Using Genetic Algorithms to LSP Setup in MPLS Networks

Adriana Oliveira [1] **and Geraldo Robson Mateus**[1]

[1] Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, Minas Gerais - Brazil

{dri,mateus}@dcc.ufmg.br

Abstract. *This work addresses the problem of physical route selection for Label Switched Paths (LSPs) in Multi-Protocol Label Switching (MPLS) networks. The route selection problem consists of defining routes for LSPs trying to minimize the network rejection rate and the total number of hops necessary to route the requests. We propose a mathematical model to represent the problem and we present a genetic algorithm (GA) to solve the model. The experiments show that the GA is able to obtain very good solutions using a very short amount of time.*

1. Introduction

In recent years there has been a tremendous growth of the Internet. Various real-time services are being deployed and new applications such as streaming media and voice over IP present new traffic patterns and new demands to the network. Pressures are being placed on Internet protocols to support quality of service (QoS).

The Internet is currently based on the best-effort paradigm, which, despite being highly scalable, cannot provide the hard guarantees that is desired by most time-critical bandwidth intensive applications. Normally, Internet protocol (IP) traffic follows rules established by routing protocols, such as Open Shortest Path First (OSPF). There is no service differentiation in IP networks. Besides no service differentiation, shortest path destination based routing often leads to unbalanced traffic distribution across the network [Fortz and Thorup 2000].

The ability to control the traffic is precisely what the MPLS [Awduche et al. 1999] technology provides. MPLS is a packet label-based switching technique where packets are assigned a short and fixed-length data header, a label, which identifies the path the packets will follow in the network as well as the treatment the packets will receive in the network. These paths are called Label Switched Paths and are basically explicit routes from a source to a destination. Therefore MPLS plays a key role by providing services unsupported by the IP protocol. It allows sophisticated routing control capabilities to be introduced into IP networks, such as the explicit routing feature mentioned before, and can help build backbone networks that better support QoS traffic[Davie and Rekhter 2000, Black 2001, Awduche 1999].

This work addresses the problem of defining the route configuration for Label Switched Paths (LSPs) in a MPLS capable network. We are particularly interested in using optimization techniques to find the best routes available, since a MPLS network only provides the explicit route feature but the network operator/administrator has to configure these explicit routes in an optimal manner.

We proposed an ILP (Integer Linear Programming) model to solve the offline problem in an exact manner. This model balance the network load while minimizing the network rejection rate and the total number of hops. The goal is to optimize the overall network performance by routing requests through under-utilized links improving the utilization of the installed infrastructure. Other models were already proposed but normally they only take into consideration one network aspect (delay, number of hops) and to our knowledge there is no other model that allow rejections[Dias et al. 2003, Dias and Camponogara 2003, Resende and Ribeiro 2003, Figueiredo et al. 2004]. The main advantage of allowing rejections is that it can be extended in the future to differentiated services and on-line requests.

The model was executed in CPLEX [ILOG 2002], a commercial optimization tool. However, various issues concerning the execution time necessary to solve the model were noticed and to address these issues we developed a genetic algorithm. Our GA is an evolutionary heuristic [Michalewicz and Fogel 2000] based on the combination of routing policies and adaptive route movements, which will be described later.

GA have been used before in the context of OSPF in several works such as in [Buriol 2003, Buriol et al. 2005, Ericsson et al. 2002]. Then the GA is used to set weights to the links and does not differentiate routes when the request has the same origin-destination pair.

An important work using GA in the context of MPLS networks is [Hong et al. 2003]. Their work differs in all important aspects of genetic algorithms: their objective function does not take into account the possibility of rejections, their definition and implementation of the genetic representation is different from ours as well as the definition and implementation of the genetic operators. Our work has a simple and clear definition and implementation of the genetic representation as well of the genetic operators and still presents fast and reliable results.

In order to analyze the genetic algorithm's performance various scenarios were studied and simulations were conducted. Simulation results show that the use of the genetic algorithm considerably decreases the amount of rejected requests in the network in comparison with the basic shortest path approach. The mathematical model and the genetic algorithm solutions were compared and the results were very good, showing that the genetic algorithm can find solutions very close to the optimal solution and it can run much faster than CPLEX.

2. Mathematical Formulation

Our model combines rejection rate and total number of hops in the objective function and the objective function can be parameterized to switch the priority of the minimization process to be the number of hops or the number of rejections.

Let $G = (V, E)$ be a directed graph representing the MPLS network under study. Let E denote the set of links that connect the backbone nodes, and $V = 1, \cdots, n$ denote the set of backbone nodes, where MPLS routers reside. For each link $(i, j) \in E$, let μ_{ij} denote the link bandwidth (the maximum kbits/sec rate) allowed to be routed on the link or the link capacity.

The set K of LSPs to be routed is represented by a list of origin-destination (O-D) pairs

$$K = \{(o_1, d_1), (o_2, d_2), \cdots, (o_n, d_n)\}$$

where we associate with each pair a bandwidth requirement. Each commodity $k \in K$ is a LSP to be routed, associated with an origin-destination pair and with a bandwidth requirement or demand (d_i^k).

A route for LSP (o, d) is a sequence of adjacent links, where the first link originates in node o and the last link terminates in node d. A set of routing assignments is feasible, if for all links $(i, j) \in E$, the total LSP effective bandwidth requirements routed on (i, j) does not exceed $\alpha\ \mu_{ij}$, where α represents the maximum utilization allowed in the links.

Network load balancing is achieved by minimizing the load on the most utilized link. Routing assignments with minimum LSP delays may not achieve the best link load balance. Likewise, routing assignments having the best link load balance may not minimize LSP delays. A compromising objective is to route all LSPs in set K such that a desired point in the trade-off curve between LSP delays and link load balancing is achieved. Our problem is modeled using a multi-objective approach consisting of two steps. In the first step, we minimize the maximum link utilization and the problem is stated as shown in Model P1.

$$P1 : Min\alpha \tag{1}$$

Subject to:

$$\sum_{k=1}^{K} x_{ij}^k \leq \alpha\mu_{ij} \qquad \forall (i, j) \in E \tag{2}$$

$$\sum_{(i,j)\in E} x_{ij}^k - \sum_{(j,i)\in E} x_{ji}^k = d_i^k \qquad \forall i \in V, \forall k \in K \tag{3}$$

$$x_{ij}^k \in 0, 1 \qquad \forall (i, j) \in E, \forall k \in K \tag{4}$$

The first group of constraints (2) imposes limits on traffic over the links, while the second group of constraints (3) guarantees flow conservation. The demand d_i^k takes the value 1 if the node i is a LSP ingress node, it takes the value -1 if the node i is a LSP egress node and it takes the value 0 otherwise. The variable x_{ij}^k takes the value 1 if, and only if, the virtual path of the $k-th$ LSP goes through the link (i,j). The ultimate objective is to minimize the load on the most utilized link.

Let α^* be the optimal value of α obtained in the first optimization step. The second optimization step is to minimize the cost subject to the constraint that all link utilization remains under α^*.

The ultimate objective is to minimize the total resource usage. If α^* is less than 1, its value will be used and no LSP rejections have to happen. If α^* is greater than 1, we will set its value to 1 and allow rejections to happen.

$$P2: Min \sum_{(i,j)\in E} \sum_{k=1}^{K} x_{ij}^k + \sum_{k=1}^{K} M(1-a^k) \tag{5}$$

Subject to:

$$\sum_{k=1}^{K} x_{ij}^k \leq \alpha^* \mu_{ij} \qquad \forall (i,j) \in E \tag{6}$$

$$\sum_{(i,j)\in E} x_{ij}^k - \sum_{(j,i)\in E} x_{ji}^k = d_i^k a^k \qquad \forall i \in V, \forall k \in K \tag{7}$$

$$\sum_{k=1}^{K} a^k >= C \tag{8}$$

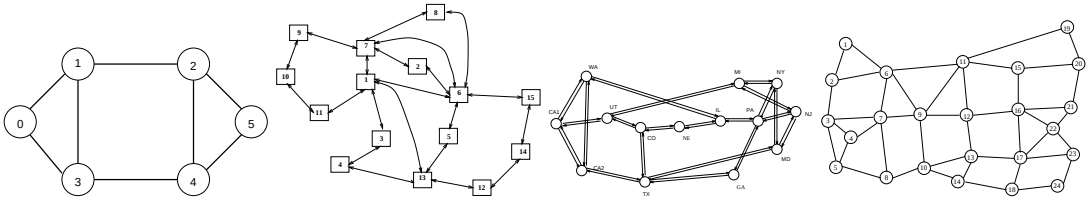
$$x_{ij}^k \in 0,1 \qquad \forall (i,j) \in E, \forall k \in K \tag{9}$$

$$a^k \in 0,1 \qquad \forall k \in K \tag{10}$$

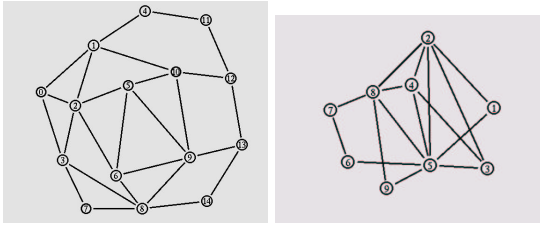
The mathematical formulation for the second step is shown in Model P2. The constraints in the second optimization step are the same as those in the first step, except for constraint (7) and (8). The variable a^k takes the value 1 if, and only if, the k^{th} LSP is accepted into the network, allowing the network to reject requests. The parameter C indicates the minimum number of LSPs that need to be provided and it is used in the constraint (8). Ideally, C should be equal to the number of LSP requests. The parameter M indicates the penalty given to rejections. In our experiments, M is set to 10.

3. Model Experiments in CPLEX

We solved the model using CPLEX [ILOG 2002]. The network topologies used in this work are shown in Figure 1.



(a) Mesh (b) Ring (c) NSF (d) Carrier



(e) Dora (f) Sul

Figure 1. Topologies Used in the Experiments

The simple 6-node network has mesh characteristics. It is used for illustrative purposes and to get the numerical examples for large complex problems. The second topology is a telecommunication metropolitan network, composed of four rings. This is a typical topology showing how today's backbone networks are interconnected. The third topology is widely used as an illustrative wide-area backbone network topology. The fourth topology is a modified version of a well connected carrier's IP backbone topology, widely used for simulation experiments. The fifth and sixth topologies were used in similar papers [Boutaba et al. 2002, Dias et al. 2003].

The sets of requests used in the experiments range from light to heavy traffic scenarios. For the simulation study, source-destination pairs were chosen by chance to represent the LSPs. Each simulation consists of a set of CBR type data flows, each with transmission rates set at 200 Kbps. The number of data flows varied from 10 up to 50. The purpose of this workload is to contrast the performance of the proposed model with that of conventional routing. Based on the LSPs definition we generated various traffic scenarios by configuring the number of flows to vary from 10 to 50 with a step of 10 flows as described in [Boutaba et al. 2002, Kar et al. 2000, Dias and Camponogara 2003].

To find the optimal solution for the test scenarios, the two models defined previously were executed on CPLEX. The Model P1 computed the minimum maximum demand rate in one link of the network for all traffic scenarios and topologies. The results are described in Table 1. Each element of the Table 1 reports the average minimum load on the most utilized link for each number of flows over 5 run trials and its standard devi-

ation (within brackets). Since the network capacity was not taken as a constraint, some demand rates are greater than 1. It means that for the traffic generated some rejections will occur.

Table 1. Minimum MaxLoad on Network Links (1.0 represents 100%)

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.72 (0.10)	1.32 (0.16)	1.96 (0.20)	2.64 (0.15)	3.04 (0.15)
Ring	0.56 (0.15)	0.84 (0.08)	1.40 (0.13)	1.64 (0.08)	2.08 (0.16)
NSF	0.48 (0.10)	0.76 (0.08)	1.04 (0.08)	1.40 (0.13)	1.84 (0.20)
Carrier	0.36 (0.08)	0.60 (0.00)	0.80 (0.13)	1.0 (0.00)	1.3 (0.10)
Dora	0.33 (0.00)	0.50 (0.00)	0.67 (0.00)	0.86 (0.07)	1.17 (0.00)
Sul	0.88 (0.16)	1.44 (0.20)	1.92 (0.39)	2.98 (0.39)	3.36 (0.54)

We noticed from Table 2 that Model P1 can take more that 30 minutes to execute in cases were the network is heavily loaded and presents various options for routes. The combinatorial nature of the problem makes the execution time very long even for small networks. These experiments showed the performance problem that we are facing; since this problem is NP-hard. The variation in the execution time can be explained by the different traffic patterns and topologies.

Table 2. Model P1 Execution Time in Seconds on CPLEX (Max. 30min)

Top.	Number of Flows				
	10	20	30	40	50
Mesh	0.04 (0.07)	0.02 (0.00)	0.02 (0.01)	363.81 (813.44)	0.05 (0.00)
Ring	0.02 (0.01)	362.19 (809.79)	723.78 (990.96)	1088.41 (888.61)	361.71 (723.13)
NSF	0.03 (0.01)	375.43 (801.44)	722.83 (989.65)	1102.77 (964.09)	361.34 (807.50)
Carrier	653.36 (746.87)	362.25 (807.24)	362.14 (809.37)	1809.59 (2.52)	1813.18 (1.38)
Dora	9.76 (4.99)	743.44 (973.78)	1084.52 (989.93)	361.92 (809.02)	1810.43 (0.44)
Sul	0.02 (0.00)	0.03 (0.01)	0.04 (0.01)	0.05 (0.01)	0.07 (0.02)

Finally, the Model P2 was executed. It uses the minimum max load (α^*) computed in Model P1 as the maximum utilization allowed for each link. In the cases were the demand is greater than 1, the utilization was set to 1.0 and the rejections start to happen. The results are presented in the Table 3.

It is important to notice that the load computed by the model does not take into account the overhead due to the protocol messages. Therefore, it is important to simulate the solutions in an operational environment as will be shown in the next section.

Table 3. Model P2 Result - Minimum Rejection and Minimum Number of Hops

Topology	Number of Flows									
	10		20		30		40		50	
	Drops	Cost	Drops	Cost	Drops	Cost	Drops	Cost	Drops	Cost
Mesh	0	20	1	48	11	148	20	240	30	340
Ring	0	35	0	81	2	137	10	231	20	331
NSF	0	30	0	70	0	103	6	184	17	289
Carrier	0	28	0	58	0	87	0	134	6	189
Dora	0	35	0	73	0	109	0	146	1	204
Sul	0	16	3	57	10	131	5	117	14	206

4. NS Simulations

A series of experiments were conducted to demonstrate the following points: (1) the need for optimization techniques to better configure the LSPs in a MPLS network and (2) the need for simulations of the network in operation taking into account protocol messages overhead.

These experiments show how a naive configuration of the LSPs leads to bad network performance when compared to a smarter one, obtained by our model. To simulate the MPLS network in the operational environment, simulations were conducted using the well known Network Simulator [McCanne and Floyd 2003].

4.1. Default MPLS x MPLS with Two-Step Model

Our hypothesis is that the routes found by the optimization method are much better than the routes found using the default approach. Even when the load in the network is under 1.0, there will be congestion and packets dropped due to delays caused by the protocols messages. The number of packets dropped will decrease substantially and the overall throughput will increase if the proposed strategy is implemented, instead of the default approach.

The experiments are conducted for all topologies and traffic scenarios where the network maximum load as computed by the model P1 is less than 1.0 - see Table 1. The metrics used to compare the solutions are: the number of packets dropped. In theory, no packet should be dropped. Each element of the Table 4 reports the percentage of packet drops for each number of flows in NS using the default routing scheme over 5 run trials and its standard deviation(within brackets).

Tables 5 and 4 shows that our approach outperforms conventional routing. Under normal operation conditions the packet discard rate is almost null for our solution, while sometimes over 10 % for conventional routing. Under heavily loaded traffic conditions, number of flows greater than 30, the packet discard rate is over 30% for conventional routing whereas below 3% if our solution is implemented.

Table 4. Percentage of Packets Dropped when using MPLS Default Scheme

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.01 (0.02)	–	–	–	–
Ring	0.00 (0.00)	0.13 (0.04)	–	–	–
NSF	0.03 (0.05)	0.22 (0.04)	0.36 (0.00)	–	–
Carrier	0.00 (0.00)	0.03 (0.04)	0.08 (0.04)	0.16 (0.10)	–
Dora	0.00 (0.00)	0.04 (0.01)	0.21 (0.04)	0.33 (0.04)	–
Sul	0.01 (0.01)	–	–	–	–

Table 5. Percentage of Packets Dropped when using Routes Computed by Two-Step Model

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.0 (0.00)	–	–	–	–
Ring	0.0 (0.00)	0.03 (0.04)	–	–	–
NSF	0.0 (0.00)	0.0 (0.00)	0.05 (0.02)	–	–
Carrier	0.0 (0.00)	0.0 (0.00)	0.02 (0.03)	0.05 (0.01)	–
Dora	0.0 (0.00)	0.0 (0.00)	0.01 (0.01)	0.05 (0.04)	–
Sul	0.0 (0.00)	–	–	–	–

5. Genetic Algorithm

Genetic algorithms (GA) are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination (or crossover) [Goldberg 1989],[Reeves 1993]. The pseudo code is shown in algorithm 1.

Initially several chromosomes are generated to form an initial pool of possible solutions, the first generation pool. During each successive generation, each organism is evaluated, and a value of fitness is returned. The pool is sorted, with those having better fitness ranked at the top. The next step is to generate a second generation pool of organisms, which is done using any or all of the genetic operators: selection, crossover and mutation. This process is repeated until an organism is produced which gives a solution that is *good enough*.

The termination condition used in this paper was (1) the program reached a fixed number of generations or (2) the highest ranking individual's fitness has reached a plateau such that successive iterations are not producing better results anymore.

The three most important aspects of using genetic algorithms are: (1) definition of the objective function, (2) definition and implementation of the genetic representation,

Algorithm 1 Genetic Algorithm Pseudo Code

```
Choose initial population of size M
repeat
    Evaluate the fitnesses of the population
    Sort the population based on fitness
    Select (C) pairs to be parent's candidates
    Mate pairs at random
    Apply crossover operator
    Keep only one child
    Select (M-C) best-ranking individuals
    to form the elite
until terminating condition
```

and (3) definition and implementation of the genetic operators. Once these three have been defined, the generic genetic algorithm should work fairly well.

In the following subsections, we will describe in detail our approach.

5.1. Genetic Representation

In our approach, a gene represents a possible path for a LSP. Each chromosome has the number of genes equal to the number of LSPs in the problem. The locus of a gene is the LSP identification $k = 1, \cdots, K$. Thus, the value of the $i * th$ gene of the chromosome represents a possible path for the $i * th$ LSP. In our approach, the chromosome represents a solution for the problem, i.e, its set of genes is the set of paths that can be used by all LSPs. Let's use as an example the network shown in Figure 2 and the LSP's requests shown in Table 6.

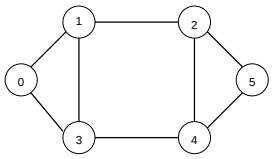


Figure 2. Example - Mesh Topology

Table 6. LSP Requests

Topology	LSPs
Mesh	(0,4) (3,2) (5,3)

The alleles represent the possible routes for each LSP. In our solution, we limited the number of possible routes to a parameter K. So, if K is 2, we can have the following genes and chromosomes.

Table 7. Possible Gene Values

LSP	Allele 1	Allele 2
1	0-3-4	0-1-2-4
2	3-1-2	3-4-2
3	5-4-3	5-2-4-3

Table 8. Possible Chromosome Values

Chromosome	Gene 1	Gene 2	Gene 3
1	0-3-4	3-1-2	5-4-3
2	0-1-2-4	3-4-2	5-4-3
3	0-3-4	3-4-2	5-2-4-3

The K-shortest paths for each LSP are computed in a pre-processing phase, using Djikstra's algorithm.

5.2. Population Initialization

Our first attempt was to define the first generation pool using a random function. Therefore, in a pre-processing phase of the algorithm, the first K shortest paths available for each LSP were calculated, using Djikstra's algorithm. Then, for each LSP, one of the K LSPs routes was chosen by chance, defining the genes for the chromosomes.

Besides the number of alternative routes, K, the population size or number of chromosomes is also a parameter in our implementation.

After some experiments, we noticed that the initial population could be improved leading to a faster convergence of the algorithm to a better solution. Hence, three policies were defined to address the problem of initial population definition. Instead of picking a route by chance, we started using different criteria to choose the routes to compose the chromosomes. The polices implemented are: min-hop (MH), limited utilization (LU) and load balance (B) and will be described in detail.

5.2.1. MH - Min Hop

This algorithm tries to grant to each LSP request the shortest route possible for that LSP. The final solution would have the minimum number of hops possible. It works as following: for each LSP request, it is verified if the shortest route residual bandwidth is sufficient to satisfy the LSP demand. If yes, the LSP is allocated to this route, otherwise the next route is checked and this process is repeated till all possible routes are verified. In other words, in the min-hop algorithm, the path from the ingress to the egress node with the least number of *feasible* links is chosen. This algorithm is the most commonly used algorithm for routing LSPs.

5.2.2. LU - Limited Utilization

This algorithm tries to balance the network load imposing a limit on link utilization to avoid congestion. It is an extension of the MH algorithm but besides being feasible the link utilization must be below a certain limit. It works as following: for each LSP request, it is verified if the shortest route residual bandwidth is sufficient to satisfy the LSP demand and if it is below a certain limit. If yes, the LSP is allocated to this route, otherwise the next route is checked and this process is repeated till all possible routes are verified.

5.2.3. B - Load Balance

This algorithm tries to balance the load among the routes. It looks for the route with the best residual/capacity ratio to allocate the LSP. It is also an extension of the MH algorithm. We define the residual bandwidth along a link to be the difference between the bandwidth of the link and the sum of the LSP demands that are already routed on that link. It works as following: for each LSP request, it is verified which route has the maximum value for the residual/capacity ratio and has sufficient bandwidth to satisfy the LSP demand. This route is allocated to this LSP.

5.2.4. Adaptive Movements - Reducing Rejection

The policies described previously routed the LSP requests based on their arrival order. After running some experiments, we noticed that changing the routes of previously accepted requests could reduce the number of LSPs rejection. Hence an adaptive scheme was proposed to solve the problem of accommodating requests that would normally be rejected.

The adaptive movements idea came from [Salvadori and Battiti 2003]. But in their work they *reroute* or *move* some LSPs to obtain the best solution. In our work [Oliveira et al. 2005, Oliveira and Mateus 2005], we will *reroute* or *move* a LSP only to allow another LSP to be accepted in the network. We will not test all possibilities, looking for the best solution. This small modification will improve the performance of our algorithm. In summary, the adaptive movements or adaptive routing scheme (ARS) is a rerouting technique that will be combined with the policies previously described to decrease the network rejection rate.

The algorithm presented in algorithm 2 is the pseudo-code for the adaptive routing scheme. This algorithm works as following: for each LSP request, a route is calculated, based on the chosen policy: MH, LU or B. If the policy can not find a route or if the route found is too long, the ARS tries to change previously accepted LSPs routes to accept the new LSP or to install it in a better way. A route is considered too long if it is at a distance D or greater than D of the shortest path.

The adaptive scheme then searches for the congested link in the first shortest path route. For this link, it checks the LSPs that were already allocated to that link - line 5. If there is a LSP that uses more bandwidth than that requested by the new request and can be moved to another route with a length increase of maximum D hops, lines 6-8, the LSP routes are changed, otherwise nothing is done. In [Salvadori and Battiti 2003], this search will be exhaustive, compromising performance at some extent.

Algorithm 2 Adaptive Movements Pseudo Code

```
 1: route = alocateOneRoute(new LSP, demand);
 2: if route does not exist or route > minRoute+D then
 3:     find the congested link in one of the available routes;
 4:     repeat
 5:         for all  each LSP that crosses this link do
 6:             if (LSP.demand > newLSP.demand) then
 7:                 find a newroute such as newroute ≤ LSP.route +D;
 8:             end if
 9:         end for
10:     until found or LSPs are finished;
11:     if found() then
12:         changeRoutes;
13:     end if
14: end if
```

The policies previously defined are combined with the adaptive movements generated three new possible ways to establish the first generation pool: adaptive min-hop (AMH), adaptive limited utilization (ALU) and adaptive load balance (AB).

5.3. Selection Methods

The selection method determines how individuals are chosen for mating. In our implementation, individuals are picked using random selection as parents candidates. Then, for each mating iteration, two chromosomes are selected from the parent's candidates group.

5.4. Fitness Function

In genetic algorithms, fitness is used to guide the search by deciding which chromosomes will be used as future points to look for better solutions. The fitness function represent the condition of being suitable. In our work, the fitness value of a chromosome will be the number of requests that are rejected due to lack of bandwidth in a path and the total number of links (hops) used in the solution, as shown in equation 11. M is a parameter that can be used by the network operator to give more priority to solutions with a lower number of hops or to solutions with a lower number of rejections.

$$Fitness(x) = total\ number\ of\ hops + M.total\ number\ of\ rejections \qquad (11)$$

The load balanced solutions obtained when using the first part of the ILP model will be obtained by the genetic algorithm through the use of the policies: AMH, B, AB, LU and ALU. In our implementation, the fitness function will be used for the elite definition and algorithm termination.

5.5. Elite Definition

In our approach, some of the better organisms from the first generation are carried over to the second generation unaltered. This form of genetic algorithm is known as an elite selection strategy.

5.6. Heuristic Crossover

In our approach, the crossover occurs in the following way. For each mating pair, the parent's genes are compared and the best gene is copied to the offspring. The metric used to compare the genes is their number of hops. Therefore, the gene that has the smaller number of hops is copied from the parent to the offspring.

6. GA Experiments

The experiments will show the performance yielded by the heuristic against that obtained solving the integer linear formulation using CPLEX. Our hypothesis is that the genetic algorithm will be able to find a good solution much faster than using CPLEX.

Since the GA can obtain different results depending on the parameters configuration, we first determined the best GA solution and then compared this solution to the solution obtained by CPLEX.

In our implementation, the genetic algorithm receives four parameters as the input:

- number of possible routes (K),
- population initialization strategy,
- population size,
- number of iterations.

The number of possible routes in the experiments varied from 2 to 5. The population initialization strategy worked as follows: if the initialization strategy is 0, all the chromosomes in the population are chosen randomly. If the initialization strategy is set to 1, the first 6 chromosomes were initialized using the policies and the rest was chosen randomly.

The population size varied from 5 to 25 with a step of 5 as well as the number of iterations. The solutions found by the best execution of the genetic algorithm are shown in Table 9 and in Figure 3. The graph shows CPLEX results for the different models described before:

- MinRej - model that takes into account the number of rejections but does not try to balance the load. It is a lower bound for our two-step model.

- MinRejBal - model that takes into account the number of rejections and tries to balance the load. Gives the results for our two-step model.
- Genetico - heuristic that takes into account the number of rejections and tries to balance the load using policies.

Table 9. Number of Rejections and Number of Hops Using the Genetic Algorithm

Topology	Number of Flows									
	10		20		30		40		50	
	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops
Mesh	0	20	1	48	11	148	20	240	30	340
Ring	0	29	0	68	4	139	15	248	24	340
NSF	0	30	0	65	2	118	11	211	20	307
Carrier	0	25	0	54	0	77	1	144	7	212
Dora	0	33	0	71	3	118	8	193	22	314
Sul	0	14	1	43	8	120	8	136	14	210

We notice that the genetic algorithm is able to find the best solution for the vast majority of cases and is very close to the optimal solution in the other cases. The number of iterations necessary to reach the best solution of the GA is generally less than 5. The execution time is on the magnitude of milliseconds for all instances executed.

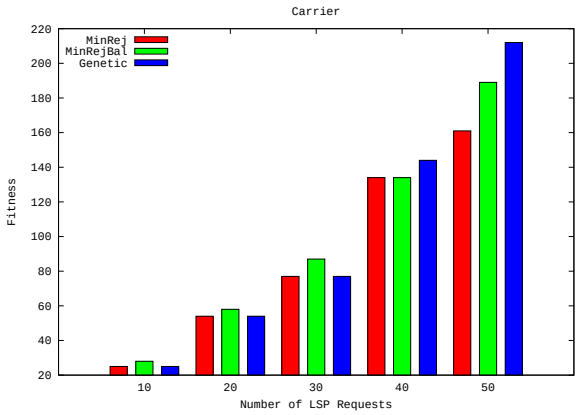


Figure 3. CPLEX and GA Results Comparison for the Carrier Network

For the scenarios with light traffic, the GA results are similar to the lower bound, where there is no need for load balancing. When the traffic load starts to increase, the GA solution is between the ones found when the traffic is balanced and the solutions found when it is not balanced.

When the network is overloaded, the GA results found are on average 20% worse than the best solution, but this is not very problematic, since our goal is to obtain good solutions when the load is within the network capacity and the network is still congested.

7. Concluding Remarks

This work demonstrates the effectiveness of the application of optimization techniques to traffic management in computer networks. It validated the use of optimization techniques for improving performance of IP networks over MPLS and the use of heuristics to obtain the solutions in a time period practical in real life. We presented a formal model that works offline and considers number of hops and rejection rates.

We presented a genetic algorithm for adaptive routing, whose main goal was minimizing the network rejection rate. It was possible, in the population initialization phase, to reroute a LSP to avoid congested links even choosing longer paths, in order to balance the overall link loads and to allow a better use of the networks resources.

The results showed an improvement on the solutions when compared to traditional shortest-path approaches and the heuristic manages to solve the mathematical model much faster than the optimization tool CPLEX. The execution of the exact models takes longer and it is very complicated since it has to occur in two phases. The packet discard rate is considerably reduced by the use of the optimization techniques.

Another advantage of the GA is the fact that it is very easy to change the objective function to change the priorities (rejection rate or minimum delay) according to administrative interests.

References

Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M., and McManus, J. (1999). RFC 2702: Requirements for Traffic Engineering Over MPLS.

Awduche, D. O. (1999). MPLS and Traffic Engineering in IP Networks. *IEEE Communications Magazine*, 37(12):42–47.

Black, U. (2001). *MPLS and Label Switching Networks*. Prentice Hall.

Boutaba, R., Szeto, W., and Iraqi, Y. (2002). DORA: Efficient Routing for MPLS Traffic Engineering. *Journal of Network and Systems Management*, 10(3):309–325.

Buriol, L. S. (2003). *Roteamento do Tráfego na Internet: Algoritmos para Projeto e Operação de Redes com Protocolo OSPF*. PhD thesis, Unicamp.

Buriol, L. S., Resende, M. G. C., Ribeiro, C. C., and Thorup, M. (2005). A hybrid genetic algorithm for the weight setting problem in ospf/is-is routing. *Networks*, 46(1):36–56.

Davie, B. and Rekhter, Y. (2000). *MPLS: Technology and Applications*. Morgan Kaufmann.

Dias, R. and Camponogara, e. a. (2003). Otimização Lagrangeana Em Engenharia de Tráfego para Redes IP sobre MPLS. *XXI Simpósio Brasileiro de Redes de Computadores*, pages 475–490.

Dias, R. A., Camponogara, E., Farines, J.-M., Willrich, R., and Campestrini, A. (2003). Implementing Traffic Engineering in MPLS-Based IP Networks with Lagrangean Relaxation. *IEEE ISCC*, pages 373–378.

Ericsson, M., Resende, M. G. C., and Pardalos, P. M. (2002). A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *J. Comb. Optim.*, 6(3):299–333.

Figueiredo, G. B., da Fonseca, N. L. S., and Monteiro, J. A. S. (2004). A Minimum Interference Routing Algorithm. *IEEE Communications Society*, 4:1942–1947.

Fortz, B. and Thorup, M. (2000). Internet Traffic Engineering by Optimizing OSPF Weights. *IEEE INFOCOM*, pages 519–528.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Hong, L., Dong, B., and Wei, D. (2003). An Explicit Routing Optimization Algorithm for Internet Traffic Engineering. *Proceedings of the International Conference on Communication Technology - ICCT 2003*, 1:445–449.

ILOG (2002). Ilog ampl cplex system 8.0 - user's guide.

Kar, K., Kodialam, M., and Lakshman, T. V. (2000). Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. *IEEE Journal on Selected Areas in Communications*, 18(12):921–940.

McCanne, S. and Floyd, S. (2003). Network simulator 2.

Michalewicz, Z. and Fogel, D. B. (2000). *How to Solve It: Modern Heuristics*. Springer-Verlag.

Oliveira, A. and Mateus, G. (2005). Traffic Engineering: Control Charts and LSPs. *12th International Conference on Telecommunications - ICT 2005*.

Oliveira, A., Ravetti, M., and Mateus, G. (2005). Traffic Engineering in MPLS Networks: ARS An Adaptive Routing Scheme Based on Control Charts. *XXIII Simpósio Brasileiro de Redes de Computadores*.

Reeves, C. R. (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Press.

Resende, M. G. C. and Ribeiro, C. C. (2003). A GRASP with Path-Relinking for Private Virtual Circuit Routing. *Networks*, 41(3):104–114.

Salvadori, E. and Battiti, R. (2003). A Load Balancing Scheme for Congestion Control in MPLS Networks. *IEEE ISCC*, pages 951–956.