Noname manuscript No.
(will be inserted by the editor)

Multi-objective Traffic Engineering for Data Center Networks

Trong-Viet Ho · Yves Deville · Olivier Bonaventure

Received: date / Accepted: date

Abstract Data centers are now the basis for many Internet and cloud computing services. The Spanning Tree Protocol and its variants have been widely used in data center networks for a couple of decades. An efficient use of the limited spanning tree links would enable to solve the traffic engineering problem in data centers. In this paper, we propose five local search approaches for generating a good set of spanning trees in data centers using multiple VLANs. The quality of these algorithms is evaluated by different multi-criteria assessment methods. The performance of each algorithm is assessed based on three standard measures: maximal link utilization, sum load, and the number of used links.

Keywords Traffic Engineering · Multi-objective Optimization · Data Center Traffic · Local Search · Multiple Spanning Tree Protocol.

1 Introduction

Currently, data centers containing thousands of servers are used to support a growing number of services. In many of these environments, the required bandwidth is growing quickly and network operators need to find solutions which ensure that their network can sustain the traffic demand without having overloaded links.

Trong-Viet Ho
ICTEAM Institute, Université catholique de Louvain,
Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgium
Tel.: +32 (0)10/47.91.16
Fax: +32 (0)10/45.03.45
E-mail: trong.ho@uclouvain.be

Yves Deville
E-mail: yves.deville@uclouvain.be

Olivier Bonaventure
E-mail: olivier.bonaventure@uclouvain.be

Current enterprise data centers contain a large number of Ethernet switches [29–31]. The IEEE 802.1s Multiple Spanning Tree Protocol (MSTP) [2] is widely deployed in large Ethernet networks. Network operators often divide the network into different regions, called Virtual Local Area Networks (VLANs) [3]. Note that two VLANs can share some nodes. MSTP configures the spanning tree for each VLAN by computing the least-weight path from each switch to an elected root switch. From a traffic engineering (TE) viewpoint, the major drawback is that MSTP itself cannot ensure a good set of spanning trees for a given set of traffic demand matrices.

Solving the TE problem in large Ethernet networks can be defined as the optimization of the network based on many evaluation metrics, such as link utilization, link delay, sum load, and number of used links. It is impossible to optimize all these metrics at the same time because an improvement in one metric can degrade other metrics. In practice, network operators would like to provide the best network performance based on a compromise among a given set of metrics. Multi-objective optimization [4] is an interesting approach for solving this kind of TE problem, where each objective is related to one metric. Solving the single objective problem is already very complex, as the search space is exponential even for networks of reasonable size. Approximate methods, such as Constraint-Based Local Search (CBLS)[5], have been proved to be well suited for solving complex combinatorial problems with huge search spaces such as this TE problem. In this paper, we focus on approximate methods based on local search.

Online TE [6] approaches are supposed to dynamically adapt the routing when traffic changes in real-time. However, the network devices are required to be more intelligent and support more flexibles policies rather than standard switches to perform the local adjustments due to the actual traffic situation of the network (e.g. OpenFlow switches [8]). Traditional "offline" TE is used to improve the performance based on the given traffic demand matrices. The goal is to provide global solutions for the network design and configuration. The advantage is that the implementation of offline TE approaches is often perfomed on cheap and normal switches. Hence, we choose an offline approach for solving the TE problem in enterprise data center networks. Offline TE approaches are suitable in public datacenters where companies rent a large number of servers. In such datacenters, each customer defines its contractual traffic matrix and the network needs to be configured to sustain this matrix. The traffic generated by each customer can vary within the bounds of the contracted traffic matrix. The Offline Traffic Engineering techniques proposed in this paper are suitable for environments were the traffic demand is stable or can be correctly estimated. Such environments include enterprise datacenter that are running long batch jobs such as scientific computing. Public cloud providers could also benefit from the techniques. In public clouds, the virtual machines that are used by the customers are often group in one VLAN per customer for traffic isolation reasons. Furthermore, many cloud providers use shaping to limit the bandwidth consummed by each virtual machine [7]. The

shaping configuration of each virtual machine could provide the traffic demand matrix used by our Offline TE technique to optimise the network.

In this paper, we extend the TE problem defined in [9], but with two bi-objective optimization problems. The first one is the minimization of the maximal link utilization and the sum load. The second one is the minimization of the maximal link utilization and the number of used links. The maximal link utilization appears in both problems because it is the essential metric for assessing the quality of a TE technique. In this paper, we propose different local search algorithms which are evaluated on different topologies. Our contribution is mainly focused on the design, development, analysis, and experimentation of multi-objective methods for the TE problem in data centers.

This paper is organized as follows. We first present related research in Section 2. We describe the theoretical background of MSTP, Local Search, and Multi-objective optimization in Section 3. Section 4 presents the problem statement. Section 5 describes our five proposed local search algorithms. Our evaluation is presented in Section 6 with an analysis of the experimental results, and we conclude the paper in Section 7.

2 Related Work

Many TE techniques have been proposed for improving the performance of 802.1s MSTP in switched Ethernet networks. The proposed approaches can be divided into four main classes. First, [10–12] map a set of VLANs to a given number of spanning trees. Second, [13–15] assign a set of flows to a set of given spanning trees. Third, [16] addressed the TE problem by building multiple source-based spanning trees (constructing a spanning tree for each of the given source nodes). Last, [17,18] proposes solving the TE for MSTP by finding the set of spanning trees described by customer traffic demands and a given network topology. But these techniques only cope with networks of small size, containing fewer than 30 switches. Our local search algorithm in [9] follows the same approach as in [17,18], but aims at minimizing the maximal link utilization in large Ethernet networks containing more than 500 switches. This paper is an expanded version of [9] and [19].

From the multi-objective optimization view point, [11] took into account link load balancing and admission fairness. The framework in [12] considered both network throughput and delay. Recently, [15] aimed at minimizing the n worst link loads (with n ranging up to the total number of network links) and the average link load in the network. However, the final output of these approaches is a unique solution where the second metric is only considered as a constraint or a lexicographical objective. In practice, the outcome of multi-objective algorithms is often a set of non-dominated solutions, called a Pareto front [4]. Many assessment methods for evaluating the quality of a Pareto front have been presented, such as in [20–22]. In the present paper, our local search algorithms will be evaluated using five different assessment methods (described in Section 6).

3 Background

Before describing, analyzing, and evaluating our multi-objective algorithms, we would like to clarify the concepts of Multiple Spanning Tree Protocol 802.1s, local search (LS), and multi-objective optimization (MO), which will be used throughout this paper.

3.1 Multiple Spanning Tree Protocol 802.1s

The multiple spanning tree protocol allows the utilization of several spanning trees inside a single Ethernet network. It can be used in large networks such as data centers relying on switched Ethernet. We restrict here our analysis to VLANs and Ethernet in data centers. Recently, the TRILL [23] (TRansparent Interconnection of Lots of Links) protocol has been proposed by Perlman: it has the ease of configuration of the Ethernet, while benefitting from the routing techniques provided at Layer 3 with the ability to multicast and broadcast. TRILL is currently supported by some high-end switches. In contrast, the spanning tree protocol is implemented in all commodity Ethernet switches. In this section, we describe the Spanning Tree Protocol and its variants.

First, the IEEE 802.1d Spanning Tree Protocol (STP) [24] reduces the network topology to a single spanning tree. To compute the spanning tree, special messages called Bridge Protocol Data Units (BPDUs) are exchanged between switches. Each BPDU message contains three pieces of information: the root ID (ID of the switch assumed to be the root), the transmitting bridge ID (ID of the switch transmitting the BPDU), and the weight (the weight of the least-weight path to the root from the transmitting switch). By default, the STP assigns a weight to each port and the link weight is obtained from the weights of one of its two ports. Consequently, the creation of the spanning tree is based on the least weight (shortest) path from each switch to an elected root switch. Thus, when the switch IDs and link weights are fixed, STP generates a specific spanning tree independently of the traffic demand matrices. Fortunately, one can configure the switch IDs and link weights to guide the STP to generate an intended spanning tree [9,19].

The IEEE 802.1w Rapid Spanning Tree Protocol (RSTP) [25] extends that of 802.1d to accelerate the convergence of the alternate spanning tree in the case of link failures or network topology changes. The drawback of 802.1d and 802.1w is that only a small number of links are included in a single spanning tree, since many available links are disabled. This is unfortunate in a data center network that contains many redundant links [29–31]. The IEEE 802.1q Virtual LANs (VLANs) [3] enables large Ethernet networks to be logically divided into many VLANs. VLANs are mainly used to isolate one application or one data center customer from the others. The servers (or virtual machines) which belong to a given VLAN can only communicate with other servers that belong to the same VLAN. The IEEE 802.1s Multiple Spanning Tree Protocol (MSTP)[2] is a combination of 802.1w and 802.1q, enabling service providers

to profit from the available links by spreading different spanning trees (one SP per VLAN) over a single physical topology. In practice, however, it should be noted that while some implementations of MSTP can only compute up to 16 different spanning trees, and map each VLAN onto one of these spanning trees, this does not restrict the number of VLANs suported. Most recent switches can handle up to 4K VLANs.

3.2 Local Search and Multi-objective Optimization

Solving a TE problem (such as load balancing) with the traditional exact methods in data centers with hundreds of switches is usually an impossible mission, as the search space is exponential. Local search is a powerful technique that can find high quality solutions for computationally complicated optimization problems (Vertex Cover, Traveling Salesman, or Boolean Satisfiability) in polynomial time. LS can choose an interesting set of solutions to be explored from a huge search space, by performing intelligent moves. LS starts with an initial solution and improves its quality by iteratively moving from one solution to one of the neighboring solutions. The search process is finished when some termination criterion is satisfied. An LS algorithm relies on a neighborhood function, which defines the set of neighbors for each solution. An LS algorithm also uses a heuristic (such as best improvement, first improvement, and random) to choose the neighbor. It can also use meta-heuristics, such as Tabu Search, Simulated Annealing, or Variable Neighborhood Search, to guide the search.

In multiple-objective optimization, a solution improving all the objectives should be preferred. But what if a solution's improving one of the objective values degrades the value of other objectives? For example, for some solutions, it is possible to modify them such that the number of used links and the sum load is increased, but with a reduced maximal link utilization in the network. In the context of the TE problem, network operators prefer to provide the best (possible) QoS by striking a balance between the different objectives. Thus, it is necessary to define a different concept of "optimal" for multiple-objective optimization.

Given a minimization problem $\mathcal{P}$ with m objective functions, assume that there is no priority between these m objectives. Let $s_1 = (f_{1,1}, f_{2,1}, .., f_{m,1})$ and $s_2 = (f_{1,2}, f_{2,2}, .., f_{m,2})$ be the evaluations of two solutions for $\mathcal{P}$. We say that s_1 *dominates* s_2 (*Pareto dominance*) when s_1 is at least as good as s_2 for every objective and strictly better in at least one objective:

$$s_1 \succ s_2 \Leftrightarrow \forall i \in \{1, .., m\} : f_{i,1} \leq f_{i,2}, \exists k \in \{1, .., m\} : f_{k,1} < f_{k,2}.$$

Let $\mathcal{A}$ be a multi-objective algorithm for $\mathcal{P}$ and $\mathcal{F}$ denote the set of feasible solutions generated by $\mathcal{A}$ for $\mathcal{P}$. A solution $s^* \in \mathcal{F}$ is called *Pareto optimal* if s^* is not dominated by any solution in $\mathcal{F}$. The *Pareto front* (PF) of $\mathcal{A}$ for $\mathcal{P}$ is the set of all Pareto-optimal solutions in $\mathcal{F}$. Hence, the output of multi-objective optimization is no longer a unique solution, but a set of non-dominated solutions, i.e., the Pareto front.

Different methods have been proposed for obtaining or approximating the Pareto front, such as Stochastic, Linear Programing, Preemptive Optimization, Weighting Objectives, Constraint, Goal Programming, and Genetic Algorithms. For multi-objective problems with exponential search space, it is often impossible to obtain the global Pareto front. Stochastic methods (such as LS) are very efficient at approximating the Pareto front. In this paper, we use the term Pareto front also to denote an approximate Pareto front obtained with an LS algorithm.

4 Problem Statement

We consider our Ethernet network as an undirected graph $G = (N, E)$, where N is the set of switches and E is the set of links between switches. We assume there is at most one link between each pair of points. If it were not the case, the multiple links are aggregated into a single link. We call BW the matrix of link bandwidths (note that $BW[i,j] = BW[j,i]$). Let $V = \{V_1, V_2, \ldots, V_k\}$ be the set of k given VLANs in the network, with $V_r = (N_r, E_r)(N_r \subseteq N$, $E_r = \{(i,j) \in E | i,j \in N_r\}$, $1 \leq r \leq k)$. Each graph V_r is assumed to be connected. We call $TD = \{TD_1, TD_2, \ldots, TD_k\}$ the set of k traffic demand matrices where $TD_r[i,j]$ represents the traffic that switch i sends to switch j in VLAN V_r $(1 \leq r \leq k;\ i,j \in N_r)$.
Let $W = \{W_1, W_2, \ldots, W_k\}$ be the set of link weight matrices for the k given VLANs. We denote by $MSP(G,V,W)$ the set of k spanning trees $SP_1(G,V_1,W_1)$, $SP_2(G,V_2,W_2)$,..., $SP_k(G,V_k,W_k)$ obtained by the Multiple Spanning Tree Protocol [2] on graph G, with set of VLANs V and set of link weight matrices W.
The Ethernet switching problem here is to distribute each traffic demand $TD_r[i,j] > 0$ $(1 \leq r \leq k;\ i,j \in N_r)$ over a unique path from i to j in $SP_r(G,V_r,W_r)$. Obviously, the only set that can be configured to change the solution (set of k spanning trees) is W. Our three measures (U_{max}, $SumL$ and $\#Links$) on each solution are defined as follows:
First, assume that $L[i,j]$ $(i,j \in N)$ denotes the load (sum of traffic flow) on link (i,j). For the computation of $L[i,j]$, the traffic flow is directed ($L[i,j] \neq L[j,i]$). The utilization of a link (i,j) is the ratio between its load and its bandwidth:

$$U[i,j] = L[i,j]/BW[i,j]\ (i,j \in N)$$

The link utilization is also directed. The link (i,j) is overloaded if its load is greater than its bandwidth ($U[i,j] > 1$ or $U[j,i] > 1$). In practice, a link will be called overloaded when its utilization reaches 0.9. Let U_{max} be the maximal link utilization:

$$U_{max} = max\{U[i,j]\}\ (i,j \in N)$$

Second, we denote by $SumL$ the sum of all link loads in the network: $SumL = \sum L[i,j]\ (i,j \in N)$.

Third, let $\#Links$ denote the number of used links in the network. A link (i, j) is considered to be a used link if $L[i, j] + L[j, i] > 0$.

Among these three measures, the maximal link utilization value U_{max} is the most important for evaluating a TE solution. The ultimate objective is to avoid congestion in the network (keep the value of U_{max} under 100%). The second measure of the sum of all link loads $SumL$ is an important criterion for assessing the total traffic that flows on each link in the network and also to assess the network delay in some sense. The number of used links $\#Links$ is very significant in terms of network backup in case of link failures and energy efficiency in data centers [26]. It is impossible to obtain an optimal solution which minimizes U_{max}, $SumL$, and $\#Links$ at the same time. In our experiments, an improvement in $SumL$ or $\#Links$ often leads to a degradation of U_{max}. In addition, taking simultaneously three objective functions into account for a TE problem on large data centers is a very complex task. For this reason, we divide these three objectives into two pairs of objectives: (U_{max} , $SumL$) and (U_{max} , $\#Links$). The formulations of these two bi-objective optimizations are the following:

Input: Graph $G = (N, E)$, set of k VLANs V, bandwidth matrix BW, set of k traffic demand matrices TD

Output:

- Bi-objective 1 (Minimization of U_{max} and $SumL$): A Pareto front $PF_{(U_{max},SumL)}$, each solution in $PF_{(U_{max},SumL)}$ is a set of k spanning trees minimizing the pair of objectives U_{max} and $SumL$.
- Bi-objective 2 (Minimization of U_{max} and $\#Links$): A Pareto front $PF_{(U_{max},\#Links)}$, each solution in $PF_{(U_{max},\#Links)}$ is a set of k spanning trees minimizing the pair of objectives U_{max} and $\#Links$.
- Given a set of k spanning trees SP_i, determine a set of k weight matrices W_i $(1 \leq i \leq k)$ such that $MSP(W_i)$ generates the spanning tree SP_i.

We describe in Section 6 the assessment methods for evaluating the quality of the Pareto front obtained by our different LS algorithms (in Section 5) for solving these two bi-objective problems.

5 Proposed Algorithms

In this paper, we present five LS algorithms dealing with the TE problems described in Section 4. We first implement three LS algorithms that take into account each of the three objectives to be minimized (U_{max}, $SumL$ and $\#Links$). To obtain a bi-objective Pareto front with these algorithms, we add an observation to store the non-dominated solutions in their search process. Next, two bi-objective algorithms are proposed based on the three single objective methods. We will refer to the three single-objective minimization algorithms as $Min_{U_{max}}$, Min_{SumL}, and $Min_{\#Links}$; the two bi-objective algorithms will be referred to as $Min_{(U_{max},SumL)}$ and $Min_{(U_{max},\#Links)}$.

5.1 Generating Weight Matrices

In Section 3.1, we have stated that the MSTP 802.1s creates the spanning trees based on two set of parameters: the switch IDs and the link weights. With 802.1s, the link weight is an integer in $[1..2^{16}-1]$. Given a network with k VLANs, m links, and n switches in each VLAN, even if we do not consider the selection of the root for each spanning tree, the size of the search space if we perform a search on the link weights is $(2^{16}-1)^{mk}$. In addition, it is difficult to control the change of link weights on the spanning tree. Dealing with this link weight problem, we found that in each VLAN, we can force 802.1s to yield any desired spanning tree simply by assigning a unit weight to all the links in a spanning tree and assigning a weight of n to all the other links in the VLAN.
Furthermore, we showed in [19] that the root determination has no impact on the single path between any pair of nodes in the spanning tree and thus has no impact on U_{max}, $SumL$, or $\#Links$. By fixing a unique root for each VLAN and performing the search on spanning trees, we can reduce the size of the search space to $\binom{m}{n-1}^k$ solutions. This is the choice we made for all the algorithms described in this section. The solution obtained by using the default 802.1s MSTP is used as the initial solution for all five of our LS algorithms.

5.2 Single-objective Approaches

Minimization of U_{max}

Our LS algorithm in [9] deals with the first objective, of minimizing U_{max} in large data centers. Experiments were conducted on different data centers topologies containing up to 564 switches and 16 VLANs with different traffic demand matrices that are synthesized from the data center studies in [27]. The algorithm can however handle a larger number of VLANs as its complexity is not related to this parameter. The solutions obtained with $Min_{U_{max}}$ could reduce by 133.26% the value of U_{max} compared to the solution given by the 802.1s Multiple Spanning Tree Protocol standard in our experiments (see Table 1, reduction of U_{max}: 233.26-100 = 133.26%).

This $Min_{U_{max}}$ LS relied on an edge exchange in each search iteration. First, we find the most congested link (s_{max}, t_{max}) (having U_{max}) in the network. Then, we apply a heuristic for selecting a spanning tree $SP_{selected}$ of a VLAN that contains (s_{max}, t_{max}) based on its load on this most congested link. Other heuristics are applied to select an edge $(s_0, t_0) \in SP_{selected}$ to be removed and to select an edge $(s_I, t_I) \in SP_{selected}$ to be added that maintains the spanning tree $SP_{selected}$ and minimizes the value of U_{max}. This edge exchange is iterated until a given time limit is reached. We also use a tabu list [5] for preventing the replacement of the same couple of edges in successive iterations.

By proving that for each edge replacement, the load changes only on the links belonged to the cycle created by adding an edge into the spanning tree

[19], we can speed up the link load recomputation cost in each search iteration. Other speeding up techniques are also applied to representing, querying, and updating the spanning trees.

Minimization of SumL

The Optimum Communication Spanning Tree Problem (OCT) [28] can be considered as a sub problem of this minimization of sum load in network with one VLAN containing all its switches. OCT is a very challenging combinatorial optimization problem and has been proven to be NP-hard even with a unit traffic demand matrix ($TD[i,j] = 1, \forall i \neq j$). Our Min_{SumL} LS is inspired by the $Min_{U_{max}}$ algorithm with the same definitions of the search space and speeding up techniques.

The key design of each LS algorithm is the definition of the neighborhood. Min_{SumL} also relies on an edge exchange in each search iteration to move from one solution to another. To find a heuristic for this LS, we represent $SumL$ by another formula:

$$SumL = \sum_{i,j \in N} L[i,j] = \sum_{\substack{r \in [1..k] \\ i,j \in V_r}} TD_r[i,j] * PathLength_r[i,j],$$

where k is the number of VLANs, and $PathLength_r[i,j]$ is the number of links on the path from i to j in the VLAN V_r.

With this new definition, we can state that the overall sum load $SumL$ can be changed by modifying the most contributing pair (i,j), in which the sum load over path $TD_r[i,j] * PathLength_r[i,j]$ is large. In each search iteration, we first choose a pair (i,j) from x pairs having the largest sum load over path $TD_r[i,j] * PathLength_r[i,j]$. Next, we apply a heuristic for selecting the edge to be removed on the path from i to j and the edge to be added for minimizing the value of $SumL$. We also use time as the termination criterion, and a tabu list for this LS.

The speeding up techniques in $Min_{U_{max}}$ are reused to accelerate the search. The main computation in this Min_{SumL} is the computation and comparing $SumL$ in each search iteration. Obviously, the traffic demand matrices are unchangeable, so the computation of path length is the most important task. Fortunately, we can maintain incrementally the path length between each pair of switches, relying on the "magic cycle" described in [19].

When the traffic demand matrices are nearly uniform, the minimization of $SumL$ is equivalent to the minimization of the average path length between each pair of switches in the network. For this reason, the balance spanning tree construction of 802.1d STP [24] with the root in the center is also a very efficient method to reduce the overall sum load. Thus, with the¿ initial solution obtained by 802.1s, Min_{SumL} reduces by by 0.57% the value of $SumL$ (see Table 1, reduction of $SumL$: 100.57-100 = 0.57%), but compared to a random initial solution, this reduction is about 12% in our experiments.

Minimization of #Links

For this $Min_{\#Links}$, we also iterate a move to a neighbor solution by performing an edge exchange in each search iteration. Our heuristic for defining the neighborhood for $Min_{\#Links}$ is based on the following observation.

The number of links used by a set of k spanning trees can be reduced only if in a spanning tree SP_i, we replace an edge used only by SP_i by an edge used by another spanning tree SP_j.

Let the $numVLAN$ of an edge (s,t) be the number of VLANs that have (s,t) in their spanning trees. From the initial solution obtained with 802.1s, a set of removable edge candidates RM is created from the edges with the first x minimal values of $numVLAN$. In each search iteration, we randomly select an edge (s_0,t_0) in RM to be replaced by an edge (s_I,t_I). We accept only a replacement that reduces or maintains the current $\#Links$. A random selection of (s_0,t_0) is performed after a given number of non-improving iterations.

In our experiments, the solutions obtained by $Min_{\#Links}$ can reduce by about 18.51% the value of $\#Links$ compared to the solution given by 802.1s (see table 1, reduction of $\#Links$: 118.51-100 = 18.51%).

Solution quality

In Table 1, we measure the performance of each single objective algorithm compared to the default configuration of the MSTP 802.1s standard. The data set for these experiments is described in Section 6.2. In our tests, a triple measure of $(U_{max}, SumL, \#Links)$ is performed on the best solution given by the three single objective LS. We assume a value of 100% for the minimum of U_{max}, $SumL$, and $\#Links$. If we only consider the visible improvement in each single objective compared to the MSTP 802.1s standard, then $Min_{U_{max}}$ seems to be the best algorithm and Min_{SumL} is possibly the worst one. However, in order to reduce by 233.26% the value of U_{max} compared to 802.1s, $Min_{U_{max}}$ uses 125.6% more links than $Min_{\#Links}$ (about 500 more links in a data center with 2000 links) and increases $SumL$ by 120.25% more than Min_{SumL}.

Table 1: Single objective algorithm overall performance

%\Alg.	802.1s	$Min_{U_{max}}$	Min_{SumL}	$Min_{\#Links}$
U_{max}	233.26	**100**	221.69	346.32
$SumL$	100.57	120.25	**100**	131.69
$\#Links$	118.51	125.6	119.29	**100**

It is even worse for $Min_{\#Links}$, where its best solution increases U_{max} by 346.32% over that given by $Min_{U_{max}}$, and $SumL$ by 131.69% compared to that obtained by Min_{SumL} in order to reduce 118.51% $\#Links$ from the initial solution given by 802.1s. Hence, the overall solution quality of an algorithm cannot be evaluated by its best solution for a single objective.

5.3 Bi-objective Optimization

Of our three single objective LS, we found that $Min_{U_{max}}$ converges much faster than Min_{SumL} or $Min_{\#Links}$. Fig. 1 depicts the improvement of U_{max} over time of $Min_{U_{max}}$ for a data center with 564 switches and 16 VLANs with a time limit of 15 minutes. $Min_{U_{max}}$ reduces U_{max} by 52.2% compared to that of 802.1s (from 0.69 to 0.33) after only 10 s. With the same test, the improvement within the first 10 s of Min_{SumL} is 3.6% and for $Min_{\#Links}$, 1.77%.

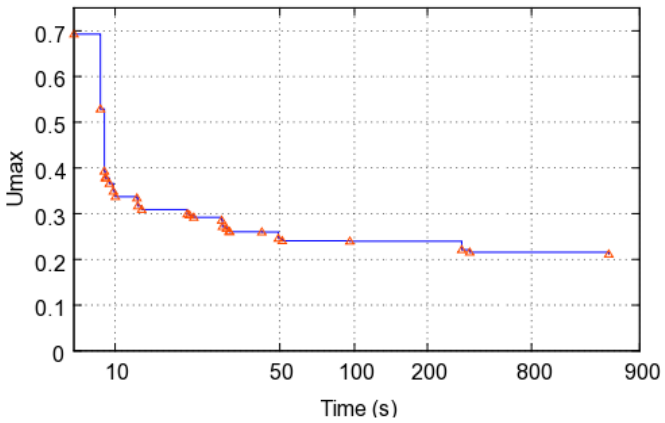


Fig. 1: Improvement of U_{max} over execution time

When we look at the number of improvements that each algorithm makes within the first 10 s, $Min_{U_{max}}$ improved its objective value only 6 times while this number is 18 for Min_{SumL} and 13 for $Min_{\#Links}$. This convergence observation motivates our algorithmic approach to the two bi-objective optimizations: $Min_{(U_{max},SumL)}$ and $Min_{(U_{max},\#Links)}$. Here, we do not consider the bi-objective $(SumL, \#Links)$ because U_{max} is the essential metric in a TE problem and we cannot guarantee network congestion avoidance with only an optimization of $(SumL, \#Links)$.

Minimization of U_{max} and SumL

Algorithm 1 describes how the heuristics of two single objective LSs are included in the search process of the bi-objective LS. In this case, we combine $Min_{U_{max}}$ with Min_{SumL} to define the $Min_{(U_{max},SumL)}$ algorithm. In $Min_{(U_{max},SumL)}$, $SumL$ is considered as Objective 1 to be regularly minimized (Line 8). The $edgeExchangeHeuristicObj1(MSP, G, V)$ returns a neighboring solution of the current solution MSP that minimizes the $SumL$ value

by applying the edge exchange heuristic of Min_{SumL}. With its fast convergence, the edge exchange heuristics of $Min_{U_{max}}$ are periodically applied in the method $edgeExchangeHeuristicObj2(MSP, G, V)$ (Line 10) after each $\#stepObj1$ iterations to guide the search by minimizing the value of Objective 2—U_{max}. The $\#stepObj1$ in Line 7 is an important parameter for configuring to balance the convergence of U_{max} and $SumL$.

Algorithm 1: Bi-objective Algorithms

```
1  MSP = getInitialSolution(G, V);                    /* by MSTP 802.1s standard */
2  PF = {MSP};                                        /* Insert MSP to Pareto front */
3  #iterations = 0;
4  #nonImprovements = 0;
5  while time_exec < time_windows do
6  |   #iterations = #iterations + 1;
7  |   if #iterations % #stepObj1 ≠ 0 then
8  |   |   MSP = edgeExchangeHeuristicObj1(MSP, G, V) ;
9  |   else
10 |   |   MSP = edgeExchangeHeuristicObj2(MSP, G, V) ;
11 |   end
12 |   if MSP is dominated by PF then
13 |   |   #nonImprovements = #nonImprovements + 1;
14 |   |   if #nonImprovements > #acceptIterations then
15 |   |   |   MSP = edgeExchangeRandom(MSP, G, V);
16 |   |   end
17 |   else
18 |   |   #nonImprovements = 0;
19 |   |   PF = updateParetoFront(PF, MSP);
20 |   end
21 end
```

The Pareto front is updated with the non-dominated solutions found during the search process (Line 19). We diversify the search with a random edge replacement after a given number of non-improving search iterations (Lines 13–16).

Minimization of U_{max} and $\#Links$

The $Min_{(U_{max},\#Links)}$ LS is also defined by the pseudo code in Algorithm 1. Here, $\#Links$ becomes Objective 1 and we apply the $Min_{U_{max}}$ edge exchange heuristic to guide the search after every $\#stepObj1$. A random move is also applied to diversify the search when no new non-dominated solutions are found in a given number of successive search iterations.

Both $Min_{(U_{max},SumL)}$ and $Min_{(U_{max},\#Links)}$ share the same design described in Algorithm 1 but they are very different in their exploration of the huge search space: they apply different definitions of neighbors. In our experiments, the configuration of the parameter $\#acceptIterations$ is the same for both problems (50), but different for $\#stepObj1$ (50 for $Min_{(U_{max},SumL)}$ and 10 for $Min_{(U_{max},\#Links)}$).

Finding a Pareto front with a single objective algorithm

In the assessment of our bi-objective optimization algorithms, we would like to compare their performance with single objective algorithms. Such a comparison is possible by extending our single objective algorithms to compute a Pareto front. The idea is just to observe a second objective and compute a Pareto front. In Algorithm 2, we present how a Pareto front PF can be obtained with a single objective algorithm. Here, an improvement in the unique objective is normally stored (Lines 8–10) but a dominance check is also added in each search iteration to update the Pareto front PF with the second objective (Lines 12, 13). By adding this observation, we have four more Pareto fronts in each experiment: two PFs of (U_{max}, $SumL$) obtained with $Min_{U_{max}}$ and Min_{SumL}; and two PFs of (U_{max}, $\#Links$) obtained with $Min_{U_{max}}$ and $Min_{\#Links}$.

In the next section, we will use different assessment methods for evaluating our proposed algorithms on different network topologies.

Algorithm 2: Finding PF with Single Objective Algorithms

```
1  MSP = getInitialSolution(G, V);                    /* by MSTP 802.1s standard */
2  PF = {MSP};                                        /* insert MSP to Pareto front */
3  MSP* = MSP;                                        /* MSP* best solution wrt obj1 */
4  bestObj1Value = getObj1Value(MSP);
5  while time_exec < time_windows do
6  |   MSP = edgeExchangeHeuristicObj1(MSP, G, V) ;
7  |   obj1Value = getObj1Value(MSP);
8  |   if bestObj1Value > obj1Value then              /* stock improvement */
9  |   |   bestObj1Value=obj1Value;
10 |   |   MSP* = MSP;
11 |   end
12 |   if MSP is not dominated by PF then    /* check dominance to update PF */
13 |   |   PF = updateParetoFront(PF, MSP);
14 |   end
15 end
```

6 Performance Comparisons

This section is organized as follows. We first describe the evaluation methods for multi-objective approaches in Section 6.1. Next, the network topologies and traffic demand matrices for our tests are presented in Section 6.2. Last, our experimental results are discussed in Section 6.3.

6.1 Assessment Metrics for Multi-objective Algorithms

Many measurements can be made for evaluating the outcomes of multi-objective algorithms. In this paper, we consider five metrics to use in comparing the

performances of our local search approaches: Purity [20], Spread Metric [20], Hypervolume [21], Hypervolume-Star, and Binary ϵ-indicator [22]. These five metrics are very useful for assessing the quality of an approximation set or a Pareto front. In this paper, we denote by PF_{obt} the Pareto front produced by the considered algorithm and PF_{ref} the reference (or best-known) Pareto front. PF_{ref} is the best Pareto front, combining the Pareto fronts PF_{obt} of all the considered algorithms. In this section, we assume without loss of generality that all the multi-objective algorithms aim at solving the minimization problem with two objectives.

6.1.1 Purity

The purity metric [20] measures the percentage of solutions of PF_{obt} that are both in PF_{obt} and in PF_{ref}:

$$Purity = \frac{|PF_{\text{obt}} \bigcap PF_{\text{ref}}|}{|PF_{\text{obt}}|},$$

where $|PF_{\text{obt}}|$ is the number of solutions of PF_{obt} and $|PF_{\text{obt}} \bigcap PF_{\text{ref}}|$ is the number of solutions contained both in PF_{obt} and in PF_{ref}.

Obviously, $Purity \in [0, 1]$, $Purity$ equals 0 when PF_{obt} and PF_{ref} do not share any common solution (all the solutions of PF_{obt} are dominated by some solutions of PF_{ref}), $Purity$ equals 1 if PF_{obt} is completely included in PF_{ref} (no other algorithm was able to compute a solution dominating a solution of PF_{obt}), and a large $Purity$ means a good multi-objective algorithm. This metric is significant when both PF_{obt} and PF_{ref} have many solutions. However, if $PF_{\text{obt}} = \{s^*\}$ and $s^* \in PF_{\text{ref}}$, then $Purity = 1$. In this case, we have no guarantee of the quality of the considered algorithm.

6.1.2 Spread Metric

The spread metric (Γ) [20] measures the maximum size of the "holes" of a Pareto front. Suppose given $PF_{\text{obt}} = \{s_1, s_2, .., s_n\}$ with $s_i = (f_{i,1}, f_{i,2})$. We assume that the objective values are normalized and are sorted in increasing order for objective 1 and in decreasing order for objective 2. We consider two extreme points s_0 and s_{n+1} of PF_{obt} such that

$$f_{0,1} \leq f_{1,1} \text{ and } f_{0,2} \geq f_{1,2},$$
$$f_{n+1,1} \geq f_{n,1} \text{ and } f_{n+1,2} \leq f_{n,2}.$$

The δ metric is defined by

$$\delta_{i,j} = |f_{i,j} - f_{i-1,j}|,\ i \in \{1, .., n+1\},\ j \in \{1, 2\},$$

where $f_{i,j}$ is the value of objective function j of solution $s_i \in PF_{\text{obt}} \bigcup \{s_0, s_{n+1}\}$. The spread metric Γ is defined by

$$\Gamma = max\{i \in \{1, .., n+1\}, j \in \{1, 2\}; \delta_{i,j}\}.$$

Fig. 2b depicts an example of a Γ measurement with $m = 3$, Γ takes the largest value $\delta_{4,1}$.

By this definition, PF_{obt} is good if its maximal hole Γ is small. However, we found that PF_{obt1} could have a smaller Γ than PF_{obt2} even if all the solutions of PF_{obt1} are dominated by some solutions of PF_{obt2}. Thus, we define a new spread metric Γ^*, measuring the spread (or coverage) of PF_{obt} over a global Pareto front PF_{ref}.
We use two extreme points s_0 and s_{n+1} of PF_{ref} to compute both $\Gamma(PF_{\text{obt}} \bigcap PF_{\text{ref}})$ and $\Gamma(PF_{\text{ref}})$. The Γ^* metric is defined by

$$\Gamma^* = \frac{\Gamma(PF_{\text{ref}})}{\Gamma(PF_{\text{obt}} \bigcap PF_{\text{ref}})}.$$

If $PF_{\text{obt}} \bigcap PF_{\text{ref}} = \emptyset$, then Γ^* is undefined. So we set $\Gamma^* = 0$ as the worst case when PF_{obt} does not contain any solution in PF_{ref}. With this assumption, $\Gamma^* \in [0, 1]$. Γ^* equals 1 (ideal) if $PF_{\text{obt}} = PF_{\text{ref}}$.

Finding Extreme Points for Γ^*
In [20], an experimental method was proposed for computing the two extreme points for Γ by selecting the pair of objective values corresponding to the highest pairwise distance. This selection must be performed on the non-dominated solution set over all runs in order to obtain the two final extreme points. This method is expensive and incompatible with the computation of our new metric Γ^*.
For the computation of Γ^*, we found that the two extreme points cannot belong to PF_{obt} or PF_{ref}. Let $s_0 = (f_{0,1}, f_{0,2})$ and $s_{n+1} = (f_{n+1,1}, f_{n+1,2})$ be two extreme points for $PF_{\text{ref}} = \{s_1, s_2, .., s_n\}$ with $s_i = (f_{i,1}, f_{i,2})$ and $n > 1$. If $n = 1$, the extreme points do not need to be defined, $\Gamma^* = 0$ if $PF_{\text{obt}} \bigcap PF_{\text{ref}} = \emptyset$ and $\Gamma^* = 1$ if $PF_{\text{obt}} \bigcap PF_{\text{ref}} = PF_{\text{ref}}$. We assume that the objective values are normalized and are sorted in increasing order for objective 1 and in decreasing order for objective 2. We put $Dist_1 = \frac{f_{n,1} - f_{1,1}}{n-1}$ and $Dist_2 = \frac{f_{1,2} - f_{n,2}}{n-1}$. The extreme point values $f_{0,1}, f_{0,2}, f_{n+1,1}$ and $f_{n+1,2}$ are defined by

$$f_{1,1} - Dist_1 \leq f_{0,1} \leq f_{1,1} \text{ and } f_{1,2} < f_{0,2} \leq f_{1,2} + Dist_2,$$
$$f_{n,1} < f_{n+1,1} \leq f_{n,1} + Dist_1 \text{ and } f_{n,2} - Dist_2 \leq f_{n+1,2} \leq f_{n,2} \quad (1)$$

Our definition of $f_{0,1}, f_{0,2}, f_{n+1,1}$ and $f_{n+1,2}$ in (1) ensures that the distance between s_0 - s_1 and s_n - s_{n+1} is always less than or equal Γ. In addition, this definition of extreme points is more compatible with Γ^* because it allows us to better represent the contribution of PF_{obt} to PF_{ref}. In this paper, we fix $s_0 = (f_{1,1}, f_{1,2} + Dist_2)$ and $s_{n+1} = (f_{n,1} + Dist_1, f_{n,2})$.

6.1.3 Hypervolume

The hypervolume (HP) metric [21] measures the "volume" of the dominated region defined by a Pareto front and a reference point $s^{\max}$. Suppose given $PF_{\text{obt}} = \{s_1, s_2, .., s_n\}$ with $s_i = (f_{i,1}, f_{i,2})$. Let $s^{\max} = (f_1^{max}, f_2^{max})$ be a reference point such that $f_1^{max} \geq max_{i \in \{1,..,n\}}\{f_{i,1}\}$ and $f_2^{max} \geq max_{i \in \{1,..,n\}}\{f_{i,2}\}$.

HP is defined as the surface area of the union of all the rectangles defined by each of the solutions in PF_{obt} and the reference point $s^{\max}$.

Fig. 2a presents an example of the hypervolume computation of a Pareto front obtained from a bi-objective algorithm. In this paper, we normalize the values to [0,1] for each objective ($s^{\max} = (1,1)$). Thus, $HP \in [0,1]$. With HP, the quality of a Pareto front is evaluated by the surface of the dominated region which it covers. A method with a large HP is considered better than a method with a smaller HP. HP is useful for comparing the performance of different algorithms solving a specific multi-objective problem. However, with two different reference points, a Pareto front PF_{obt1} could cover a bigger region than a Pareto front PF_{obt2} even when the solutions of PF_{obt1} are completely dominated by the solutions of PF_{obt2}. So, a unique reference point is required when comparing different Pareto fronts.

Finding the Reference Point s^{max}

In [21], the authors simply used the point (0,0) in the context of a maximization problem as the reference point for computing the hypervolume HP of its Pareto front. However, if the considered Pareto fronts are too far from (0,0), the "volume" of HP consists of a large surface of the insignificant region that can make the HP metric inexact. We have the same problem with the minimization problem if the reference point is too far from the Pareto front considered.

For a Pareto front $PF_{\text{obt}} = \{s_1, s_2, .., s_n\}$ with $s_i = (f_{i,1}, f_{i,2})$, we define its upper bound point $s^* = (f_1^*, f_2^*)$ as

$$\begin{aligned} &f_1^* = f_{n,1} + Dist_1,\ f_2^* = f_{1,2} + Dist_2, \text{ if } n > 1, \\ &f_1^* = f_{1,1} + \tfrac{1}{2}.max(f_{1,1}, f_{1,2}),\ f_2^* = f_{1,2} + \tfrac{1}{2}.max(f_{1,1}, f_{1,2}), \text{ if } n = 1 \end{aligned} \quad (2)$$

Given m Pareto fronts to be compared, $PF_{\text{obt1}}, PF_{\text{obt2}}, ..., PF_{\text{obt}m}$, let $s_i^* = (f_{i,1}^*, f_{i,2}^*)$ be the upper bound point of $PF_{\text{obt}i}$. The unique reference point $s^{\max} = (f_1^{max}, f_2^{max})$ for computing the HP of these Pareto fronts is defined by

$$f_1^{max} = max_{i \in [1..m]}(f_{i,1}^*) \text{ and } f_2^{max} = max_{i \in [1..m]}(f_{i,2}^*).$$

Our definition of reference point for HP is based on the solution values of all the Pareto fronts considered. In (2), an additional distance is added to the maximal values of each objective ensuring that s_1 and s_n contribute to the HP of each Pareto front considered.

6.1.4 Hypervolume-Star

The hypervolume-star (HP^*) metric is a combination of the purity and the hypervolume metric. We introduce this new hypervolume metric to disable the dependence of the purity on the number of solutions. We define the HP^* metric of a Pareto front PF_{obt} over a global Pareto front PF_{ref} by

$$HP^* = \frac{HP(PF_{\text{obt}} \bigcap PF_{\text{ref}})}{HP(PF_{\text{ref}})}.$$

We have $HP^* \in [0,1]$ because $PF_{\text{obt}} \bigcap PF_{\text{ref}} \subseteq PF_{\text{ref}}$. HP^* measures the contribution of PF_{obt} in the hypervolume PF_{ref}. HP^* equals 0 if $PF_{\text{obt}} \bigcap PF_{\text{ref}} = \emptyset$ (PF_{obt} and PF_{ref} do not share any common solution). HP^* equals 1 if $PF_{\text{obt}} \bigcap PF_{\text{ref}} = PF_{\text{ref}}$ (ideal when PF_{obt} is the global Pareto front PF_{ref}). In the computation of HP^*, one should use the same reference point $s^{\max}$ for computing $HP(PF_{\text{obt}} \bigcap PF_{\text{ref}})$ as for $HP(PF_{\text{ref}})$.

For computing HP*, we only consider the subset $PF_{\text{obt}} \bigcap PF_{\text{ref}}$ of PF_{ref}. Thus, the upper bound point s^*_{ref} of PF_{ref} (computed by formula (2)) is used as its reference point $s^{\max}$ ($s^{\max} = s^*_{ref}$).

6.1.5 Binary ϵ-indicator

For a bi-objective minimization problem, we define when a solution $s_1 = (f_{1,1}, f_{1,2})$ $\epsilon-$ dominates a solution $s_2 = (f_{2,1}, f_{2,2})$, denoted as $s_1 \succeq_\epsilon s_2$ as follows:

$$(s_1 \succeq_\epsilon s_2) \Leftrightarrow \forall i \in \{1,2\} : f_{i,1} \leq \epsilon . f_{i,2}$$

The ϵ-indicator $I(PF_{\text{obt}}, PF_{\text{ref}})$ is the minimum factor ϵ such that for any solution s_2 in PF_{ref} there exist at least one solution s_1 in PF_{obt} that is not worse by a factor ϵ in all objectives [22].

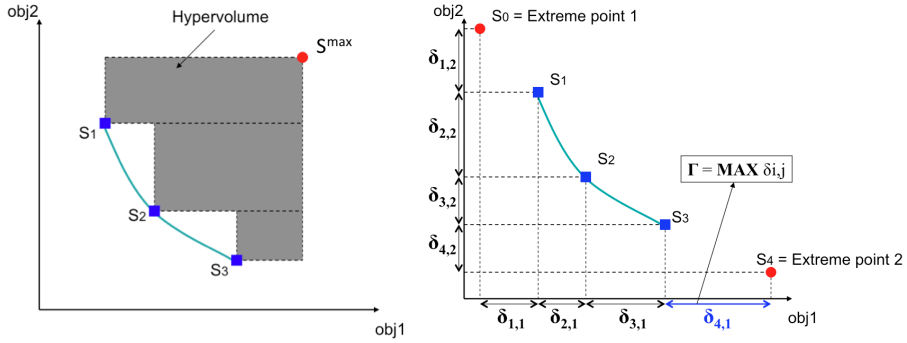
$$I(PF_{\text{obt}}, PF_{\text{ref}}) = \inf_{\epsilon \in \mathbb{R}} \{\forall s_2 \in PF_{\text{ref}}, \exists s_1 \in PF_{\text{obt}} : s_1 \succeq_\epsilon s_2\}$$

We have $I(PF_{\text{obt}}, PF_{\text{ref}}) \in (0,1]$. The ϵ-indicator $I(PF_{\text{ref}}, PF_{\text{obt}})$ compares the distance of PF_{obt} with PF_{ref}. With this metric, when comparing with PF_{ref}, the ideal value is $I(PF_{\text{ref}}, PF_{\text{obt}}) = 1$. This occurs when $PF_{\text{obt}} \subseteq PF_{\text{ref}}$: none of the solutions of PF_{obt} is dominated by a solution of PF_{ref}. All of the solutions of PF_{obt} are thus in the global Pareto front PF_{ref}. As to the purity, the limitation of this measure is that it does not consider the number of solutions of PF_{obt}. If $PF_{\text{obt}} = \{s^*\}$ and $s^* \in PF_{\text{ref}}$, then $I(PF_{\text{ref}}, PF_{\text{obt}}) = 1$ although one cannot thus assess the quality of PF_{obt}.

6.2 Data Set Composition

The seven data center topologies for testing our algorithms are described in Table 2. The private enterprise topology is a 3-Tier Cisco architecture [29] while the cloud data center uses the 3-Tier textbook data center architecture in [30]. In Table 2, the two VLAN generation types Geographic (Geo.) and Random (Ran.) are used to create different tests with the same network topology (same nodes, same edges, but different VLAN generation). In the Geographic case, each VLAN is generated geographically by grouping a set of neighboring racks of switches (i.e., the racks of servers in the same or neighboring buildings). In the Random case, we assume that the racks are assigned randomly to the different VLANs depending on their increasing need. The details of the Private (PR) and Cloud (CL) architectures are presented in [9].

We also use the data center topologies Fat Tree (FT) [31], PortLand (PL) [32], and the generic topology Expanded Tree (ET) in [19]. To simplify the



(a) Hypervolume (b) Gamma Measurement

Fig. 2: Measurement methods

Table 2: Data Set Composition

Size \ Type	PR Geo\Ran.	CL Geo.\Ran.	FT Ran.	PL Ran.	ET Ran.
Num. Nodes	242	564	320	313	200
Num. Links	549	2024	2048	602	398

observations with five different assessment metrics, we generate 16 VLANs and a near-uniform traffic demand matrix (see [9]) for each VLAN for every topology type in our experiments. These data sets are available online in [33]. The time limit for running our algorithms was fixed at 30 minutes for all the network topologies considered. As our algorithms are non deterministic, each algorithm has been executed 10 times on each instance. Note that all our algorithms can handle a larger number of VLANs and this parameter does not influence the complexity of the algorithms.

6.3 Results and Evaluation

In this section, the Pareto fronts obtained by our five proposed LS algorithms in Section 5 are evaluated on the same data sets (Table 2). For each bi-objective problem, PF_{ref} is the Pareto front which is the union of all the solutions produced by all the algorithms considered. In each test, the Pareto fronts PF_{obt} obtained by $Min_{U_{max}}$, Min_{SumL}, and $Min_{(U_{max},SumL)}$ are compared in the bi-objective 1 problem: minimization of U_{max} and $SumL$. In the same manner, the Pareto fronts PF_{obt} obtained by $Min_{U_{max}}$, $Min_{\#Links}$, and $Min_{(U_{max},\#Links)}$ are evaluated in the bi-objective 2 problem: minimization of U_{max} and $\#Links$. In this section, we assume without loss of generality that all the objective values are normalized to [0,1]. Thus, in each test, the extreme point 1 is $s_0(0,1)$, the extreme point 2 is $s_{n+1}(1,0)$ and the reference point is $s^{\max}(1,1)$. Each Pareto front is assessed by the five metrics described

in Section 6.1: Purity, Hypervolume (HP), Hypervolum-Star (HP*), Binary ϵ-indicator (I), and Spread Metric (Γ^*). The values of each metric are in [0,1], and the value 0 is the worst and 1 is the ideal.

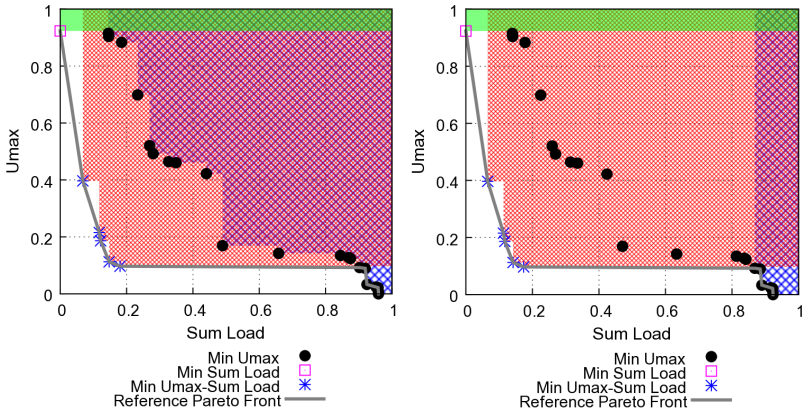
Our detailed results for the seven network topologies in Table 2 are presented in Tables 5 to 18 (see the Appendix) with these five assessment metrics. Each table describes the evaluation for one bi-objective problem $(U_{max}, SumL)$ or $(U_{max}, \#Links)$ on a given test. Table 3 represents the overall performance for the bi-objective problem 1 $(U_{max}, SumL)$ on all the network topologies. Table 4 provides the same information for $(U_{max}, \#Links)$. In the tables, we report the number of solutions in each Pareto front ($\#Sol$) as a reference assessment metric for better understanding the outcomes given by the five principal metrics. In each table, we also provide the average value (μ) and the standard deviation (σ) of each metric over the 10 executions of the algorithms. We also provide the classical interval $[\mu-2\sigma, \mu+2\sigma]$. When the metric belongs to $[0, 1]$, this interval must be included in $[0, 1]$.

Bi-objective problem 1: Minimization of U_{max} and $SumL$

In Table 3, we can see that none of the five metrics of $Min_{U_{max}}$ is the best, although its PF_{obt} has more than 24 solutions. Min_{SumL} has the best value for the metrics $Purity$ and I. However, these metrics are significant only when PF_{obt} has many solutions but PF_{obt} of Min_{SumL} has on average only 1.5 solutions. $Min_{(U_{max},SumL)}$ has the best value for three metrics: HP, HP*, and Γ^*, and its values for $Purity$ and I are very close to those of Min_{SumL}, with more than 12 solutions in its PF_{obt}. Except for HP, $Min_{(U_{max},SumL)}$ has a smaller standard deviation than $Min_{U_{max}}$ on the different metrics. The $Min_{(U_{max},SumL)}$ algorithm is thus more stable. As expected, $Min_{(U_{max},SumL)}$ is the best method for solving this bi-objective problem.

Table 3: Performance Comparison U_{max}, $SumL$

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	24.3	0.2884	0.6788	0.3971	0.8830	0.4986
	σ	7.8	0.1947	0.0926	0.2024	0.0967	0.2606
	$\mu-2\sigma$	8.7	0	0.4936	0	0.6896	0
	$\mu+2\sigma$	39.9	0.6777	0.8640	0.8019	1	1
$Min_{(U_{max},SumL)}$	μ	12.7	0.8946	**0.8683**	**0.9158**	0.9846	**0.6640**
	σ	4.5	0.1153	0.1514	0.1275	0.0650	0.1856
	$\mu-2\sigma$	3.7	0.6640	0.5655	0.6609	0.8545	0.2929
	$\mu+2\sigma$	21.7	1	1	1	1	1
Min_{SumL}	μ	1.5	**0.8961**	0.2075	0.1650	**0.9884**	0.3883
	σ	0.7	0.2724	0.1631	0.0941	0.0558	0.0288
	$\mu-2\sigma$	0.1	0.3513	0	0	0.8767	0.3308
	$\mu+2\sigma$	2.9	1	0.5337	0.3533	1	0.4459



(a) Hypervolume HP (b) Hypervolume-Star HP*

Fig. 3: $(U_{max}, SumL)$ measurement on expanded tree

Fig. 3 depicts the assessment metrics for $(U_{max}, SumL)$ on an expanded tree topology. None of the five metrics of $Min_{U_{max}}$ is the best because most of its solutions in PF_{ref} have a large value for $SumL$, and its PF_{obt} is mainly contained in the dominated region ($Purity = 0.2884$). Thus, the surface of the dominated region covered by $Min_{U_{max}}$ measured with HP* (in Fig. 3b) is much smaller than this surface measured with HP (in Fig. 3a).

In Fig. 3, we can state that Min_{SumL} has the best value for the metrics Purity and I, because its PF_{obt} has only one solution, which is contained in PF_{ref}. The PF_{obt} of $Min_{(U_{max},SumL)}$ is completely included in the center of PF_{ref} and it dominates many solutions of $Min_{U_{max}}$. Its $Purity$, 0.8946, is very large compared to this value for $Min_{U_{max}}$. However, $Min_{U_{max}}$ and Min_{SumL} are clearly better than $Min_{(U_{max},SumL)}$ from the single objective performance.

The $Min_{(U_{max},SumL)}$ LS provides a good Pareto front by ensuring a small value of $SumL$ with a significant decrease of U_{max} compared to the solution obtained by 802.1s (see Table 1). Furthermore, the PF_{ref} obtained by these three LSs offers a wide range of solutions for solving this TE problem.

Bi-objective problem 2: Minimization of U_{max} and #Links

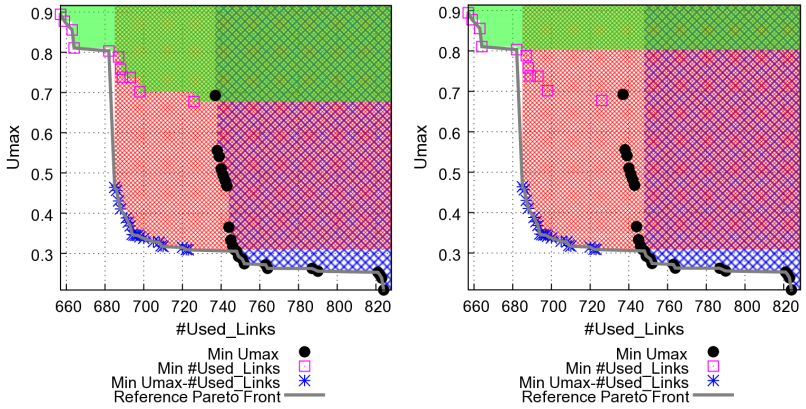
In Table 4, $Min_{(U_{max},\#Links)}$ has the best average value for all the metrics. The HP and *purity* metrics are much larger than those of $Min_{U_{max}}$ and $Min_{\#Links}$. Except for HP*, it also has smaller standard deviation than the two other algorithms, underlying a better stability of $Min_{(U_{max},\#Links)}$. As expected, $Min_{(U_{max},\#Links)}$ is thus the best method.

Fig. 4 depicts a measurement for $(U_{max}, \#Links)$ on a cloud data center topology with geographic VLAN generation. In this test, all the non-dominated solutions given by $Min_{(U_{max},\#Links)}$ can be found at the center of PF_{ref}.

As shown in Fig. 4b, the hypervolume covered by $Min_{(U_{max},\#Links)}$ is the largest. We can state that by keeping a small value of U_{max} and $\#Links$ for all its solutions, $Min_{(U_{max},\#Links)}$ provides good choices for network operators. In Fig. 4, $Min_{(U_{max},\#Links)}$ can reduce the value of U_{max} from 0.7 to 0.3 using only 720 links, while $Min_{U_{max}}$ needs 750 links to approximate U_{max}. In addition, we can easily choose a solution in PF_{obt} of $Min_{(U_{max},\#Links)}$ with a reasonable U_{max} (less than 0.5) which uses a small number of links in the network.

Table 4: Performance Comparison U_{max}, $\#Links$

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	I^*
$Min_{U_{max}}$	μ	18.3	0.6611	0.2667	0.3515	0.8729	0.5812
	σ	9.4	0.2209	0.1240	0.1081	0.1777	0.0661
	$\mu - 2\sigma$	0	0.2193	0.0186	0.1353	0.5175	0.4491
	$\mu + 2\sigma$	37.1	1	0.5147	0.5677	1	0.7134
$Min_{(U_{max},\#Links)}$	μ	11.8	**0.9964**	**0.6972**	**0.9255**	**0.9992**	**0.7602**
	σ	4.6	0.0172	0.0939	0.1190	0.0036	0.1043
	$\mu - 2\sigma$	2.6	0.9620	0.5095	0.6875	0.9920	0.5515
	$\mu + 2\sigma$	21	1	0.8849	1	1	0.9689
$Min_{\#Links}$	μ	7.0	0.3412	0.3090	0.1356	0.8745	0.5279
	σ	3.3	0.3499	0.1594	0.0989	0.1891	0.0519
	$\mu - 2\sigma$	0.4	0	0	0	0.4963	0.4241
	$\mu + 2\sigma$	13.6	1	0.6278	0.3335	1	0.6317



(a) Hypervolume HP (b) Hypervolume-Star HP*

Fig. 4: $(U_{max}, \#Links)$ measurement on cloud data center

In this test, compared to the solution obtained with 802.1s MSTP standard, $Min_{\#Links}$ can reduce about 80 links while $Min_{U_{max}}$ markedly reduces the value of U_{max}: from 0.7 to about 0.2. Especially, the PF_{ref} is composed of the important contributions of all the PF_{obt} obtained with these three LSs.

7 Conclusion

In this paper, we develop a novel design of multi-objective algorithms, based on local search, for solving the TE problem in data center networks. Our multi-objective algorithms show good performance for large data centers, by integrating the heuristics of each single objective into the search process. There are few existing assessment methods for evaluating the performance of multi-objective TE techniques in data center networks. The state of the art approaches often evaluate their outcomes based on each single objective observation. In this paper, we introduce a strong assessment method using three metrics: maximal link utilization, sum load, and number of used links, for evaluating the multi-objective overall performance. The non-dominated set of solutions (Pareto front) obtained by our multi-objective algorithms offer the network operators a wide range of good solutions for solving the TE problem in data center networks.

Acknowledgements Ho Trong Viet was supported by the FRIA (Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture, Belgium).

References

1. Cisco, "White Paper: Understanding Multiple Spanning Tree Protocol (802.1s)", *Document ID: 24248*, April 2007, Accessed October 2013. http://www.cisco.com/image/gif/paws/24248/147.pdf
2. IEEE Standard 802.1S, "Virtual Bridged Local Area Networks - Amendment 3: Multiple Spanning Trees", 2002
3. IEEE Standard 802.1Q, "Virtual LANs", 2006.
4. Sawaragi. Y, Nakayama. H. and Tanino. T, "Theory of Multiobjective Optimization", in *Academic Press Inc*, 1985.
5. P. V. Hentenryck, and L. Michel, "Constraint-based Local Search", *MIT Press*, 2005.
6. Kandula, Srikanth and Katabi, Dina and Davie, Bruce and Charny, Anna, "Walking the tightrope: responsive yet stable traffic engineering", in *SIGCOMM Comput. Commun. Rev.*, 2005.
7. L. Cheng and C.-L. Wang, "Network performance isolation for latency-sensitive cloud applications", in *Future Gen- eration Computer Systems,* 2012.
8. McKeown, Nick and Anderson, Tom and Balakrishnan, Hari and Parulkar, Guru and Peterson, Larry and Rexford, Jennifer and Shenker, Scott and Turner, Jonathan, "OpenFlow: enabling innovation in campus networks", *SIGCOMM Comput. Commun. Rev.*, 2008.
9. T. V. Ho, Y. Deville, O. Bonaventure and P. Francois, "Traffic Engineering for Multiple Spanning Tree Protocol in Large Data Centers", in *Proceedings of the* 23^{th} *ITC Conference*, San Francisco, USA, 2011.
10. A. Meddeb, "Multiple Spanning Tree Generation and Mapping Algorithms for Carrier Class Ethernets", in *Proceedings of the IEEE GLOBECOM Conference*, 2006.

11. Xiaoming He, Mingying Zhu, and Qingxin Chu, "Traffic Engineering for Metro Ethernet Based on Multiple Spanning Trees", in *Proceedings of the International Conference ICNICONSMCL*, 2006.
12. Y. Lim, H. Yu, S. Das, S. S. Lee, M. Gerla, "QoS-aware Multiple Spanning Tree Mechanism over a Bridged LAN Environment", in *Proceedings of the IEEE GLOBECOM Conference*, 2003
13. D. Santos, et al., "Traffic Engineering of Multiple Spanning Tree Routing Networks: the Load Balancing case", in *Proceedings of the 5th Euro-NGI conference on Next Generation Internet networks*, 2009.
14. A. F. de Sousa, G. Soares, "Improving Load Balance of Ethernet Carrier Networks using IEEE 802.1S MSTP with Multiple Regions", in *IFIP TC6 Networking Conf*, 2006.
15. D. Santos, A. F. de Sousa, et al., "Optimization of link load balancing in multiple spanning tree routing networks", in *Telecommunication Systems Journal*, 2011.
16. W. Chen, D. Jin, L. Zeng, "Design of Multiple Spanning Trees for Traffic Engineering in Metro Ethernet", in *Proceedings of the International Conference on Communication Technology ICCT*, 2006.
17. G. Mirjalily, F. A. Sigari, R. Saadat, "Best Multiple Spanning Tree in Metro Ethernet Networks", in *Proceedings of the Second International Conference on Computer and Electrical Engineering*, 2009.
18. M. Padmaraj , et al., "Metro Ethernet traffic engineering based on optimal multiple spanning trees", in *Wireless and Optical Communications Networks*, 2005.
19. T. V. Ho, O. Bonaventure, Y. Deville, Q. D. Pham, and P. Francois, "Using Local Search for Traffic Engineering in Switched Ethernet Networks", in *Proceedings of the* 22^{th} *ITC Conference*, Amsterdam, The Netherlands, 2010.
20. A. L. Custdio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, "Direct multisearch for multiobjective optimization", preprint 10-18, Dept. of Mathematics, Univ. Coimbra, 2010
21. E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms A comparative case study", in *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*. Springer Verlag, Berlin, Germany, 1998, pp. 292 - 301
22. Eckart Zitzler, Lothar Thiele, et al., "Performance Assessment of Multiobjective Optimizers: An Analysis and Review", Swiss Federal Institute of Technology, Computer Engineering and Networks Laboratory, 2002
23. R. Perlman, D. Eastlake, D. Dutt, S. Gai, and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification," RFC 6325, July 2011.
24. IEEE Standard 802.1D, "Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Common specifications-Part 3: Media Access Control (MAC) Bridges" ,1998.
25. IEEE Standard 802.1W, "Rapid spanning tree configuration", 2001.
26. J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power Awareness in Network Design and Routing", The 27th Conference on Computer Communications IEEE INFOCOM 2008.
27. T. Benson, Aditya Akella, and David A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proceedings of the Internet Measurement Conference (IMC)*, Melbourne, Australia, Nov 2010.
28. Hu T. C., "Optimum communication spanning trees", in *SIAM Journal on Computing:188-195*, 1974.
29. Cisco Systems, "Cisco data center infrastructure 2.5 design guide", http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf
30. Albert Greenberg, et al., "VL2: A Scalable and Flexible Data Center Network, " in *Proceedings of SIGCOMM* 2009.
31. M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *SIGCOMM*, 2008.
32. R. N. Mysore, and al., "PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," in *SIGCOMM*, 2009.
33. T. V. Ho et al, "Multi-objective Traffic Engineering for Data Center Networks", Data Sets are available for downloading at: http://becool.info.ucl.ac.be/resources/multi-objective-traffic-engineering-data-center-networks

Biographies

Ho Trong-Viet received a B.E. degree in Computer Science from Hanoi University of Science and Technology in 2006, and a M.S. degree in Systems and Networks from the Universite Claude Bernard de Lyon in 2008. He is currently a PhD student in the ICTEAM Institute at the Universite Catholique de Louvain. His research interests are focussed on traffic engineering problems in data center networks using spanning trees.

Yves Deville graduated from the University of Namur in 1983 where he obtained his PhD degree in 1987. He also spent one year at Syracuse University (USA), where he got an MS degree in computer science in 1986. He received the IBM Belgium Computer Science Award for his PhD thesis. He is a full professor at the University of Louvain, where he leads the Constraint Programming Lab (http://becool.info.ucl.ac.be). He spent a sabbatical at Glasgow University (2003) and at Brown University (2010). He has published more than one hundred papers. He has been elected to the executive committee of the Association for Constraint Programming (2011–2014), and serves on the editorial board of the Journal of Constraint Programming.

Olivier Bonaventure graduated from the University of Liege in 1992. He obtained his PhD in 1999 and spent one year at Alcatel in Antwerp. He is now a full professor at the Universite Catholique de Louvain, where he leads the IP Networking Lab (http://inl.info.ucl.ac.be) and is Vice-President of the ICTEAM Institute. He has published more than eighty papers, contributes to IETF, was granted several patents, and served on the editorial board of IEEE/ACM Transactions on Networking. He currently serves as Education Director within ACM SIGCOMM and is a member of the CoNEXT steering committee. He received several awards, including the INFOCOM 2007 best paper award.

APPENDIX

Table 5: Performance Comparison U_{max}, $SumL$ - Cloud Geographic

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	23.2	0.0690	0.7459	0.1313	0.8733	0.5612
	σ	3.04	0.0955	0.0265	0.0569	0.1734	0.1432
	$\mu - 2\sigma$	17.12	0	0.6929	0.0175	0.5265	0.2748
	$\mu + 2\sigma$	29.28	0.2560	0.7988	0.2451	1	0.8476
$Min_{(U_{max},SumL)}$	μ	12	0.9417	0.9722	0.9773	0.9995	0.8704
	σ	2.32	0.0452	0.0263	0.0313	0.0011	0.0487
	$\mu - 2\sigma$	7.35	0.8512	0.9196	0.9148	0.9974	0.7731
	$\mu + 2\sigma$	16.65	1	1	1	1	0.9677
Min_{SumL}	μ	1	1	0.0852	0.0965	1	0.5109
	σ	0	0	0.0195	0.0114	0	0
	$\mu - 2\sigma$	1	1	0.0462	0.0737	1	0.5109
	$\mu + 2\sigma$	1	1	0.1243	0.1194	1	0.5109

Table 6: Performance Comparison U_{max}, $\#Links$ - Cloud Geographic

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	24.6	0.6016	0.4101	0.5117	0.9036	0.4910
	σ	2.5	0.1413	0.0579	0.1058	0.2039	0.0487
	$\mu - 2\sigma$	19.6	0.3191	0.29443	0.3002	0.4957	0,3935
	$\mu + 2\sigma$	29.6	0.8842	0.5258	0.7232	1	0,5884
$Min_{(U_{max},\#Links)}$	μ	16	1	0.6458	0.9179	1	0.7212
	σ	2.03	0	0.0353	0.0409	0	0.0424
	$\mu - 2\sigma$	11.94	1	0.5752	0.8362	1	0.6364
	$\mu + 2\sigma$	20.06	1	0.7165	0.9997	1	0,8060
$Min_{\#Links}$	μ	6.4	0.5313	0.2006	0.1614	0.9426	0.4408
	σ	3.93	0.2546	0.0825	0.0696	0.1240	0.0601
	$\mu - 2\sigma$	0.54	0.0220	0.0357	0.0223	0.6946	0,32068
	$\mu + 2\sigma$	12.26	1	0.3655	0.3005	1	0.5610

Table 7: Performance Comparison U_{max}, $SumL$ - Cloud Random

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	19.1	0.0628	0.6376	0.1732	0.8721	0.4749
	σ	2.11	0.1106	0.0474	0.1183	0.0154	0.0188
	$\mu - 2\sigma$	14.87	0	0.5428	0	0.8413	0.4373
	$\mu + 2\sigma$	23.33	0.2840	0.7324	0.4098	0.9029	0.5125
$Min_{(U_{max},SumL)}$	μ	9	0.8778	0.9741	0.9475	0.9999	0.8603
	σ	1.79	0.0666	0.0155	0.1568	0	0.0089
	$\mu - 2\sigma$	5.41	0.7446	0.9430	0.6339	0.9999	0.8424
	$\mu + 2\sigma$	12.59	1	1	1	0.9999	0.8781
Min_{SumL}	μ	1	1	0.1250	0.1608	1	0.4684
	σ	0	0	0.0457	0.0437	0	0
	$\mu - 2\sigma$	1	1	0.0336	0.0735	1	0.4684
	$\mu + 2\sigma$	1	1	0.2164	0.2482	1	0.4684

Table 8: Performance Comparison U_{max}, $\#Links$ - Cloud Random

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	18.6	0.8065	0.3467	0.4602	0.9181	0.4357
	σ	2.2	0.1321	0.0312	0.0334	0.0447	0.0311
	$\mu - 2\sigma$	14.2	0.5422	0.2842	0.3933	0.8287	0.3735
	$\mu + 2\sigma$	23	1	0.4092	0.5271	1	0.4979
$Min_{(U_{max},\#Links)}$	μ	17.3	0.9942	0.6744	0.9276	0.9992	0.6461
	σ	2.86	0.0188	0.0308	0.0314	0.0025	0.0550
	$\mu - 2\sigma$	11.57	0.9567	0.6128	0.8649	0.9941	0.5362
	$\mu + 2\sigma$	23.03	1	0.7360	0.9904	1	0.7560
$Min_{\#Links}$	μ	13.3	0.4361	0.4762	0.2547	0.8994	0.4142
	σ	3.13	0.1385	0.0562	0.0877	0.0141	0.0563
	$\mu - 2\sigma$	7.04	0.15903	0.3639	0.07931	0.8713	0.3015
	$\mu + 2\sigma$	19.56	0.7131	0.5886	0.4301	0.9276	0.5268

Table 9: Performance Comparison U_{max}, $SumL$ - Private Geographic

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	15.7	0.0701	0.6720	0.1048	0.8720	0.4289
	σ	2.69	0.0426	0.0269	0.0467	0.0265	0.0147
	$\mu - 2\sigma$	10.32	0	0.6181	0.0113	0.8189	0.3995
	$\mu + 2\sigma$	21.08	0.1553	0.7258	0.1982	0.9250	0.4583
$Min_{(U_{max},SumL)}$	μ	16.6	0.9217	0.9858	0.9236	0.9811	0.7713
	σ	2.17	0.0517	0.0027	0.0767	0.0559	0.0799
	$\mu - 2\sigma$	12.26	0.8183	0.9804	0.7701	0.8692	0.6115
	$\mu + 2\sigma$	20.94	1	0.9913	1	1	0.9312
Min_{SumL}	μ	1.9	1	0.3826	0.3841	1	0.4544
	σ	0.3	0	0.0318	0.0664	0	0.0433
	$\mu - 2\sigma$	1.3	1	0.3189	0.2512	1	0.3678
	$\mu + 2\sigma$	2.5	1	0.4463	0.5169	1	0.5411

Table 10: Performance Comparison U_{max}, $\#Links$ - Private Geographic

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	13.3	0.7368	0.3516	0.4340	0.9008	0.4376
	σ	1.85	0.1169	0.0388	0.0465	0.0075	0.0907
	$\mu - 2\sigma$	9.6	0.5031	0.2741	0.3409	0.8857	0.2561
	$\mu + 2\sigma$	17	0.9705	0.4292	0.5269	0.9159	0.6190
$Min_{(U_{max},\#Links)}$	μ	16.3	0.9877	0.7514	0.9253	0.9981	0.6685
	σ	2.44	0.0299	0.0498	0.0891	0.0044	0.0885
	$\mu - 2\sigma$	11.42	0.9278	0.6518	0.7472	0.9894	0.4914
	$\mu + 2\sigma$	21.18	1	0.8511	1	1	0.8456
$Min_{\#Links}$	μ	10.9	0.1743	0.4662	0.1507	0.8001	0.3642
	σ	2.24	0.1713	0.0839	0.1303	0.2972	0.0264
	$\mu - 2\sigma$	6.42	0	0.2985	0	0.2057	0.3113
	$\mu + 2\sigma$	15.38	0.5169	0.6339	0.4114	1	0.4170

Table 11: Performance Comparison U_{max}, $SumL$ - Private Random

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	18.3	0.0601	0.5252	0.1416	0.8846	0.4481
	σ	3.2	0.0405	0.0457	0.0613	0.0272	0.0668
	$\mu - 2\sigma$	11.9	0	0.4338	0.0189	0.8302	0.3146
	$\mu + 2\sigma$	24.7	0.1410	0.6167	0.2642	0.9390	0.5817
$Min_{(U_{max},SumL)}$	μ	21.1	0.9384	0.9778	0.9694	0.9990	0.9521
	σ	3.1	0.0296	0.0082	0.0614	0.0004	0.1426
	$\mu - 2\sigma$	14.9	0.8791	0.9614	0.8466	0.9983	0.6669
	$\mu + 2\sigma$	27.3	0.9977	0.9942	1	0.9998	1
Min_{SumL}	μ	1	1	0.0519	0.0782	1	0.4067
	σ	0	0	0.0089	0.0095	0	0
	$\mu - 2\sigma$	1	1	0.0341	0.0592	1	0.4067
	$\mu + 2\sigma$	1	1	0.0697	0.0972	1	0.4067

Table 12: Performance Comparison U_{max}, $\#Links$ - Private Random

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	16.3	0.7669	0.3356	0.4071	0.9048	0.4835
	σ	2.3	0.0683	0.0233	0.0287	0.0066	0.0332
	$\mu - 2\sigma$	11.7	0.6304	0.2889	0.3497	0.8916	0.4170
	$\mu + 2\sigma$	20.9	0.9034	0.3823	0.4645	0.9180	0.5499
$Min_{(U_{max},\#Links)}$	μ	14.2	0.9930	0.7297	0.9407	0.9974	0.7609
	σ	2.2	0.025	0.0307	0.0327	0.0078	0.0667
	$\mu - 2\sigma$	9.8	0.9430	0.6683	0.8752	0.9819	0.6275
	$\mu + 2\sigma$	18.6	1	0.7911	1	1	0.8942
$Min_{\#Links}$	μ	9.8	0.2041	0.4296	0.1203	0.8962	0.4159
	σ	1.4	0.1432	0.0522	0.0835	0.0090	0.0252
	$\mu - 2\sigma$	7	0	0.3251	0	0.8783	0.3655
	$\mu + 2\sigma$	12.6	0.4904	0.5340	0.2872	0.9142	0.4662

Table 13: Performance Comparison U_{max}, $SumL$ - Fat Tree

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	41.7	0.6595	0.8034	0.8424	0.9185	0.5236
	σ	2.9	0.1070	0.0322	0.0916	0.0726	0.1314
	$\mu - 2\sigma$	35.9	0.4455	0.7391	0.659	0.7733	0.2607
	$\mu + 2\sigma$	47.5	0.8735	0.8678	1	1	0.7864
$Min_{(U_{max},SumL)}$	μ	15.2	0.8618	0.8011	0.9253	0.9711	0.3834
	σ	2.2	0.1041	0.0408	0.0579	0.0571	0.1059
	$\mu - 2\sigma$	10.8	0.6536	0.7195	0.8096	0.8570	0.1715
	$\mu + 2\sigma$	19.6	1	0.8828	1	1	0.5953
Min_{SumL}	μ	2.6	0.7308	0.3416	0.2261	0.9604	0.2354
	σ	0.9	0.3177	0.0404	0.1061	0.0786	0.0042
	$\mu - 2\sigma$	0.8	0.0954	0.2609	0.0139	0.8033	0.2269
	$\mu + 2\sigma$	4.4	1	0.4223	0.4383	1	0.2438

Table 14: Performance Comparison U_{max}, $\#Links$ - Fat Tree

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	43.7	0.4966	0.1335	0.2133	0.6338	0.7383
	σ	4.2	0.091	0.0166	0.0151	0.0579	0.0231
	$\mu - 2\sigma$	35.3	0.3143	0.1004	0.1830	0.5180	0.6922
	$\mu + 2\sigma$	52.1	0.6789	0.1666	0.2436	0.7496	0.7844
$Min_{(U_{max},\#Links)}$	μ	5.3	1	0.5568	0.9148	1	0.8538
	σ	1.3	0	0.0196	0.0306	0	0.0681
	$\mu - 2\sigma$	2.7	1	0.5177	0.8536	1	0.7176
	$\mu + 2\sigma$	7.9	1	0.5960	0.9759	1	0.9900
$Min_{\#Links}$	μ	4.2	0.9286	0.2790	0.1833	0.9730	0.6955
	σ	1.96	0.128571429	0.0685	0.0733	0.0832	0.0470
	$\mu - 2\sigma$	0.28	0.6714	0.1420	0.0367	0.8066	0.6014
	$\mu + 2\sigma$	8.12	1	0.4159	0.3300	1	0.7895

Table 15: Performance Comparison U_{max}, $SumL$ - PortLand

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	29.4	0.7211	0.7055	0.9064	0.9674	0.5559
	σ	3.7	0.1263	0.0437	0.0666	0.0207	0.1715
	$\mu - 2\sigma$	22	0.4684	0.6181	0.7732	0.9260	0.2128
	$\mu + 2\sigma$	36.8	0.9738	0.7928	1	1	0.8990
$Min_{(U_{max},SumL)}$	μ	7.6	0.7895	0.5674	0.7476	0.9918	0.2644
	σ	1.9	0.0852	0.0750	0.0826	0.0078	0.0604
	$\mu - 2\sigma$	3.8	0.6190	0.4173	0.5824	0.9763	0.1437
	$\mu + 2\sigma$	11.4	0.9600	0.7174	0.9127	1	0.3851
Min_{SumL}	μ	1	0.7000	0.0856	0.0347	0.9985	0.2102
	σ	0	0.4583	0.0171	0.0170	0.0023	0
	$\mu - 2\sigma$	1	0	0.0514	0.0008	0.9938	0.2102
	$\mu + 2\sigma$	1	1	0.1198	0.0687	1	0.2102

Table 16: Performance Comparison U_{max}, $\#Links$ - PortLand

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	6.9	0.3043	0.2260	0.1909	0.9226	0.6181
	σ	0.7	0.0868	0.0340	0.0321	0.0743	0.0374
	$\mu - 2\sigma$	5.5	0.1308	0.1579	0.1268	0.7741	0.5434
	$\mu + 2\sigma$	8.3	0.4779	0.2940	0.2551	1	0.6929
$Min_{(U_{max},\#Links)}$	μ	7.3	1	0.8452	0.9707	1	0.8251
	σ	1	0	0.0527	0.0799	0	0.0916
	$\mu - 2\sigma$	5.3	1	0.7399	0.8108	1	0.6420
	$\mu + 2\sigma$	9.3	1	0.9506	1	1	1
$Min_{\#Links}$	μ	1.2	0	0.1136	0	0.8589	0.5907
	σ	0.4	0	0.0801	0	0.1306	0
	$\mu - 2\sigma$	0.4	0	0	0	0.5977	0.5907
	$\mu + 2\sigma$	2	0	0.2738	0	1	0.5907

Table 17: Performance Comparison U_{max}, $SumL$ - Expanded Tree 200

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	22.6	0.3761	0.6619	0.4800	0.7933	0.4978
	σ	3	0.1320	0.0703	0.1444	0.1695	0.0737
	$\mu - 2\sigma$	16.6	0.1120	0.5212	0.1912	0.4542	0.3504
	$\mu + 2\sigma$	28.6	0.6402	0.8025	0.7688	1	0.6452
$Min_{(U_{max},SumL)}$	μ	7.3	0.9315	0.7997	0.9200	0.9494	0.5461
	σ	2.2	0.1139	0.0676	0.1092	0.1442	0.0603
	$\mu - 2\sigma$	2.9	0.7040	0.6645	0.7017	0.6610	0.4255
	$\mu + 2\sigma$	11.7	1	0.9349	1	1	0.6667
Min_{SumL}	μ	1.9	0.8421	0.3808	0.1747	0.9600	0.4324
	σ	0.5	0.3202	0.2029	0.0758	0.1150	0.0210
	$\mu - 2\sigma$	0.9	0.2018	0	0.0232	0.7300	0.3905
	$\mu + 2\sigma$	2.9	1	0.7867	0.3263	1	0.4744

Table 18: Performance Comparison U_{max}, $\#Links$ - Expanded Tree 200

PF \ Metric	**Val.**	**#Sol**	**Purity**	**HP**	**HP***	**I**	Γ^*
$Min_{U_{max}}$	μ	4.7	0.9149	0.0631	0.2435	0.9263	0.8645
	σ	1	0.1018	0.0100	0.0186	0.0980	0.0435
	$\mu - 2\sigma$	2.7	0.7113	0.0431	0.2063	0.7302	0.7775
	$\mu + 2\sigma$	6.7	1.1184	0.0831	0.2807	1	0.9515
$Min_{(U_{max},\#Links)}$	μ	6.1	1	0.6770	0.8814	1	0.8457
	σ	0.97	0	0.0609	0.0805	0	0.0331
	$\mu - 2\sigma$	4.16	1	0.5551	0.7204	1	0.7794
	$\mu + 2\sigma$	8.04	1	0.7989	1	1	0.9120
$Min_{\#Links}$	μ	3.5	0.1143	0.1976	0.0789	0.7512	0.7742
	σ	1.7	0.175	0.1288	0.0793	0.2605	0.0038
	$\mu - 2\sigma$	0.1	0	0	0	0.2303	0.7666
	$\mu + 2\sigma$	6.9	0.4643	0.4552	0.2375	1	0.7819