Very Large-Scale Neighborhood Search Algorithms for the Design of Service Overlay Networks

Jocelyne Elias[*], Fabio Martignon[†], Giuliana Carello[*]

Abstract

Service Overlay Networks (SONs) allow virtual operators to create and deploy value-added Internet services with Quality of Service guarantees, while leaving the underlying network infrastructure unchanged. The deployment of a SON can be very expensive, and hence its planning requires careful decisions, including the overlay nodes' placement and the capacity provisioning of the access links that connect the end-users to the SON infrastructure.

In this work we first propose a novel Integer Linear Programming (ILP) model for the overlay network design problem which selects the optimal number and position of overlay nodes, the capacity reserved on each overlay link, as well as the optimal routing of the incoming traffic demands.

Since such model can be solved to the optimum only for small network instances, we further propose an efficient and novel *tabu search* based heuristic for the planning of SONs that combines polynomial size and very large-scale neighborhoods. The very large-scale neighborhood of the solution given by tabu search is explored efficiently to obtain in a short time a new one that is both far from the current solution and cost-decreasing.

We provide numerical results of the proposed heuristic on a set of realistic, large-size instances, including real ISP topologies, and discuss the effect of different parameters on the characteristics of the planned networks. Furthermore, we compare such results with the bound obtained solving our ILP model in small network scenarios. We show that in the considered network topologies the proposed heuristic performs very close to the optimum with a short computation time, thus providing a promising framework for the design of SONs.

Keywords: - Service Overlay Networks, Network Design, Optimization, Heuristic, Very Large-Scale Neighborhood, Tabu Search.

1 Introduction

The extraordinary success of the Internet has brought us fresh new challenges. Many applications have emerged with Quality of Service (QoS) requirements that are very different from those for which the Internet was originally designed.

Service Overlay Networks (SONs) have been recently introduced as an effective way to provide end-to-end QoS guarantees without any changes in the underlying Internet structure [1, 2, 3, 4, 5, 6].

A SON is an application-layer network built on top of the traditional IP-layer networks. In general, the SON is operated by a third-party ISP that owns a set of overlay nodes residing in the underlying ISP domains. These overlay nodes perform service-specific data forwarding and control functions, and are interconnected by virtual overlay links which correspond to one or more IP-layer links [1].

The service overlay architecture is based on business relationships between the SON, the underlying ISPs, and the users. The SON establishes bilateral service level agreements with the individual underlying ISPs to install overlay nodes and purchase the bandwidth needed for serving its users. On the other hand, the users subscribe to SON services, which will be guaranteed regardless of how many IP domains are crossed by the users' connection. The SON gains from users' subscriptions. Although the quality requirements that a SON must satisfy may be different (e.g., bandwidth, delay, delay jitter, packet loss), we assume they are mapped to an equivalent bandwidth [1, 6]. To assure the bandwidth for the SON, the underlying ISPs have several technical options: they can lease a transmission line to the SON, use

[*]Jocelyne Elias and Giuliana Carello are with the Department of Electronics and Information, Politecnico di Milano, e-mail: {elias, carello}@elet.polimi.it

[†]Fabio Martignon is with the Department of Information Technology and Mathematical Methods, University of Bergamo, Dalmine (BG) 24044, Italy. E-mail: fabio.martignon@unibg.it

bandwidth reservation mechanisms or create a separate Label Switched Path if MPLS [7] is available in their networks.

Obviously, the deployment of Service Overlay Networks can be a capital-intensive investment. It is therefore imperative to develop network design tools that consider the cost recovery issue for a SON. The main costs of SON deployment include the overlay nodes installation cost and the cost of the bandwidth that the SON must purchase from the underlying network domains to support its services.

The topology design problem for Service Overlay Networks is considered by very few works [6, 8, 9, 10] which make several limiting assumptions: (1) the number and location of overlay nodes are predetermined, while the overlay nodes placement is a critical issue in the deployment of the SON architecture, (2) the capacities of overlay nodes are unlimited, thus assuming that the underlying ISPs will always be able to provide bandwidth to the SON, and (3) only small network instances are considered, with a limited number of connections and overlay nodes.

In this paper we overcome these limitations by first proposing a novel Integer Linear Programming model for the overlay network design problem which selects the optimal number and position of overlay nodes, the capacity reserved on each overlay link as well as the optimal routing of incoming traffic demands.

Since such model can be solved to the optimum only for small network instances, a practical approach is to employ heuristic algorithms that can find good solutions within a reasonable amount of computation time.

Neighborhood search algorithms are a wide class of improvement algorithms that find at each iteration an improving solution by searching the "neighborhood" of the current solution. The larger the neighborhood, the better is the quality of locally optimal solutions. At the same time, the larger the neighborhood, the longer it takes to search it at each iteration. For this reason, a larger neighborhood does not necessarily produce a more effective heuristic unless one can search it in a very efficient manner. A very large-scale neighborhood search is a technique in which the size of the neighborhood is "very large" with respect to the size of the input data.

To solve very large instances of the SON design problem, we further propose a novel and efficient tabu search based approach that uses polynomial size and very large-scale neighborhoods. A very large-scale neighborhood search is used to improve the best solution found by tabu search, widening the set of explored solutions; it can therefore be seen as a diversification step for the tabu search. Our approach is novel in that it combines tabu search and very large-scale neighborhood search, which represents in itself a contribution of our paper for both the network design and the Hub Location problem (which is a special case of our problem).

We provide numerical results for a set of realistic and very large-size instances, including real ISP topologies, and investigate the impact of different parameters on the SON design problem, such as number and installation cost of overlay nodes, bandwidth costs and traffic demands. The numerical results we gathered show that our heuristic obtains good solutions even for large-scale instances in a short computation time. Furthermore, in small-size network scenarios the proposed heuristic performs very close to the optimal solution provided by the integer model.

The main contributions of this paper can therefore be summarized as follows:

- a novel Integer Linear Programming model for the Service Overlay Network design problem that jointly optimizes the overlay topology as well as the traffic routing.
- A novel and efficient heuristic that obtains good solutions for very large-scale instances in a reasonable computation time; such heuristic is based on a new approach that combines tabu search and a very large-scale neighborhood search.
- An extensive performance evaluation of the proposed network design framework in several realistic scenarios, including real ISP topologies.

The paper is structured as follows: Section 2 discusses related work, while Section 3 presents the SON design model. Section 4 defines the polynomial size and very large-scale neighborhoods considered in this work, as well as the methods used to search them efficiently. Section 5 introduces the proposed heuristic that finds good solutions to the SON design problem. Section 6 presents numerical results that show the effect of different parameters on the characteristics of the planned network. Finally, Section 7 concludes this paper.

2 Related Work

The network design problem has been considered in different contexts, such as wired backbone networks [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21], wireless networks [22, 23] and recently Service Overlay Networks [6, 8, 9, 10, 24, 25, 26, 27, 28]. The problem consists in deciding the number and position of the network nodes, the links between such nodes and the capacity installed on each link, as well as the traffic routing.

The overlay node placement problem is addressed in [26, 27, 28]. In [26] the authors consider how to place service nodes optimally in a network, balancing the need to minimize the number of nodes while limiting the distance between users and service nodes. This work, however, only proposes optimization algorithms for the uncapacitated version of the coverage problem. The work in [27] focuses on designing an overlay network that maximizes the number of unicast and multicast connections with deterministic delay requirements, without considering link costs. Finally, the overlay node placement problem is investigated in [28] to improve routing reliability and TCP performance. This paper, however, assumes that overlay nodes have infinite capacities, and does not take into account the costs involved in the deployment of the overlay network.

A similar problem is considered in [8], where end-systems and overlay nodes are connected through ISPs that support bandwidth reservations; large-size network topologies are solved using simulated annealing. However, this work formulates the design problem assuming that locations of overlay nodes are given and there are no capacity constraints on overlay nodes.

The work in [29] considers a generalized cost model in the formulation of the design problem, and provides both exact and approximate solutions.

A generalized framework for the SON design problem is proposed in [30] where different characteristics are considered: 1) single-homed/multi-homed end-systems 2) usage-based/fixed cost model and 3) capacitated/uncapacitated networks. Optimal and approximation algorithms are also provided.

Dynamic topology design frameworks for SONs are introduced in [6, 9, 10, 24, 25].

In [6], an adaptive framework for the design of overlay networks is presented to ensure inter-domain QoS, and several heuristics are proposed to solve the least-cost topology design problem. Another set of heuristics for SON design is proposed in [9]; these heuristics aim to construct an overlay topology maintaining the connectivity between overlay nodes under various IP-layer path failure scenarios.

Reference [24] deals with dynamic topology construction to adapt to the underlying network topology changes. An architecture for topology-aware overlay networks is proposed to enhance the availability and performance of end-to-end applications by exploring the dependency between overlay paths. Several clustering-based heuristics for overlay node placement and a routing mechanism are also introduced.

The dynamic overlay network reconfiguration issue is addressed in [25], where the main goal is to find the optimal reconfiguration policies that can both accommodate time-varying communication requirements and minimize the total overlay network cost.

Two heuristics, based on the linear relaxation of an ILP model and randomized rounding, are proposed in [10] for the SON design problem. However, such algorithms solve only small and medium-size instances, since optimizing large-scale SON topologies with such approach is too cumbersome both from the computational and memory occupation point of view.

The problem of designing survivable communication networks (i.e., networks that are still functional after the failure of certain network components) is investigated in [15], where the authors introduce a general model that includes the minimum spanning tree, Steiner tree and minimum cost k-connected network design problems, as well as heuristics that work well with such problems.

Bley et al. [17, 18] study the network design problem with survivability constraints, and present mixed-integer programming approaches for traffic engineering and network design problems with non-bifurcated shortest path routing and a modular link and node hardware model. Furthermore, both a mathematical optimization model and a Lagrangian relaxation approach are presented in [19] for the integrated network design and routing problem in IP networks. Such problem consists in deciding simultaneously the network's topology, node and link capacities, along with the routing weights which determine the traffic flows. The objective is to minimize the overall network cost.

The network design problem is an attractive candidate for integer programming decomposition due to its embedded network flow structure. Benders decomposition for the uncapacitated network design problem (called also *fixed-charge* problem) has been studied in [16], and its potential flexibility is also illustrated therein. In particular, the authors illustrate their computational experience in using Benders decomposition with a dual ascent and variable elimination preprocessing procedure to solve the uncapacitated network design problem with several binary and continuous variables.

A family of dual-ascent algorithms for the uncapacitated network design problem is developed in [20]. In particular, Balakrishnan et al. [20] present a dual-ascent procedure which, in conjunction with an add-drop heuristic, is capable of efficiently solving realistic-size instances of less-than-truckload consolidation problems. The authors demonstrate the computational advantages of using the more disaggregate multicommodity flow formulation for the uncapacitated network design problem.

In summary, the most recent works dealing with the topology design problem in telecommunication networks [6, 8, 9, 10, 11, 12, 13, 14, 22, 23, 24, 25, 26, 27, 28] are less general than our current work since they consider at least one of the following special cases: 1) the number and location of overlay nodes are pre-determined, 2) there are no capacity constraints on overlay nodes, and 3) only small and medium-size topologies are considered, with a limited number of connections and overlay nodes. In this work, on the contrary, we take into consideration all these issues in the formulation of the overlay network design problem.

Furthermore, none of the above works uses very large-scale neighborhood search techniques to solve the SON design problem. A survey of such techniques can be found in [31, 32].

3 Service Overlay Network Design Model

In this section, we introduce a mathematical formulation of the Minimum Cost SON Design (MCSD) problem that jointly optimizes the users' traffic routing and the overlay network topology, in terms of the number and location of the overlay nodes, as well as of the capacity reserved on each overlay link.

A common approach to such problem is to consider feasible positions of traffic concentration points in the service area (Test Points, TPs), which exchange traffic among themselves, and feasible positions where overlay nodes can be installed (Candidate Sites, CSs) [11]. Each TP can represent a single network user or a centroid of a discrete/continuous traffic distribution, according to the planning scenario. The placement of TPs depends on the expected traffic distribution, while the placement of CSs depends on the underlying network topology and the agreements of the SON operator with the ISPs. Although the concept of *test point* is distinguished from *end-user* (formally, the end-user is the traffic generation agent that is placed in a TP), we use the two terms as synonyms throughout the paper.

Let $S = \{1, 2 \ldots, m\}$ denote the set of CSs and $I = \{1, 2 \ldots, n\}$ the set of TPs.

The cost associated with installing an overlay node in CS j is denoted by c^I_j; c^B_{jl} denotes the cost for the SON operator to buy one bandwidth unit between CSs j and l from the underlying ISPs, and c^A_{ij} is the access cost per bandwidth unit required between TP i and CS j; finally, c^E_{jk} represents the cost per bandwidth unit for the traffic transmitted on the egress link between CS j and TP $k \in I$. An egress link is a link that connects an overlay node j to a test point k, carrying a positive amount of flow destined for k, i.e. traffic that is leaving the SON.

The traffic generated by TP i towards TP k is given by the parameter w_{ik}, $i, k \in I$. In the following we assume that all the quality parameters requested by incoming traffic can be controlled by defining an equivalent flow bandwidth requirement as discussed in [33, 34]. This assumption allows us to focus only on bandwidth constraints.

The maximum access capacity of overlay node j is denoted by Γ_j, which restricts the volume of the total traffic flow incoming from the test points that are assigned to such overlay node. To simplify the notation, we define $o_i = \sum_{k \in I} w_{ik}$ and $d_i = \sum_{k \in I} w_{ki}$, which represent, respectively, the total traffic originating from and destined for TP i.

According to the geographic location of TPs and CSs, and the underlying physical topology, the following connectivity parameters can be calculated.

Let a_{ij}, $i \in I, j \in S$ denote the test point coverage parameters:

$$a_{ij} = \begin{cases} 1 & \text{if TP } i \text{ can access the SON through an overlay node installed in CS } j \\ 0 & \text{otherwise} \end{cases}$$

Obviously, a_{ij} depends on the proximity of TP i to CS j, that is on the access coverage provided by the SON operator with CS j through agreements with local network operators.

Moreover, let b_{jl}, $j, l \in S$ denote the connectivity parameters between two different CSs, which may depend on the proximity of the overlay nodes j and l in the underlay network, as well as on the agreements between the SON and the different ISPs:

$$b_{jl} = \begin{cases} 1 & \text{if CSs } j \text{ and } l \text{ can be connected with an overlay link} \\ 0 & \text{otherwise} \end{cases}$$

Decision variables of the problem include overlay nodes' installation variables z_j, $j \in S$:

$$z_j = \begin{cases} 1 & \text{if an overlay node is installed in CS } j \\ 0 & \text{otherwise} \end{cases}$$

TP assignment variables x_{ij}, $i \in I, j \in S$:

$$x_{ij} = \begin{cases} 1 & \text{if TP } i \text{ is assigned to an overlay node installed in CS } j \\ 0 & \text{otherwise} \end{cases}$$

and finally flow variables f^i_{jl}, which are introduced only if $b_{jl} = 1$, and denote the traffic flow originating from TP $i \in I$ and routed on link (j, l).

Given the above parameters and variables, the Minimum Cost SON Design (MCSD) problem can be formulated as follows:

$$\min\left\{ \sum_{j\in S} c^I_j z_j + \sum_{i\in I}\sum_{j\in S}(o_i c^A_{ij} + d_i c^E_{ji})x_{ij} + \sum_{i\in I}\sum_{j\in S}\sum_{l\in S} c^B_{jl} f^i_{jl} \right\} \tag{1}$$

$$\text{s.t.} \sum_{j\in S} x_{ij} = 1 \qquad \forall i \in I \tag{2}$$

$$x_{ij} \le z_j a_{ij} \qquad \forall i \in I, j \in S \tag{3}$$

$$\sum_{i\in I} o_i x_{ij} \le \Gamma_j z_j \qquad \forall j \in S \tag{4}$$

$$\sum_{l\in S} f^i_{jl} = \sum_{l\in S} f^i_{lj} + \left(o_i x_{ij} - \sum_{k\in I} w_{ik} x_{kj} \right) \qquad \forall i \in I, j \in S \tag{5}$$

$$z_j, x_{ij} \in \{0, 1\} \qquad \forall i \in I, j \in S \tag{6}$$

$$f^i_{jl} \ge 0 \qquad \forall i \in I, j, l \in S \tag{7}$$

The objective function (1) imposes the minimization of the sum of the nodes' installation costs and the flow transportation costs (access, egress and transport costs). Constraints (2) and (3) dictate that each test point is assigned to one overlay node subject to the test point coverage parameters. Constraints (4) require that the access capacity of each overlay node is not violated. Constraints (5) are the flow conservation equations at the overlay nodes; more specifically, the left-hand side of constraints (5), $\sum_{l\in S} f^i_{jl}$, represents the total traffic originated at TP i going out of CS j, while the right-hand side represents the total traffic entering in CS j, and it is composed of two terms: $\sum_{l\in S} f^i_{lj}$ is the total incoming traffic (originated at TP i) that arrives from other CSs, while the second term, $o_i x_{ij} - \sum_{k\in I} w_{ik} x_{kj}$, is the total traffic injected in CS j by the i-th TP. This latter is computed by subtracting from the total traffic originated in i (o_i) all the *local* traffic, $\sum_{k\in I} w_{ik} x_{kj}$, i.e., all traffic that is destined for all TPs k that are connected to the same CS j. Finally, constraints (6) and (7) are integrality and non-negative constraints on the problem variables.

Note that capacity over-provisioning is one of the most widely accepted dimensioning criterion for providing QoS in the Internet backbone. For this reason, we do not consider capacity bounds for the SON backbone links since we assume that enough capacity is always available between overlay nodes. However, it is easy to extend the MCSD model taking into account also overlay link capacities, simply introducing the following constraints:

$$f^i_{jl} \le b_{jl} U_{jl}, \quad \forall i \in I, j, l \in S \tag{8}$$

where U_{jl} represents the capacity of overlay link (j, l).

Finally, we observe that the above model is NP-hard since it includes the Hub Location problem as special case (see Hamacher et al. [35]).

In the MCSD model we consider bifurcated routing since we assume that test points represent traffic concentration points, as explained before, which correspond to aggregates of single network users. Therefore, the traffic aggregate originating from a TP can be split by the SON operator over multiple paths without impacting on the performance of single users' connections.

4 Neighborhoods Definition

The MCSD model defined in Section 3 can be solved in a reasonable computation time only for small network instances, as we will show in the numerical results section. Furthermore, the overlay traffic pattern is quite fluctuating and changes on relatively short timescales during the operation of an overlay network [6, 36]. Hence, there is a need for redesigning quite frequently the virtual topology of the SON, rerouting the traffic accordingly [6]. Therefore, in the following we describe the proposed heuristic for solving the Minimum Cost SON Design problem (named TS-MCSD), which is based on a tabu search approach.

In TS-MCSD, the SON design problem is split into two subproblems: a *location problem*, in which it is decided where the overlay nodes should be installed, and an *allocation problem*, in which it is determined to which overlay node each test point should be allocated. The rationale of this approach is that a neighborhood search both on the location and the allocation problem would be too expensive in terms of computational effort.

Since TS-MCSD is based on the concept of search in the neighborhood of a given solution, in this section we define the neighborhoods considered in our work and the techniques used to search them efficiently. Then we proceed by illustrating in detail the TS-MCSD heuristic (Section 5).

We consider two types of neighborhoods: polynomial size and very large-scale neighborhoods. The first neighborhood is applied both to the *location* and to the *allocation problem*. The very large-scale neighborhood is applied only to the *allocation problem*, as we will explain later in this section.

4.1 Polynomial Size Neighborhoods

We define two polynomial size neighborhoods: the first is considered for the *location problem*, while the second for the *allocation problem*.

4.1.1 Polynomial Size Location Neighborhood (PSLN)

For the location problem, given a set of installed overlay nodes N, the neighborhood is generated by applying three moves:

- $m1$: remove one overlay node from N, i.e. close one overlay node;
- $m2$: add a candidate overlay node to N, i.e. install a new overlay node;
- $m3$: swap an overlay node in the set N with an overlay node which is in the candidate site set S but not in N.

The size of the neighborhood corresponding to applying $m1$, $m2$ and $m3$ is $O(|S|^2)$.

Note that when an overlay node is removed, or a swap is performed, it can be necessary to re-define the allocation of TPs to overlay nodes. Such allocation is performed by means of the algorithms detailed in Section 5.2.1.

4.1.2 Polynomial Size Allocation Neighborhood (PSAN)

For the allocation problem, the set of installed overlay nodes is fixed and the solution is represented by the allocation of each test point to an overlay node. The neighborhood is generated by applying two moves:

- $a1$: allocate a TP i to an overlay node j different from the one to which it is allocated in the current solution;
- $a2$: swap the overlay nodes to which any two TPs are allocated; for example, supposing that in the current solution TP i is allocated to overlay node j and TP k is allocated to overlay node l, in the neighboring solution, i is allocated to l and k to j.

The size of this neighborhood is $O(|I|^2)$. Note that we accept only swap moves that lead to an improvement in the sum of the allocation costs (i.e., the access and egress costs).

4.2 Very Large-Scale Neighborhood (VLSN) for the Allocation Problem

We now define the *Cyclic Exchange Neighborhood* (CEN) [37, 38], which is the particular type of Very Large-Scale Neighborhood used in our work. CEN is used to perform cyclic exchanges of test points among a fixed set of overlay nodes.

We first introduce some notation, then we describe how to build the *test point improvement graph* and how to explore the *single test point cyclic exchange neighborhood.*

4.2.1 Basic Notation for VLSN

Let F be a feasible solution to the MCSD problem (1)-(7). The set F can be represented as a partition of the test point set I into m subsets, i.e., $F = \{I_1, I_2, ..., I_m\}$, where each subset $I_j = \{i \in I : x_{ij} = 1\}$, corresponds to a potential overlay node and contains the set of TPs assigned to that overlay node in the solution F. Eventually, I_j can be empty if no overlay node is installed at candidate site j.

The cost of each subset I_j, in terms of the installation and allocation costs (i.e., the access and egress costs), is given by

$$C(I_j) = \begin{cases} c^I_j + \sum_{i \in I_j} (o_i c^A_{ij} + d_i c^E_{ji}) & \text{if } I_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

The cost of the whole partition F is therefore

$$C(F) = \sum_{j=1}^{m} C(I_j).$$

A solution F is feasible if the following inequality holds for $j = 1, ..., m$:

$$O(I_j) = \sum_{i \in I_j} o_i \leq \Gamma_j.$$

The residual capacity r_j of each subset I_j, i.e., the capacity still available at an overlay node installed at j, is defined as:

$$r_j = \Gamma_j - O(I_j) \geq 0.$$

In the following, we denote by $H(i)$, $i \in I$, the overlay node serving TP i in the current solution, which we refer to as *host* node.

4.2.2 Test Point Exchanges

Given a solution F to the SON design problem, the *single test point cyclic exchange neighborhood* of F is defined as the set of new solutions obtainable from F by exchanging test points among overlay nodes in a cyclic manner.

Such a neighborhood is constructed by moving among the overlay nodes single TPs conveniently chosen. The single test point neighborhood is defined in terms of single test point cyclic exchanges as follows.

A single test point cyclic exchange with respect to a solution F is defined as a TP sequence $R = (i_1, i_2, ..., i_q)$, such that each TP i_r in the sequence is assigned to a different overlay node in F, i.e., $H(i_r) \neq H(i_s)$ for $r \neq s$, $r, s = 1, ..., q$. The cyclic exchange represents therefore the following exchange of TPs: TP i_1 is relinquished from overlay node $H(i_1)$ and assigned to overlay node $H(i_2)$, TP i_2 is relinquished from overlay node $H(i_2)$ and assigned to overlay node $H(i_3)$, and so on until TP i_q is relinquished from overlay node $H(i_q)$ and assigned to overlay node $H(i_1)$.

A cyclic exchange R is feasible if the capacity constraint of each overlay node involved in the cycle is still respected after the exchange, i.e., if $o_{i_{r-1}} \leq r_{H(i_r)} + o_{i_r}$ for $r = 2, ..., q$, and $o_{i_q} \leq r_{H(i_1)} + o_{i_1}$.

A cyclic exchange R is profitable, with respect to the total allocation cost, if the new solution F' obtained from F through R is such that $C(F') < C(F)$.

Note that the test point cyclic exchange R can be profitable for the allocation problem but not for the MCSD problem as a whole since this latter includes further the inter-overlay node bandwidth costs. Hence, once a profitable cyclic exchange R is found, the total cost of the new solution must be compared with the cost of the current solution to decide whether the cyclic exchange R is really profitable or not for the SON design problem.

Each feasible single test point cyclic exchange generates a new feasible solution F' from the current solution F. The set of all the new solutions obtainable from F through a feasible exchange R defines the single test point cyclic exchange neighborhood of F.

In order to efficiently detect improving single test point moves, such a neighborhood is searched only partially using a heuristic approach which detects special negative cost cycles in a suitably defined graph, called the *test point improvement graph* [37, 38]. The improvement graph along with negative subset-disjoint cycles are described in the following subsections.

4.2.3 The Test Point Improvement Graph

Given a solution F of the MCSD problem, the *test point improvement graph* relative to F is a directed graph $G(F) = (N(F), A(F))$ defined as follows. The set of nodes $N(F)$ contains a node, i, for each TP $i \in I$.

The set of arcs $A(F)$ contains an arc (i, k) for each pair of nodes i and k in $N(F)$, provided that test points i and k are assigned to two different overlay nodes in F, i.e., $H(i) \neq H(k)$, and that after the insertion of i in $I_{H(k)}$ and the removal of k from it, the capacity of overlay node $H(k)$ is not exceeded, i.e., $o_i - o_k \leq r_{H(k)}$. An arc connecting two nodes i and k, in fact, signifies that TP i leaves overlay node $H(i)$ and is assigned to overlay node $H(k)$, which in turn relinquishes TP k.

The cost α_{ik} of the arc (i, k) reflects the allocation cost variation of overlay node $H(k)$ and it is therefore defined as:

$$\alpha_{ik} = C_{iH(k)} - C_{kH(k)},$$

where $C_{iH(k)} = o_i c^A_{iH(k)} + d_i c^E_{H(k)i}$.

4.2.4 Negative subset-disjoint cycles

Having provided a description of the test point improvement graph, we can now define negative subset-disjoint cycles in it. A directed cycle $(n_1, ..., n_q)$, with $n_r \in N(F), r = 1, ..., q$, in the improvement graph $G(F)$ associated with solution F, is a negative subset-disjoint cycle if each one of its nodes is associated with a different overlay node and if the sum of the costs of its arcs is negative. It can be shown that the problem of detecting a negative subset-disjoint cycle in $G(F)$ is equivalent to the problem of finding a feasible profitable multi-exchange for F [39].

There is a one-to-one correspondence between the set of feasible cyclic exchanges relative to the current solution F and the set of subset-disjoint cycles in $G(F)$. Each cyclic exchange is profitable if and only if the corresponding subset-disjoint cycle is negative. In particular, the fact that the test point improvement graph includes only feasible arcs with respect to the capacity constraints ensures that each exchange corresponding to a cycle in $G(F)$ is a feasible exchange.

Note that the way we define the costs of the arcs in $A(F)$, considering only test points' allocation costs (access and egress costs), implies that the cost of each cycle in $G(F)$ reflects only the allocation cost variation of the solution due to the corresponding test point exchanges. Hence, if the cycle cost is negative, the associated exchange is said to be profitable for the SON design problem if and only if the total cost of the corresponding solution, including installation costs, allocation (access/egress) costs and transfer costs, is lower than the cost of the current solution.

4.2.5 Search for negative subset-disjoint cycles

As previously stated, the problem of finding an improving move in the single test point neighborhood can be reformulated as the problem of identifying a negative-cost subset-disjoint cycle in $G(F)$. This problem is known to be NP-complete [39].

A heuristic algorithm has been proposed in [38] to detect negative-cost subset-disjoint cycles for the Capacitated Minimum Spanning Tree problem and such algorithm is then used in the Facility Location problem for the same goal [37]. The algorithm is reviewed in the following, and it is based on a modification of the label-correcting algorithm for the shortest path problem where changes are introduced to check for path subset-disjointness.

A label-correcting algorithm determines a shortest path from a specified node s to every other node in the network with arc costs given by α_{ij}.

Two indices are maintained for each node j: $d(j)$, the distance label of node j, and $pred(j)$, the predecessor index of node j. The distance label $d(j)$ is either ∞, indicating that the algorithm has yet to discover a directed path from s to j, or it is the length of some directed path from the origin to node j. The predecessor index, $pred(j)$, records the node prior to node j in the current directed path of length $d(j)$. Let $P[j]$ denote the current directed path from s to j. The optimality conditions for the shortest path problem require that $d(j) \leq d(i) + \alpha_{ij}$ for each arc (i, j) in the network.

The label correcting algorithm starts with node s as the starting node, and sets $d(s) = 0$ and $d(j) = \infty$ for all $i \in N \setminus \{s\}$. The main idea behind such algorithm is to identify an arc (i, j) violating its optimality condition, that is, $d(j) > d(i) + \alpha_{ij}$, and decrease the distance label $d(j)$ to $d(i) + \alpha_{ij}$; this step is called the *distance update* step.

The algorithm repeatedly performs distance update steps and terminates when all the arcs satisfy their optimality conditions. To efficiently identify an arc (i, j) violating its optimality condition, the algorithm maintains a list of nodes (LIST) with the property that if an arc (i, j) violates its optimality conditions then LIST must contain node i.

At each iteration, the algorithm selects a node i belonging to LIST, removes it from LIST, and examines it by performing a distance update step for all arcs emanating from such node. To identify valid cycles in the improvement graph, one modification is made in the label-correcting algorithm, enforcing the property that the current directed path $P[j]$ from node s to each node j is a subset-disjoint path, that is, the nodes in the path $P[j]$ belong to different rooted subtrees. To accomplish this, each subtree is assigned a unique label and each node in the subtree gets that label; let $L(i)$ denote this label for node i. While executing the label-correcting algorithm, whenever a node i is removed from LIST, a check is done whether its current path $P[i]$ is a subset-disjoint path, i.e., all nodes in it have different labels. If not, then node i is not examined; otherwise $d(i)$ is set equal to the length of the path $P[i]$, and the arcs emanating from it are examined one by one. While examining the arc (i, j), a check is performed whether $d(j) < d(i) + \alpha_{ij}$. If this condition is satisfied, the arc (i, j) is skipped and another arc emanating from node i is examined. If $d(j) > d(i) + \alpha_{ij}$, then the three following cases must be considered:

For each node j:

1. $j \in P[i]$. In this case, a subset-disjoint cycle $\{(i, j)\} \cup P[i] \setminus P[j]$ has been discovered. Since $d(j) > d(i) + \alpha_{ij}$, it is easy to see that this cycle has a negative cost. Hence the cycle identified is a valid cycle.

2. $j \notin P[i]$ and $L(j) \neq L(k)$ for every $k \in P[i]$. In this case, adding arc (i, j) to $P[i]$ creates a subset-disjoint path $P[i] \cup \{(i, j)\}$. We update $d(j) = d(i) + \alpha_{ij}$, add node j to LIST, and continue the label-correcting algorithm.

3. $j \notin P[i]$ and $L(j) = L(k)$ for some $k \in P[i]$. In this case, adding arc (i, j) to $P[i]$ does not create a subset-disjoint path. We do not update $d(j)$ and continue the label-correcting algorithm.

The modified label-correcting algorithm is applied, as described above, with some node as node s. The algorithm discovers, during its execution, several profitable cyclic or path exchanges or none of them. The algorithm can be further enhanced so that it selects the first profitable exchange, or let it run to completion and select the most profitable exchange.

5 TS-MCSD: a Very Large-Scale Neighborhood Search Heuristic to Solve the MCSD Problem

To solve the SON design problem, we develop a novel and efficient heuristic (named TS-MCSD) that uses the tabu search approach along with the polynomial size (PSAN and PSLN) and the Cyclic Exchange (CEN) Neighborhood concepts described in Section 4.

As stated before, the SON design problem is split into two subproblems: a *location problem* (i.e., the overlay node installation problem) and an *allocation problem* (i.e., the problem of allocating a TP to an overlay node). In the *location* problem it is decided where the overlay nodes should be installed, i.e., using the notation of Section 3, the values of the z_j variables are set. In the *allocation* problem it is determined to which overlay node each TP should be allocated, and at the same time the traffic routing; therefore, in this subproblem, the x_{ij} and f^i_{jl} values are set.

In summary, the TS-MCSD heuristic (illustrated in the flow diagram of Figure 1) proceeds as follows:

- Computing a feasible initial solution.
- Performing an iterative procedure (called Iterative Very Large-Scale Tabu Search, I-VLS-TS) which consists of:
    1. an *optimization step* where the location problem is solved using a tabu search on the polynomial size location neighborhood (PSLN, Section 4.1.1);
    2. given the solution obtained in the optimization step, a *diversification step* is performed using the very large-scale neighborhood search (Section 4.2).
- Performing a post optimization step which consists in applying a local search to the polynomial size allocation neighborhood (PSAN) of the best solution found by the iterative procedure.

The final step is performed only on the PSAN to strike a balance between limiting the overall computation time and trying to improve the solution found so far.

In the following we describe in detail the steps which compose the proposed heuristic.

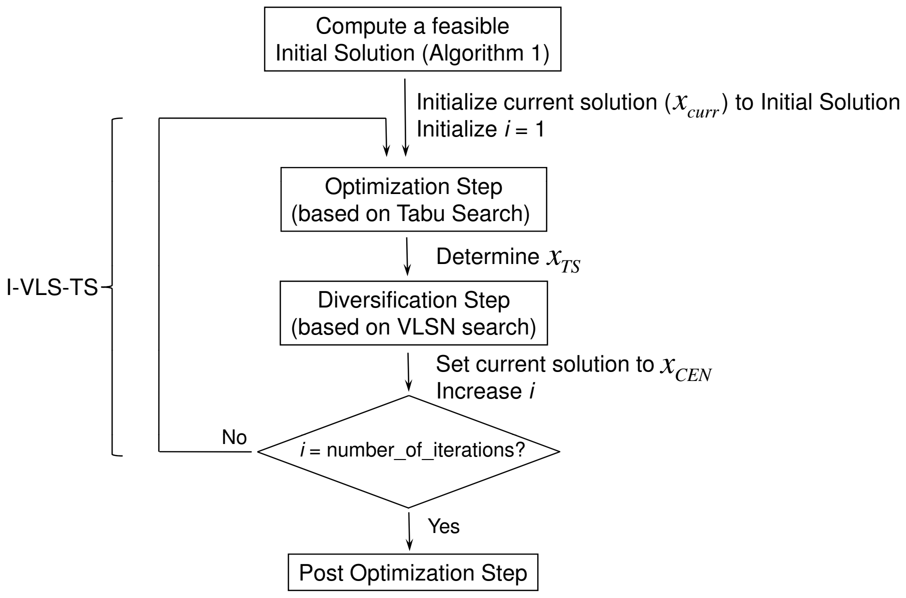

Compute a feasible Initial Solution (Algorithm 1)
Initialize current solution (x_{curr}) to Initial Solution
Initialize i = 1
Optimization Step (based on Tabu Search)
Determine x_{TS}
Diversification Step (based on VLSN search)
Set current solution to x_{CEN}
Increase i
i = number_of_iterations?
No
Yes
Post Optimization Step
I-VLS-TS

Figure 1: Flow diagram of the TS-MCSD heuristic

5.1 Initial Solution

The initial solution is computed by means of the greedy algorithm described in the following.

Given as input the problem parameters (namely: the traffic demand matrix, the costs and the capacity bounds), the algorithm provides an initial feasible solution in which all the test points are allocated. The pseudo-code in Algorithm 1 details how this procedure works.

Starting with an empty set of overlay nodes, a new overlay node is added until we reach a set that provides enough capacity to accommodate the whole flow originated at test points.

Overlay nodes are added according to a function that considers the cost of allocating all TPs to the set of chosen overlay nodes. For each overlay node j, TP i is ordered according to non-decreasing access cost (Access Cost Order, $AC_{ord}[i]$) and according to non-decreasing amount of flow offered to the network (Traffic Order, $T_{ord}[i]$). Then, TP i is considered according to a weighted combination of the positions in the described orders, $W_{pos}[i]$ (see Algorithm 1 for details).

All the TPs that can be assigned to the overlay node j without violating its capacity constraint are labeled as covered by j, unless they are already labeled as covered by another overlay node. The overlay node providing the maximum number of covered TPs is chosen and installed.

Once a set of overlay nodes providing enough capacity to accommodate the whole flow is obtained, the allocations are computed by means of a greedy procedure based on the algorithm proposed in [40] to solve the General Assignment Problem. Such greedy procedure is detailed in Algorithm 3 (Section 5.2.1), since it is also used in the optimization step.

If the capacity constraint is satisfied, an initial feasible solution is found, otherwise the solution evaluated is taken as infeasible. In this latter case, other overlay nodes are added to the set of chosen overlay nodes until a feasible solution is obtained.

Algorithm 1 Pseudo-code specification for computing the initial solution

```
for each candidate overlay node $j$ do
    order test points according to non-decreasing access cost
    $AC_{ord}[i]$ = position of test point $i$ in the order
    order test points according to non-decreasing amount of the total flow (provided by the traffic
    demand matrix)
    $T_{ord}[i]$ = position of test point $i$ in the order
    $W_{pos}[i] = |I| - T_{ord}[i] + 2 \cdot AC_{ord}[i]$
    order test points according to non-decreasing value of $W_{pos}[i]$
    while solution is not feasible do
        determine the number ($NNCov$) of non covered test points that each overlay node can cover,
        according to $W_{pos}[i]$
        open an overlay node in location $j$ such that $NNCov[j]$ is maximum
        if not all test points are covered then
            continue
        else
            allocate test points (using the Greedy Allocation algorithm, Algorithm 3)
            check if the capacity constraints are satisfied (check the feasibility of the solution)
        end if
    end while
end for
```

5.2 Iterative Very Large-Scale Tabu Search

The Iterative Very Large-Scale Tabu Search (I-VLS-TS) is an iterative procedure which consists of an optimization step, where the location problem is solved using a tabu search on the polynomial size location neighborhood (PSLN), and of a diversification step, which is performed applying the cyclic exchange neighborhood (CEN) search to the allocation problem.

CEN is applied to the best solution determined by the tabu search, and it is searched only once in an attempt to obtain a new solution which is both far from the local optimum and cost-decreasing. Note that if the new solution is not profitable in terms of the total cost (indicated by function $Cost(\cdot)$ in Algorithm 2) with respect to the current one, such solution is discarded and the algorithm stops. The pseudo-code of I-VLS-TS is given in Algorithm 2.

Algorithm 2 Pseudo-code specification of I-VLS-TS

```
Initialize the current solution ($x_{curr}$) to the initial solution found using Algorithm 1
Initialize tabu_list_size and number_of_iterations
for $i = 1$ to number_of_iterations do
    (1) Compute the local minimum ($x_{TS}$) using tabu search: $x_{TS}$ = TabuSearch($x_{curr}$)
        Allocate TPs to overlay nodes using Algorithm 3
    (2) Apply once the Cyclic Exchange Neighborhood search to $x_{TS}$: $x_{CEN}$ = CENSearch($x_{TS}$)
        If ($Cost(x_{CEN}) < Cost(x_{curr})$) then set $x_{curr} = x_{CEN}$
        Else STOP.
end for
```

In the following, we describe the optimization step which is based on tabu search.

5.2.1 Optimization Step

The optimization step is performed using a tabu search approach and the PSLN neighborhood. The solution is represented by the set of sites in which an overlay node is opened.

Tabu search is a local search based optimization method [41, 42] that can accept cost-increasing solutions in order to escape from local minima. This feature allows the neighborhood search to explore other parts of the solution space.

For each neighborhood, the best neighbor solution is used as new current solution and, if it improves upon the best solution found so far, the best solution is updated. This may cause the algorithm to cycle. To avoid cycles, a memory is kept of the last solutions accepted as current solution, in term of the moves used to transform one solution to the next. When a move is performed, it is considered tabu for the next T iterations, where T is the tabu list length.

A solution produced by a move belonging to the tabu list, which we refer to as *tabu move*, is discarded. To avoid discarding good solutions, an aspiration criterion is defined: a solution generated by a tabu move is accepted if it satisfies the aspiration criterion. Tabu search stops when it reaches its stopping condition, for instance after a given number of visited neighborhoods without improvement in the best solution found. The tabu search parameters (tabu list dimension, stopping and aspiration criteria) play a fundamental role in implementing such metaheuristic, and in this work they have been set according to computational experience:

- Tabu moves: a tabu move is represented by the node or the nodes involved in the move to be forbidden.
- Tabu list size: a static tabu list is considered containing up to $T = 6$ moves.
- Stopping criterion: the algorithm stops after 10 consecutive neighborhood searches without improvements in the best solution found.
- Aspiration criterion: a solution generated by a tabu move is accepted if it improves upon the best solution found so far.

Given the set of overlay nodes individuated by tabu search, the allocation of test points to such nodes is performed using the greedy algorithm, presented as Algorithm 3 (see reference [40]).

Algorithm 3 Pseudo-code specification of the Greedy Allocation algorithm

```
while not all test points are allocated do
    for each test point $i$ not allocated do
        order overlay nodes according to non-decreasing value of $v_{ij} = \frac{c^A_{ij}+c^E_{ji}}{o_i+d_i}$
        $t_1[i]$ = first overlay node in the order
        $t_2[i]$ = second overlay node in the order
        $\Delta[i] = v_{it_2(i)} - v_{it_1(i)}$
    end for
    select test point $i$ such that $\Delta[i]$ is maximum
    if no capacity constraint is violated then
        allocate $i$ to $t_1[i]$
    end if
end while
```

We recall that c^A_{ij} and c^E_{ji} are, respectively, the per-unit flow access and egress costs between test point i and overlay node j, while $o_i + d_i$ represents the total amount of flow originating from and destined for test point i. The algorithm allocates the test points one by one according to a parameter $v_{ij} = \frac{c^A_{ij}+c^E_{ji}}{o_i+d_i}$, which takes into account both the allocation costs and the total amount of flow originating from and destined for the test point.

More specifically, the rationale behind dividing access and egress costs by traffic demands is the following: when c^A_{ij} and/or c^E_{ji} increase, node j should be chosen with less probability, since the access/egress costs deriving from its installation are high; on the other hand, if TP i has a high amount of flow (incoming and/or outgoing, $o_i + d_i$), then the cost of node j in Algorithm 3 is reduced, so that it will be installed with higher probability in order to try to accommodate such traffic.

A test point is allocated to an overlay node if and only if the allocation does not violate the capacity constraint.

For each test point, not yet assigned, the overlay nodes are ordered according to non-decreasing value of v_{ij}. Then, for each test point i the first and the second overlay nodes in the order, $t_1(i)$ and $t_2(i)$, are considered. The test point with the maximum value of $v_{it_2(i)} - v_{it_1(i)}$ is selected and assigned to $t_1(i)$ if the capacity constraint is satisfied. The main idea behind this procedure is to assign first those test points which would suffer the greatest disadvantage in case they are not assigned to the closest overlay node.

Given as input the set of overlay nodes, the algorithm attempts to allocate all test points to these nodes. If it fails, the set is discarded as infeasible.

Note that we can also compute the exact solution to the allocation problem simply minimizing the allocation costs (access and egress costs) as follows: minimize $\left\{ \sum_{i\in I} \sum_{j\in S} (o_i c^A_{ij} + d_i c^E_{ji}) x_{ij} \right\}$ subject to constraints (2), (3), (4) and (6), where z_j is not any more a variable, but a parameter determined solving the location problem: $z_j = 1$ if an overlay node is installed in CS j, $z_j = 0$ otherwise. It turns out that the new network costs computed with the exact model are in all cases at most 0.4% lower than the costs of the networks planned using Algorithm 3, and in several instances they even coincide, at the expense of a higher computation time. For this reason, in the numerical results section we will consider only this latter algorithm for the allocation of TPs to overlay nodes.

5.2.2 Diversification Step

In this step, the Cyclic Exchange Neighborhood of the local optimum determined in the optimization step is searched using the techniques described in Section 4.2, trying to find a new solution that is both far from such local optimum and cost-decreasing.

5.3 Post Optimization Step

Finally, in this step we perform a local search of the polynomial size allocation neighborhood (PSAN) of the best solution found by I-VLS-TS in order to improve the chosen assignments. More specifically, starting from the best solution found by the I-VLS-TS procedure, the PSAN is generated and then visited by the local search with the aim of finding a new solution that improves the objective function value.

6 Numerical Results

In this section we test the sensitivity of the proposed heuristic to different parameters like the number of candidate sites and test points, the traffic demands, the installation as well as the bandwidth costs. We compare the performance of the exact and heuristic approaches for small-size network instances in terms of the obtained results and computation time.

To this end we consider both randomly generated network instances and real ISP topologies provided by Telecom Italia Lab (TILab). Random network topologies are obtained using a custom generator as well as a degree-based generator (BRITE [43, 44]) which produces topologies with node degree power laws.

To generate random network instances, we have implemented a topology generator which considers a square area with edge equal to 1000, and randomly extracts the position of m candidate sites (CSs) and n test points (TPs). The area is divided into N_{ISP} Internet Service Providers (ISPs); for the sake of simplicity in this paper we consider $N_{ISP} = 25$ ISPs obtained by dividing the whole area into $L \times L$ squares, with $L = 200$.

We assume that each TP can be connected to a CS only if the CS is at a distance not greater than r from the TP, with $r = 200$ and 400 for sparse and dense networks, respectively. As for the connectivity parameters between different CSs, we assume that each CS can be directly connected with an overlay link to any other CS (i.e., $b_{jl} = 1, \forall j, l \in S$); this allows our model to investigate all possible link configurations to find the optimal overlay topology.

The cost matrix for bandwidth (c_{jl}^B) is then generated. If CSs j and l belong to the same ISP, we assume that c_{jl}^B is fixed and equal to 1 monetary unit per Mb/s. On the other hand, if CSs j and l belong to different ISPs, c_{jl}^B depends on the peering agreements between such ISPs. For the sake of simplicity, we assume that in this case c_{jl}^B is a random variable uniformly distributed between $C/2$ and $3C/2$, with C being equal to $\frac{L_{jl}}{L}$, that is the distance between j and l (L_{jl}) divided by the width of an ISP domain (L), i.e. 200 with the above settings.

If not specified differently, the installation cost of an overlay node is equal to 10 monetary units. As for the access and egress cost, we assume they are fixed and equal to 1 monetary unit per Mb/s. The maximum access capacity of an overlay node j (Γ_j) is set to 50 Mb/s for all $j \in S$.

Finally, the maximum number of iterations used in the I-VLS-TS algorithm (the "number_of_iterations" parameter in Algorithm 2) is set to 10.

Obviously, none of the above assumptions affects the proposed model and heuristic which are general and can be applied to any problem instance and network topology.

Identifying candidate sites can be a difficult task in real ISP networks. To solve this problem, a topology-aware node placement heuristic could be used, as proposed in [24], to decide the potential locations for overlay nodes inside an ISP. Such techniques can be used together with our heuristic and model, thus representing a further research topic worth pursuing.

All the results reported hereafter are the optimal and approximate solutions of the considered instances obtained, respectively, by formalizing the proposed model in AMPL [45] and solving it with CPLEX 9.1 [46], or using the proposed heuristic approach. The workstations used were equipped with an Intel Pentium 4 (TM) processor with CPUs operating at 3 GHz, and with 1024 Mbyte of RAM. For each random network scenario, the results are obtained averaging each point on 10 network instances, while for real ISP topologies we provide the exact results obtained by solving the instances provided by the ISP.

6.1 Effect of the Traffic Demands: Random network instances

We first consider the SON design problem in a random network scenario with $n = 20$ TPs. Each test point offers the same amount of traffic w_{ik}.

Figure 2 reports an example of the planned networks when applying the SON design model to the same instance with $m = 40$ candidate sites and with two different requirements on the end-user traffic, $w_{ik} = 500$ kb/s and $w_{ik} = 1$ Mb/s for all TPs. The test point coverage radius r is equal to 200. CSs and TPs are represented with circles and triangles, respectively. As expected, increasing the traffic demands forces the model to install a higher number of overlay nodes and links to convey the traffic towards the destinations.

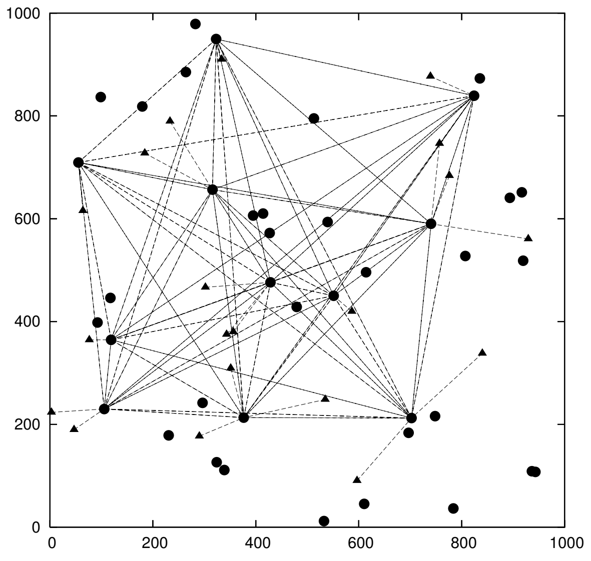
Table 1 analyzes the characteristics of the solutions of the MCSD model and the TS-MCSD heuristic in the same scenario when varying the number of candidate sites. For each couple (m, w_{ik}) and optimization algorithm, the table reports the number of installed overlay nodes (N_R) and links (N_L), the total network cost and the computation time (measured in seconds) to obtain the solution.

For small instances ($m = 30 - 80$) we could obtain the exact solutions with the MCSD model, which enabled a comparison between the TS-MCSD heuristic and optimal integer solutions; column *gap* reports the average percentage gap between the cost provided by TS-MCSD and the optimal cost obtained with the MCSD model. This gap shows how close is TS-MCSD to the optimum obtained by the integer model, in terms of the objective function value. Furthermore, column "Max *gap*" reports the maximum percentage gap observed for each scenario, to account for worst-case scenarios. Note that solving the MCSD model for dense networks (i.e. $r = 400$), requires an extremely long computation time even for very small m values, so that only the proposed heuristic can be applied in practice.

Three main results come from the observation of the table: first, the very same effect of traffic increase observed in Figure 2 is evident also on averaged results; in fact, the number of installed nodes and links increases when increasing the traffic demands.

Second, for a given traffic value, increasing the number of CSs (m) increases the solution space; as a consequence, the model favors the solutions providing connectivity that have a lower impact on the network cost, which in turn decreases with m.

Finally, the proposed heuristic performs very close to the optimal solutions (less than 2% for $w_{ik} = 500$ kb/s and less than 4% for $w_{ik} =$ 1 Mb/s), and the computation time is small for all the tested instances.


(a) 500 kb/s

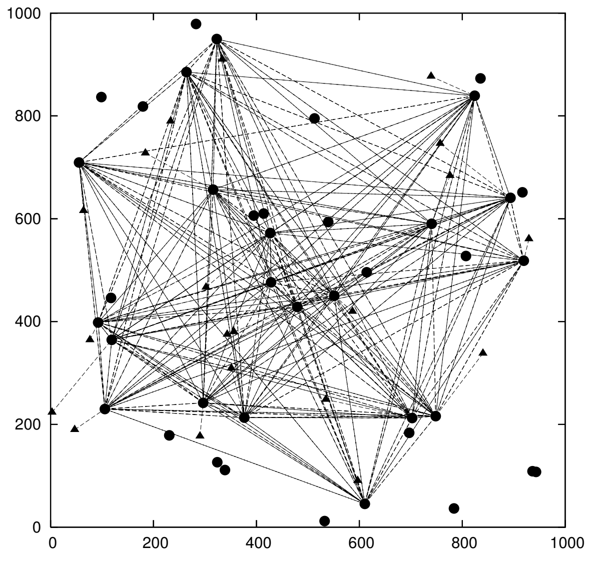

(b) 1000 kb/s

Figure 2: Sample SONs planned by the MCSD model with increasing traffic demands (500 and 1000 kb/s). The number of TPs is 20, while the number of CSs is 40. The test point coverage radius r is equal to 200. CSs and TPs are represented with circles and triangles, respectively.

Furthermore, we verified that in several network instances TS-MCSD solved the SON design problem to the optimum.

Table 1: Solutions provided by the TS-MCSD heuristic and the MCSD model, with 20 TPs and $r = 200$ (sparse networks).

w_{ik}=500 kb/s

	MCSD				TS-MCSD					
m	N_R	N_L	Cost	Time	N_R	N_L	Cost	Time	*gap* %	Max *gap* %
30	10.8	147.5	955.1	3.7	10.2	136.1	963.0	1.0	0.8	3.3
40	10.9	150.6	926.9	47.7	10.7	145.5	943.5	2.0	1.8	6.2
50	11.4	160.3	890.1	146.1	11.0	151.2	904.1	3.4	1.6	3.7
60	11.5	163.4	880.3	380.1	11.1	155.4	888.9	5.3	1.0	2.0
80	12.1	192.0	851.7	4785.3	11.5	161.1	867.7	11.3	1.9	7.7
100					11.8	170.1	846.0	21.3		
200					11.9	172.8	799.3	151.5		
500					12.3	182.0	763.9	2798.7		

w_{ik}=1000 kb/s

	MCSD				TS-MCSD					
m	N_R	N_L	Cost	Time	N_R	N_L	Cost	Time	*gap* %	Max *gap* %
30	13.5	208.3	1847.2	10.7	13.7	215.4	1882.0	1.0	1.9	4.1
40	13.7	214.5	1772.3	96.0	13.8	218.6	1818.0	2.1	2.6	4.9
50	14.1	225.2	1683.6	357.8	13.8	218.8	1729.4	3.7	2.8	5.7
60	14.5	238.5	1668.7	5482.8	13.9	218.8	1715.2	5.9	2.8	4.4
80	14.0	222.6	1590.2	17003.0	13.8	218.8	1648.0	12.9	3.6	7.9
100					13.8	218.6	1603.6	23.7		
200					14.0	222.6	1511.5	177.1		
500					13.8	218.9	1404.7	3245.8		

We then recomputed the optimal solutions with a 2% gap using CPLEX. Table 2 reports the corresponding numerical results, which show that the computation time saving can be consistent, but the overall computation time tends to increase dramatically for increasing m values, so that solving the MCSD model becomes impractical.

Table 2: Solutions provided by the MCSD model with a 2% gap with respect to the optimal solution, with 20 TPs and $r = 200$ (sparse networks).

	w_{ik}=500 kb/s				w_{ik}=1000 kb/s			
m	N_R	N_L	Cost	Time	N_R	N_L	Cost	Time
30	10.8	147.7	957.2	2.2	13.4	206.3	1853.2	4.8
40	11.0	152.0	927.7	21.8	13.5	210.1	1775.4	51.9
50	11.6	165.4	890.8	63.6	14.1	225.1	1684.9	237.0
60	11.7	166.3	880.8	281.4	14.6	240.5	1669.8	1589.2
80	12.1	175.1	851.7	2758.1	14.0	222.6	1590.5	7260.6

In the same network scenario we compared our proposed heuristic with two other variants, viz. tabu search (without VLSN), which is implemented simply dropping step (2), the CEN search, from Algorithm 2, as well as a VLSN local search (without tabu search). Numerical results indicate that TS-MCSD achieves an average cost reduction of 14% with respect to the first approach, and of approximately 20% with respect to the second. The maximum gaps (which were measured for $w_{ik} = 1Mb/s$ and for large network instances) were equal to 31% and 42%, respectively, thus confirming the effectiveness of our proposed approach in planning cost-efficient SONs.

Finally, we considered a variation of this network scenario with large-size instances containing $n = 100$ TPs. The results obtained using the TS-MCSD heuristic are shown in Tables 3 and 4 for $r = 200$ and 400, respectively, with m ranging from 100 to 500 and for different w_{ik} values. These results are in line with the observations reported above. Since for $r = 400$ the solution space is larger than in the $r = 200$ case, TS-MCSD plans Service Overlay Networks with a lower cost.

Table 3: Large-size instances: solutions provided by the TS-MCSD heuristic, with 100 TPs and $r = 200$ (sparse networks).

	w_{ik}=50 kb/s				w_{ik}=100 kb/s			
m	N_R	N_L	Cost	Time	N_R	N_L	Cost	Time
100	22.7	696.3	2159.2	217.4	27.6	938.1	4158.7	234.9
200	25.6	835.2	2069.9	959.7	30.2	1092.9	3951.6	1034.5
500	26.4	872.9	1980.9	1555.8	33.0	1264.3	3753.5	2600.4

Table 4: Large-size instances: solutions provided by the TS-MCSD heuristic, with 100 TPs and $r = 400$ (dense networks).

	w_{ik}=50 kb/s				w_{ik}=100 kb/s			
m	N_R	N_L	Cost	Time	N_R	N_L	Cost	Time
100	20.5	600.9	2142.6	210.6	27.1	908.8	4145.2	260.1
200	22.2	678.3	2065.1	933.4	29.8	1066.6	3933.8	1180.0
500	24.8	799.2	1969.0	2144.7	32.1	1207.4	3748.5	2452.4

6.2 Effect of the Traffic Demands: Power Law topologies

To better capture the power law distributions for node degrees that characterize Internet topologies, we further considered BRITE, a degree-based topology generator [43, 44]. BRITE was used to generate power law topologies with 500 CSs and 20 TPs on a square area with edge equal to 1000; the Barabási − Albert model [47] was adopted with default parameters provided by BRITE and an average node degree equal to 10. The bandwidth cost matrix was generated as for the custom topology generator.

Table 5 illustrates the numerical results obtained with such topologies, with an offered traffic w_{ik} equal to 500 and 1000 kb/s. The results confirm the behavior already observed for the random instances obtained with the custom generator, where the number of installed overlay nodes and links increases with increasing traffic demands.

Table 5: Power Law topologies: solutions provided by the TS-MCSD heuristic in the topologies with node degree power laws generated using BRITE, with 500 CSs and 20 TPs.

w_{ik} (kb/s)	N_R	N_L	Cost	Time
500	4.0	52.0	823.6	1297.7
1000	10.0	130.0	2914.9	2493.4

Note that doubling the offered traffic in power law topologies forces the heuristic to install proportionally more nodes and links with respect to random topology scenarios. This is mainly due to the fact that when traffic increases, the capacity of heavily clustered nodes is exhausted, and the heuristic must search entirely new paths (installing new overlay nodes and links) to accommodate and route the traffic from origin to destination. In fact, in power law topologies created by the Barabási − Albert model, paths typically converge through few "hub" nodes, with large node degree, and less alternate paths exist with respect to random topologies generated as we detailed at the beginning of this section. In this latter case, the heuristic can find more easily alternate nodes and links (already installed) so that the incremental number of nodes and links is less important in these topologies.

6.3 Effect of the Bandwidth and Installation Costs

We evaluate the effect of the bandwidth cost as well as the nodes' installation cost on our model's performance, considering a random network scenario with $n = 100$ TPs and $m = 100$ CSs. The offered traffic w_{ik} is equal to 50 kb/s and we consider dense networks with $r = 400$. The solution, and in particular the number of installed nodes and links, intuitively depends on two factors: the ratio β between the overlay nodes' installation cost and the bandwidth reservation cost, and the ratio δ between the cost of a link connecting two CSs which belong to different ISPs and that of a link connecting two CSs belonging to the same ISP.

Tables 6 and 7 report the results obtained varying, respectively, the parameters β and δ. Table 6 shows that if the cost of installing an overlay node decreases with respect to the bandwidth reservation cost, the proposed heuristic tends to install more overlay nodes. Note that the computation time of the TS-MCSD heuristic is short.

Similarly, when δ increases, we observe that our heuristic tends to deploy less overlay nodes and links, thus diminishing the number of longer (and costlier) links that traverse different ISPs in the planned SON. This behavior is reflected in the numerical results reported in Table 7.

Table 6: Variable cost ratio β: solutions provided by the TS-MCSD heuristic with 100 TPs, 100 CSs, w_{ik}=50 kb/s and $r = 400$ (dense networks).

β	N_R	N_L	Cost	Time
10	20.5	600.9	2159.2	210.6
5	26.8	896.9	2041.5	233.8
1	33.9	1325.5	1914.6	255.4
1/10	36.3	1494.6	1880.0	256.4

6.4 Real ISP Topologies

Finally, we considered three diverse ISP topologies which have been provided by Telecom Italia Lab (TILab). Table 8 lists, for each topology, the number of test points (n) and candidate sites (m), the overlay node installation cost (c^I_j) and capacity (Γ_j). All candidate sites have the same installation cost

Table 7: Variable cost ratio δ: solutions provided by the TS-MCSD heuristic with 100 TPs, 100 CSs, w_{ik}=50 kb/s and $r = 400$ (dense networks).

δ	N_R	N_L	Cost	Time
1	20.5	600.9	2159.2	210.6
2	14.2	390.9	3032.4	177.5
5	12.3	340.0	4870.0	179.0
10	11.7	326.1	7400.8	186.8

and the same capacity. The traffic demands (which are not symmetric, i.e. w_{ik} is in general different from w_{ki}), the number and location of TPs and CSs, the access/egress costs and the transport costs are those provided by TILab for each network instance.

Table 9 shows the results obtained using the TS-MCSD heuristic on the TILab network instances. Note that in all such real scenarios the proposed heuristic solved the MCSD problem with almost negligible computation time, thus representing a practical solution for the planning of SONs. The table further reports, in the last column (gap_{IS}), the percentage improvement obtained by I-VLS-TS (the iterative procedure) with respect to the initial solution computed using Algorithm 1. The measured improvements (up to 45%) were obtained in average within the first 2 steps, with a maximum of 6 steps, which justifies our choice of setting to 10 the number of iterations of I-VLS-TS.

Table 8: TILab Network instances: number of test points and candidate sites, overlay node installation cost and capacity.

Network	n	m	c_j^I	Γ_j
TILab1	15	12	20	2000
TILab2	39	17	100	500
TILab3	49	49	60000	10000

Table 9: TILab topologies: solutions provided by the TS-MCSD heuristic.

Network	N_R	N_L	Cost	Time	gap_{IS} %
TILab1	3.0	36.0	6412.3	0.1	23.8
TILab2	4.0	90.0	13612.9	0.8	45.3
TILab3	7.0	140.0	46590955.4	11.1	7.8

We were also able to compute the exact solution using the MCSD model for the small TILab1 topology. Such solution contains exactly the same number of overlay nodes ($N_R = 3$) and links ($N_L = 36$) planned by TS-MCSD, and has a cost equal to 6226.9, i.e., approximately 2.9% cheaper than the network planned by our heuristic. The computation time, however, is 805.6 s, hence much larger than that necessary for TS-MCSD (0.1 s). On the other hand, we were unable to solve to the optimum the two larger instances (TILab2 and TILab3) due to their large number of TPs, and consequently of network flows.

We then derived other instances by reducing the overlay nodes capacities by a factor of 0.6 and 0.8, and by reducing the installation costs by a factor of 0.1, 0.01 and 0.001. Tables 10 and 11 show the numerical results obtained in the three TILab topologies with, respectively, the overlay node capacity reduction of 0.6 and 0.8, and the installation cost reduction of 0.1, 0.01 and 0.001. For comparison reasons, in these Tables we also reported the numerical results obtained without applying any reduction factor (which are those already shown in Table 9).

When the overlay nodes capacity decreases, the heuristic is forced to install more overlay nodes, and the total SON cost increases (see Table 10). On the other hand, when the overlay nodes installation cost decreases, TS-MCSD tends to install more overlay nodes (see Table 11, network topology TILab2), and the overall network cost decreases. It can be further observed that the cost changes due to the capacity and cost reduction factors are quite limited, thus suggesting that these are not the major cost factors for such network topologies. Finally, note that also for such network instances the computation time of our proposed heuristic is very short.

Table 10: TILab topologies: solutions provided by the TS-MCSD heuristic. An overlay node capacity reduction of 0.6 and 0.8 has been applied. For comparison reasons, the results obtained without capacity reduction (i.e., capacity reduction factor = 1) are also reported.

Network	*Cap. Reduction Factor*	N_R	N_L	Cost	Time
TILab1	0.6	4.0	42.0	7261.1	0.2
TILab1	0.8	3.0	36.0	6538.3	0.2
TILab1	1	3.0	36.0	6412.3	0.1
TILab2	0.6	5.0	98.0	13691.9	1.0
TILab2	0.8	5.0	98.0	13691.9	1.1
TILab2	1	4.0	90.0	13612.9	0.8
TILab3	0.6	10.0	188.0	49374604.8	11.5
TILab3	0.8	8.0	154.0	47310633.0	13.2
TILab3	1	7.0	140.0	46590955.4	11.1

Table 11: TILab topologies: solutions provided by the TS-MCSD heuristic. The overlay nodes installation costs have been reduced by a factor of 0.1, 0.01 and 0.001. For comparison reasons, the results obtained without cost reduction (i.e., cost reduction factor = 1) are also reported.

Network	*Cost Reduction Factor*	N_R	N_L	Cost	Time
TILab1	1	3.0	36.0	6412.3	0.1
TILab1	0.1	3.0	36.0	6358.3	0.2
TILab1	0.01	3.0	36.0	6352.9	0.2
TILab1	0.001	3.0	36.0	6352.3	0.2
TILab2	1	4.0	90.0	13612.9	0.8
TILab2	0.1	10.0	168.0	13057.0	1.0
TILab2	0.01	10.0	168.0	12967.0	1.0
TILab2	0.001	10.0	168.0	12958.0	1.0
TILab3	1	7.0	140.0	46590955.4	11.1
TILab3	0.1	7.0	140.0	46241653.3	12.1
TILab3	0.01	7.0	140.0	46203853.3	12.1
TILab3	0.001	7.0	140.0	46200073.3	12.1

7 Conclusion

In this paper we addressed the topology design problem for Service Overlay Networks in terms of deciding the number and location of the overlay nodes to be deployed and the assignment of users to overlay nodes, in order to minimize the total network installation cost.

To this end, we proposed a novel optimization model based on mathematical programming that provides a bound to the performance achievable by any optimization algorithm.

Furthermore, we developed a novel and efficient heuristic approach that uses tabu search along with polynomial size and very large-scale neighborhoods. The very large-scale neighborhood is used to perform the diversification step on the best solution obtained by tabu search.

The numerical results we gathered show that our heuristic obtains good solutions even for large-scale instances in a short computation time, and that it is able to capture the effect on the overlay topology configuration of all the considered network parameters, thus providing a promising framework for the design of SONs.

References

[1] Z. Duan, Z.-L. Zhang, and Y.T. Hou. Service Overlay Networks: SLAs, QoS, and Bandwidth Provisioning. *IEEE/ACM Transactions on Networking*, pages 870–883, vol. 11, no. 6, December 2003.

[2] Z. Li and P. Mohapatra. QRON: QoS-aware Routing in Overlay Networks. *IEEE Journal on Selected Areas in Communications*, pages 29–40, vol. 22, no. 1, January 2004.

[3] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. OverQoS: Offering Internet QoS Using Overlays. In *Proceedings of the 1st Workshop on Hot Topics in Networks HotNets-I*, Princeton, New Jersey, USA, October 2002.

[4] X. Gu, K. Nahrstedt, R.N. Chang, and C. Ward. QoS-assured service composition in managed service overlay networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems, 2003*, Providence, Rhode Island, USA, May 2003.

[5] J. Touch and S. Hotz. The X-Bone. In *Proceedings of the third Global Internet Mini-Conference*, pages 75–83, Sydney, Australia, 1998.

[6] H.T. Tran and T. Ziegler. A design framework towards the profitable operation of service overlay networks. *Computer Networks*, pages 94–113, vol. 51, no. 1, 2007.

[7] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. In *IETF RFC 3031*, January 2001.

[8] S.L. Vieira and J. Liebeherr. Topology Design for Service Overlay Networks with Bandwidth Guarantees. In *the 12th IEEE International Workshop on Quality of Service, IWQoS*, pages 211–220, Montreal, Canada, June 2004.

[9] Z. Li and P. Mohapatra. On investigating overlay service topologies. *Computer Networks*, pages 54–68, vol. 51, no. 1, 2007.

[10] A. Capone, J. Elias, and F. Martignon. Models and Algorithms for the Design of Service Overlay Networks. *IEEE Transactions on Network and Service Management*, article in press, September 2008.

[11] R.R. Boorstyn and H. Frank. Large-Scale Network Topological Optimization. *IEEE Transactions on Communications*, pages 29–47, vol. 25, no. 1, January 1977.

[12] M. Pioro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers, 2004.

[13] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE Infocom'02*, pages 1190–1199, vol.3, New York, USA, June 2002.

[14] A. Kershenbaum, P. Kermani, and G.A. Grover. MENTOR: an algorithm for mesh network topological optimization and routing. *IEEE Trans. Commun.*, pages 503–513, vol. 39, no. 4, April 1991.

[15] M. Grötschel, C.L. Monma, and M. Stoer. Design of survivable networks. *Handbooks in Operations Research and Management Science*, 7:617–672, 1995.

[16] T.L. Magnanti, P. Mireault, and R.T. Wong. Tailoring Benders decomposition for uncapacitated network design. *Mathematical Programming Study*, 26(1):112–154, 1986.

[17] A. Bley, M. Grötschel, and R. Wessäly. Design of broadband virtual private networks: Model and heuristics for the B-WiN. *Robust communication networks: Interconnection and survivability*, 53:1–16.

[18] A. Bley and T. Koch. Integer programming approaches to access and backbone IP-network planning. *ZIB Preprint ZR-02-41, Konrad-Zuse-Zentrum für Informationstechnik Berlin*, 2002.

[19] A. Bley. A Lagrangian approach for integrated network design and routing in IP networks. In *Proceedings of the First International Network Optimization Conference (INOC 2003), Paris*, pages 107–113. Citeseer, 2003.

[20] A. Balakrishnan, T.L. Magnanti, and R. T. Wong. A Dual-Ascent Procedure for Large Scale Uncapacitated Network Design. *Operations Research*, pages 716–740, vol. 37, n. 5, September-October 1989.

[21] M. Minoux. Network synthesis and optimum network design problems: models, solution methods and applications. *Networks*, pages 313–360, vol. 19, 1989.

[22] A. Hills. Large-Scale Wireless LAN Design. *IEEE Communications Magazine*, pages 98–107, vol. 39, no. 11, November 2001.

[23] E. Amaldi, A. Capone, M. Cesana, and F. Malucelli. Optimization Models for the Radio Planning of Wireless Mesh Networks. In *Proceedings of Networking 2007*, Atlanta, Georgia, USA, 14-18 May 2007.

[24] J. Han, D. Waston, and F. Jahanian. Topology Aware Overlay Networks. In *Proceedings of IEEE Infocom'05*, Miami, FL, 13-17 March 2005.

[25] J. Fan and M.H. Ammar. Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. In *Proceedings of IEEE Infocom'06*, Barcelona, Spain, April 2006.

[26] S. Shi and J. Turner. Placing servers in overlay networks. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS) 2002*, San Diego, CA, July 2002.

[27] B.D. Vleeschauwer, F.D. Turck, B. Dhoedt, and P. Demeester. On the construction of QoS enabled overlay networks. In *Proceedings of the 5th International Workshop on Quality of future Internet Services (QofIS04)*, pages 164–173, Barcelona, Spain, October 2004.

[28] S. Roy, H. Pucha, Z. Zhang, Y.C. Hu, and L. Qiu. Overlay Node Placement: Analysis, Algorithms and Impact on Applications. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, Toronto, Canada, June 2007.

[29] L. Zhou and A. Sen. Topology Design of Service Overlay Network with a Generalized Cost Model. In *Proceedings of IEEE Global Telecommunications Conference, GLOBECOM*, pages 75–80, November 2007.

[30] A. Sen, L. Zhou, B. Hao, B.H. Shen, and S. Ganguly. On Topological Design of Service Overlay Networks. In *Proceedings of the thirteenth International Workshop on Quality of Service, IWQoS*, pages 54–68, 2005.

[31] R.K. Ahuja, Ö. Ergun, J.B. Orlin, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, pages 75–102, vol. 123, 2002.

[32] R.K. Ahuja, J.B. Orlin, and D. Sharma. Very large-scale neighborhood search. *International Transactions in Operational Research*, pages 301–317, vol. 7, 2000.

[33] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high speed networks. *IEEE Journal on Selected Areas in Communications*, pages 968–981, September 1991.

[34] J.A. Schormans, J. Pitts, K. Williams, and L. Cuthbert. Equivalent capacity for on/off sources in ATM, Electronic. *Electronic Letters*, pages 1740–1741, vol. 30, no. 21, 1994.

[35] H.W. Hamacher, M. Labb, S. Nickel, and T. Sonneborn. Adapting polyhedral properties from facility to hub location problems. *Discrete Applied Mathematics*, pages 104–116, vol. 145, n. 1, December 2004.

[36] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 137–150, Marseille, France, 2002.

[37] R.K. Ahuja, J.B. Orlin, S. Pallottino, M.P. Scaparra, and M.G. Scutellà. A multi-exchange heuristic for the single source capacitated facility location problem. *Management Science*, pages 749–760, vol. 50, no. 6, June 2004.

[38] R.K. Ahuja, J.B. Orlin, and D. Sharma. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. *Mathematical Programming*, pages 71–97, vol. 91, 2001.

[39] P. Thompson and J.B. Orlin. Theory of cyclic transfers. Working paper, Operations Research Center, MIT, 1989.

[40] S. Martello and P. Toth. *Knapsack Problems - Algorithms and Computer Implementations.* Wiley and Sons, 1990.

[41] F. Glover. Tabu Search. Part I. *Orsa Journal on Computing*, pages 190–206, vol. 1, 1989.

[42] F. Glover. Tabu Search. Part II. *Orsa Journal on Computing*, pages 4–32, vol. 2, 1990.

[43] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings of MASCOTS 2001*, Cincinnati, OH, August 2001.

[44] A. Medina, I. Matta, and J. Byers. On the Origin of Power-Laws in Internet Topologies. ACM Computer. *ACM Communications Review*, pages 18–28, vol. 30, no. 2, April 2000.

[45] *AMPL: A Modeling Language for Mathematical Programming.* Available at http://www.ampl.com.

[46] ILOG Optimization Products. ILOG CPLEX. http://www.ilog.com/products/cplex/.

[47] A.L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, pages 509–512, vol. 286, no. 5439, 1999.