Evolutionary Algorithm for Provisioning VPN Trees based on Pipe and Hose Workload Models

Boulbaba Thabti[ab], Habib Yousse f[a], Aref Meddeb[a] and Ridha Mahjoub[b]

[a]PRINCE Research Unit, ISITCom Hammam Sousse, University of Sousse, Tunisia
[b]LAMSADE Research Lab, Paris Dauphine University France
b_thabti@yahoo.fr ; habib.youssef@fsm.rnu.tn ; Aref.Meddeb@infcom.rnu.tn ; mahjoub@lamsade.dauphine.fr

Abstract— **With the worldwide acceptance of the Internet as a de-facto public network, there is great interest in the deployment of Virtual Private Networks (VPNs) across IP networks. VPNs provide an economic alternative to the expensive private leased lines. The main purpose of a VPN is to provide a company secure communication among multiple sites through the shared Internet. The allocation of bandwidth for VPNs to meet the requirements specified by customers is an important traffic engineering research issue. This paper addresses the problem of provisioning VPN services based on pipe and hose workload models over shared network infrastructure with bandwidth guarantees, an issue studied only by different resource allocation methods [1] and not studied before by general optimization methods. The general problem of computing a constrained VPN tree with optimum bandwidth allocation is a hard combinatorial optimization problem [2]. Our proposed optimization method for provisioning VPNs is based on the Simulated Evolution (SE) meta-heuristic originally introduced in [3]. The proposed Evolutionary Algorithm (EA) connects VPN nodes using a tree structure while seeking to optimize the total bandwidth reserved on the edges of the VPN tree. Experimental results with Waxman network graphs [4] show that tree bandwidth costs obtained with hose workloads are higher by a factor of a maximum 2.5 compared to those obtained with pipe workloads.**

Keywords- VPN, Traffic engineering, Simulated Evolution, Hose-Model, Pipe Model, Over-provisioning.

I. INTRODUCTION

Virtual Private Network (VPN) establishes connectivity between a set of geographically dispersed endpoints over a shared network infrastructure. Providing VPN service is playing an important role in the revenue stream of Internet service provider. In order to be able to support a wide variety of customer requirements, network operators need a flexible and bandwidth efficient model, comparable to a private dedicated network established with leased lines.

Traditionally, provisioning VPN, *i.e.* setting up paths between every VPN customer pair, is based on the pipe model [5], in which the traffic demand is specified for each customer pair, and the bandwidth is reserved for point-to-point connection tunnel. For the pipe model, a traffic matrix which describes the required bandwidth between each VPN endpoint pair must be known a priori. However, the communication pattern between the endpoints is difficult to predict [6]. It is almost impossible to estimate the exact traffic matrix required by the pipe model. Figure 1 illustrates an example of using the pipe model.

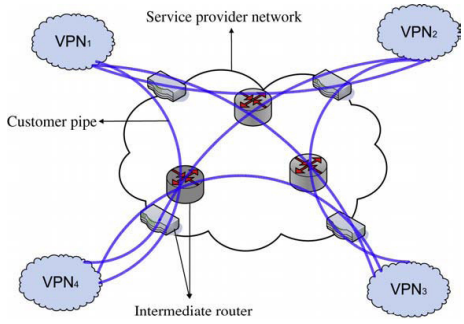

Figure 1. VPN pipe model

A more convenient and practical workload model for provisioning VPNs is the hose model, which was first introduced in [7] and subsequently refined in [8]. The hose model specifies, instead of a complete traffic matrix, the total amount of traffic which a customer injects into the network and the total amount of traffic which it receives from the network. This VPN specification is in fact backed up by the service level agreement (SLA). Figure 2 illustrates an example of using the hose model. Wherein four advantages were discussed: ease of specification, flexibility, multiplexing gain and characterization.

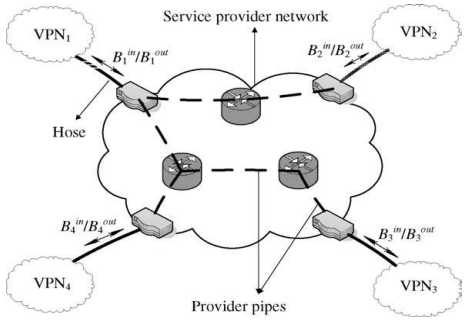

Figure 2. VPN hose model

978-1-4244-9953-3/11/$26.00 ©2011 IEEE

Provisioning a VPN requires identifying a sub-graph that connects the VPN endpoints; and reserving the necessary bandwidth on the corresponding physical links. It was shown that the optimum topology is a tree [2]. Other arguments that favour the use of tree topologies are: First, they are minimal networks; they provide connectivity without any unnecessary additional links; second, by providing a unique path between each pair of nodes, they eliminate the routing problem (i.e., deciding how traffic should flow between nodes). This simplifies both the network and its design. However, since trees are minimally connected they are also minimally reliable and robust [2]. This is why actual networks provide usually higher connectivity. Nevertheless, the design of the network often starts with a tree.

In this paper, we propose an Evolutionary Algorithm (EA) that finds least cost feasible provisioned VPN trees connecting the VPN endpoints based on the traditional pipe model and the hose model. We confirm results presented in [1], which showed that VPN trees obtained with the hose workload model are over-provisioned by a factor of 2 to 3 with respect to those obtained using a pipe workload model. Extensive simulations of the EA heuristic showed that the bandwidth cost of VPN trees designed based using a hose workload model are higher by a factor of 2 to 2.5 with those obtained with a pipe workload model.

The remainder of this paper is structured as follows. In Section 2, we review related work. Section 3 formally describes the VPN bandwidth provisioning problem. In Section 4, we describe our proposed Evolutionary Algorithm (EA). In Section 5, we present experimental results and a detailed comparison of the bandwidth reservation for VPN trees designed based on the pipe and the hose workload models. Section 6 concludes the paper.

II. MOTIVATION AND RELATED WORK

Reference [8] was the first to propose this concept for provisioning IP Virtual Private Networks. In their paper an analysis of the bandwidth efficiency of the hose model is presented. The evaluation is based on trace driven simulations of traffic derived from a voice network and from a large corporate network.

The work presented in [8] inspired research work on developing algorithms for designing minimum cost networks based on hose specifications. A number of provisioning algorithms for VPNs based on the hose model have been proposed [9-14].

Reference [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 a bandwidth-optimized 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 (the same amount of traffic sent and received by the hose), 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 (different amount of traffic sent and received by the hose).

Reference [1] 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. The authors concluded that the hose model performs better in reducing blocking probability, decreasing traffic loss, and ease of implementation over the pipe model. They also showed 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.

Reference [15] proposed a multi-path routing provisioning approach for the hose model and compared the performance of several approximation algorithms. The authors ran 6200 series of experiments with small connected random graphs with 3–5 nodes. Their results indicate that multi-path routing had reduced reservation cost compared to tree routing sharing.

Reference [16] investigate the VPN provisioning using multi-path routing with QoS guarantees such as bandwidth and propagation delay over shared network infrastructure based on the hose model and the pipe model. They present and reformulate MILP formulations which can be efficiently solved by standard MILP packages.

In [17], the authors enhanced the original hose model to allow for specification of delay limits between VPN endpoints. They proposed three provisioning approaches for the enhanced hose model: the pipe mesh approach, the multiple source-based trees approach, and the shared tree approach. Using theoretical analysis and simulation results the authors concluded that the shared tree approach is appealing because of its low provisioning cost and ease of routing and restoration.

To handle on-line multiple VPN setup requests, the authors in [18] propose a hose model VPN provisioning algorithm called MTRA to handle on-line multiple VPN setup requests rapidly and reduce the rejection ratio.

The focus of our paper is on the provisioning the VPN trees using the pipe and hose workload models. Similar works are conducted for VPN provisioning but using different topology such as multi path routing [16] and graph topology [19]. We propose an evolutionary algorithm based on the Simulated Evolution (SE) Meta heuristic introduced in [20] to optimize the total bandwidth reserved on edges of the VPN tree. Comparisons of the bandwidth reservation of VPN trees obtained with the hose model show an over-provisioning factor between 2 and 2.5 compared to those generated with the pipe model.

III. PROVISIONING VPN OVER SHARED NETWORK INFRASTRUCTURE

A. Problem Statement

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. 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.

B. VPN Provisioning using the Pipe Model

For the pipe model, a traffic matrix $D=(d_{uv})_{u,v\in P}$ describes the required bandwidth between each VPN endpoint pair. Traffic between each pair of customer access points is carried through the customer pipes (point-to-point connections) with a given pre-allocated bandwidth according to d_{uv}. 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 $c(i,j)$ to be reserved on link (i,j) of T is given by Equation (1):

$$c(i,j)=\sum_{u\in P_i^{(i,j)},\, v\in P_j^{(i,j)}} d_{u,v} \qquad (1)$$

C. VPN Provisioning using the Hose Model

In the hose model, only the ingress bandwidth $b^-(v)$ and the egress bandwidth $b^+(v)$ are specified for each customer access point. The traffic to and from a customer endpoint is arbitrarily distributed to other VPN endpoints, while satisfying the ingress and egress bandwidth requirements. We address the case when VPN endpoint bandwidth requirements are asymmetric, that is, for a VPN endpoint v, $b^-(v)$ 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]:

$$c(i,j)=\mathrm{Min}\Big(\sum_{l\in P_i^{(i,j)}} b^+(l),\ \sum_{l\in P_j^{(i,j)}} b^-(l)\Big) \qquad (2)$$

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

$$C_T=\sum_{(i,j)\in T} c(i,j) \qquad (3)$$

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.

D. Example

Consider Figure 3, which depicts a VPN tree T having three endpoints (1, 8 and 10). The sub trees $T_4^{(4,5)}$ and $T_5^{(4,5)}$ resulting from the removal of the link (4,5) from T. We have $P_4^{(4,5)}=\{1\}$ and $P_5^{(4,5)}=\{8,10\}$.

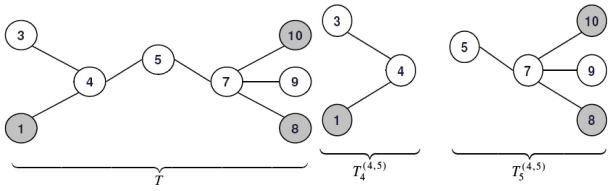

Figure 3. A tree T and two sub trees $T_4^{(4,5)}$ and $T_5^{(4,5)}$ resulting from the removal of the link (4, 5) from T.

For the hose model, the bandwidth requirements for the endpoints are as follows: for endpoint 1, $b^-(1)=3$ and $b^+(1)=6$, and for endpoint 8, $b^-(8)=6$ and $b^+(8)=3$ and for endpoint 10, $b^-(10)=2$ and $b^+(10)=3$. The bandwidth reserved on edge (4,5) is $C_T(4,5)=9$ where c(4,5)=6 and c(5,4)=3.

Let us consider for the pipe model the following traffic matrix $D=(d_{1,10}=1,\ d_{10,1}=2,\ d_{1,8}=2,\ d_{8,1}=2,\ d_{8,10}=3,\ d_{10,8}=4)$. The bandwidth reserved on (4,5) is $C_T(4,5)=7$ where c(4,5)=3 and c(5,4)=4.

E. Network Dimensioning

Our goal when trying to characterize the required bandwidth reservation of the hose model was to be able to compare its total network capacity requirement to the pipe model. Naturally, we wanted to have network examples with "comparable" traffic volumes when doing the evaluation, which is not trivial if we recall that the main difference of the hose and the pipe dimensioning models is in the traffic description. To generate comparable pipe and hose workloads, we proceed as follows.

- We generate a pipe workload in the form of a traffic matrix $D=(d_{uv})_{u,v\in P}$ containing point to point bandwidth values required to dimension according to the pipe model.
- As a "comparable" hose characterization we calculate the hose parameters egress and ingress bandwidths as the respective sums of the rows and columns of the traffic matrix D. This way we calculate the smallest possible hose that can accommodate the given traffic matrix.

After dimensioning the same network according to these inputs with both models, we calculate the ratio between the resource requirements of the hose and the customer–pipe models, which we consider a good indicator of the required extra capacity. In the following we call this ratio as the *over-provisioning factor*.

IV. The Evolutionary Algorithm (EA)

In this section, we present our Evolutionary Algorithm (EA). EA is an Evolutionary algorithm for optimized VPN tree provisioning. It is an approximation algorithm based on the Simulated Evolution (SE) meta-heuristic introduced in [3]. SE is a general iterative heuristic proposed by Ralph Kling [20]. It falls in the category of algorithms which emphasizes the behavioral link between parents and offspring, or between reproductive populations rather than genetic link [21]. This scheme combines iterative improvement and constructive perturbation and saves itself from getting trapped in local minima by following a stochastic perturbation approach. It iteratively operates a sequence of evaluation, selection and allocation steps, starting from an initial solution (see Figure 4). The selection and allocation steps constitute a compound move from current solution to another feasible solution of the solution space. SE proceeds as follows. It starts with a randomly or constructively generated valid initial solution. A solution is

seen as a set of movable elements. Each element m has an associated goodness measure g_m in the interval $[0,1]$. The main loop of the algorithm consists of three steps: **evaluation**, **selection** and **allocation**. These steps are carried out repetitively until some stopping condition is satisfied. In the evaluation step, the goodness of each element is estimated. It is a measure of how near each element is to its optimum position. In the selection step, the algorithm probabilistically selects a subset of elements for rearrangement from current solution. Elements with low goodness values have higher probabilities of getting selected. A bias parameter 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 low 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]. Finally, the allocation step tries to assign the selected elements to better locations. *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. Other than these three steps, some input parameters for the algorithm are set in an earlier step know as **initialization**.

To apply this heuristic to enumerate 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.

```
ALGORITHM Simulated_Evolution (M,L,B)
        M: Set of movables elements
        L: set of locations
        B: Selection Bias
Repeat
      EVALUATION

        For Each m ∈ M Evaluate g_m
      SELECTION
        For Each  m∈ M
          IF (g_m + B) < Random  THEN
              P_s = P_s ∪ {m}
      ALLOCATION
        For Each m ∈ P_s
          Re-Allocate( m )
Until (Stopping Condition)
Return Best Solution
(Simulated_Evolution)
```

Figure 4. Structure of the Simulated Evolution algorithm

A. 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 not 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 5, $P = \{2,5,7,8,9\}$.

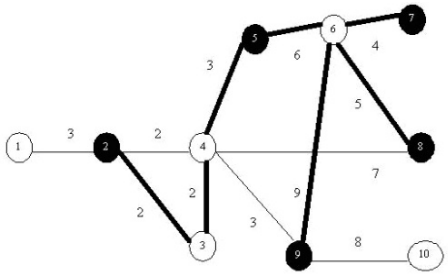

Figure 5. A network example with 10 nodes and 5 VPN endpoints: 2, 5, 7, 8, and 9.

The tree found initially by Kruskal is shown in bold in Figure 5. 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 $\pi_{ij} = \langle v_i \ldots v_j \rangle$ representing a branch of the VPN tree, where v_i and v_j are VPN endpoints.

B. Evaluation

Usually, the quality of a VPN tree is strongly correlated with the quality of its paths. EA 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:

$$g_{\pi_{ij}} = 1 - \frac{C(\pi_{ij})}{\text{Max}\,(C(\pi_{kl}))\pi_{kl} \in T} \qquad (4)$$

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

C. 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.

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.

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, we chose to $K = n_P$, 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:

$$g_T = 1 - x_T \quad (5)$$

Where,

$$x_T = \frac{C_T}{C_{max}} \quad (6)$$

Let C_{max} be 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. All trial trees (for each removed path) are evaluated based on Equation (5) and the best tree among all perturbations will be chosen to continue the following iteration. During the search, the algorithm remembers the VPN tree with maximum g_T, which is returned when the algorithm stops.

3) Stopping criterion

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

4) Pseudo code of EA

A step-by-step description of our Pseudo code of EA is described in the following.

Step 1: Initialization.
- 1.1. Compute the minimum bandwidth tree by Kruskal's algorithm,
- 1.2. Prune leaves of T that do not correspond to VPN endpoints (that is, do not belong to P).
- 1.3. Compute bandwidth reservation $cost(e), \forall e \in T$.
- 1.4. Set Best_Tree=Kruskal_Tree,

Step 2: Evaluation.
- 2.1 Set Temp_Tree= Best_Tree and iteration=0.
- 2.2 Compute members M from Temp_Tree.
- 2.3 Compute bandwidth reservation $cost(m), \forall m \in M$.
- 2.4 Set $\theta = Max(cost(m)), \forall m \in M$.
- 2.5 Compute the goodness $g(m)$ as $g(m) = 1 - [cost(m)/\theta]$.
- 2.6 Repeat 2.4 and 2.5, $\forall m \in M$.

Step 3: Selection.
- 3.1. Generate random float R witch $R = random(1)$.
- 3.2. If $g(m) + B < R$, then queue m in the Selected Set for Allocation (SSA)
- 3.3. Repeat 3.1 and 3.2, $\forall m \in M$.
- 3.4. Sort the element of SSA.

Step 4: Allocation.
- 4.1. Set Best_Allocated_Tree= NULL.
- 4.2. Prune path m from Temp_Tree. We obtain two sub trees T_1 and T_2.
- 4.3. Let $u, v \in P$ as $u = T_1 \cap m$ and $v = T_2 \cap m$.
- 4.4. Compute temporary shortest path Temp_Path as Temp_Path=Shortest_Path (G, u, v) by Dijkstra's algorithm.
- 4.5. Set Temp_Tree= $T_1 \cup T_2 \cup$ Temp_Path.
- 4.6. If $cost(Temp\_Tree) < cost(Best\_Tree)$, then set Best_Allocated_Tree = Temp_Tree.
- 4.7. If $cost(Best\_Allocated\_Tree) < cost(Best\_Tree)$, then set Best_ Tree = Best_Allocated_Tree.
- 4.8. Repeat 4.1- 4.7, $\forall m \in SSA$.

Step 5: Termination check.
- 5.1 Set iteration = iteration + 1.
- 5.2 If iteration < Stopping_criteria go to Step 2; otherwise, stop.

5) Time Complexity

EA consists of a main loop composed of three steps, Evaluation, Selection, and Allocation. The worst time complexity of the three steps are $O(n_p)$, $O(n_p \log n_P)$ and $O(n_p n_v^2)$ respectively. Then, since $n_v >> n_p$, the worst time complexity of EA iteration is $O(n_p n_v^2)$. The number of iterations is usually less than 100.

V. Simulation Results

Our proposed EA heuristic has been tested on several randomly generated networks. The WANMAN simulator [22] was used to generate random topologies generated using the well-known Waxman model [4]. The Waxman model was selected to obtain random networks with close resemblance to real networks because it verifies the power law proposed by [23]. The simulations are implemented using the C language and all tests were performed on a dual Core processor of 2 GHz and with 2 GB of RAM. EA was tested for three network sizes of 10, 20, and 30 nodes. Each simulation result given below is the average of 10 rounds of simulation runs for every category of network size and traffic patterns as described in Table I.

TABLE I. Test Network Characteristics

Networks	Network Characteristics			
	Number of Nodes	Maximum Node Degree	Number of Edge	Number of endpoints
Net1	10	7	32	5
Net2	20	12	147	10
Net3	30	18	233	15

We run sets of experiments to evaluate the quality of the search performed by EA and also to compare the bandwidth reservation for provisioning VPN trees using the pipe model and the hose model.

Figure 6 tracks with time the total number of solutions found by the proposed EA 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, EA keeps evolving toward better solution subspaces and avoids the lower quality solutions.

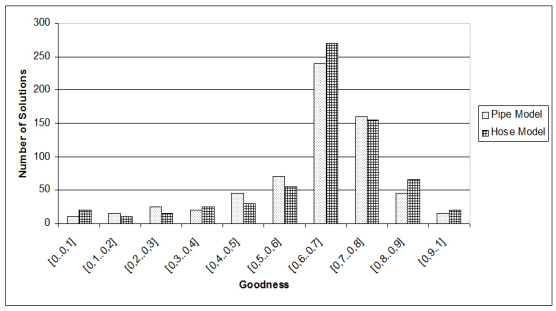

Figure 6. Total number of solutions evaluated by EA for various intervals of goodness using pipe model and hose model.

As it is clear from Figure 6, more than 60% of the VPN trees found and evaluated by EA for both models were in the good solution sub-space (bar chart highly skewed towards the right), i.e. in the goodness interval [0.6-0.8]. This confirms the evolutionary nature of the algorithm.

Figure 7 tracks the cost of the best solution over time. As is clear, EA converges within a maximum of 40 iterations for the hose model and the pipe model.

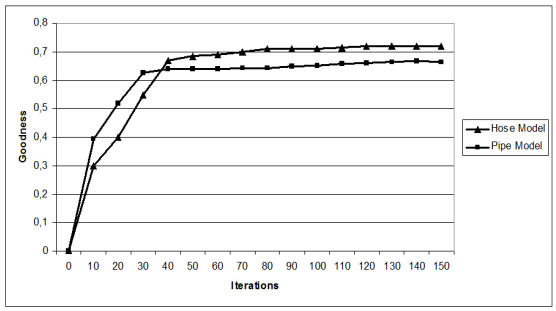

Figure 7. Goodness of best solution found by EA vs. iteration number for network Net3 using the pipe model and hose model.

Figure 8 represents the cost of the obtained VPN trees. We observe that the cost increases as well as decreases, confirming the stochastic nature of EA. It explores a subset of possible-solutions and elects the best one observed during the entire search.

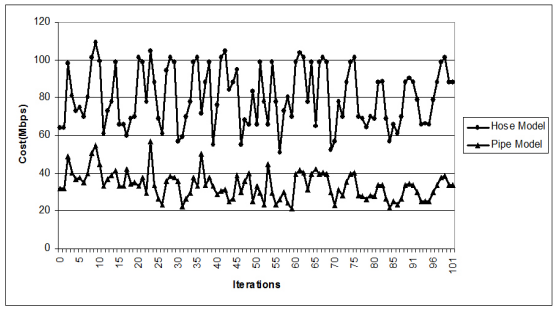

Figure 8. The cost variation during the iterations in pipe and hose models.

As illustrated in Figure 9, in practice, the execution time grows linearly with the network size. However, theoretically the worst time complexity is cubic in the network size. As it is clear, EA perform better in terms of execution time for provisioning VPN tree using the pipe model than using the hose model.

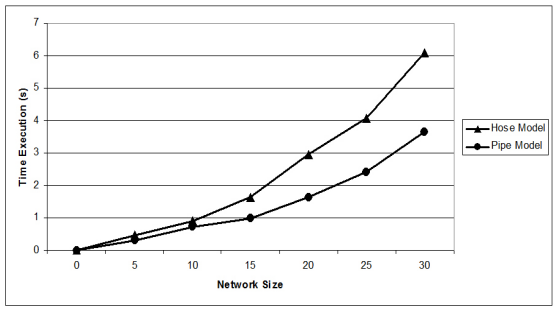

Figure 9. Execution time comparison for different network sizes for the pipe and hose models.

Figure 10 shows the tree cost for varying network sizes. EA performs better in terms of tree bandwidth reservation cost using the pipe model than the hose model.

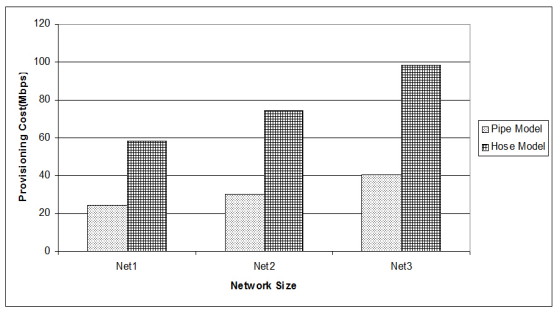

Figure 10. Bandwidth cost comparison of VPN trees obtained with the pipe model and the hose model.

Figure 11 shows the over-provisioning factor for varying network sizes of the provisioned VPN trees by the hose model compared with those obtained by the pipe model. We can see that the over-provisioning factor is below 2.5.

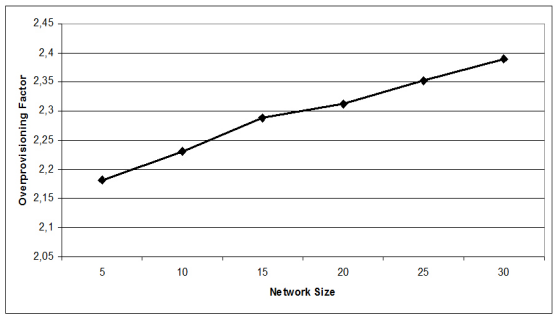

Figure 11. Bandwidth over-provisioning factor of VPN trees obtained by the pipe model and the hose model.

VI. CONCLUSIONS

This paper proposes an evolutionary based algorithm for the provisioning of VPN trees using either a pipe or a hose workload. Our proposed Evolutionary Algorithm (EA) is

based on the Simulated Evolution Meta heuristic introduced in [20]. EA was always able to find a low cost feasible VPN tree if one exists. As time elapsed, EA progressively zoomed towards a better solution subspace, a desirable characteristic of approximation iterative heuristics. Numerical results on a set of test cases show an over-provisioning factor of at most 3 achieved by the hose model in bandwidth reservation over the pipe model. But the over-provisioning of the hose model can be well offset by the benefits it delivers in terms of ease of specification, flexibility, reduced blocking, decreased traffic loss, multiplexing gain and characterization.

REFERENCES

[1] A. Júttner, I. Szabo and Á Szentesi,"On Bandwidth Efficiency of the Hose Resource Management Model in Virtual Private Networks", in Proceedings of IEEE INFOCOM, 2003.

[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] Sadiq M. Sait and Habib Youssef. Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems. John Wiley & Sons, 2003.

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

[5] B. Davie, Y. Rekhter, MPLS Technology and Applications, Morgan Kaufmann, San Mateo, CA, 2000.

[6] Medina, A., Taft, N., Salamatian, K., Bhattacharyya S. and C. Diot: "Traffic matrices estimation: existing techniques and new directions", ACM SIGCOMM'02, August 19-23 (2002), Pittsburgh, USA

[7] J. A. Fingerhut, S. Suri, and J. S. Turner, "Designing Least-Cost Nonblocking Broadband Networks", Journal of Algorithms, 24(2), pp. 287-309, 1997.

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

[9] R. Yuan, W.T. Strayer, Virtual Private Networks: Technologies and Solutions, Addison-Wesley, 2001.

[10] S. Raghunath, K.K. Ramakrishnan, Resource management for virtual private networks, IEEE Commun. Magazine 45 (4) (2007) 38–44.

[11] A. Gupta, A. Kumar, T. Roughgarden, Simpler and better algorithms for network design", ACM Symp. Theory Comput. (2003).

[12] W.C. Tat, L. King-Shan, K.L. Yeung, P.W. Chi, Routing algorithm for provisioning symmetric virtual private networks in the hose model, IEEE Global Telecommun. Conf. 2 (2005) 1–5.

[13] Monia Ghobadi, Sudhakar Ganti, Gholamali C. Shoja, "Resource optimization algorithms for virtual private networks using the hose model",Computer Networks, Vol. 52, Issue 16, 13 November 2008, Pages 3130-3147

[14] Boulbaba Thabti and Habib Youssef , "EVT: Evolutionary Algorithm for VPN Tree Provisioning", in Proceeding of International Network Optimization Conference INOC2009, 26-29 April 2009 Pisa, Italy.

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

[16] Quanshi Xia , "Provisioning VPN over Shared Network Infrastructure", 4th International Conference on Networking, Reunion Island, France, April 17-21, ICN 2005, LNCS 3420, pp. 500–507, 2005.

[17] L. Zhang, J. Muppala, S. Chanson, Provisioning virtual private networks in the hose model with delay limits, Hong Kong University, International Conference on Parallel Processing, 2005, pp. 211–218.

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

[19] A. Altin, E. Almadi, P. Beloti, M.C Pinar, " Provisioning Virtual Private Networks under Traffic Uncertainty", Special Issue: on Multicommodity Flows and Network Design, Networks Volume 49, Issue 1, pages 100–115; October 2006.

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

[21] D.B. Forgel. An introduction to Simulated Evolution optimization. IEEE Transaction on Neural Networks, 5(1) :3-14, Jan 1994

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

[23] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. In ACM SIGCOMM, Cambridge, MA, September 1999.