Computer Networks 57 (2013) 2766–2774

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Hose workload based exact algorithm for the optimal design of virtual private networks

Ibrahima Diarrassouba [c], Ali Lourimi [a,*], Ali Ridha Mahjoub [b], Habib Youssef [a]

[a] PRINCE Research Unit, ISITCom Hammam Sousse, University of Sousse, Tunisia
[b] Université Paris Dauphine, Laboratoire LAMSADE, Paris Place De-Lattre-de-Tassigny, 75775 Paris Cedex 16, France
[c] Université du Havre, Laboratoire de Mathématiques Appliquées du Havre, 25 rue Philippe Lebon, BP 1123, 76063 Le Havre Cedex, France

ARTICLE INFO

Article history:
Received 17 February 2013
Received in revised form 21 May 2013
Accepted 21 June 2013
Available online 13 July 2013

Keywords:
Virtual private network
Hose model
Branch-and-Cut
Maximum flow problem
Cutting plane
Separation problem

ABSTRACT

The Virtual Private Networks (VPN) optimal bandwidth allocation problem with tree topology has been widely discussed in the literature. Most of the algorithms proposed by researchers to solve this problem use approximation schemes. In this paper, we propose an exact and efficient Branch-and-Cut algorithm for the problem in the context of a hose workload model. In particular, we consider the case when the ingress and egress traffic at VPN endpoints are asymmetric and the links of the network have unbounded capacities. The algorithm proposed here is based on a linear integer programming formulation for the problem introduced by Kumar et al. (2002) [2]. Using this and a deep investigation of the polyhedral structure of that formulation, our algorithm permits to solve large instances of the problem having up to 120 nodes and 10 terminals.

1. Introduction

A Virtual Private Network (VPN) allows to securely connect distant clients (e.g. via VPN tunnels) over a public network. The logical connections constitute an overlay network. Such a network is said virtual because it connects local networks by secured logical tunnels over a physical public infrastructure such as the Internet, and private because only the end routers of the local networks on terminal sides of the tunnels can "see" the data.

Today network virtualization represents the most significant innovation in networking. The basic entity offered by network virtualization is a virtual network (VNet), that is, an abstraction of a network over a given subset of terminals [8,6,7]. In general, establishing a VNet requires embedding a subnetwork that spans the terminals of the VNet in the physical network.

A central challenge in network virtualization is the efficient use of the given physical resources. Companies found in network virtualization the ideal solution to establish on-demand overlay networks that enable their customers to securely access company resources.

VPN design has been the subject of numerous studies over the last years [1–5,9–13]. Although the security issue has been settled, bandwidth allocation, survivability and QoS provisioning remain open research issues. The survivability here is to install spare resources (nodes, links) in order to ensure that the VPN continue to deliver the traffic in case of failure. The resource management concerns the allocation of the bandwidth on the links of a VPN so that all the traffic can be delivered to the VPN clients. In this paper, we deal with the optimal bandwidth allocation problem assuming a tree topology and a hose workload with asymmetric ingress and egress traffic.

The hose workload model has been introduced by Duffield et al. [17] as a flexible model for specifying the

* Corresponding author. Tel.: +216 23655353.
E-mail addresses: diarrasi@univ-lehavre.fr (I. Diarrassouba), Ali.Lourimi@infcom.rnu.tn (A. Lourimi), mahjoub@lamsade.dauphine.fr (A.R. Mahjoub), Habib.youssef@fsm.rnu.tn (H. Youssef).

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2013.06.009

bandwidth requirements of VPN endpoints. According to this model, each VPN specification consists of the following.

- The set of endpoints P.
- The hose ingress bandwidth B_i^{in} and the hose egress B_i^{out}, for each endpoint $i \in P$, expressed in Kbps.

B_i^{in} and B_i^{out} represent the expected traffic flows that the endpoint i receives, and sends, respectively. In contrast to the pipe model, where the workload is specified in the form of a matrix giving the expected traffic between any two VPN endpoints, the hose model only requires the specification of ingress and egress bandwidth of each VPN endpoint. The hose model has several advantages such as the ease of specification, scalability, multiplexing gain and simple characterization [2].

We consider tree structures to connect the VPN endpoints in P. With tree routing, the VPN endpoints are connected by a Steiner tree and all VPN traffic is sent along the unique paths of this tree, from senders to receivers. Kumar et al. [2] showed that the optimal tree can be obtained by solving a set of integer programs.

1.1. Our results

In this paper, we consider the integer programming formulation introduced by Kumar et al. [2]. This formulation contains an exponential number of constraints. We show that the linear relaxation of this problem can be solved in polynomial time. For this, we show that the so-called separation problem (that is whether a constraint is violated or not by a given solution) is polynomially solvable. In fact, we show that this problem reduces to a maximum flow problem in a special graph, which can be solved in polynomial time.

We also investigate the inequalities of that formulation and show that they are all nonredundant, and hence, they all have to be considered in the resolution of the problem. Using this, we develop a Branch-and-Cut algorithm (that is, a linear programming-based Branch-and-Cut algorithm reinforced by the addition of violated inequalities) for the problem. Our computational study shows that the algorithm we proposed is very efficient and permits to solve large instances with up to 120 nodes and 10 terminals.

1.2. Related work

The VPN optimal bandwidth allocation problem has been widely studied in the literature. In [2], Kumar et al. studied the problem of computing optimal VPN trees in the context of a hose workload and when the links of the network have unbounded capacities. The authors considered this problem in the symmetric case, that is, when $B_i^{in} = B_i^{out}$ for every endpoint $i \in P$ and devised a polynomial algorithm for the problem in this case.

In [4], Erlebach and Ruegg consider the bandwidth allocation problem for hose model VPN under multi-path routing. They showed that considering multi-path routing instead of tree and single-path routing offers significant advantages. In fact, they showed that the problem can be solved in polynomial time in both cases with and without finite capacities on the links, whereas it is NP-hard for tree and single-path routing. For this, they proposed a linear programming formulation for the problem and showed that this linear program can be solved in polynomial time.

Most of algorithms proposed in the literature for the optimal bandwidth allocation problem are heuristic approaches. Also, a huge work has been considered to develop approximation algorithms. In [2], Kumar et al. also considered the problem in the asymmetric case, that is where egress and ingress bandwidth may be different. They showed that variant of the problem is NP-Hard. The authors also gave an integer programming formulation and proposed a 10-approximation algorithm for the problem.

Gupta et al. [1] extended the work presented in [2] and proposed, in the asymmetric case with infinite capacities on the edges, an algorithm whose approximation ratio equals 9.002. Their algorithm is based on the linear relaxation of the formulation of [2] and rounding procedure. Swamy et al. [21] improved the approximation ratio to 5.

In [3], Giuseppe et al. showed that in the case of sum-symmetric, that is when $\sum_{i\in P} B_i^{in} = \sum_{i\in P} B_i^{out}$ and infinite capacities on edges, a solution can be computed in polynomial time with approximation ratio 3. They also gave a new integer programming formulation for the problem.

In [24] Eisenbrand et al. considered a variant of the virtual private network design problem which generalizes the previously studied symmetric and asymmetric case. In their model the terminal set is partitioned into a number of groups, where terminals of each group do not communicate with each other. Their main result is a 4.74 approximation algorithm for this problem.

In [25], Chu and Lea explored the optimal weight setting to support hose-model VPN traffic in an IP-based hop-by-hop routing network. They presented a mixed-integer programming formulation to compute the optimal link weights that can maximize the ingress and egress VPN traffic admissible to a hop-by-hop routing network. They also presented a heuristic algorithm for solving the link weight searching problem for large networks.

1.3. Notations

Here we give the basic notations we will use all along the paper. A network is modelled by an undirected graph $G = (V, E)$, where V and E are the node and edge sets respectively. An edge between two nodes i and j will be denoted by ij. We will denote by $P \subseteq V$ the set of terminals (VPN endpoints). A tree is a connected subgraph of G with no cycle and is denoted by T. A tree T is said to be a *Steiner tree* with respect to the terminal nodes of P if it connects all the nodes of P. Note that a Steiner tree may not connect all the nodes of G. We will also denote by T_v a tree rooted at the node $v \in V$.

Given an edge ij in a tree T, we denote by T_i^{ij} the connected component of T containing node i when the edge ij is deleted from T, and P_i^{ij} the set of VPN endpoints in T_i^{ij}. We also denote by $C_T(ij)$ the bandwidth revered on

the link ij of the VPN tree T, and by C_T the total bandwidth reserved on all the links of the VPN tree T. The distance between two nodes i and j in a graph G is the length of a shortest path (in terms of number of edges) between i and j, and is denoted by $d_G(ij)$. $d_T(ij)$ will denote the distance between nodes i and j in the tree T.

Given a node set $W \subseteq V$, we will denote by $\overline{W} = V \setminus W$. If W and W' are two disjoint node sets, $[W, W']$ will denote the set of edges having one endnode in W and the other one in W'. For $W \subseteq V, \delta(W) = [W, \overline{W}]$, is the *cut set* induced by the node set W.

We will denote by $H = (V, A)$ a directed graph with V the node set and A the arc set. An arc from a node i to a node j will be denoted (i, j). Given two node sets W and W', $[W, W']$ will denote the set of arcs having their origin in W and their destination in W'. We will have $\delta^+(W) = [W, \overline{W}]$ and $\delta^-(W) = [\overline{W}, W]$.

2. Integer programming formulation for the problem

In this section, we present the integer programming formulation introduced by Kumar et al. [2] for the VPN optimal bandwidth allocation, when the egress and ingress traffic at every VPN endpoint may be different. They show that the problem is NP-hard in this case. In the following, we give some structural properties of VPN trees, and describe how one can obtain an optimal VPN tree by computing a series of Steiner trees. In fact, each Steiner tree is computed by solving a linear integer program. All the results given here can be found in [2].

Given a VPN tree T, the only traffic that crosses a link ij from node i to j is the traffic coming from endpoints in P_i^{ij} and going towards endpoints in P_j^{ij}. The traffic from node i to node j cannot exceed $min\{\sum_{l\in P_i^{ij}} B_l^{out}, \sum_{l\in P_j^{ij}} B_l^{in}\}$. Thus the bandwidth to be reserved on an edge ij of T from i to j is given by

$$C_T^{ij} = min\left\{\sum_{l\in P_i^{ij}} B_l^{out}, \sum_{l\in P_j^{ij}} B_l^{in}\right\}.$$

The total bandwidth reserved for T is

$$C_T = \sum_{ij\in T} C_T^{ij}.$$

An edge ij of VPN tree T is *biased towards* i if the following two conditions hold:

- $\left(\sum_{l\in P_i^{ij}} B_l^{in} < \sum_{l\in P_j^{ij}} B_l^{out}\right)$ or $(\sum_{l\in P_i^{ij}} B_l^{in} = \sum_{l\in P_j^{ij}} B_l^{out}$ and P_i^{ij} contains a special node, say $\widehat{l})$.
- $\left(\sum_{l\in P_i^{ij}} B_l^{out} < \sum_{l\in P_j^{ij}} B_l^{in}\right)$ or $(\sum_{l\in P_i^{ij}} B_l^{out} = \sum_{l\in P_j^{ij}} B_l^{in}$ and P_i^{ij} contains a special node, say $\widehat{l})$.

An edge ij is said to be *biased* if it is biased towards either i or j. An edge is said to be *balanced* if it is not biased. A node of T is a *core node* if a balanced edge is incident on it.

Kumar et al. [2] showed that the core nodes associated with a VPN tree T are connected by a tree consisting entirely of balanced edges, each of these edges having a cost equal to $M = min\left\{\sum_{l\in P} B_l^{in}, \sum_{l\in P} B_l^{out}\right\}$. If the balanced edges are removed from T, then, in each of the resulting connected components, say C_v, there is a single core node v. The cost of C_v, that is the bandwidth reserved to the edges of C_v, is $\sum_{l\in C_v\cap P} d_T(v, l) \times (B_l^{in} + B_l^{out})$.

If $core(T)$ denotes the set of core nodes in T and by $bal(T)$ the set of balanced edges in T, then the cost of T is

$$C_T = M * |bal(T)| + \sum_{v\in core(T)} \sum_{l\in C_v\cap P} d_T(v, l) \times (B_l^{in} + B_l^{out}).$$

To compute the optimal VPN tree T^*, we simply need to compute a set of nodes S^* for whom the quantity

$$M \times b + \sum_{l\in P} min_{v\in S}\{d_G(v, l)\} \times (B_l^{in} + B_l^{out}) \quad (1)$$

is minimum, where b is the number of edges in the Steiner tree connecting the nodes in S [2].

If we suppose that a given node $v \in V$ belongs to the optimal node set S^*, then one can compute S^* by computing a Steiner tree rooted in v. Consequently, to compute S^*, one can compute, for each node $v \in V$, a Steiner tree rooted in v, say S_v, whose cost is minimum with respect to Eq. (1), and then, choose S^* as the tree whose cost is minimum among all the trees S_v.

The problem of computing a Steiner tree S_v rooted in v, for all $v \in V$, can be formulated as a linear integer program. Let x_{ij}, y_i and z_e be 0–1 variables, where $y_i = 1$ if $i \in S_v$ and $y_i = 0$ otherwise, $x_{ij} = 1$ if VPN endpoint j is assigned to node i and $x_{ij} = 0$ otherwise, and finally $z_e = 1$ if edge e is in the Steiner tree connecting the nodes in S_v. For all $j \in P$, we let $B_j = B_j^{in} + B_j^{out}$.

Computing S_v is equivalent to solving

$$\text{Minimize} \quad \sum_{i\in V, j\in P} d_G(i, j) \times B_j \times x_{ij} + M \times \sum_{e\in E} z_e$$

$$\text{subject to} \quad \sum_{i\in V} x_{ij} \geqslant 1, \quad \forall j \in P, \quad (2)$$

$$y_i - x_{ij} \geqslant 0, \quad \forall i \in V \text{ and } j \in P, \quad (3)$$

$$\sum_{e\in\delta(\widehat{V})} z_e - \sum_{i\in\widehat{V}} x_{ij} \geqslant 0, \quad \forall \widehat{V} \subset V, \quad (4)$$

$$x_{ij},\ y_i \text{ and } z_e \in \{0, 1\}. \quad (5)$$

In this formulation, each VPN endpoint j must be assigned to at least one node in S_v (constraints Eqs. (2) and (3)). Constraints Eq. (4) are the so-called *cut constraints* and ensure that the nodes of S_v are connected by a Steiner tree.

After computing the optimal core node set S^*, the optimal VPN tree T^* is obtained from S^* by first adding in T^* the edges of the Steiner tree of S^*. Then, we contract in G the nodes of S^* and construct a Breadth-First-Search (BFS) tree rooted at that supernode and connecting all the VPN endpoints in P (as the leaves). Finally, the edges of that BFS tree are added in T^*.

The whole algorithm is summarized below.

Algorithm 1. Algorithm to compute the optimal VPN tree

Algorithm 1: Algorithm to compute the optimal VPN tree

Data: $G = (V, E)$, P, $\{(B_i^{in}, B_i^{out}),$ for all $i \in P\}$
Result: Optimal VPN tree T^*
begin
  $C_T = \infty$; **forall** $v \in V$ **do**
    Compute the optimal set S_v; **if** *Cost of* $S_v < C_T$ **then**
      $S^* = S_v$; $C_T =$ Cost of S_v;
  // Computing the optimal tree T^*
  Add in T^* the edges of S^*; Contract in G the nodes of S^*; Compute a BFS tree rooted at supernode obtained after step 1 and connecting all the nodes of P; Add in T^* the edges of this BFS tree;
end

As explained before, each node set S_v, $v \in V$, is computed by solving the linear integer program given above. Since the optimal Steiner tree problem is NP-hard in the general case, we need to use integer programming techniques like Branch-and-Cut techniques to solve it. Moreover, it contains an exponential number of constraints (the cut constraints Eq. (4)). In the next section, we describe our Branch-and-Cut algorithm for solving the problem.

3. Branch-and-Cut algorithm

3.1. Description

Branch-and-Cut method is one of the most effective method for solving combinatorial optimization problems. It has been widely used to solve both real word and theoretical combinatorial problems which are formulated as integer linear programs. The Branch-and-Cut method consists of solving the problem by applying the Branch-and-Bound method upon linear programming techniques. Each node of the Branch-and-Bound tree consists in a linear program obtained by considering the linear relaxation of the problem.

The linear relaxation of the problem can be solved by using the so-called *cutting plane method*. This method allows to solve a linear program (LP for short) by using a subset of inequalities of the original LP. The cutting plane method is particularly effective when the number of inequalities of the original LP is large (e.g. exponential) since it permits to solve the LP by using only a few (polynomially bounded) number of inequalities.

For more details on Branch-and-Cut and cutting planes algorithms, the reader can refer to [18,19,23].

The main ingredient of the cutting plane method is the so-called separation problem which consists, given a solution $\bar{x}$ of an LP and a class of inequalities $\mathcal{F}$, in saying if there exists an inequality of $\mathcal{F}$ which is violated by $\bar{x}$ or not. An algorithm used to solve a separation problem is called a *separation algorithm*.

The integer programming formulation described above to solve the Steiner tree problems contains an exponential number of cut constraints Eq. (4). To solve its linear relaxation, we use the cutting plane method. In the next section, we describe the separation algorithm we use for the cut inequalities Eq. (4).

3.2. Separation of the cut constraints

In this section, we discuss the separation problem of the cut constraints Eq. (4). We show that the problem reduces to maximum flow problem with lower and upper bounds, and hence, can be solved in polynomial time but before, we introduce the following notions.

Let $G = (V, A)$ be a directed graph with s and t two nodes of G. An *st-flow from source s to destination t* is a function.

$f : A \to \mathbb{R}_+$ which satisfies the so-called *flow conservation constraints*

$$\sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a) = 0, \quad \text{for all } v \in V \setminus \{s, t\}.$$

Associated with a flow function, the quantity

$$f_0 = \sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a) = \sum_{a \in \delta^-(t)} f(a) - \sum_{a \in \delta^+(t)} f(a)$$

is the *value of the st-flow*.

The maximum flow problem with lower and upper bounds is the problem of finding a flow function which maximizes the flow value f_0 and such that the flow value $f(a)$ on every arc $a \in A$ is restricted to be lower than a positive capacity $U(a)$ and greater than a positive value $L(a)$. In the classical version of the maximum flow problem, we have $L(a) = 0$, for all $a \in A$.

The maximum flow problem with lower and upper bounds is closely related to the so-called circulation problem. A *circulation* on the graph G is a flow function for which the flow conservation constraints hold on every node of the graph, that is

$$\sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a) = 0, \quad \text{for all } v \in V.$$

An *st*-flow function with lower and upper bounds can be transformed into a circulation by adding in G an arc from t to s with 0 and ∞ as lower and upper bounds, respectively. The maximum *st*-flow problem with lower and upper bounds is then feasible if and only if the associated circulation problem is feasible. We have the following fundamental theorem which can be found in [14].

Theorem 3.1. *Given a directed graph $G = (V, A)$, the circulation problem in G with nonnegative lower and upper bounds on arc flows is feasible if and only if for every node set W*

$$\sum_{(i,j) \in \delta^-(W)} L_{ij} \leqslant \sum_{(i,j) \in \delta^+(W)} U_{ij}. \tag{6}$$

Conditions Eq. (6) described in Theorem 3.1 are the so-called *cut conditions*.

To find a feasible circulation in a graph, one can apply a maximum flow-based algorithm. In fact, we first choose two nodes s and t and then compute a classical maximum

st-flow in G, that is with upper bounds U_{ij} and 0 as lower bounds for every arc $(i, j) \in A$. If for every arc $(i, j) \in A$, we have $L_{ij} \leqslant f(i, j)$, then this flow is feasible with respect to lower and upper bounds L_{ij} and U_{ij}. If, on the contrary, there exists some infeasible arcs, that is $f(i, j) < L_{ij}$, for some arcs $(i, j) \in A$, then we try to make these arcs feasible. For this, we try, for each infeasible arc (p, q) to gradually increase the flow value on an augmenting cycle (in the residual graph) containing arc (p, q), until the flow $f(p, q)$ becomes greater than L_{pq} or until there is no augmenting cycle. In this latter case, if (p, q) has become feasible, then we choose another infeasible arc and repeat that procedure. If there is no augmenting cycle and (p, q) is still infeasible, then there is no feasible circulation in G, with respect to bounds L_{ij} and U_{ij} and by Theorem 3.1, one can find a node set W which does not satisfy the cut conditions Eq. (6). This algorithm can be implemented in polynomial time by using any efficient maximum flow algorithm and any cycle finding method. For more details on this procedure and on maximum flow algorithms, see [14].

In the rest of the paper, we will denote by $\overline{\Gamma} = (\overline{x}, \overline{y}, \overline{z})$ a solution of the problem and by $G(\overline{z})$ the support graph of this solution, that is the graph obtained from G by removing every edge $e \in E$ having $\overline{z}(e) = 0$. We will also let $\overline{z}([W, \overline{W}]) = \sum_{e \in [W, \overline{W}]} \overline{z}_e$, for all $W \subseteq V$.

Before describing the separation procedure of the cut constraints Eq. (4), we need to distinguish three cases which are summarized in the three following lemmas.

Proofs of the following lemmas are given in Appendix A.

Lemma 3.1. *Suppose that the graph $G(\overline{z})$ is not connected and that $C_1, \ldots, C_r$ are its connected components. Suppose also that $v \in C_1$. Then, the cut constraints induced by the node sets C_k, $k = 2, \ldots, r$, are satisfied by $(\overline{x}, \overline{y}, \overline{z})$ if and only if, for all $j \in P, \overline{x}_{ij} = 0$, for all $i \in C_k$, $k = 2, \ldots, r$.*

Lemma 3.2. *Suppose that $G(\overline{z})$ is not connected and that $C_1, \ldots, C_r$ are its connected components, with $v \in C_1$. Suppose also that the cut constraints induced by node sets C_k, $k = 2, \ldots, r$, are satisfied by $(\overline{x}, \overline{y}, \overline{z})$, for all $j \in P$. Then, there exists a violated cut constraint induced by $(\overline{x}, \overline{y}, \overline{z})$ if and only if there exists a violated cut constraint induced by a node set $\widehat{V}$ such that $\widehat{V} \subsetneq C_1$ and $v \in C_1 \setminus \widehat{V}$.*

In the next lemma, we show that if the graph $G(\overline{z})$ is connected, then the separation problem of the cut constraints Eq. (4), for a given terminal $j \in P$, is equivalent to finding a maximum flow with lower and upper bounds in a special graph, described below (note that G_j has the same node set as G).

Let $j \in P$ be a VPN endpoint and let $G_j = (V, A_j)$ be the directed graph obtained as follows. For every edge $ab \in E$, we add in G_j two arcs (a, b) and (b, a) with bounds $(0, \overline{z}_{ab})$. For every node $i \in V \backslash \{v\}$, we add an arc (v, i) with bounds $(\overline{x}_{ij}, \infty)$. We have the following lemma.

Lemma 3.3. *Suppose that the graph $G(\overline{z})$ is connected and let $j \in P$ be a terminal node. The solution $(\overline{x}, \overline{y}, \overline{z})$ satisfies all the cut constraints associated with j if and only if for all $i_0 \in V \backslash \{v\}$, there exists a feasible flow with lower and upper bounds in the graph G_j between i_0 and v.*

Lemmas 3.1–3.3 allows to devise an exact separation algorithm for the cut constraints Eq. (4). We first look if the graph $G(\overline{z})$ is connected or not. If it is not connected and $C_1, \ldots, C_r$ are its connected components, with $v \in C_1$, we check if for all $j \in P$ and for every node $i \in C_k$, $k = 2, \ldots, r$, $\overline{x}_{ij} > 0$. If yes, then, by Lemma 3.1, the cut constraint induced by C_k and j is violated. If $G(\overline{z})$ is not connected and for every $k = 2, \ldots, r$, and $j \in P, \overline{x}_{ij} = 0$, then, by Lemma 3.2, the separation problem of the reduces to looking for violated cut constraints in the subgraph induced by the component C_1. Thus, this latter case is similar to the case when $G(\overline{z})$ is connected. Now if $G(\overline{z})$ is connected, then, by Lemma 3.3, the separation problem is equivalent to finding a feasible flow between i and v, for all $i \in V \backslash \{v\}$. To do this, we build, for all $j \in P$, the graph G_j as described above. Then, for all $i \in V \backslash \{v\}$, we look for a feasible flow between i and v, using the algorithm described above (and in [14]). If there is no feasible flow between i and v, then the algorithm returns a node set W which does not satisfies the cut conditions. By Lemma 3.3, this node set, together with j induces a violated cut constraint. If for all $i \in V \backslash \{v\}$ and $j \in P$, there exists a feasible flow in G_j between i and v, then the solution $(\overline{x}, \overline{y}, \overline{z})$ satisfies all the cut constraints. The computation of the node set W can be done by using the maximum flow algorithm of Ford and Fulkerson which runs in $(|V||A_j|)$. The separation algorithm for cut constraints can, hence, be implemented to run in polynomial time.

4. Experimental study

4.1. Network topology

In this section we give an analysis of the effectiveness of the proposed algorithm in terms of execution time and number of constraints generated. The proposed separation procedure is also evaluated in terms of solution quality found.

In this study, two methods are used for generating graphs. The first is the Waxman method [22] where a flat random graph is constructed. This method defines the probability of having an edge between two nodes u and v as

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

where $0 < \alpha$, $\beta < 1$, d is the Euclidian distance between nodes u and v and L is the maximum distance between the two freely selected nodes. An increase in the parameter α results in an increase of the number of edges in the graph, while a decrease in the parameter β increases the ratio of the long edges against the short ones.

The second method used to generate graphs is the one proposed by Barabasi and Albert [15]. This model suggests two possible causes for the emergence of the power law in the frequency of outdegrees in network topologies: incremental growth and preferential connectivity. The network growth process consists of an incremental addition of new nodes. The preferential connectivity refers to the tendency of a new node to connect to existing nodes that are highly connected or popular. When a node u connects to the

network, the probability that it connects to a node v (already belonging to the network) is given by:

$$P(u, v) = \frac{d_v}{\sum_{k \in V} d_k},$$

where d_v is the degree of the node belonging to the network, V is the set of nodes connected to the network and $\sum_{k \in V} d_k$ is the sum of outdegrees of the nodes previously connected.

BRITE (Boston university Representative Internet Topology gEnerator) [20] was used as a tool for generation of realistic network topologies based on Waxman and Barabasi graphs methods.

4.2. Computational results

We have implemented our algorithm for computing the optimal VPN tree in C++ and used COIN-OR [16] to solve it. We have used an M40-331 TOSHIBA laptop with an Intel Pentium M processor of 1.73 GHz and 1 GB of RAM.

We have used BRITE generator [20] to generate graphs with 10–120 nodes, and using Waxman and Barabasi methods, respectively. For each type of instances (Waxman or Barabasi), we have considered 5, 7 and 10 terminals nodes in the graph. BRITE generator has been configured to generate graphs whose number of edges equals twice the number of nodes of the graph. The experimental results obtained are given in Tables 2 and 3 for Waxman and Barabasi instances, respectively.

Gap is a measure of the relative error between the cost of the optimal solution *COpt* and the cost of the linear relaxation of the optimal Steiner tree solution (say *Cr*). Here *COpt* and *Cr* are expressed in Mbps. *Gap* is computed as follows,

$$Gap = \left| \frac{COpt - Cr}{Cr} \right| * 100. \quad (7)$$

The entries of Tables 2 and 3 are defined in Table 1.

Recall that the algorithm solves a series of Steiner tree problems, each of them corresponding to a node of the graph. The optimal value COpt of the problem is the best value among all the optimal solutions of the Steiner tree problems.

First we observe from Table 2 that the algorithm has been able to solve to optimality all the instances of the Waxman test set. A large number of cut inequalities is generated for all the instances but the total CPU time is relatively small. The total CPU time varies from 1 s (Waxman 23: 10 nodes and 10 terminals) to 57 min (Waxman 10: 110 nodes and 5 terminals). Also, the Gap (the relative error between the optimal bandwidth and the linear relaxation of the optimal Steiner tree problem) is relatively small. The Gap is less than 2.57% for all the instances. For 11 instances, the Gap is 0, that is the optimal Steiner tree problem is solved with no branching during the Branch-and-Cut process.

For Table 3, we also observe that the algorithm is able to solve to optimality all the instances of the Barabasi test set. For these instances also, the number of generated cut inequalities is relatively important but the total CPU time is relatively small (between 1.25 s and 1h11min for Barabasi 23 and Barabasi 11, respectively). Also for these instances, the Gap is relatively small (less than 3.35%) except for two instances, Barabasi 9 and Barabasi 33, having respectively a Gap of 16.28% and 16.47%. Also, for 8 instances the Gap achieved is 0% indicating that the Branch-and-Cut algorithm for the optimal Steiner tree problem is solved to optimality without branching. The small CPU time obtained for both Waxman and Barabasi instances shows that the algorithm can be used to efficiently solve and quickly design a VPN tree with given endpoints and bandwidths. The efficiency of the algorithm is also shown by the small Gap achieved during the resolution process. This points out that the formulation of the problem as a series of Steiner tree problems produces good lower bounds and, in many cases, can quickly obtain an optimal solution of the problem.

Table 1
Notations.

$\|V\|$	Number of nodes of the graph
$\|E\|$	Number of edges of the graph
$\|P\|$	Number of VPN endpoints (terminals) of the graph
NCut	Total number of generated cut inequalities
COpt	Optimal solution (optimal bandwidth to be allocated)
Gap	The relative error between the optimal solution and the linear relaxation of the optimal Steiner tree problem (that is the Steiner tree problem giving the value COpt)
TT	Total CPU time in seconds

Table 2
Results for Waxman graphs.

$\|V\|$	$\|E\|$	$\|P\|$	NCut	COpt	Gap	*TT*
40	80	5	9443	94	0.01	72.15
45	90	5	15,163	83	0.15	138.83
50	100	5	11,784	77	0.15	107.51
55	110	5	23,549	98	0.02	294.92
60	120	5	32,358	102	0.19	434.37
70	140	5	2,826,529	97	0.00	498.82
80	160	5	3,232,413	103	0.01	1515.31
90	180	5	3,936,736	107	0.04	1020.52
100	200	5	548,294	113	0.69	2135.00
110	220	5	1,837,571	12,060	0.06	3421.00
120	240	5	2,644,177	10,273	2.50	1226.68
30	60	7	3606	116	0.31	24.53
35	70	7	4775	132	0.03	39.53
40	80	7	13,007	164	0.00	150.51
45	90	7	18,124	164	0.00	227.11
50	100	7	26,461	154	0.12	360.56
55	110	7	28,357	143	0.06	428.35
60	120	7	33,125	155	0.00	506.23
70	140	7	2,951,349	17,362	0.00	517.48
80	160	7	3,542,104	20,103	0.03	679.00
90	180	7	4,301,294	20,827	2.57	1949.00
100	200	7	1,193,158	25,781	0.05	1164.75
10	20	10	225	105	0.00	0.71
15	30	10	666	137	0.00	3.29
20	40	10	1943	152	0.00	12.22
25	50	10	2691	134	0.00	22.07
30	60	10	6237	208	0.52	66.75
35	70	10	5645	184	0.12	60.94
40	80	10	18,258	194	0.15	235.23
45	90	10	12,535	201	0.00	250.33
50	100	10	9403	218	0.34	154.17
55	110	10	15,525	241	0.25	322.15
60	120	10	28,331	220	0.00	583.23

Table 3
Results for Barabasi graphs.

$\|V\|$	$\|E\|$	$\|P\|$	NCut	COpt	Gap	*TT*
40	80	5	6722	70	0.11	50.91
45	90	5	5521	68	0.00	41.44
50	100	5	6762	74	0.00	59.43
55	110	5	27,828	91	0.08	324.30
60	120	5	8971	73	2.02	85.33
70	140	5	2,756,710	11,218	0.04	232.85
80	160	5	3,094,710	10,050	0.06	194.02
90	180	5	3,732,625	9464	0.09	472.67
100	200	5	314,101	7703	16.28	720.85
110	220	5	1,452,690	12,918	0.05	1139.74
120	240	5	2,138,990	15,421	0.13	4313.00
40	80	7	6057	115	1.33	53.12
45	90	7	10,827	131	0.02	119.77
50	100	7	11,986	109	0.46	136.02
55	110	7	12,261	114	0.10	153.22
60	120	7	22,523	134	0.24	340.90
70	140	7	2,871,571	10,804	1.53	116.09
80	160	7	3,320,956	15,563	0.22	341.70
90	180	7	4,050,091	13,262	0.23	543.25
100	200	7	722,739	17,556	0.03	1144.15
110	220	7	157,643	12,778	0.00	842.00
120	240	7	195,791	19,547	0.00	1681.00
10	20	10	356	119	0.48	1.25
15	30	10	523	136	0.00	2.39
20	40	10	1366	171	0.39	8.55
25	50	10	1925	158	3.35	14.08
30	60	10	2486	171	0.29	21.30
35	70	10	5407	167	0.45	53.17
40	80	10	6402	219	0.61	79.35
45	90	10	8936	202	0.09	123.95
50	100	10	15,518	197	0.00	246.58
55	110	10	20,213	228	0.00	392.73
60	120	10	14,694	249	16.47	289.47

Now we compare the results obtained with respect to the number of nodes and the number of VPN endpoints. We can see that the CPU time increases with the number of terminals. For example, for Waxman 5, Waxman 18 and Waxman 27 (60 nodes and respectively 5, 7 and 10 terminals), the CPU time is respectively 434 s, 506 s and 583 s. This can be explained by the number of valid cut inequalities which increases with the number of VPN endpoints. The algorithm may then spend more time in the cut constraints separation phase. Note that, even if the number of cut inequalities increases with the VPN endpoints, the polyhedral investigation of the problem yields a reduction of the necessary cut inequalities generated during the Branch-and-Cut algorithm. Thus, the small total CPU time let us suppose that the algorithm can solve to optimality large instances (with more than 120 nodes and 10 demands) in a reasonable amount of time, and give, in the worst case good lower and upper bound to the optimal solution.

5. Conclusions

The problem of designing virtual private networks has been widely studied in the literature but its resolution is actually limited to heuristic algorithms.

In this paper we give an efficient exact algorithm to solve this problem. The newly proposed algorithm gives an exact optimal solution, i.e. a VPN tree which connects all the terminals of the VPN and having the minimal allocated bandwidth. Our algorithm relies on a Branch-and-Cut method with an efficient separation procedure based on the problem of maximum flow with nonnegative lower bounds algorithm. It also relies on a polyhedral investigation of the problem for describing and using facet-defining inequalities in the algorithm. The small CPU time and Gap achieved during the computational study show that the proposed algorithm can efficiently solve the problem.

The Branch-and-Cut algorithm can be improved by performing a deep polyhedral investigation and find new valid inequalities for the Steiner tree problems. Also, the algorithm may be improved by dividing primal heuristics and solve to optimality very large sized instances.

Appendix A. Proofs for separation results

Proof 5.1 (*Proof of Lemma 3.1*). First, suppose that, for all $k = 2,\dots,r$, the cut constraint induced by the node set C_k and any $j \in P$, is satisfied by $(\overline{x},\overline{y},\overline{z})$. Then, we have

$$\overline{z}_e \geqslant \sum_{i\in C_k} \overline{x}_{ij}, \quad \text{for all } j \in P,\ k = 2,\dots,r.$$

Since C_k is a connected component of $G(_b arz)$, we have that $\sum_{e\in\delta(C_k)}\overline{z}_e = 0 \geqslant \sum_{i\in C_k}\overline{x}_{ij}$. As $\overline{x}_{ij} \geqslant 0$, for all $i \in V$, $j \in P$, we obtain $\overline{x}_{ij} = 0$, for all $i \in C_k$, $k = 2,\dots,r$ and for all $j \in P$.

Now suppose that for some $k_0 \in \{2,\dots,r\}$ and $j_0 \in P$, the cut constraint induced by C_{k_0} and j_0 is violated by $(\overline{x},\overline{y},\overline{z})$. Then, we have that $\sum_{e\in\delta(C_{k_0})}\overline{z}_e < \sum_{i\in C_{k_0}}\overline{x}_{ij}$. Since C_{k_0} is a connected component of $G(\overline{z})$, and hence $\sum_{e\in\delta(C_{k_0})}\overline{z}_e = 0$, we get $\sum_{i\in C_{k_0}}\overline{x}_{ij} > 0$. Thus, $\overline{x}_{i_0 j_0} > 0$ for some node $i_0 \in C_{k_0}$. □

Proof 5.2 (*Proof of Lemma 3.2*). If there exists a node set $\widehat{V}\subsetneq C_1$, with $v \in C_1 \setminus \widehat{V}$ and a terminal $j_0 \in P$, such that the cut constraint induced by $\widehat{V}$ and j_0 is violated by $(\overline{x},\overline{y},\overline{z})$, then the result is obvious. Thus, suppose that there exists a node set $\widehat{V} \subseteq V$, with $v \notin \widehat{V}$, and a terminal $j_0 \in P$ which induce a violated cut constraint for $(\overline{x},\overline{y},\overline{z})$, that is

$$\sum_{e\in\delta(\widehat{V})} \overline{z}_e < \sum_{i\in C_{k_0}} \overline{x}_{ij}. \tag{A.1}$$

Let $\widehat{V}_k = \widehat{V} \cap C_k, k = 1,\dots,r$. Clearly, we have that

$$\widehat{V} = \bigcup_{k=1}^{r} \widehat{V}_k.$$

Thus, $\sum_{e\in\delta(\widehat{V})}\overline{z}_e$ and $\sum_{i\in\delta(\widehat{V})}\overline{x}_{ij}$ can be written as

$$\sum_{e\in\delta(\widehat{V})} \overline{z}_e = \sum_{k=1}^{r} \overline{z}[\widehat{V}_k, C_k \setminus \widehat{V}_k] \tag{A.2}$$

and

$$\sum_{i\in\widehat{V}} \overline{x}_{ij_0} = \sum_{k=1}^{r}\sum_{i\in\widehat{V}_k} \overline{x}_{ij_0}. \tag{A.3}$$

Since the cut constraints induced by C_k, $k = 2,\dots,r$, by Lemma 3.1, we have that $\overline{x}_{ij_0} = 0$, for all $i \in \widehat{V}_k, k = 2,\dots,r$. Thus,

$$\sum_{i\in\widehat{V}}\overline{x}_{ij_0}=\sum_{i\in\widehat{V}_1}\overline{x}_{ij_0}. \tag{A.4}$$

By combining Eqs. (A.1), (A.2) and (A.4), we get

$$\sum_{k=1}^{r}\overline{z}[\widehat{V}_k,C_k\setminus\widehat{V}_k]<\sum_{i\in\widehat{V}_1}\overline{x}_{ij_0}.$$

As $\overline{z}_e\geqslant 0$, for all $e\in E$, we get

$$\overline{z}([\widehat{V}_1,C_1\setminus\widehat{V}_1])=\overline{z}(\delta(\widehat{V}_1))\leqslant\sum_{k=1}^{r}\overline{z}[\widehat{V}_k,C_k\setminus\widehat{V}_k]<\sum_{i\in\widehat{V}_1}\overline{x}_{ij_0}.$$

Moreover, $v\notin\widehat{V}_1$ since $v\notin\widehat{V}$. This implies, that the cut constraint induced by $\widehat{V}_1$ and j_0 is violated by $(\overline{x},\overline{y},\overline{z})$, and the result holds. □

Proof 5.3 (*Proof of Lemma 3.3*). First suppose that $(\overline{x},\overline{y},\overline{z})$ satisfies all the cut constraints. We are going to show that there exists a feasible flow with lower and upper bounds between every node $i_0\in V\setminus\{v\}$ and v. For this, we transform the flow problem into a circulation problem by adding in G_j an arc (v,i_0) from v to i_0 with bounds $(0,\infty)$. Note that by, Theorem 3.1, there is a feasible circulation if and only if the cut conditions Eq. (6) hold for every node set $W\subseteq V$.

Let $W\subseteq V$. If $v\in W$, then $\delta^+(W)$ contains at least one arc (v,i_1), with $i_1\neq i_0$. Hence, at least one arc of $\delta^+(W)$ have an infinite capacity. Since the lower bound of every arc of G_j is finite, we have that

$$\sum_{(a,b)\in[W,\overline{W}]}U_{ab}>\sum_{(a,b)\in[\overline{W},W]}L_{ab}. \tag{A.5}$$

If $v\in\overline{W}$, then $\delta^+(W)=[W,\overline{W}]$ is composed only of arcs with bounds $(0,\overline{z}_e)$, while $\delta^-(W)=[\overline{W},W]$ is composed of arcs with bounds $(0,\overline{z}_e)$ and $(\overline{x}_{ij})$. Also, if $i_0\in W$, then $\delta^-(W)$ may contain the extra arc (v,i_0), with bounds $(0,\infty)$. Thus, we have that

$$\sum_{(a,b)\in[W,\overline{W}]}U_{ab}=\sum_{ab\in[W,\overline{W}]}\overline{z}_{ab}$$

and

$$\sum_{(a,b)\in[\overline{W},W]}L_{ab}=\sum_{i\in W}\overline{x}_{ij}.$$

Since $(\overline{x},\overline{y},\overline{z})$ satisfies all the cut constraints, then we have that

$$\overline{z}(\delta(W))=\sum_{ab\in[W,\overline{W}]}\overline{z}_{ab}\geqslant\sum_{i\in W}\overline{x}_{ij}.$$

Thus, we get

$$\sum_{(a,b)\in[W,\overline{W}]}U_{ab}\geqslant\sum_{(a,b)\in[\overline{W},W]}L_{ab}. \tag{A.6}$$

Therefore, Eqs. (A.5) and (A.6) implies that, for all $W\subseteq V$, the cut condition holds, and hence, there is a feasible flow between i_0 and v in G_j.

Now suppose that there exist a cut constraint, induced by a node set W and a terminal $j\in P$, and violated by $(\overline{x},\overline{y},\overline{z})$. We will show that the flow with lower and upper bounds between some node $i_0\in V\setminus\{v\}$ and v is not feasible. Let i_0 be a node of W. As W induce a valid cut constraint for the problem, we have that $v\notin W$ and

$$\sum_{ab\in[W,\overline{W}]_G}\overline{z}_{ab}<\sum_{i\in W}\overline{x}_{ij}. \tag{A.7}$$

Since $v\notin W$, the cut corresponding to W in the graph G_j satisfies

$$\sum_{(a,b)\in[W,\overline{W}]_{G_j}}U_{ab}=\sum_{ab\in[W,\overline{W}]_G}\overline{z}_{ab}$$

and

$$\sum_{(a,b)\in[\overline{W},W]_{G_j}}L_{ab}=\sum_{i\in W}\overline{z}_{ab}.$$

With inequality Eq. (A.7), we get

$$\sum_{(a,b)\in[W,\overline{W}]_{G_j}}U_{ab}<\sum_{(a,b)\in[\overline{W},W]_{G_j}}L_{ab},$$

implying that the cut conditions induced by W is not satisfied and the flow problem with lower and upper bounds is not feasible. □

References

[1] A. Gupta, J. Kleinburg, A. Kumar, R. Rastogi, B. Yener, Provisioning a virtual private network: a network design problem for multicommodity flow, in: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), 2001.

[2] A. Kumar, R. Rastogi, A. Silberschatz, B. Yener, Algorithms for provisioning virtual private networks in the hose model, IEEE/ACM Transactions on Networking 10 (4) (2002) 565–578.

[3] Giuseppe F. Italiano, Stefano Leonardi, Gianpaolo Oriolo, Design of networks in the hose model, in: Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE), 2002, pp. 65–76.

[4] T. Erlebach, M. Ruegg, Optimal bandwidth reservation in hose model VPNs with multi-path routing, in: IEEE Infocom, 2004.

[5] Haibo Wang, Gee-Swee Poo, Availability guaranteed service provisioning in hose model VPNs with multi-path routing, in: ICC '06 3, 2006, pp. 1014–1019.

[6] Guy Even, Moti Medina, Gregor Schaffrath, Stefan Schmid, Competitive and deterministic embeddings of virtual networks, in: ICDCN, 2012, pp. 06–121.

[7] M. Chowdhury, M.R. Rahman, R. Boutaba, ViNEYard: virtual network embedding algorithms with coordinated node and link mapping, in: ACM/IEEE ToN, 2011.

[8] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, S. Shenker, Extending networking into the virtualization layer, in: Proc. 8th ACM Workshop on Hot Topics in Networks (HotNets), 2009.

[9] M. Ghobadi, S. Ganti,G.C. Shoja, Hierarchical provisioning algorithm for virtual private networks using the hose model, in: Global Telecommunications Conference, 2007, pp. 2467–2471.

[10] M. Ghobadi, S. Ganti, G.C. Shoja, Resource optimization to provision a virtual private network using the hose model, in: Proceedings of the IEEE International Conference on Communications, 2007, pp. 512–517.

[11] Yu-Liang Liu, Yu-Ting Chin, Traffic engineering for provisioning VPNs with time-varying bandwidth requirements, in: Electronics and Information Engineering (ICEIE), 2010 International Conference, vol. 2, 2010, pp. 309–313.

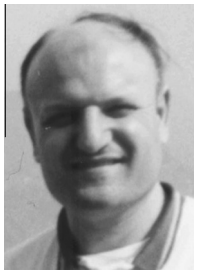
[12] Dong Wang, Yunfeng Peng, Keping Long, A novel iterative clustering steiner tree algorithm for optimal resource reservation in hose based VPN, Optical Internet (COIN) (2010) 1–3.

[13] B. Thabti, H. Youssef, A. Meddeb, A.R. Mahjoub, Evolutionary algorithm for provisioning VPN trees based on pipe and hose workload models, Natural Computation (ICNC) 4 (2011) 2058–2064.

[14] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory Algorithms and Applications, Prentice Hall, New Jersey, 1993.

[15] A.L. Barabasi, R. Albert, Emergence of scaling in random networks, Science (1999) 509–512.
[16] COIN-OR. <www.coin-or.org>.
[17] N.G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K.K. Ramakrishnan, Jacobus E. van der Merwe, A flexible model for resource management in virtual private networks, in: Proceedings of ACM SIGCOMM, 1999.
[18] B. Korte, J. Vygen, Combinatorial Optimization: Theory and Algorithms, third ed., Springer, 2006.
[19] A.R. Mahjoub, Polyhedral approaches, in: V. Paschos (Ed.), Concepts of Combinatorial Optimization, ISTE-WIELY, 2010, pp. 261–324.
[20] A. Medina, A. Lakhina, I. Matta, J. Byers, BRITE: an approach to universal topology generation, in: IEEE/ACM MASCOTS, 2001, pp. 346–356.
[21] C. Swamy, A. Kumar, Primal–dual algorithms for connected facility location problems, in: Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization, 2002.
[22] B.M. Waxman, Routing of multipoint connections, IEEE Journal on Selected Areas in Communications 6 (9) (1988) 1617–1622.
[23] L. A. Wolsey, Integer Programming, Wiley-Interscience, first ed., 1998.
[24] F. Eisenbrand, E. Happ, Provisioning a virtual private network under the presence of non-communicating groups, in: Proceedings of the 6th International Conference on Algorithms and Complexity, Lecture Notes in Computer Science, vol. 3998, 2006, pp. 105–114.
[25] J. Chu, C.-T. Lea, Optimal link weights for IP-based networks supporting hose-model VPNs, IEEE/ACM Transactions on Networking 17 (3) (2009) 778–788.

Ali Lourimi was born in Sousse, Tunisia, in 1978. He received the bachelor's degree in applied mathematics from the University of Monastir, Tunisia, in 2000 and master's degree in Telecommunications from the Higher School of Communications of Tunis, Tunisia, in 2002. In 2006, he joined the PRINCE Research Unit, University of Sousse, Tunisia. His current research interests include VPN design and Combinatorial Optimization techniques. He is actually an assistant in the University of Gafsa.



A. Ridha Mahjoub is a Professor of Operation Research and Combinatorial Optimisation at the LAMSADE laboratory of the University of Paris-Dauphine. Prior to that, he was a Professor at the University of Brest, France, from 1991 to 1998, then at the University of Clermont, France, from 1998 to 2007. Professor Mahjoub earned a Maîtrise in mathematics from the Faculty of Sciences of the University of Tunis, a Doctorat 3éme Cycle and a Doctorat d'Etat, both from the University of Grenoble, France. His research interests are related to the theory and applications of polyhedral approaches for hard combinatorial optimisation problems, linear programming and integer programming approaches, graph theory and the complexity of algorithms. His research works have addressed numerous combinatorial problems such as the maximum cut problem or bipartite subgraph problem. Some of his recent work has focused on the development of efficient branching cut algorithms for network design problems. Professor Mahjoub has published expensively in top rated journals like Mathematical Programming, Mathematics of Operations Research, SIAM Journal on Discrete Mathematics, Discrete Mathematics, Discrete Applied Mathematics, Operations Research Letters, Networks, Discrete Optimization. Further, Professor Mahjoub has held several positions. He is curently the Co-Director of Département de Formation et de Recherche de Mathématiques et Informatique at Dauphine. He is also the Editor of the journal RAIRO-Operations Research.

Dr. Ibrahima Diarrassouba recieves his Ph.D. in Operations Research from Blaise Pascal University of Clermont-Ferrand, France. He is also graduated in Computer Engineering from ISIMA Engineering School. Actually, He is Assistant Professor at University of Le Havre and works in the field of Combinatorial Optimization and especially in Network Design and in Logistics and Transportation.

Dr. Habib Youssef received a Diplôme d'Ingénieur en Informatique from the Faculté des Sciences de Tunis, University of El-Manar, Tunisia in June 1982 and a Ph.D. in computer science from the University of Minnesota, USA, in January 1990. From September 1990 to January 2001 he was a Faculty member of the computer engineering department of King Fahd University of Petroleum & Minerals (KFUPM), Saudi Arabia (Assistant Professor from 1990 to 1995 and Associate Professor from September 1995 to January 2001). From February 2001 to June 2002, he was a Maitre de Conférences en informatique at the Faculté des Sciences de Monastir (FSM), University of Monastir, Tunisia. From July 2002 to August 2005, he served as the Director of the Institut Supérieur d'Informatique et Mathematiques of the University of Monastir, and from July 2007 to August 2011 as the Director of the Institut Supérieur d'Informatique et des Technolgies de Communication of the University of Sousse, where he is currently serving as a Professor of computer science.

He has over 160 publications to his credit in the form of books, book chapters, and journal and conference papers. He is the author with S. Sait of two books, (1) "VLSI Physical Design Automation: Theory and Practice", McGraw-Hill 1995, (also co-published by IEEE Press 1995), and reprinted with corrections by World Scientific in 1999, and (2) "Iterative Computer Algorithms with Applications in Engineering", IEEE CS Press 1999, and since 2003 published by John Wiley & Sons, which has also been translated into Japanese. His current research interests are computer networks, performance evaluation of computer systems, and combinatorial optimization.