Proceedings of the 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION, Corfu, Greece, August 17-19, 2005 (pp482-487)

Genetic-based Traffic Engineering in GMPLS networks

FRANCESCO PALMIERI and UGO FIORE
Centro Servizi Didattico Scientifico
Università degli studi di Napoli "Federico II"
Complesso Universitario di Monte S. Angelo,Via Cinthia 5
NAPOLI - ITALY

Abstract: - The exponential growth of the Internet and the trend toward providing differentiated classes of service has placed heavy burdens on network management and control operations. Adding more resources may temporarily relieve congestion conditions, but it is not a cost-effective solution in solving resource contention problems in the long run. What service providers need are effective traffic engineering mechanisms to coordinate, control, and efficiently utilize existing resources to satisfy customer demand. MPLS has emerged as a potential enabling technology for those tasks by realizing, with native tunneling techniques, multiple explicit label switched paths throughout the network. In this paper the optimal label switched path displacement problem is formulated as a multi-objective optimization problem and solved using a genetic algorithm-based approach. Several solutions can be proposed and the one that best matches the traffic engineering requirements be chosen.

Key-Words: - Traffic Engineering, genetic Algorithms, network optimization, MPLS.

1 Introduction

High-speed optical networks are expected to support a wide variety of communication-intensive real-time multimedia applications. Consequently, network carriers are facing the challenge of designing and managing their networks to support fast, reliable and quality-differentiated services. At the same time, they wish to maximize the usage efficiency of their network infrastructure, by optimizing connectivity resource usage while meeting customer service level agreements. To accomplish these goals, Internet Traffic Engineering (TE) draws on modeling, characterization, control and performance optimization of the network traffic. Optimization in this context refers to the transport of IP packets in the most efficient, reliable, and expeditious manner possible, according to the customer resources (i.e. bandwidth) and QoS requirements (latency, jitter etc.), through a given network [1]. Essentially, TE is required mainly because the current dynamic routing protocols always use the shortest paths to forward traffic. This practice conserves network resources, but it causes some resources of the network to be over utilized while the other resources remain under-utilized. Furthermore, the actual most used dynamic routing protocols never take into account specific traffic flow requirements such as bandwidth and QoS needs. Multi-protocol label switching (MPLS) and constraint-based routing (CBR) are the two key elements of all the modern TE frameworks in the Internet since they can easily be used to ensure fair traffic distribution, minimizing resource contentions and improving overall network utilization by realizing, with several tunneling techniques, multiple explicit alternative paths throughout the network. However, both these technologies require, for optimal tunnel/path displacement, dynamical network optimization based on multiple and apparently unrelated metrics and constraints with the objective of optimizing the network resource utilization according to cost and performance criteria. This optimization problem can be formulated as follows: given a dimensioned network and a traffic demand matrix, we would like to find a multi-path routing solution, which optimizes a certain network QoS measure. The combination of multiple different metrics and constraints makes this kind of routing optimization problem NP-complete. There are two approaches to solve such problems. One is to directly construct the correct solution based on the constraints, using brute force. Although this approach yields the accurate solution satisfying all the constraints, often it is not feasible for large problems, simply because they cannot be solved in computationally tractable time.

The other approach is to use a heuristic-based solution that can be computed in feasible time and may produce a near-optimal solution. The paradigm of "evolutionary" programming that mimics the evolution process of the nature, provides the necessary means to solve such complex problems. An important example of evolutionary programming model is the class of genetic algorithms (GA). It is basically a learning technique, in which the less fit solutions are removed from a representative solution space (evolution) and are replaced by better ones produced by applying some operators on the existing solutions. This, progressively refines the search space and makes the problem computationally manageable, by enabling the searching procedure to converge quickly to yield a near-optimal solution in computationally feasible execution time. Consequently, in order to deal with the high computational power required by the above problem, we propose the use of a Genetic optimization approach to dynamically obtain effective engineered routing solutions generated through multi-point searching based on the species evolution model. By using the genetic optimization method, we considered the network as a multistage process of chromosome reproduction based on a properly crafted fitness function, and solved the MPLS LSP optimization problem as a multi-objective optimization problem by genetic crossover and mutation steps to find an optimal engineered network layout of maximum performance score. The metrics characterizing the objectives are simultaneously minimized to obtain optimal paths and have never been combined into one, so that several solutions can be proposed and the one that best matches the TE requirements be chosen. The proposed solution has been evaluated through simulation to demonstrate that the approach is a significantly effective and robust method to overcome the disadvantages of the classic heuristic-based solutions.

2 Basic Concepts

This section briefly illustrates the background concepts needed to explain the proposed framework.

2.1 TE in next generation networks

Roughly speaking, it is often asserted by practitioners in the field that TE in large scale IP networks essentially boils down to the ability to place traffic where the capacity exists to accommodate it; whereas network engineering, on the other hand, boils down to the ability to install capacity where the traffic exists. MPLS has emerged as a potential enabling technology for TE in connection-oriented packet networks [2]. The signaling protocol (e.g., RSVP-TE) provides mechanisms for establishing multiple alternative label switched paths (LSPs) to facilitate explicit routing [3]. Stimulated by recent progress in optical networking, there has also been a growing interest in designing the control plane (i.e., routing and signaling) for the optical layer based on reusing and leveraging existing control-plane protocols. To this end, Generalized MPLS (GMPLS), which is an extension of MPLS, is emerging as the candidate control-plane solution for next-generation optical networks [4] based on constraint-based routing optimization methods.

2.2 Genetic Algorithms

Genetic Algorithms (GAs) are stochastic algorithms whose search methods model a natural phenomenon, the genetic evolution. In evolution, the problem each species faces is searching for beneficial adaptations to a changing environment. The "knowledge" that each species has gained is embodied in the makeup of the chromosomes of its members. In this light, GAs can be essentially classified as learning techniques. GAs have been successfully applied to a wide class of optimization problems. So it is evident that network optimization for TE is a major field of GAs' applicability. Genetic algorithms are also based on the mechanics of natural selection. They work with a population of genetic entities, each representing a possible solution to a given problem and hence the whole search space in GA-based problem is composed of any possible solutions to the problem. Each entity, as a solution in the search space is represented by a sequence of simple or complex series of numeric values also called "gene". This solution string is also referred as a chromosome in the search space. Each chromosome has an associated objective function called the fitness. A good chromosome is the one that has a high/low fitness value, depending upon the nature of the problem (maximization/minimization). A set of chromosomes and associated fitness values is called the population. This population at a given stage of GA is referred to as a generation. The highly fit chromosomes are given opportunities to reproduce by cross breeding with others in the population. A new population of possible solutions

is thus produced by selecting the best chromosomes from the current generation and mating them to produce a new set of chromosomes. Consequently, fitness values indicate also which chromosomes are to be carried to the next generation. There are three main processes in the while loop for a typical GA:

1. The process of selecting good chromosomes from the current generation to be carried to the next generation. This process is called selection/reproduction and its main task is to originate new generations starting from existing generations.
2. The process of shuffling two randomly selected chromosomes to generate new offspring is called crossover. Sometimes, one or more bits of a chromosome are complemented or randomly changed to generate a new offspring. This process of complementation is called mutation since it introduces new genetic material into a chromosome by randomly selecting and changing single genes.
3. The process of replacing worst performing chromosomes based on the fitness value. The population size is finite in each generation of GA, which implies that only relatively fit chromosomes in generation *(i)* will be carried to the next generation *(i + 1)*.

The power of a GA comes from the fact that the algorithm terminates rapidly to an optimal or near optimal solution, when there is little or no change in the quality or fitness of solutions in the population, so that with successive application of GA operators the best one in the population is selected as final solution. It is suggested that an individual's strength to survive in the world is determined by its gene structure and that over many generations only "good" genes prevail, whereas "bad" ones are rejected. Genetic algorithms apply this principle to optimization problems by representing possible solution alternatives through appropriate gene strings and performing operations of "natural selection" on these strings [5].

3 Evolutionary routing optimization

Our approach to routing optimization for TE is based on two theories: the multi-objective optimization and the genetic-based solution search theories. Here, multi-objective optimization aims to obtain an efficient solution for several objectives, where any improvement in one objective can only be achieved at the expense of another. Pragmatically, in our problem of determining, according to traffic demands, optimal (G)MPLS paths in a complex network to strengthen traffic management capabilities, prevent congestion and achieve differential Class of service and QoS requirements, we are not interested in the optimal solution but in one of the possible solutions that optimizes at their best all our individual objectives, in respect of the superimposed problem constraints. In detail, a general Multi-objective Optimization Problem includes a set of *n* decision variables, *k* objective functions, and *m* restrictions. Objective functions and restrictions are functions of decision variables. This can be expressed as the optimization of $\mathbf{y} = \boldsymbol{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))$ subject to $\boldsymbol{e}(\mathbf{x}) = (e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_m(\mathbf{x})) \geq \mathbf{0}$, where $\mathbf{x} = (x_1, x_2, ..., x_n) \in \mathbf{X}$ is the decision vector, and $\mathbf{y} = (y_1, y_2, ..., y_k) \in \mathbf{Y}$ is the objective vector. $\mathbf{X}$ denotes the decision space while the objective space is denoted by $\mathbf{Y}$. The set of restrictions $\mathbf{e}(\mathbf{x}) \geq \mathbf{0}$ determines the set of feasible solutions $\mathbf{X}_f$ and its corresponding set of objective vectors $\mathbf{Y}_f$. The problem consists of finding $\mathbf{x}$ that optimizes $\mathbf{f}(\mathbf{x})$. In general, there is no unique "best" solution but a set of solutions. Thus, a component-wise order relation should be defined between objective vectors. Given two decision vectors $\mathbf{u}$ and $\mathbf{v}$, we say that $\mathbf{u}$ *dominates* $\mathbf{v}$ and write $\mathbf{u} \prec \mathbf{v}$ *iff* $\mathbf{f}_i(\mathbf{u}) \leq \mathbf{f}_i(\mathbf{v})$, $i = 1, \ldots, k$ and $\mathbf{u} \neq \mathbf{v}$. The *Pareto optimal set* is the set of non-dominated decision vectors, i.e., the points $\mathbf{v}^* \in \mathbf{X}_f$ such that there is no $\mathbf{v} \in \mathbf{X}_f$ such that $f_i(\mathbf{v}) \leq f_i(\mathbf{v}^*)$ $\forall i = 1, \ldots, k$ and $f_j(\mathbf{v}) < f_j(\mathbf{v}^*)$ for at least one *j*. The corresponding objective vectors form the *Pareto front*. The set of best alternatives can be obtained through a progressive multi-step refinement process in which new feasible and possibly better solutions are continuously generated and evaluated according to a cumulative objective, or fitness function and finally accepted as optimal, or near optimal solutions when there is little or no change in their quality. The above approach, resembling the process of species evolution with multi-points searching to find an optimized solution is typically proper of the evolutionary programming model and in particular of the Genetic Algorithms, that can be used to generate in an acceptable time solutions belonging to the *Pareto front*. Accordingly, in our specific problem the network can be

modelled as a direct graph, $G = (V, E)$ where V is the set of nodes and E is the set of links. Let $(i, j) \in E$, be the link from node i to node j. For each link (i,j), let $z(i,j)$, $c(i,j)$, $d(i,j)$ and $t(i,j)$ be its capacity, cost per byte/sec., delay and current traffic, respectively. Let $s \in V$ denote a source, $N \subseteq V - \{s\}$ denote the set of all the possible destinations, and ϕ the traffic demand of the current connection requests. Let $P_T(s, n)$, denote a feasible path in the network that connects a source node s with a destination node $n \in N$. The routing problem may be stated as a Multi-objective Optimization Problem that tries to find a set of paths that minimizes for each path P_T, the maximum delay D_M, the path cost C, the maximum traffic α_T flowing on the path, that can be seen as the overall link utilization, and the average delay D_A, expressed as:

$$D_M = \underset{n \in N}{Max}\left\{ \sum_{(i,j) \in P_T(s,n)} d(i,j) \right\} \quad (1)$$

$$C = \sum_{(i,j) \in P_T(s,n)} c(i,j) \quad (2)$$

$$\alpha_T = \underset{(i,j) \in P_T(s,n)}{Max}\left\{ \frac{\phi + t(i,j)}{z(i,j)} \right\} \quad (3)$$

$$D_A = \frac{1}{|N|} \sum_{n \in N} \left[\sum_{(i,j) \in P_T(s,n)} d(i,j) \right] \quad (4)$$

subject to:

$$\phi + t(i,j) \le z(i,j) \quad \forall (i,j) \in P_T(s,n) \quad (5)$$

3.1 The fitness function

As we would like to minimize some or all the above values in the network, according to our TE requirements, we can combine them, with some fixed weights depending from the importance we give to each and choose the inverse of this value as our fitness function. In this way, solutions with smaller results receive higher fitness values and, thus, have an higher chance to be reproduced when setting up a new generation. Let M_i each maximum value to be minimized and d_i its corresponding weight, in order to influence the reproduction process in our GA we apply power scaling to the following evolution function that plays the role of the environment, rating solutions in terms of their "*fitness*":

$$fitness = \left(\frac{1}{\sum_i d_i M_i} \right)^p \quad p > 0 \quad (6)$$

With $p < 1$ we can achieve that fitness values of bad solutions are increased relatively to the best ones, thus, avoiding that they die out too fast and that the optimization process converges too early. For $p > 1$, the gap between good and bad solutions is increased, forcing the process to converge faster.

3.2 Chromosome representation

The string representation of possible solution alternatives is a crucial point of every genetic algorithm thus, in order to apply a GA to solve the problem defined in the previous sections, a suitable encoding of possible solutions in a vector (i.e. chromosome) representation is needed. A chromosome is represented by a string of length $\lceil \log_2 k \rceil \cdot n$, in which each element, the gene g_i, represents a path between a node s in V and a node n in N. If g_i assumes a *null* value then traffic from s to n should be routed according to the plain IGP-determined shortest path. A hybrid approach is adopted for the generation of the initial population, so that the exploration of the search space is significantly reduced. For every source-destination pair, the k-shortest paths connecting them are computed by a constrained SP algorithm. Each gene in the chromosome represents one of the k-shortest paths chosen at random. A single chromosome contains thus a set of feasible paths for all the source-destination pairs.

3.3 Selection

All chromosomes will be selected according to their fitness and consequently we want a solutions with a fitness value as high as possible. There are two selection mechanisms, respectively to select parent chromosomes for a new generation and to remove some of bad chromosomes from the current population. For the first task we implement so called "rank selection" to make the probability to be selected a little bit more balanced for all chromosomes in the population. We first rank the population and then every chromosome receives a probability value from this ranking. The probability value is measured relative to the probability value of the last

(worst) chromosome i.e. the last but one will have twice that probability etc. For the second task we simply sort the chromosomes according to their fitness from good to bad and then remove some of the last chromosomes.

3.4 Crossover and Mutation

The reproduction process creates a new generation. Starting from an existing generation, chromosomes are reproduced with a probability proportional to the quality of the corresponding solution. Chromosomes representing solutions with good properties have a higher chance to survive than those depicting solution points with bad characteristics ("survival of the fittest"). The crossover operator chooses pairs of strings, breaks up their gene sequence at random places, and exchanges the genetic information to produce new chromosomes. This means that all offspring's genes will be inherited either from the first parent or from the second one. Finally, the mutation operator introduces new genetic material by randomly selecting and changing single genes. Mutation is important to partially shift the overall search process to new locations within the solution space. Otherwise, the search process would converge to a local optimum without having the chance to consider any further points. In our implementation, single-point crossover is performed by generating a random crossover point into the chromosome structure. Each gene g_i of offspring o_1 will be inherited from the gene g_i from parent p_1 if its position is less of equal than the crossover point otherwise the gene g_i of o_1 will be inherited from the gene g_i from p_2. The complementary rule exists for the gene g_i of offspring o_2. For mutation we generate another real number; if this number is lower than a fixed *mutation probability* the offspring's genes will be arbitrarily mutated by randomly modifying some of its genes. At every crossover or mutation step the GA has to be able to turn the resulting chromosome into valid solutions.

4 The GA implementation

The genetic algorithm requires that certain input parameters be set, such as the maximum number of generations, number of chromosomes, crossover probability, and mutation probability. Several studies indicates that a crossover probability that is too high could destroy good solutions faster than they are produced, while a crossover probability that is too low may inactivate the search process. In addition, a small value of mutation probability is always used because a high value of mutation probability is essentially equal to a random search. The general structure of the GA implemented is sketched in Figure 1 below.

```
GA_RTTE( RC , t , p , c , m , n ) {
P = empty; Initialization (); Evaluation ();
while ( Max( fitness(g_n in P ) ) > t ) {
    PS = empty;  Select (); Crossover (); Mutation ();
    P = PS; Evaluation (); }
for ( i = 1;  i ≤ p;  i++ )
    if ( Rank(g_i in P ) == 1 ) then
        send g_i to control plane;
}
Initialization () { // generate initial population
while (| g_i | ≤ p)
    for (j  = 1; j ≤ n; j++ )
        generate new g_i using r_j in RC ;
        if (g_i not in P ) then add g_i to P ;
}
Evaluation () { // compute fitness for each gene
for (i = 1;  i ≤ p;  i++ )
    for (j  = 1; j ≤ n; j++ ) {
        calculate fitness(r_j in g_i);
        fitness(g_i )=Max(fitness (r_j in g_i ) ); }
}
Select () { // select members to add to new generation
for (i = 1;  i ≤ p;  i++ )
   if (1  ≤ Rank(g_i ) ≤ (1 − c )* p ) then add g_i to PS ;
}
Crossover () { // produce two offspring
for (i = 1;  i ≤ c*p/100;  i++ ) {
    select (g_m, g_n)  pair from PS at random;
    find crossover point c_p (0 < c_p < n) at random;
    for (k = 1;  k ≤ c_p;  k++ )
        add r_k from g_m to g_n; add r_k from g_n to g_m ;
    for (k = c_p;  k ≤ n;  k++ )
        add r_k from g_m to g_n; add r_k from g_n to g_m ;
    }
}
Mutation () { //handle mutation
for (i = 1;  i ≤ m*p/100;  i++ ) {
    select g_m from PS at random;
    for (j  = 1; j ≤ n; j++ )
        if ( Random( 1, 0 ) == 1) then
        replace in g_m r_j by r_k r (r_j≠ r_k, r_k in RC ); }
}
```

Figure 1. GA pseudo-code implementation.

Here, RC is the candidate solution, t is the Fitness acceptability threshold value, p is the number of hypotheses to be included in population, c is the fraction of the population to be replaced by Crossover at each step or crossover probability, m is the mutation probability, $n = |N|$ is the number of genes in each chromosome, represented as $g_n = \{ r_0, r_1, \ldots r_{n-1} \}$., i.e. the length of a chromosome is n, PS is the offspring made by applying the Crossover operator.

5 Simulation and results

The basic genetic algorithm employed was a steady state GA with a population size of 20. The probability of a single bit being changed by mutation is the reciprocal of the chromosome length, so that one bit is changed on average. The crossover probability is 40% and the mutation probability is 5%. Recombination is achieved through uniform crossover. The offspring created replaces the worst-fit member of the population. All the parameter settings were derived empirically.

5.1 Simulation environment

In the simulations the backbone of NSFnet was chosen as the sample network topology. It consists of 16 nodes representing states in the USA. The network cost of a link joining two states is the driving distance between them. The link capacities were assigned equally. The traffic requests are allowed to use paths with at most 5 hops. The bandwidth demands of the traffic requests are also generated randomly. To establish a baseline, the standard shortest path routing algorithm has been run with the same set of requests as the GA. Table 1 below summarizes the comparison.

Algorithm	Standard deviation of link occupation
SP	42,70
GA	23,89

Table 1. Comparison versus the baseline.

In the GA solutions, all links are at or below 100% utilization, whereas shortest path routing spreads the load far less efficiently. The evolution curve is shown in Figure 2 as the fitness of the best individuals in percentage of the best fitness achieved, in function of the number of generations. As it can be seen, there is little improvement after 25 generations. Since random uniform mutation was used, it was unlikely that the algorithm could get trapped in a local minimum. This behavior results from the fact that both the mathematical model and the proposed GA optimize the use of network resources for traffic exceeding the capacity of single SPF-generated paths, also when multiple objectives are optimized and while some are improved, others are slightly worsened. The proposed algorithm behaves in a satisfactory way, minimizing some variables and sacrificing others but, in all cases, with a performance close to that indicated by the Pareto front-based multi-objective optimization model.

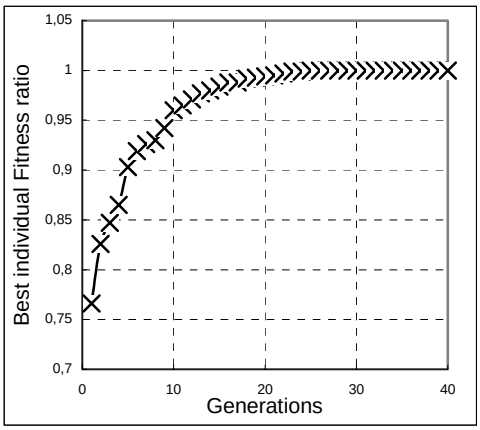

Figure 2. Evolution curve.

6 Conclusions

In this paper we have considered the problem of optimizing LSP layout in an MPLS traffic engineering scenario and proposed a multi-objective approach based on genetic algorithms. The end-to-end delay, cost, link utilization are simultaneously minimized to obtain optimal paths. The metrics have not been combined into one, so that several solutions can be proposed and the one that best matches requirements be chosen. The evaluation result has shown that the approach is a significantly useful and robust method to overcome the disadvantages of the classic heuristic-based solutions.

References

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, Requirements for traffic engineering over MPLS, RFC 2702, Sep. 1999.

[2] E. Rosen, A. Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001

[3] D. O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, Dec. 2001.

[4] E. Mannie et al., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture," IETF Draft, Work in Progress, Mar. 2002.

[5] D.E. Goldberg, "Genetic Algorithms in Search, Optimization & Machine Learning," Addison-Wesley, Massachusetts, 1989