EVT: Evolutionary Algorithm for VPN Tree Provisioning

Boulbaba Thabti, Habib Youssef

Research Unit PRINCE
ISITCom of Hammam Sousse, Tunisia
University of Sousse
b_thabti@yahoo.fr ; habib.youssef@fsm.rnu.tn

Abstract

A Virtual Private Network (VPN) is an overlay network defined over a public network infrastructure like the Internet. A VPN provides customers with a secure, low cost, and more flexible communication environment than leased lines. The allocation of bandwidth for VPNs to meet the requirements specified by customers is an important traffic engineering research issue. A VPN workload model called *hose-model*, first introduced in [1], was developed to provide customers with a flexible and convenient way of specifying the bandwidth requirements of VPN endpoints. The VPN endpoints are connected according to a tree topology. The general problem of computing a constrained VPN tree with optimum bandwidth allocation is a hard combinatorial optimization problem [2]. In this work, we propose an evolutionary algorithm called *EVT* to optimize the total bandwidth reserved on edges of the VPN tree. *EVT* is based on the Simulated Evolution meta-heuristic originally introduced in [3]. Experimental results with Waxman network graphs show that *EVT* performs significantly better than the *Primal-Dual* algorithm reported in [2] with respect to final solution quality.

Keywords --- Virtual Private Network, Traffic engineering, Hose-Model, Simulated Evolution.

1 Introduction

Virtual private Networks (VPNs) that make use of existing (possibly public) network infrastructure are a way of procuring the equivalent of a private leased-lines network that is relatively more expensive and less flexible. Permanently dedicated resources are not required, and the topology and size of the VPN can be altered as required. These VPNs are most commonly deployed on the Internet, often to provide connectivity between corporate Local Area Networks (LANs), or to enable individual mobile users to access a private LAN. The establishment of the Internet VPNs does not require a large overhead: tunneling is used to provide routing, addressing, bandwidth guarantees, and security measures. Bandwidth guarantees mean that the network provider guarantees to the VPN users that a certain amount of bandwidth will be available to them at any time, unaffected by the traffic sent through the physical network by other users. From the perspective of the network provider, it is an important optimization problem to satisfy the bandwidth guarantees required by a VPN while minimizing the amount of network resources (link bandwidth) that are reserved for it.

The two most common VPN resource provisioning models are the *customer pipe-model* and the *hose-model*. In the *customer pipe-model*, customers must have precise predictions about the complete traffic requirement of each pair of VPN endpoints in advance.
The *hose-model* was proposed as a flexible and user-friendly model for specifying the bandwidth requirement of a VPN. A *hose* specifies bandwidth limits on the ingress and egress traffic at every VPN endpoint. Therefore, customers need not specify point to point loads but rather the ingress bandwidth requirement $b^{-}(v)$ and egress bandwidth requirement $b^{+}(v)$ for each endpoint v of a VPN. The value $b^{-}(v)$ define the maximum rate of traffic that endpoint v will receive from the network at any time, and the value $b^{+}(v)$ is the maximum rate of traffic that endpoint v sends to the network.

In this paper, we present a new algorithm for provisioning VPNs assuming a hose model. The algorithm finds a least cost feasible VPN tree connecting the VPN endpoints. Trees offer several advantages as described in [2] such as sharing of reserved bandwidth, scalability, simplicity of routing, and ease of restoration. A least cost feasible VPN tree is a tree structure connecting the VPN endpoints while satisfying the rate traffic limits specified by the hose and minimizing the total bandwidth reserved on the tree edges.

The remainder of this paper is structured as follows. In Section 2, we review related work. In Section 3, we formally state the constrained VPN tree optimization problem. In Section 4, we describe our *EVT* algorithm. In Section 5, we present experimental results and compare the performance of *EVT* with the VPN provisioning algorithm reported in [3]. We conclude and give directions of future work in Section 6.

2 Previous work

Duffield et al first presented the hose model in [1]. In their paper, they proposed *provider-pipes, hose-specific state and VPN-specific* provisioning algorithms for hose-model VPNs. They also proposed several *hose-model* VPN provisioning algorithms for the non-failure case. In [4], Liang et al propose a hose model VPN provisioning algorithm called MTRA to handle on-line multiple VPN setup requests rapidly and reduce the rejection ratio. Ruëgg et al [5] proposed an algorithm to calculate multi-path topologies and compared the performance of several approximation algorithms. They found that running times of these topology computation algorithms increase very quickly with the number of nodes and can be in the order of minutes for large networks. Balasubramanlan et al [6] compared the bandwidth requirement of single-link failure recovery algorithms by experimental simulations and showed that the path protection algorithm has much lower bandwidth then either line protection algorithms. Junter et al [7] compared the bandwidth efficiency of the hose-model with that of the customer pipe-model and concluded that the hose model results in bandwidth over provisioning by a factor of 2 to 3. They also show that in order to achieve reasonably low over-provisioning factors, the computation of a tree-structured resource-sharing topology for the whole VPN using explicit routing is the only viable candidate among the statically provisioned models without multi-path routing. Italiano et al [8] consider the problem of making a VPN, tolerant to single edge failures. Their paper suggests a 16-approximation for the problem of minimizing the total capacity of the edges in backup paths for a given VPN tree reservation. The goal of their work was to find a set of backup paths for a given *VPN tree* such that it could be restored under any single-link failure. The proposed algorithm tries to find the backup path set that needs the minimum total *protected bandwidth* allocation. In [9], Rebecca studies the problem of VPN provisioning in a programmable network environment. In particular, it considered automated VPN design and the distribution of resources across network nodes in an optimal manner. Gupta et al [10] investigated the problems concerning MPLS labels allocation and routing protocol on a *VPN tree*. Kumar et al [2] argued that bandwidth optimization of hose-model VPNs should be based on a tree topology (hereafter called: *VPN tree*). They presented a $O(mn)$ time algorithm to compute the bandwidth-optimization VPN tree where the links have infinite capacity and, given the bandwidth requirements of each endpoint in a VPN. The tree routing algorithm tries to find the *VPN tree* that needs minimum total bandwidth allocation on its edges. In the case of symmetric bandwidth requirements (i.e., $b^-(v) = b^+(v)$ for all VPN endpoints v), the tree routing algorithm is optimal and is guaranteed to find a *bandwidth-optimum VPN tree*. They proved that it is *NP*-hard to compute the optimal tree routing for general bandwidth values (in the case of asymmetric bandwidth requirements).

3 Problem formulation

The network backbone is modeled by an undirected graph $G(V,E)$, where V and E are the set of nodes and the set of bidirectional links, respectively. Let n_V and n_E denote the cardinality of V and E, respectively. In the hose model, each VPN specification consists of a set of nodes $P \subseteq V$ corresponding to the VPN endpoints, where for each node $i \in P$, the hose ingress and egress bandwidths are $b^-(v)$ and $b^+(v)$, respectively. As mentioned in Section 1, we consider tree structures to connect the VPN endpoints in P since trees have several desirable properties: (1) when link capacities are unbounded, we can find a minimum cost VPN tree [2]; (2) routing is simple as well as restoration; (3) heuristics to design a mesh topology usually start from a tree topology.

For the link (i, j) in tree T, we denote by $T_i^{(i,j)}$ and by $T_j^{(i,j)}$ the connected components of T containing, respectively, node i and node j when the link (i, j) is deleted from T. Also, let $P_i^{(i,j)}$ and $P_j^{(i,j)}$ denote the set of VPN endpoints in $T_i^{(i,j)}$ and $T_j^{(i,j)}$, respectively. Observe that all traffic from one VPN endpoint to another traverses the unique path in the VPN tree T between the two endpoints. In this section, we address the case when VPN endpoint bandwidth requirements are asymmetric, that is, for a VPN i, $b^-(v)$ and $b^+(v)$ may be unequal. Asymmetric ingress and egress bandwidths make the VPN tree design problem NP-hard.

Let us consider the link (i, j) that connects the two sets of endpoints $P_i^{(i,j)}$ and $P_j^{(i,j)}$. The bandwidth to be reserved on link (i, j) of T is given by [2]:

$$C(i, j) = Min(\sum_{l \in P_i^{(i,j)}} b^+(l), \sum_{l \in P_j^{(i,j)}} b^-(l)) \tag{1}$$

Thus, the total bandwidth reserved for tree T is given by:

$$C_T = \sum_{(i,j)\in T} C(i,j) \tag{2}$$

We seek to find a tree $T(V_T, E_T)$ where $V_T \subseteq V$ and $E_T \subseteq E$ for which C_T is minimum.

4 EVT algorithm

This section describes a new Evolutionary algorithm for optimized VPN tree provisioning in the hose model. It is an approximation algorithm based on the Simulated Evolution (SE) meta-heuristic proposed by [11]. For the SE meta-heuristic, a solution is seen as a population of movable elements. Each element m is characterized by a measure $g_m \in [0,1]$ that estimates the quality of its actual state goodness. Starting from a given initial solution, SE repetitively executes the following three steps in sequences, Evaluation, Selection, and Allocation, until certain stopping conditions are met (see Figure 1).

The Evaluation step estimates the goodness of each element in its current location. It is a measure of how near each element is to its optimum position. In the Selection step, the algorithm probabilistically selects elements for rearrangement. Elements with low goodness values have higher probabilities of getting selected.

A selection bias (B) is used to compensate for errors made in the estimation of goodness. Its objective is to inflate or deflate the goodness of elements. A high positive value of bias decreases the probability of selection and vice versa. Large selection sets degrade the solution quality due to uncertainties created by large perturbations. On the other hand, for high bias values the size of the selection set is small, which may degrade the quality of solution due to limitations of the algorithm to escape local minima. A carefully tuned bias value results in good solution quality and reduced execution time [3].

Elements selected during the *Selection* step are assigned to new locations in the *Allocation* step with the hope of increasing their goodness values and thereby improving the overall fitness of the solution (reducing the cost of the solution). *Allocation* is the step that has most impact on the quality of the search performed by the SE algorithm. A completely random allocation makes the SE algorithm behave like a random walk. Therefore, this operator should be carefully engineered to the problem instance and must include domain-specific knowledge. The *Selection* and *Allocation* steps allow the search to evolve toward more fit solutions.

```
ALGORITHM Simulated Evolution (M, L, B)

M: Set of movables elements
L: set of locations
B: Selection Bias
REPEAT
    EVALUATION
        FOR EACH $m \in M$   Evaluate $g_m$
    SELECTION
        FOR EACH $m \in M$
            IF $(g_m + B) < Random$ THEN
                $P_s = P_s \cup \{m\}$
    ALLOCATION
        FOR EACH $m \in P_s$
            Re-Allocate( $m$ )
UNTIL (Stopping Condition)
```

Figure 1: Pseudo code of Simulated Evolution

To apply this heuristic to identify an optimized VPN tree, we must adopt an appropriate solution encoding (how is a solution represented and what constitutes a movable element), choose a utility function to evaluate the solution quality, select an initial solution to start the exploration of the solution space; evaluate the quality of a movable element to be able to allocate new positions to the selected elements, and finally choose a stopping condition.

4.1 Initial Solution and solution encoding

The algorithm starts with an initial feasible solution built as follow: A minimum cost Steiner tree is constructed in two steps. First a minimum cost tree spanning all the nodes of the network is determined using

Kruskal algorithm. In a second step, all sub-branches non containing VPN terminals are removed. The solution encoding is path based, where a solution is coded as a vector of $(n_P - 1)$ elements, where $n_P = |P|$, where P is the set of VPN endpoints. For example consider the network of Figure 2, $P = \{2,5,7,8 and 9\}$.

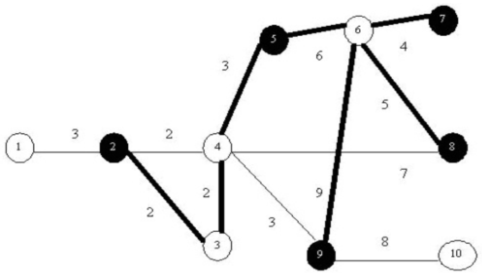

Figure 2: A network example with 10 nodes and 5 VPN endpoints: 2, 5, 7, 8, and 9.

This tree is encoded as a set of segment paths $T = \{\pi_{2345}, \pi_{567}, \pi_{768}, \pi_{869}\}$. The label on each link is the total cost in terms of bandwidth reserved on the link.

Each movable element is a path $\pi_{ij} = \langle v_i ... v_j \rangle$ representing a branch of the VPN tree, where v_i and v_j are VPN endpoints.

4.2 Evaluation

Usually, the quality of a VPN tree is strongly correlated with the quality of its paths. *EVT* proceeds by repetitively evolving a tree formed by paths π_{ij} of good quality.

Let $C(\pi_{ij})$ be the cost of path π_{ij}. The quality or goodness of the path π_{ij} connecting two endpoints v_i and v_j is evaluated by the following equation:

$$g_{\pi_{ij}} = 1 - \frac{C(\pi_{ij})}{Max(C(\pi_{kl}))_{\pi_{kl} \in T}} \qquad (3)$$

Thus, the higher the cost of a path π_{ij}, the closer to zero is its goodness.

4.3 Evolution step

In this step, the heuristic forces the search to evolve to a neighboring solution. This is achieved in two steps: Selection followed by Allocation.

4.3.1 Selection

The objective of this operation is to select the part of the VPN tree to change. In this work, a number of segment paths are selected for reallocation as follows. For each segment path π_{ij} of the current VPN tree, the goodness $g_{\pi_{ij}}$ of the segment path π_{ij} is compared to a randomly generated number $R \in [0,1]$. If $R > g_{\pi_{ij}} + B$, then π_{ij} is selected for reallocation. Note that according to our tree encoding, the number of path segments in any VPN tree is equal to $n_P - 1$.

We also experimented with the following selection strategy. The selection set consists of only the longest path segment. As shall be seen in the experimental section, this approach resulted in equally good solutions, with a sizeable reduction in run time.

4.3.2 Allocation

Before allocation, selected members are sorted in ascending order according to their goodness. During the *Allocation* step, the selected segment path π_{ij} is removed from the tree. This operation partitions the tree into two sub-trees. The VPN tree is reconstructed by identifying the best path connecting the two sub trees. For each path deleted, other candidate paths are tested.

To reduce the size of the solution space to be searched, we construct a trial tree by testing K paths between VPN terminals of the first and second sub-trees. The lowest cost path to connect the two sub trees is selected. In this

work, $K = n_P$ is the number of VPN terminals.
Figure 3 shows the trial trees generated for the network of Figure 2.

The goodness g_T of a VPN tree T is evaluated as follows:

$$g_T = 1 - x_T \tag{4}$$

Where,

$$x_T = \frac{C_T}{C_{\max}} \tag{5}$$

$C_{\max}$ is the cost of the maximum tree cost among all trees enumerated by the algorithm. Initially, $C_{\max}$ is equal to the cost of the initial tree; $C_{\max}$ gets updated as new trees are enumerated.
Figure 3 shows the trial trees generated for the network of Figure 2 as follow:

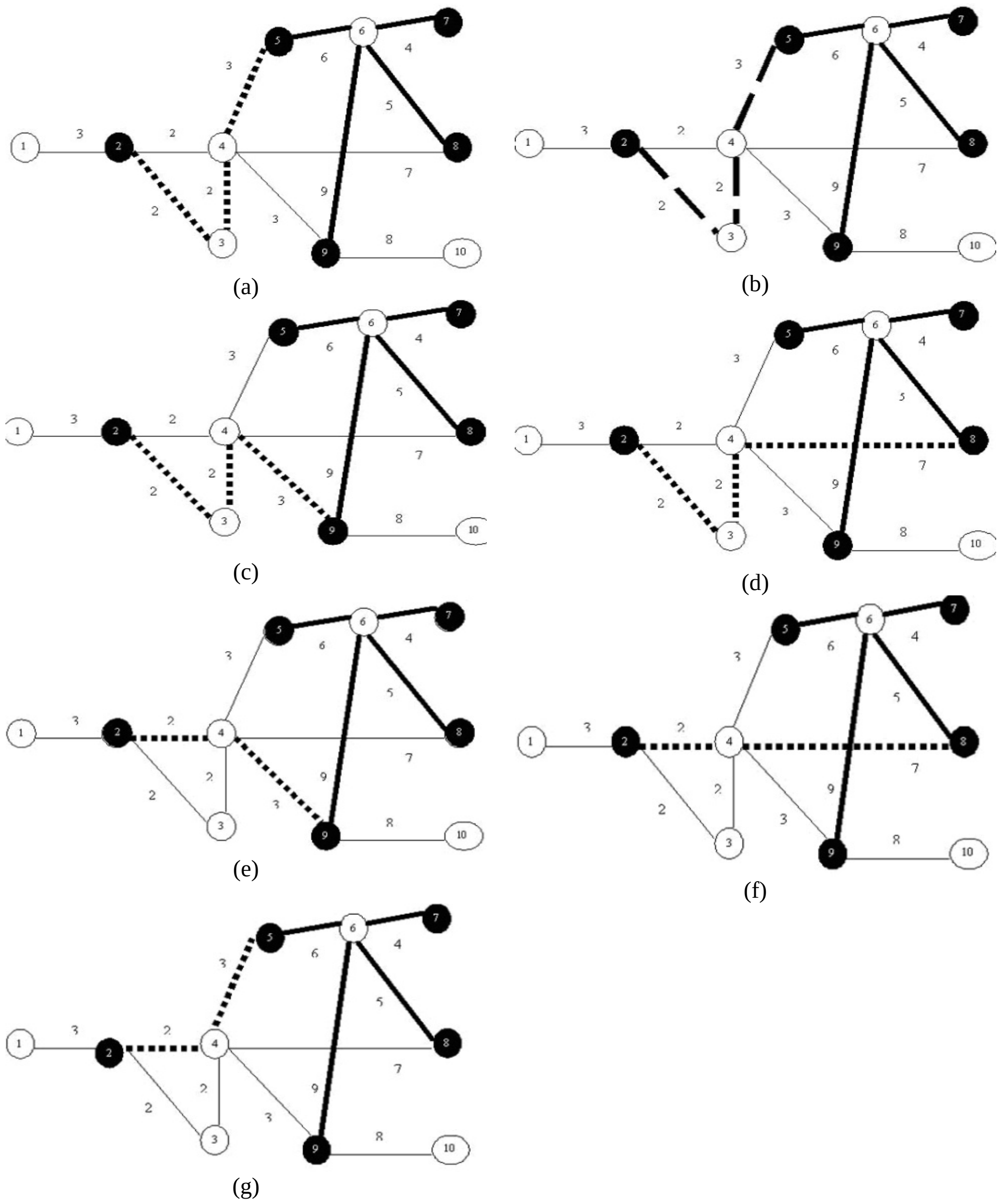

Figure 3: *Allocation step: (a) The segment path* π_{2345} *is removed, (b) two sub trees that result from previous removal* $T' = \{\pi_{2345}\}$ *and* $T'' = \{\pi_{567}, \pi_{768}, \pi_{869}\}$ *are generated, (c) first alternative to reconnect* T' *and* T'' *via* π_{2349}*, (d) second alternative via* π_{2348}*, Third alternative in (e) to reconnect* T' *and* T'' *via* π_{249}*, (f) The fourth alternative to reconnect* T' *and* T'' *via* π_{248}*, (g) The best and the fifth alternative to reconnect* T' *and* T'' *via* π_{245}*.*

During the search, the algorithm remembers the VPN tree with maximum g_T , which is returned when the algorithm stops.
All trial trees (for each removed path) are evaluated based on Equation (4) and the best tree among all perturbations will be chosen to continue the following iteration.

To further reduce the runtime of *EVT* in each iteration, we limited the selection set to consist of the segment path with the lowest goodness (the *longest path segment* that has the maximum bandwidth cost among all the members to be allocated).
As we shall see in the experimental section, this lower complexity strategy did not degrade the search strategy of the heuristic.

4.3.3 Stopping criterion

For *EVT* the search is stopped after a maximum of 100 iterations or if no improvement is observed for twenty consecutive iterations.

4.4 Pseudo code of *EVT* algorithm

The pseudo code of *EVT* algorithm is presented in Table 1 below.

Table 1: Pseudo code of *EVT* algorithm

Evolutionary Vpn Tree algorithm

```
 A: network graph G (V, E);
 Q: set of VPN endpoints.
 B: A selection bias
EVT (G(V, E), Q)
{
                // INITIALIZATION STEP
  for each q ∈ Q Do
     { Generate (b⁺(q), b⁻(q) }
  for each e ∈ E do
     { ComputeCost(e)}
  BestTree ← KruskalCost (G,Q)
  TempTree← BestTree
  Bool ← 1
  Repeat
                // EVALUATION STEP
    {   If ( Bool ←1)
        { Bool ← 0
         M ← Members( TempTree, Cost)
         For each m in M do
           { $\theta$ := (Max$_{m\in M}$ (Cost(m))
              g(m) ← 1- (Cost(m) / $\theta$ ) } }
                // SELECTION  STEP
     i ← 0
     For each m in M do
         { R← rand(1)
           If ( g(m) +B< R)
           { Ps(i) ← m  }  }
     Sort(Ps)
                // ALLOCATION STEP
     For each m in Ps do
     {  BestAlloc ← NULL
        PrunePath (m)
        Partition_Tree_into_Subtrees( Tree1, Tree2)
        Endpoint1 ← Tree1 ∩ m
        Endpoint2 ←Tree2 ∩ m
        TempPath ← Shortest_Path (G, Endpoint1, Endpoint2)
        TempTree ← replace_m_by_TempPath(TempTree)
        If (Cost (tempTree) > Cost (BestTree))
          { BestAlloc ← TempTree  }
        If ( BestAlloc < > Null)
          { Bool ← 1
            If (Cost(BestAlloc ) < Cost( BestTree)
              { BestTree ← BestAlloc }  }  }
  Until (Stopping criteria)
Return (BestTree) }
```

4.5 Time Complexity

The time complexity of *EVT* can be shown to be in the worst case $O(n_P {n_v}^2)$. Indeed, *EVT* consists of a main loop composed of three steps, Evaluation, Selection, and Allocation. The worst time complexity of the three steps are respectively, $O(n_P)$, $O(n_P \log n_P)$, and $O(n_P {n_v}^2)$. Then, since $n_v >> n_P$, the worst time complexity of one iteration of *EVT* is $O(n_P {n_v}^2)$. The number of iterations is usually less than 100.
The time complexity of *EVT* based *longest segment path* algorithm in the worst case becomes $O({n_v}^2)$.

5 Simulation Results

EVT has been extensively tested on several randomly generated networks. The WANMAN simulator [12] was used to generate random Waxman graphs [13]. Test networks are created as follows: First, n nodes are randomly distributed over a rectangular grid. Each node is placed with integer coordinates, and the Euclidean distance is determined between each pair of nodes. Then, edges are introduced between pairs of nodes (u,v) with a probability that depends on the distance between them.

$$P(u,v) = \beta \exp^{\frac{-d(u,v)}{L\alpha}} \tag{6}$$

Where $d(u,v)$ is the distance between nodes u and v, L is the maximum distance between any two nodes, α and β are parameters in the range $[0;1]$.
Larger values of α result in graphs with higher edge densities, while small values of β increase the density of short edges relative to longer ones. *EVT* was tested on different networks sizes of 10, 20, 30 and 50 nodes with $\alpha = 0.25$ and $\beta = 0.25$.

We run two sets of experiments. In the first set of experiments, we evaluate the quality of the search performed by *EVT*.
Figure 5 shows how well *EVT* is focused on the good solution subspace. As it is clear from the figure, more than 60% of the VPN trees found and evaluated by *EVT* were in the good solution sub-space (bar chart highly skewed towards the right), i.e. in the goodness interval [0.6-0.8].
Figure 6 tracks with time the total number of solutions found by the proposed *EVT* algorithm for various goodness cost intervals [0-0.25], [0.25-0.5], [0.5-0.75] and [0.75-0.1]. The plot clearly indicates that as the search progresses, *EVT* keeps evolving toward better solution subspaces and avoids the less quality solutions. These confirm the evolutionary nature of the algorithm. For example, as illustrated in Figure 6, beyond iteration 30, all solutions explored have goodness in the interval [0.75; 1].
Figures 5 and 6 clearly indicate that *EVT* has been well tuned to the problem addressed in this work.
Figure 7 tracks the cost of the best solution over time. As is clear, *EVT* converges within a maximum of 40 iterations.
Figure 8 represents the cost of the VPN tree. We noted that the cost vary through good solutions, as well as non-optimal solutions confirming the stochastic nature of *EVT*. It explores a subset of possible-solutions and elects the best one observed during the entire search.

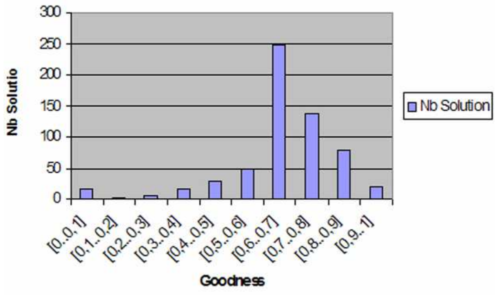

Figure 5: Total number of solutions evaluated by *EVT* for various intervals of goodness.

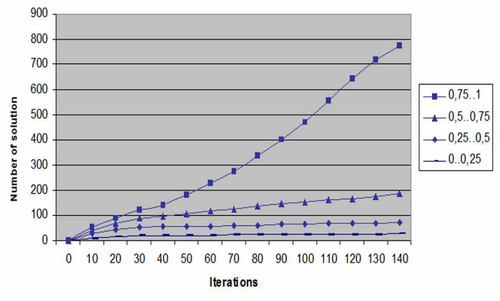

Figure 6: Total number of solutions enumerated by *EVT* vs. iteration count for different goodness ranges.

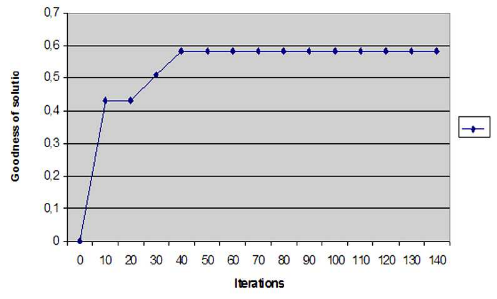

Figure 7: Goodness of best solution found by *EVT* vs. iteration number for network with 29 nodes and 20 endpoints.

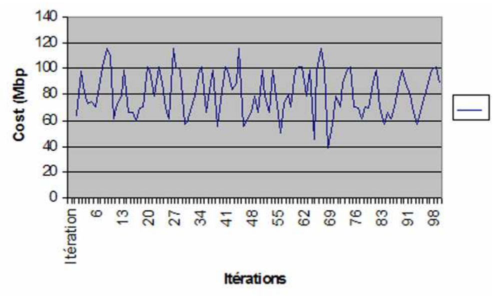

Figure 8: Variation of the cost during the iterations

In the second set of experiments, *EVT* is compared with the *primal-dual* algorithm introduced in [2]. We conducted 100 runs for every category of network size ((12 nodes, 8 endpoints), (20 nodes, 12 endpoints), (29 nodes, 22 endpoints).
Figure 9 shows the tree cost for varying network sizes *EVT* performs better in terms of tree bandwidth reservation cost. On all test cases, the solution cost found by *EVT* is lower than the primal-dual algorithm.

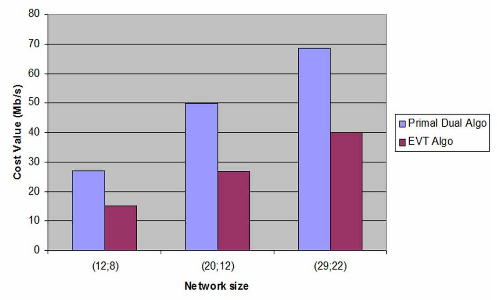

Figure 9: Cost comparison of *EVT* and *primal-dual* algorithms.

EVT based *longest segment path* algorithm was also evaluated. Experiments show that algorithm gives the same values as *EVT* based original allocation algorithm as shown in Figure 10.

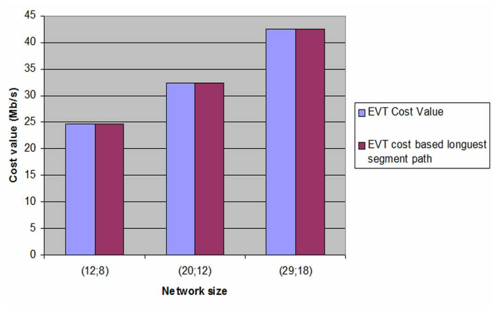

Figure 10: *EVT* algorithm vs. *EVT* based *longest segment path* algorithm.

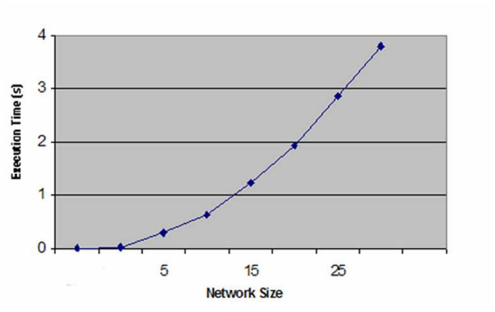

Figure 11: Execution time for different network sizes

As indicated in Figure 11, in practice, the execution time grows linearly with the network size. However, theoretically the worst time complexity is cubic in the network size.

6 Conclusions

In this paper, we presented a new algorithm for provisioning VPNs assuming a hose model based on the simulated evolution meta-heuristic. We connected VPN endpoints using a tree structure and our algorithm attempted to optimize the total bandwidth reserved on edges of the VPN tree. The proposed *EVT* algorithm was always able to find a low cost feasible VPN tree if one exists. As time elapsed, *EVT* progressively zoomed towards a better solution subspace, a desirable characteristic of approximation iterative heuristics. Simulation results show that *EVT* was always capable of finding better trees as compared to the *Primal-dual* algorithm reported in [2] and zoomed toward a better search subspace in a maximum of 30 iterations on all test networks.

References

[1] N.G. Duffield, Pawan Goyal, Albert Greenberg, *"A Flexible Model for Resource Management in Virtual Private Networks''*. *Proceedings of ACM SIGCOMM*, 1999.

[2] A. Kumar, R. Rastogi, A. Silberschatz and B. Yener, *"Algorithms for Provisioning Virtual Private Networks in the Hose Model"*, IEEE ACM Transactions on Networking, vol. 10, no. 4, August 2002.

[3] Habib Youssef and Sadiq M. Sait. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems.* IEEE Computer Science Press, 1999.

[4] Y. Liang Liu, Y. S.Sun and M. Chang Chen, *"MTRA: An On Line Hose Model VPN Provisioning Algorithm"*, IEEE INFOCOM 2004

[5] T. Erlebach and M. Rüegg, *"Optimal Bandwidth Reservation in Hose-Model VPNs with Multi-Path Routing"*, in Proceedings of IEEE INFOCOM, 2004.

[6] A. Balasubramanlan and G. Sasaki, *"Bandwidth Requirement for Protected VPNs in the Hose Model"*, Proceeding of IEEE International Symposium on Information Theory 2003.

[7] A. Jűttner, I. Szabơ and Á Szentesi,*"On Bandwidth Efficiency of the Hose Resource Management Model in Virtual Private Networks"*, in Proceedings of IEEE INFOCOM, 2003.

[8] G. Italiano, R. Rastogi and B. Yener, *"Restoration Algorithms for Virtual Private Networks in the Hose Model"*, in Proceedings of IEEE INFOCOM, 2002.

[9] Rebecca Isaacs. "*Lightweight, Dynamic and Programmable Virtual Private Networks*" Proceedings of the IEEE Third Conference on Open Architectures and Network Programming (OPENARCH 2000)

[10] A. Gupta, A. Kumar and R. Rastogi, *“Exploring the Trade-off Between Label Size and Stack Depth in MPLS Routing”*, in Proceedings of IEEE INFOCOM, 2003.

[11] Ralph.Michael. Kling. *Optimization by simulated evolution and its application to cell placement*. Ph.D. thesis, University of Illinois, Urbana, 1990

[12] Raouf Elyousfi *Algorithmes de mise à jour dynamique de l’arbre de routage*, Master thesis, Faculté des Sciences de Monastir, Département Informatique, 2005.

[13] Bernard M. Waxman, 1988 *‘‘Routing of multipoint connections''*. IEEE Journal on Selected Areas in Communication, 6(9): 1617-1622.