Chapter 3

Solution Method: Genetic Algorithm Representation

This chapter presents the solution method of the genetic algorithm representation model that is proposed to solve the shortest path problem in OSPF hop-by-hop routing and MPLS explicit-routing.

3.1 Formulation of the Shortest Path Problem in General Routing Problem

The general routing problem is defined as follows. Our network is a directed graph, or multi-graph, \( G = (N, A) \) whose nodes and arcs represent routers and the links between them. Each arc \( a \) has a capacity \( c(a) \) which is a measure for the amount of traffic flow it can take. In addition to the capacitated network, we are given a demand matrix \( D \) that for each pair \((s, t)\) of nodes tells us how much traffic flow we will need to send from \( s \) to \( t \). We shall refer to \( s \) and \( t \) as the source and the destination of the demand. Many of the entries of \( D \) may be zero, and in particular, \( D(s,t) \) should be zero if there is no path from \( s \) to \( t \) in \( G \). The routing problem is now, for each non-zero demand \( D(s,t) \), to distribute the demanded flow over paths from \( s \) to \( t \). Here, in the general routing problem, we assume there are no limitations to how we can distribute the demanded flow over paths from \( s \) to \( t \). Here, in the general routing problem, we assume there are no limitations to how we can distribute the flow between the paths from \( s \) to \( t \). The above definition of the general routing problem is equivalent to the one used, e.g. in Awduche et al. [1].
3.2 Genetic Algorithm to Solve the Shortest Path Problem in OSPF hop-by-hop routing

In this section, we discuss the implementation of genetic algorithm in solving the shortest path problem in OSPF. Dijkstra algorithm is used to find the shortest path problem. In the first solution model, we develop a solution model to solve the shortest path problem in OSPF by using genetic algorithm instead of the traditional Dijkstra algorithm.

3.2.1 Formulation of the Shortest Path Problem in OSPF hop-by-hop routing

In OSPF, the network operator assigns a weight to each link, and shortest paths from each router to each destination are computed using these weights as lengths of the links. In each router, the next link on all shortest paths to all possible destinations is stored in a table, and a demand going in the router is sent to its destination by splitting the flow between the links that are on the shortest paths to the destination. The exact mechanic of the splitting can be somewhat complicated, depending on the implementation. Here, as a simplifying approximation, we assume that it is an even split. The quality of OSPF routing depends highly on the choice of weight.

In OSPF, the Internet is represented using weighted directed graph according to RFC 1583 [25]. A weighted directed graph $G = (V, E)$ comprises a set of nodes $V = \{v_i\}$ and a set of edges $E \in V \times V$ connecting nodes in $V$. Corresponding to each edge, there is a
non-negative number $w_{ij}$ representing the cost (distance, transit times, or others of interesting) from node $v_i$ to node $v_j$. A path from node $v_i$ to node $v_j$ is a sequence of edges $(v_i, v_l, v_m, \ldots, v_k, v_j)$ from $E$ in which no node appears more than once. A path can also be equivalently represented as a sequence of nodes $(v_i, v_l, v_m, \ldots, v_k, v_j)$. The problem is to find a path between two given nodes having minimum total cost (total weight), $\Phi$. The integer programming model is formulated as follows:

$$\Phi = \min \sum_{i} \sum_{j} w_{ij} x_{ij}$$

Let $x_{ij}$ be an indicator variable defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if edge } (i, j) \text{ is included in the path} \\ 0, & \text{otherwise} \end{cases}$$

In order to achieve Internet Traffic Engineering, we could optimize the OSPF weights as proposed in [5]. The objective function is to minimize the total cost (weight), $\Phi$ of the network, which is the summation of 2 parameters, i.e. distance, $d$ and utilization’s weight, $\Phi(u)$. The integer-programming model is formulated as follows:

$$\Phi = \min \sum_{i} \sum_{j} (d_{ij} x_{ij} + \Phi(u) y_{ij} )$$

Having decided on a routing, the load $l(a)$ on an arc $a$ is the total flow over $a$, that is $l(a)$ is the sum over all demands of the amount of flow for that demand which is sent over $a$. The utilization of a link $a$, $u(a)$ is defined as follow:

$$u(a) = l(a) / c(a)$$

51
Our objective is to keep the loads within the capacities. More precisely, our cost function sums the cost of between \( l(a) \) and \( c(a) \). In our experimental study, we had,

\[
\Phi(u) = \sum_{a \in A} \Phi_a(\ell(a))
\]

where for all \( a \in A, \Phi_a(0) = 0 \) and

\[
\Phi_a(\ell(a)) = \begin{cases} 
1 & \text{for } 0 \leq \ell(a)/c(a) < 1/3 \\
3 & \text{for } 1/3 \leq \ell(a)/c(a) < 2/3 \\
10 & \text{for } 2/3 \leq \ell(a)/c(a) < 9/10 \\
70 & \text{for } 9/10 \leq \ell(a)/c(a) < 1 \\
500 & \text{for } 1 \leq \ell(a)/c(a) < 11/10 \\
5000 & \text{for } 11/10 \leq \ell(a)/c(a) < \infty 
\end{cases}
\]

The graph, which reflects the above cost function, is illustrated in figure 3.1.
The above definition of the general routing problem and the objective function is equivalent to the one used in Bernard et. al. [5]. The idea behind $\Phi_a$ is that it is cheap to send flow over an arc with a small utilization. As the utilization approaches 100%, it becomes more expensive, for example because the algorithm becomes more sensitive to bursts. If the utilization goes above 110%, the penalty becomes so high that this should never happen.

Knowing the optimal solution for the general routing problem is an important benchmark for judging the quality of solutions based on, say, OSPF and MPLS routing. The above objective functions provide a general best effort measure. The approach here can also be used if the demand matrix is modeling service level agreements.
(SLAs) with customers, with demands representing guarantees of throughput for virtual leased lines. In this case, we are more interested in ensuring that no packet is sent across overloaded arcs. Our objective is to minimize a maximum rather than a sum over the arcs. However, due to the very high penalty for overloaded arcs, our objective function favors solutions without overloaded arcs. A general advantage to working with a sum rather than a maximum is that even if there is a bottleneck link that is forced to be heavily loaded, our objective function still cares about minimizing the leads in the rest of the network.

3.2.2 Solution Method I: Previous-node-based Encoding

How to encode a path in a graph is critical for developing a genetic algorithm to this problem. It is not as easy as the Traveling Salesmen Problem (TSP) to find out a nature representation. Special difficulties arise from: (a) a path contains a variable number of nodes and the maximal number is n-1 for a n node graph, and (b) a random sequence of edges usually does not correspond to a path. To overcome such difficulties, we adopted an indirect approach: encode some guiding information for constructing a path, but a path itself, in a chromosome. The path is generated by a sequential code appending procedure, beginning at a specified node 1 and terminating at a specified node n. As we know, a gene in a chromosome is characterized by two factors:

(a) allele, the value the gene takes, and

(b) locus, the position of the gene located within the structure of chromosome.
In the proposed previous-node-based encoding method, the position of a gene is used to represent node ID and the value is used to represent the previous node of the current node for construction a path among candidates. A shortest path tree from the start node to all the other nodes can be uniquely determined from this encoding. Let us see an example of a network, which consists of 6 nodes and 13 links as shown in Figure 3.2.

![Network diagram with nodes and links](image)

*Figure 3.2 Example of mesh topology network, which consists of 6 nodes and 13 links*

Suppose we are going to find a path from node 1, the start node to all the other nodes. From Dijkstra algorithm, the shortest path tree from the start node (N0) to all the other nodes is shown in figure 3.3. Using genetic algorithm, the solution obtained is the same as the solution obtained from Dijkstra algorithm.
To encode the solution into a chromosome, the previous-node-based encoding method is used as shown in figure 3.4, where the locus of a gene is used to represent node ID and the allele is used to represent the previous node of the current node.

\[
\begin{array}{cccccc}
N0 & N1 & N2 & N3 & N4 & N \\
0 & 4 & 4 & 0 & 0 & 4 \\
\end{array}
\]

Locus: Node ID

Allele: Previous node ID

\[ \text{Figure 3.4 Previous-node-based encoding in a chromosome} \]

### 3.2.2.1 Search Method

In this problem, the steady-state genetic algorithm is used to find the optimal solution. This algorithm uses overlapping populations; only a portion of the population is replaced in each generation. The amount of overlap (percentage of population that is replaced) may be specified when tuning the genetic algorithm.
3.2.2.2 Defining a Representation

Genetic algorithm can operate on any data type. Any representation must have appropriate genetic operators defined for it. The representation determines the bounds of the search space, but the operators determine how the space can be traversed.

For any genetic algorithm, the representation should be a minimal, complete expression of a solution to the problem. A minimal representation contains only the information needed to represent a solution to the problem. A complete representation contains enough information to represent any solution to the problem. If a representation contains more information than is needed to uniquely identify solutions to the problem, the search space will be larger than necessary.

Whenever possible, the representation should not be able to represent infeasible solutions. If a genome can represent an infeasible solution, care must be taken in the objective function to give partial credit to the genome for its good genetic material while sufficiently penalizing it for being infeasible. In general, it is much more desirable to design a representation that can only represent feasible solutions so that the objective function measures only optimality, not feasibility. A representation that includes infeasible solutions increase the size of the search space, and thus, makes the search more difficult.
The shortest path problem depends on a sequence of items. As a result, an ordered-based representation in array data type is chosen to represent the shortest path, where a genome consists of an array of genes. The previous-node-based encoding representation for the shortest path problem in OSPF is a minimal representation that can represent valid shortest path solutions.

3.2.2.3 Genetic Operators

The use of a genetic algorithm requires the definition of initialization, crossover, and mutation operators specific to the data type in the genome. Here, the roulette wheel approach is adopted as the selection procedure, which is one of the fitness-proportional selection to produce the next generation.

3.2.2.4 Initialization

The mesh network topology is initialized with a maximum of 10 nodes. The range of possible values was based upon the node table. The magnitude of the link represents the distance (which is direct proportional to weight) and is input by the user. In the standard library in Java programming language, there is a method provided in the Random class to generate a random integer number, where the range can be specified in the method. The program will then check whether the random integer number is one of the previous nodes from the adjacency matrix. If yes, then the integer, which represents the previous node ID will be assigned to the allele value of the gene of the chromosome. If not, then the next random integer number will be generated to check its validity.
3.2.2.5 Crossover

The crossover operator is the array one-point crossover with the condition that the crossover node ID of the 2 parent chromosomes must be the same number. For example, two separate chromosomes representing two different solutions of the shortest path tree (SPF) are shown in figure 3.5.

Solution 1

```
N0  N1  N2  N3  N4  N5  N6
0   0   0   1   1  14  4
```

Parent chromosome 1

Solution 2

```
N0  N1  N2  N3  N4  N5  N6
0   0   0  01  1  14  4
```

Parent chromosome 2

Figure 3.5 Two separate chromosomes representing two different solutions of the SPF
Let us take node 3 (N3) as the crossover point. From chromosome 1, the previous node for N3 is node 1 and the previous node for N3 for chromosome 2 is node 0. The crossover is done by swapping the two alleles (previous nodes) of the two chromosomes. As a result, two new children chromosomes are constructed from the parent chromosomes as shown in figure 3.6.

<table>
<thead>
<tr>
<th>N0</th>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
<th>N5</th>
<th>N6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Child chromosome 1

<table>
<thead>
<tr>
<th>N0</th>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
<th>N5</th>
<th>N6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Child chromosome 2

Figure 3.6 Two new children chromosomes are constructed from the parent chromosomes

The next step is to build a valid path from node 3 (N3) to the origin node (N0). This is done using the guiding table of the topology. Now, the previous node of node 3 (N3) for chromosome 1 is node 0 (N0) and the previous node of N3 for chromosome 2 is node 1 (N1). Now, trace back the path from N3 to the origin node (N0) to ensure the validity of the new path. Change any previous node of the path, which does not conform to a valid path using the guiding table. Here in child chromosome 1, the previous node of node 3 (N3) is node 0 (N0), which is the origin node. In this case, nothing needs to be changed. In child chromosome 2, the previous node of node 3 (N3) is node 1 (N1). So, we need to trace back by selecting N1 and checking its previous node. The previous node for node 1 is the origin node (N0). In this case, nothing needs to be changed. As a result, without changes both the new child chromosomes are valid solutions as shown in figure 3.7. So, we can insert the new chromosomes into the next generation.
3.2.2.6 Mutation

Mutation is performed by the replacement of the mutation operator. A node ID (locus) is randomly selected, then the previous node (allele) is replaced with another random node. For example, a chromosome as in figure 3.8 is selected and node 5 (N5) is selected for mutation. Then, the previous node, which is node 4 is replaced with node 1. The validity of the path is checked with the guiding table before the new chromosome is accepted.
Figure 3.8 The mutation process of a selected chromosome
3.2.2.7 Objective Function

Genetic algorithms are often more attractive than gradient search methods because they do not require complicated differential equations or a smooth search space. The genetic algorithm needs only a single measure of how good a single individual is compared to the other individuals.

One can evaluate the individuals in a population using an individual-based evaluation function, or a population-based evaluator. Here, we use an individual-based objective, then the function is assigned to each genome. The genome performance measure, often referred to as the objective function, is based upon the path the genome represents. The objective function calculation is straightforward. Each link consists of a weight. The fitness value of a genome is calculated by adding up all the weights of the links used in the path. The fitness scoring is the reverse of the fitness value. So, the higher the weight a genome has the lower its fitness scoring.

3.3 Genetic Algorithm to Solve the Shortest Path Problem in MPLS explicit-routing

MPLS is an IETF initiative that integrates Layer 2 information about network links (bandwidth, latency, utilization) into Layer 3 (IP) within a particular autonomous system or ISP in order to simplify and improve IP packet exchange. MPLS gives network operators a great deal of flexibility to divert and route traffic around link failures, congestion, and bottlenecks.
The route for a given Label Switched Path (LSP) can be established in two ways, control-driven (also called hop-by-hop LSP), or explicitly routed (ER-LSP). Here, one problem domain has been defined, i.e. MPLS explicit-routing. In the solution model, we developed a solution called previous-node-based encoding to find the shortest path ER-LSP in MPLS using genetic algorithm.

3.3.1 Formulation of the Shortest Path Problem in MPLS Explicit-routing

MPLS supports traditional hop-by-hop routing and provide explicit-routing capability for traffic engineering. While the hop-by-hop routing follows the path that normal Layer 3 routed packets will take, the explicit-routing can be specified and controlled by the network operators or network management applications to direct the network traffic, independent of the Layer 3 topology. In this way the explicit-routing gives network operators a great deal of flexibility to divert and route traffic around link failures, congestion, and bottlenecks.

A weighted directed graph $G = (V, A)$ comprises a set of nodes $V=\{v_i\}$ and a set of edges $A \in V \times V$ connecting nodes in $V$. Let $x_{ij}$ be an indicator variable defined as follows:

$$
    x_{ij} = \begin{cases} 
    1, & \text{if edge } (i, j) \text{ is included in the path} \\
    0, & \text{otherwise} 
    \end{cases}
$$
Corresponding to each edge, there is a non-negative number $d_{ij}$ (distance) and $c_{ij}$ (capacity) from node $v_i$ to node $v_j$. The objective function is to minimize the total cost (metric) of the network, which is the summation of 2 parameters, i.e. distance ($d$) and utilization's weight ($w(u)$). The integer programming model is formulated as follows:

$$\Phi = \min \sum_i \sum_j \left( d_{ij}x_{ij} + \Phi(u_{ij})x_{ij} \right)$$

Having decided on a routing, the load $l(a)$ on an arc $a$ is the total flow over $a$, that is $l(a)$ is the sum over all demands of the amount of flow for that demand which is sent over $a$. The utilization of a link $a$, $u(a)$ is defined as follow:

$$u(a) = l(a) / c(a).$$

As in the case of OSPF discussed in section 3.2.1, our objective in MPLS is also to keep the loads within the capacities. In order to achieve Internet Traffic Engineering in MPLS, we could use the optimized weight function as proposed in [5]. More precisely, our cost function sums the cost of between $l(a)$ and $c(a)$. In our experimental study, we had,

$$\Phi(u) = \sum_{a \in A} \Phi_a(\ell(a))$$

where for all $a \in A, \Phi_a(0)=0$ and

$$\Phi_a(\ell(a)) = \begin{cases} 
1 & \text{for } 0 \leq \ell(a) / c(a) < 1/3 \\
3 & \text{for } 1/3 \leq \ell(a) / c(a) < 2/3 \\
10 & \text{for } 2/3 \leq \ell(a) / c(a) < 9/10 \\
70 & \text{for } 9/10 \leq \ell(a) / c(a) < 1 \\
500 & \text{for } 1 \leq \ell(a) / c(a) < 11/10 \\
5000 & \text{for } 11/10 \leq \ell(a)/ c(a) < \infty 
\end{cases}$$

65
The graph, which reflects the above cost function, is illustrated in figure 3.9.

*Graph for the Cost versus Utilization function*

*Figure 3.9 The graph which reflects the above cost function $\Phi_0$*

The above definition of the general routing problem and the objective function is equivalent to the one used in Bernard et. al. [5]. The idea behind $\Phi_0$ is that it is cheap to send flow over an arc with a small utilization. As the utilization approaches 100%, it becomes more expensive, for example because the algorithm becomes more sensitive to bursts. If the utilization goes above 110%, the penalty becomes so high that this should never happen.
3.3.2 Solution Method II: Priority-based Encoding

To solve the shortest path problem in MPLS explicit-routing, the priority-based encoding has been proposed. How to encode a path in a graph is critical for developing a genetic algorithm to this problem. As we know, a gene in a chromosome is characterized by two factors, locus, i.e., the position of the gene located within the structure of chromosome, and allele, i.e., the value the gene takes. In the proposed priority-based encoding method, the position of a gene is used to represent node identity (ID) and this value is used to represent the priority of the node for construction a path among candidates. A path can be uniquely determined from this encoding. Priority-based encoding in a chromosome follows the following guidelines:

- The node with the biggest priority value = start node
- The node with the smallest priority value = end node
- Possible next node with bigger priority value will be selected

A simple graph is shown in figure 3.10 and its adjacency metrics are given in figure 3.11.

Figure 3.10 A simple graph

67
Let us see an example of a chromosome generated randomly as shown in figure 3.12.

Based on the guidelines of priority-based encoding, the node with the biggest priority value, which is node 5 (N5) will be the start node and the node with the smallest priority value, which is node 6 (N6) will be the end node. Suppose we are going to find a path from N5 to N6. At the beginning, we try to find a node, which is connected to node 5. Nodes 3, 4 and 6 are possible for consideration. This can be easily fixed due to adjacent relation among nodes. The priorities for them are 2, 5 and 1 respectively. Node 6 is the end node and is not considered. The node 4 has the highest priority and is put into the path. Then we form the set of nodes available for next consideration and select the one with the highest priority among them. Repeat the se steps until we obtain a complete path (5, 4, 2, 1, 3, 6). It is easy to see that any permutation of the encoding can always yield to a path as shown in figure 3.13.
3.3.2.1 Search Method

In this problem, the steady-state genetic algorithm is used to find the optimal solution. This algorithm uses overlapping populations; only a portion of the population is replaced each generation. The amount of overlap (percentage of population that is replaced) may be specified when tuning the genetic algorithm.

3.3.2.2 Defining a Representation

The shortest path problem depends on a sequence of items, in which case an ordered-based representation in array data type is chosen. The previous-node-based encoding representation for shortest path problem in MPLS is a minimal representation that can represent valid shortest path solutions, where a genome consists of an array of genes, which represent previous node for a valid path.
3.3.2.3 Genetic Operators

Use of a genetic algorithm requires the definition of initialization, crossover, and mutation operators specific to the data type in the genome. Here, the roulette wheel approach is adopted as the selection procedure, which is one of the fitness-proportional selections to produce the next generation.

3.3.2.4 Initialization

The node identity (ID) is initialized with random numbers of the type integer ranged from start node to end node. The range of possible values of the priority value (allele) is based upon the node table. The magnitude of the link represent two parameters, i.e. distance and utilization’s weight and can be input by the user.

3.3.2.5 Crossover

The crossover operator is the array two-points crossover with the condition that the crossover node ID of the 2 parent chromosomes must be the same number. For example, two separate chromosomes representing two different solution of the shortest path tree are shown in figure 3.14.
Let us take node 2 (N2) to node 5 (N5) as the two-point crossover. The crossover is done by swapping the two alleles (previous node) of the two chromosomes. As a result, two new children chromosomes are constructed from the parent chromosomes as shown in figure 3.15.

**Figure 3.14 Two parents chromosomes representing two different solutions**

**Figure 3.15 Two new solutions represented by child chromosomes**
3.3.2.6 Mutation

Mutation is performed by swapping mutation operator as shown in figure 3.16. Two nodes (locus) are randomly selected and the priority values (allele) are swap. Thus, a new chromosome is created.

![Chromosome selected for mutation](image)

![New chromosome after mutation](image)

*Figure 3.16 The mutation process of a selected chromosome*

3.3.2.7 Objective Function

In MPLS, the objective function calculation is quite straightforward. Each link consists of a total weight, which is the summation of two parameters, i.e. distance \((d)\) and utilization’s weight \((w(u))\). The fitness value of a genome is calculated by adding up all the weights of the links used in the path. The fitness scoring is the reverse of the fitness value. So, the higher the weight a genome has the lower its fitness scoring.
3.4 Genetic Algorithm Considerations

There are some considerations when implementing a genetic algorithm. The performance and effectiveness of the genetic algorithm depend on several factors, i.e. diversity and convergence. The performance issue is also very closely related to the setting of genetic algorithm parameters, i.e. population size, maximum generation, crossover rate, mutation rate and overlapping rate for the steady state genetic algorithm.

3.4.1 Diversity

The diversity of the population will determine the scope of the choice of the solution encoded in the chromosome. The higher the diversity the larger the search scope and the more choices of solution we can have. However, the higher the diversity, the longer the time it takes to converge and reach the optimal solution. Diversity will be affected by numerous factors associated with the implementation of the genetic algorithm, such as parameter encoding, population size, crossover and mutation rates, and selection technique.

3.4.2 Convergence

The goal of implementing any genetic algorithm is convergence on an optimal solution for a given search space. Convergence will also be affected by numerous factors associated with the implementation of the genetic algorithm, such as parameter
encoding, population size, crossover and mutation rates, and selection technique. Depending on the problem being solved, these factors are usually determined only by experience working with genetic algorithms of all flavors.

In the next section, experiments are carried out to study the effects of genetic algorithm parameters towards its performance in finding the optimal solution. The performance issues to be studied are diversity and convergence.