focusing on the link utilization, which is not an absolute value, but a ratio, we can easily relax the conditions by multiplying all link capacities by the same factor. Since we minimize the maximum link utilization, the solution will be the same on the real and the relaxed network.

Four different networks, with network sizes 26, 25, 35, and 50, and with link capacities of 155, 64 and 34 Mbps is used for testing. The 26 node, 41 link network is the USA network topology given in [10]. Other networks are randomly generated networks. Three different random traffic generators are used to create the traffic requirements in three different types. The first one is uniform traffic, where the demand between nodes is normally distributed. In the second type, namely the neighborhood type traffic generation, the idea in [11] is used. The nodes that are close to each other are more likely to have larger traffic requirements. The last traffic type is called community of interest. In this type, it is assumed that there are user groups in the network. The traffic between members within the same group is defined to be higher compared to the traffic between members from different groups. All tests are run on Beowulf Cluster in Boğaziçi University [asma.boun.edu.tr]. This is a network based parallel machine with 14 nodes, which are connected to each other with the speed of 10 Mbps. Each computer has the same hardware configuration, a Celeron 300A CPU and 32 MB RAM. RedHat Linux 6.0 is our operating system, and we are using mpich 1.1.2.
Table 1: Test results for the 25 node network.

<table>
<thead>
<tr>
<th>Traffic type</th>
<th>Average value (PAGA)</th>
<th>Average value (GA)</th>
<th>Average value (SA)</th>
<th>MoR</th>
<th>Average value (PAGA)</th>
<th>Average value (GA)</th>
<th>Average value (SA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>0.723</td>
<td>0.674</td>
<td>0.633</td>
<td>0.539</td>
<td>20.28</td>
<td>1.75</td>
<td>4.46</td>
</tr>
<tr>
<td>Neighborhood</td>
<td>0.728</td>
<td>0.676</td>
<td>0.663</td>
<td>0.560</td>
<td>18.35</td>
<td>1.71</td>
<td>7.89</td>
</tr>
<tr>
<td>Community of interest</td>
<td>0.705</td>
<td>0.655</td>
<td>0.638</td>
<td>0.508</td>
<td>19.19</td>
<td>2.15</td>
<td>6.52</td>
</tr>
</tbody>
</table>

Figure 3: Distribution of fitness values of one million randomly generated solutions (25 nodes).

Because of space limitations, we only present the results of the tests on the 25 node networks. Other results can be seen in [12]. In Table 1, we present the results of the tests on the 25 node network. GA column shows the results when the genetic algorithm is run on a single computer. SA column presents the results when the VP network is optimized using simulated annealing. MoR (maximum of random) column shows the maximum fitness value found in the generation of one million random solutions. Figure 3 shows the distribution of the feasible solutions when one million random solutions were generated for each traffic type. PAGA outperforms all its competitors in the quality of the result while having worse run duration. However, we will not focus on the run duration, since our testbed is not appropriate for this. Note that GA performs better than SA both in the quality of the result and the run duration. All algorithms perform better than the maximum value found in one million random generations.

Figure 4: Result ranges of algorithms for 25 node network.

Another important criterion is the range of the quality of the solutions or in other words how stable the algorithm is. That is whether the algorithm gives once a high result, once a low result, or it always gives results close to the average. In Figure 4, we see that PAGA also performs very well according to this criterion. Each line shows the minimum, the maximum and the average of the quality of results in 100 tests. Each three line belongs to a traffic type, from left to right, uniform, neighborhood, and community of interest.

V. CONCLUSIONS

In this study, a GA based method for optimizing the VP network topology is proposed. When the VP network is optimized, it becomes possible to efficiently use the network resources under QoS constraints. We tried to increase GA’s performance by using annealing in the selection mechanism of GA. Parallel programming is used to overcome the dependence of algorithm’s performance on the parameters. The resulting algorithm is called parallel annealed genetic algorithm (PAGA). We formulated a VP optimization problem, and we coded the solution into the genes. The effects of different parameters are tested on two network sizes (25 and 26 nodes) with partial factorization. Extensive computational experiments on 25, 26, 35, and 50 node networks are performed. The quality of results were compared to those of the genetic algorithm on a single computer, simulated annealing, and the maximum fitness value found in one million random generations. The algorithm performs very well if the quality of results is considered. It outperformed all of its competitors in this criterion. However, the running time (0.5-1 minute) is not suitable for a real-time optimization. On the other hand, the algorithm may be very useful in creating a new network or after adding a new node. In addition, it may be run in time slices, for example every several minutes, and then apply the new VPs to the network. In networks connecting WANS, the traffic requirements will not vary too much. Also changing the virtual paths brings a table setup time. Therefore, it may not be a good idea to change the VPs frequently. Our algorithm should be used for permanent virtual circuits, since PAGA requires information about all the traffic requirements. Therefore it must be run on a central place, and then the routing information will be distributed to nodes. PAGA is not suitable for switched virtual circuits.

Even in the case of an emergent need of new VP network configuration, e.g., in the case of link or node failures, the algorithm can be run without the exchange operation. In that case, the algorithm responds very quickly (in several seconds). Although there is a drop in the quality of results, this drop is tolerable. Even in this
case PAGA outperforms its competitors in the quality of results, while competing in the run duration. The parameter exchange rate supplies us a very elegant control mechanism. The performance of PAGA in the run duration in clusters with higher speed interconnections is a subject for further research. Since PAGA does not provide an optimal solution but a suboptimal one, the degree of its optimality is another subject for further research. We performed extensive tests for both obtaining a suitable set of parameters for the PAGA algorithm and for comparing our heuristics under different workload scenarios on different size networks. Proposed heuristic is giving promising results for constructing a virtual path set with a minimal risk of congestion since relatively less utilized links are preferred. We plan to solve similar optimization problems using PAGA and compare its performance with other alternative solution techniques.

ACKNOWLEDGEMENTS

Authors would like to thank Dr. Can Ozturan and Dr. M. Ufuk Caglayan for supplying the parallel cluster ASMA for this study; Burak Gurdag for administering and keeping the cluster up.

REFERENCES


