It's a Match! Near-Optimal and Incremental Middlebox Deployment *

Tamás Lukovszki
Eötvös Loránd University
lukovszki@inf.elte.hu

Matthias Rost
Technische Universität Berlin
mrost@inet.tu-berlin.de

Stefan Schmid
Aalborg University
schmiste@cs.aau.dk

ABSTRACT

The virtualization and softwarization of modern computer networks offers new opportunities for the simplified management and flexible placement of middleboxes as e.g. firewalls and proxies. This paper initiates the study of algorithmically exploiting the flexibilities present in virtualized and software-defined networks. Particularly, we are interested in the initial as well as the incremental deployment of middleboxes. We present a deterministic $O(\log(\min\{n, \kappa\}))$ approximation algorithm for n-node computer networks, where κ is the middlebox capacity. The algorithm is based on optimizing over a submodular function which can be computed efficiently using a fast augmenting path approach. The derived approximation bound is optimal: the underlying problem is computationally hard to approximate within sublogarithmic factors, unless $P = NP$ holds. We additionally present an exact algorithm based on integer programming, and complement our formal analysis with simulations. In particular, we consider the number of used middleboxes and highlight the benefits of the approximation algorithm in incremental deployments. Our approach also finds interesting applications, e.g., in the context of incremental deployment of software-defined networks.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; G.1.6 [**Optimization**]: Linear Programming

Keywords

Network Virtualization, NFV, Software-Defined Networking, Capacitated Set Cover, Facility Location, NP-hardness

1. INTRODUCTION

Middleboxes are ubiquitous in modern computer networks, which provide a wide spectrum of in-network functions to ensure security, performance, and policy compliance. In fact, the number of middleboxes in enterprise networks can be of the same order of magnitude as the number of routers [8].

While in-network functions were traditionally implemented in specialized hardware appliances and middleboxes, computer networks in general and middleboxes in particular become more and more software-defined and virtualized [1]: network functions can be implemented in software and deployed fast and flexibly on the virtualized network nodes, e.g., running in a virtual machine on a commodity x86 server. Virtualization is also attractive for its potential cost savings [17].

Modern computer networks also offer new flexibilities in terms of how traffic can be *routed* through middleboxes and virtualized data plane appliances (often called Virtual Network Functions, short *VNFs*) [16]. In particular, the advent of *Software-Defined Network (SDN)* technology allows network operators to steer traffic through middleboxes (or chains of middleboxes) using arbitrary routes, i.e., along routes which are not necessarily shortest paths, or not even loop-free [2, 13, 15, 18, 23]. In fact, Openflow, the standard SDN protocol today, not only introduces a more flexible routing, but itself allows to implement basic middlebox functionality, on the switches [6]: an Openflow switch can match, and perform actions upon, not only layer-2, but also layer-3 and layer-4 header fields.

However, not much is known today about how to exploit these flexibilities algorithmically. A particularly interesting problem regards the question of where to deploy a minimal number of middleboxes such that basic routing and capacity constraints are fulfilled. Intuitively, the smaller the number of deployed network functions, the longer the routes via these functions, and a good tradeoff between deployment costs and additional latency must be found. Moreover, ideally, middleboxes should be *incrementally deployable*: when additional middleboxes are deployed, existing placements do not have to be changed. This is desirable especially in deployment scenarios with budget constraints.

Our Contributions. This paper initiates the study of the problem of placing a minimum number of middleboxes or network functions, such that distance constraints between communicating node pairs as well as capacity constraints on the network nodes are satisfied.

Our main contribution is a deterministic and greedy $O(\log(\min\{\kappa, n\}))$-approximation algorithm for the middlebox placement problem in n-node networks where capacities are bounded by κ. Our algorithm supports both deployments *from scratch* as well as *incremental deployments*: It does not require any changes to the locations of existing middleboxes or the preemption of previously served communication pairs when additional middleboxes are deployed.

The approximation bound is optimal in the sense that, as we show, neither the capacitated nor the uncapacitated

*This research was supported by the EU projects UNIFY under grant n° 619609. Part of this research was done when the first author visited the Heinz Nixdorf Institute, Germany, being supported by DFG SFB 901.

problem admits a sublogarithmic polynomial-time approximation, unless $P = NP$ holds.

This paper also presents a (non-polynomial) optimal algorithm based on a 0-1 integer linear program which, together with our hardness result, implies that the problem is NP-complete: 0-1 programming is one of Karp's 21 NP-complete problems.

We complement our formal analysis with a simulation study, where we investigate the tradeoff between routing flexibilities (in terms of path stretch) and number of required middleboxes, and also highlight the benefit of incremental deployments.

We believe that our model and approach has ramifications beyond middlebox deployment. For instance, our algorithm can also be used to solve the problem of incremental SDN deployment [5, 14], which so far has only been studied heuristically.

Novelty and Related Work. Interestingly, the middlebox deployment problem has not been studied before in the algorithms literature. As we show, the problem can be seen as a novel covering problem [4, 7, 22] where: (1) we are interested in the distance *between communicating pairs*, *via* the covering nodes, and not *to* the covering nodes; (2) we aim to support incremental deployments: middlebox locations selected earlier in time as well as the supported communication pairs should not have to be changed when deploying additional middleboxes; (3) we consider a *capacitated* setting where the number of items which can be assigned to a node is bounded by κ.

Nevertheless, we can adapt algorithmic concepts introduced in prior work. Our lower bounds build upon hardness results on uncapacitated covering problems [12], and our upper bound builds upon Wolsey's study of vertex and set covering problems with hard capacities [22]. An elegant alternative proof to Wolsey's dual fitting approach, based on combinatorial arguments, is due to Chuzhoy and Naor [3]. The authors also show that using LP-relaxation approaches is generally difficult, as the integrality gap of a natural linear program for the weighted and capacitated vertex and set covering problems is unbounded.

2. MODEL

We model the computer network as a graph connecting a set V of $n = |V|$ nodes. The input to our problem is a set of communicating node pairs P: the route of each node pair $(s,t) \in P$ needs to traverse an instance of a middlebox or respective network function (e.g., a firewall). Node pairs do not have to be disjoint: nodes can participate in many communications simultaneously.

For the sake of generality, we assume that middleboxes can only be installed on a subset of nodes $U \subseteq V$. We will refer to the set of middleboxes locations (and equivalently, the set of middleboxes instances) by M, and we are interested in deploying a minimal number of middleboxes at legal locations $M \subseteq U$ such that:

1. Each pair $p = (s,t) \in P$ is assigned to an instance $m \in M$, denoted by $m = \mu(p)$.
2. For each pair $p = (s,t) \in P$, there is a route from s via $m = \mu(p)$ to t of length at most $\rho \cdot d(s,t)l$, i.e., $d(s,m) + d(m,t) \leq \rho \cdot d(s,t)$, where $d(u,v)$ denotes the length of the shortest path between nodes $u, v \in V$ in the network G, and where ρ is called the *stretch*. Our approach supports many alternative constraints, e.g., on the maximal route length.
3. Capacities are respected: at most κ node pairs can be served by any middlebox instance.

Our objective is to minimize the number of required middlebox instances, subject to the above constraints.

Use Cases. *Middlebox Placement.* Our model is mainly motivated by the middlebox placement flexibilities introduced in network function virtualized and software-defined networks. For example, network policies can often be defined in terms of adjacency matrices or big switch abstractions, specifying which traffic is allowed between an ingress port s and an outgress network port t. In order to enforce such a policy, traffic from s to t needs to traverse a middlebox instance inspecting and classifying the flows. The location of every middlebox can be optimized, but is subject to the constraint that the route from s to t via the middlebox should not be much longer than the shortest path from s to t (in terms of bounded *stretch* or maximal length).

Incremental Deployment of Software-Defined Networks. Our model also finds interesting applications in the context of hybrid Software-Defined Networks (SDNs) [5, 14, 21]. Especially the *partial deployment* of SDNs is of practical importance, due to cost constraints. It has recently been shown that by routing traffic via sparsely deployed Openflow switches (serving as "waypoints" where SDN functionality is applied), it is possible to run SDN applications also in hybrid networks, and hence to reap the benefits of SDN. In particular, our approach can be used to solve the incremental SDN deployment of Panopticon [5], which so far has only been studied heuristically.

3. APPROXIMATION ALGORITHM

We present a deterministic and polynomial-time $O(\log(\min\{n, \kappa\}))$-approximation algorithm for the incremental middlebox deployment problem. Our algorithm is based on an efficient computation of a certain submodular set function: it defines the maximum number of pairs which can be covered by a given set of middleboxes. In a nutshell, the submodular function is computed efficiently using an augmenting path method on a certain bipartite graph, which also determines the corresponding assignment of communication pairs to the middleboxes (a maximum matching). The augmenting path algorithm is based on a simple, linear-time breadth-first graph traversal.

The augmenting path method is attractive and may be of independent interest: similarly to the flow-based approaches in the literature [3], it does not require changes to previously deployed middleboxes, but removes the disadvantage of [3] that the set of served communication pairs changes over time: an attractive property for *incremental deployments*. Our solution with augmenting paths results (theoretically and practically) in significantly faster computations of the submodular function value as well as of the corresponding assignment compared to the flow based approach.

Preliminaries. For any set of middlebox instances $M \subseteq U$, we define $\phi(M)$ to be the maximum number of communication pairs that can be assigned to M, given the capacity constraints at the nodes and the route length constraints. We show that ϕ is non-decreasing and submodular, and $\phi(M)$ – and the corresponding assignment of pairs to middlebox instances in M – can be computed in polynomial

time. This allows us to use Wolsey's Theorem [22] to prove an approximation factor of $1 + O(\log \phi_{\max})$, where $\phi_{\max} = \max_{m \in U} \phi(\{m\})$. Since in our case, $\phi_{\max} = \min\{\kappa, |P|\}$ and $|P| \leq n^2$, this implies that Algorithm 1 computes an $O(\log(\min\{\kappa, n\}))$-approximation for the minimum number of middlebox instances that can cover all pairs P (i.e., all pairs can be assigned to the deployed middleboxes).

In order to compute function $\phi(M)$, for any $M \subseteq U$, we construct a bipartite graph $B(M) = (M \cup P, E)$, where P is the set of communicating pairs. We will simply refer to the middlebox instances $m \in M$ and pairs $p \in P$ in the bipartite graph as the *nodes*. The edge set E connects middlebox instances $m \in M$ to those communicating pairs $p \in P$ which can be routed via m without exceeding the route stretch constraint ρ, i.e., $E = \{(m,p) \ : \ m \in M, \ p = (s,t) \in P, \ d(s,m) + d(m,t) \leq \rho \cdot d(s,t)\}$, where $d(u,v)$ denotes the length of shortest path between nodes u and v in the network. For each p, the set of such middlebox nodes can be computed in a pre-processing step by performing an all-pairs shortest paths algorithm to compute $d(u,v)$ for each u, v in the network, and for each $p = (s,t) \in P$, selecting the nodes $m \in M$ with $d(s,m) + d(m,t) \leq \rho \cdot d(s,t)$.

A *partial assignment* $A(M) \subseteq B(M)$ of pairs $p \in P$ to middlebox instances in M is a subgraph of $B(M)$, in which each $p \in P$ is connected to at most one middlebox $m \in M$ by an edge, i.e., $\deg_{A(M)}(p) \leq 1$ where deg denotes the degree. A pair $p \in P$ with $\deg_{A(M)}(p) = 1$ is called an *assigned pair*, and a pair with $\deg_{A(M)}(p) = 0$ is called an *unassigned pair* or a *free pair*. A partial assignment $A(M)$ without free pairs is called an *assignment*. The *size* $|A(M)|$ of a (partial) assignment $A(M)$ is defined as the number of edges in $A(M)$.

Our goal is to compute a partial assignment $A(M)$ of pairs $p \in P$ to middlebox instances in M which maximizes the number of assigned pairs. Accordingly, we distinguish between *assignment edges* E_A and *non-assignment edges* $E_{\overline{A}}$, where $E_A \cup E_{\overline{A}} = E$ forms a partition of the edge set E of $B(M)$. Our algorithm ensures that at any moment of time, the partial assignments are *feasible*, i.e., the assignment fulfills the following capacity constraints. The *current load* of a middlebox $m \in M$, denoted by $\lambda(m)$, is the number of communicating pairs served by m according to the current partial assignment $A(M)$. Moreover, we define the *free capacity* $\kappa^*(m)$ of m to be $\kappa^*(m) = \kappa - \lambda(m)$. A (partial) assignment $A(M)$ is feasible if and only if it does not violate capacities, i.e., $\lambda(m) \leq \kappa$, for all $m \in M$.

In order to compute the integer function $\phi(M)$, which is the size $|A^*(M)|$ of a maximum feasible partial assignment $A^*(M)$, we make use of augmenting paths. Let $A(M)$ be a feasible partial assignment. An *augmenting path* $\pi = (v_1, v_2, \ldots, v_j)$ relative to $A(M)$ in $B(M)$ starts at a middlebox $m \in M$ with free capacity, ends at a free pair $p \in P$, and alternates between assignment edges and non-assignment edges, i.e.,

1. $v_1 \in M$ with $\kappa^*(v_1) > 0$ and $v_j \in P$ with $\deg_{A(M)}(v_j) = 0$,
2. $(v_i, v_{i+1}) \in B(M) \setminus A(M)$, for any odd i,
3. $(v_i, v_{i+1}) \in A(M)$, for any even i.

An augmenting path π relative to $A(M)$ is a witness for a better partial assignment: The symmetric difference $A'(M) = (A(M) \setminus \pi) \cup (\pi \setminus A(M)$ is also a (partial) assignment with size $|A'(M)| = |A(M)| + 1$. Due to the properties of the augmenting path, by this reassignment, one additional pair will be covered by the same set of middlebox instances, without violating node capacities: in $A(M)$, the first node had free capacity and the last node represents a free pair. Furthermore, the degree at each internal node of π remains unchanged, since one incident assignment edge gets unassigned and one incident unassigned edge gets assigned. Therefore, the load of the internal nodes in π remains unchanged.

Conversely, an augmenting path must always exist for suboptimal partial assignments. To see this, consider the following reduction to matching: replace each middlebox node with κ many clones of capacity 1, and connect each pair $p \in P$ to all corresponding clones by an edge. Feasible assignments constitute a matching in this graph, and vice versa. Given any suboptimal matching, we find a witness for a augmenting path as follows. We take the symmetric difference of the suboptimal and the optimal solution, which gives us a set of paths and cycles of even length, such that the edges on these paths and cycles are alternating between matching edges and non-matcing edges. Since the optimal solution has strictly more edges than the suboptimal one, a path must exist where the optimal solution has one additional edge. This path is an augmenting path which leads to a larger matching and thus, to a strictly better feasible assignment in the original bipartite graph.

Augmenting paths can be computed efficiently, by simply performing breadth-first searches in $B(M)$. A breadth-first search partitions the nodes of the graph $B(M)$ into layers. The nodes $m \in M$ with free capacities are used as the starting nodes of this search. These nodes form the first layer of the partitioning. At the first level of the search, we use the non-assignment edges. At subsequent levels of the search, the traversed edges are required to alternate between non-assignment and assignment edges. That is, when searching for successors of a node $m \in M$, only non-assignment edges are traversed, while when searching for successors of a node $p \in P$, only assignment edges are traversed. The search terminates at the first layer, where one or more free nodes in P are reached. Then a path from a node $m \in M$ with free capacity to a free node $p \in P$ through the breadth-first search layering is an augmenting path. We obtain such a path by starting a depth-first search from a reached free node $p \in P$.

Submodularity. The set function $\phi : 2^U \to \mathbb{N}$ is called *non-decreasing* if and only if $\phi(U_1) \leq \phi(U_2)$ for all $U_1, U_2 \subseteq U$ with $U_1 \subseteq U_2$, and *submodular* if and only if $\phi(U_1) + \phi(U_2) \geq f(U_1 \cap U_2) + f(U_1 \cup U_2)$ for all $U_1, U_2 \subseteq U$.

For each potential middlebox location $u \in U$, let $S_u \subseteq P$ be the set of communication pairs that can be routed through u in a path with stretch at most ρ, i.e. $S_u = \{p = (s,t) \in P \ : \ d(s,u) + d(u,t) \leq \rho \cdot d(s,t)\}$. For any set of middlebox instances $M \subseteq U$, $\phi(M)$ denotes the maximum number of communication pairs that can be assigned to M, given the capacity constraints at the nodes and the route length constraints. This is the maximum number of communication pairs that can be covered by the collection of sets $\{S_u \ : \ u \in M\}$, such that each set can cover at most κ pairs. The set function ϕ is known to be non-decreasing and submodular (see e.g. [11], [22]).

Summary and Algorithm. Given these insights, we can describe our greedy algorithm. The algorithm starts

Algorithm 1: Greedy Algorithm

```
1  init M ← ∅, A(M) ← empty assignment
2  while A(M) is not a feasible assignment do
3  |  set m* ← ∅, opt ← 0, and tmp ← 0
4  |  foreach u ∈ U \ M do
5  |  |  tmp ← φ(M ∪ {u})   (* augmenting paths *)
6  |  |  if tmp > opt then
7  |  |  |  opt ← tmp, m* ← u
8  |  |  end
9  |  end
10 |  set M ← M ∪ {m*} and update A(M)
11 end
```

with an empty set M and cycles through the possible middlebox locations $u \in U \setminus M$ always deploying the middlebox yielding – together with the already places middleboxes – the highest function value ϕ (see Lines 4-9 in Algorithm 1). Function ϕ can be evaluated on the pre-computed bipartite matching graph using augmenting paths.

Analysis. Given the submodularity and the augmenting path construction, we have derived our main result. The greedy algorithm performs $|M^*|$ iterations, where $|M^*|$ is the number of middleboxes in the solution. We call one iteration a *greedy step*. In each greedy step, the value $\phi(M \cup \{u\})$ is computed for all $u \in U \setminus M$, and the middlebox location u with the highest value is added to M. That is, we compute the value of ϕ exactly $|U \setminus M| \leq n$ times in each greedy step; the value of ϕ is computed $O(n \cdot |M^*|)$ times during the algorithm. For the computation of $\phi(M \cup \{u\})$, $u \in U \setminus M$, we initially have a maximum assignment $A(M)$ for M and all augmentig paths (starting at u) must be computed. One augmenting path can be computed in time linear in the size of the bipartite graph $B(M)$. Using the *Hopcroft-Karp algorithm*, we can even compute all of at most κ many augmenting paths in time $O(\min\{\kappa, \sqrt{|V(B(M))|}\} \cdot |E(B(M))|)$, where $V(B(M))$ and $E(B(M))$ denote the set of nodes and set of edges in $B(M)$. Since $B(M) \subseteq B(M^*)$, this is $O(\min\{\kappa, \sqrt{|V(B(M^*))|}\} \cdot |E(B(M^*))|)$. Therefore, the overall running time of the algorithm is $O(n \cdot |M^*| \cdot \min\{\kappa, \sqrt{|V(B(M^*))|}\} \cdot |E(B(M^*))|)$:

THEOREM 1. *Our greedy, incremental middlebox deployment algorithm computes a solution M^* and a feasible assigment of communicating pairs P to M^*, such that M^* is an $O(\log(\min\{\kappa, n\}))$-approximation of the optimal solution. The overall runtime is bounded by $O(n \cdot |M^*| \cdot \min\{\kappa, \sqrt{|V(B(M^*))|}\} \cdot |E(B(M^*))|)$.*

Hardness and Lower Bound. Theorem 1 is essentially the best we can hope for:

THEOREM 2. *The middlebox deployment problem is NP-hard and cannot be approximated within $c \log n$, for some $c > 0$ unless $P = NP$. Furthermore, it is not approximable within $(1 - \epsilon) \ln n$, for any $\epsilon > 0$, unless $NP \subset \text{DTIME}(n^{\log \log n})$.*

PROOF SKETCH. We present a polynomial time reduction from the Minimum Set Cover (MSC) problem, defined as follows: Given a finite set S of n elements and a collection C of subsets of S. A set cover for S is a subset $C' \subseteq C$ such that every element in S is contained in at least one member of C'. The objective is to minimize the cardinality of the set cover C'.

Consider an instance of the MSC problem: let $S = \{v_1, ..., v_n\}$ be a set of n elements, $C = \{S_i \subseteq S, i = 1, ..., m\}$. We define the instance of the corresponding middlebox deployment problem in a network $G = (V, E)$ with a set of communicating pairs P and stretch $\rho = 1$ as follows. For each element $v \in S$, we introduce two nodes v_s and v_t in V. For each subset $S_i \in C$, we introduce a node v_{S_i} in V as well. The edge set E of the network $G = (V, E)$ is defined by the following rule: there is an edge $(v_s, v_{S_i}) \in E$ and an edge $(v_{S_i}, v_t) \in E$ if and only if the corresponding element v is contained in S_i. The set of communicating pairs is defined as $P = \{(v_s, v_t) : v \in S\}$ and the set of potential middlebox locations is defined as $U = \{v_{S_i} : S_i \in C\}$. $G = (V, E)$ is a bipartite graph with partitions U and $\{v_s : v \in S\} \cup \{v_t : v \in S\}$. If $v \in S$ is contained in a set $S_i \in C$ then there is a path of length 2 between the corresponding pair (v_s, v_t) in G. This is also the shortest path between v_s and v_t. In the middlebox deployment problem with stretch $\rho = 1$, a set of nodes $M \subseteq U$ of minimum cardinality must be selected such that between each pair $(v_s, v_t) \in P$ there is a route of length of at most 2 and it contains at least one node of M. By the construction of the network, for each pair $(v_s, v_t) \in P$, there is a route v_s, v_{S_i}, v_t of length 2 in G if and only if $v \in S_i$. Let $M \subseteq U$ be a minimum cardinality solution of the middlebox deployment problem. The node set M implies a minimum cardinality solution for the MSC problem and vice versa. This proves the NP-hardness of the problem.

The inapproximability results follow from the combination of the above reduction and the inapproximability results of the minimum set cover problem by Raz and Safra [19] and by Feige [7]. Raz and Safra [19] proved that the minimum set cover problem is not approximable within $c \log n$, for some $c > 0$, unless $P = NP$. Feige [7] showed the inapproximability within $(1 - \epsilon) \ln n$, for any $\epsilon > 0$, unless $NP \subset \text{DTIME}(n^{\log \log n})$. If we had a better approximation for the middlebox deployment problem, by the above reduction, we would have a better approximation factor for the minimum set cover problem, as well. □

4. EXACT 0-1 PROGRAM

The middlebox deployment problem can be formulated as the 0-1 integer program (IP) given in Algorithm 1. Together with the NP-hardness result this proves the problem's NP-completeness [9].

For all potential middlebox locations $u \in U$, let S_u be the set of pairs P that can be routed through u on a path of stretch at most ρ. i.e. $S_u = \{p = (s, t) \in P : d(s, u) + d(u, t) \leq \rho \cdot d(s, t)\}$, where $d(v, w)$ denotes the length of the shortest path between nodes $v, w \in V$ in the network. S_u can be precomputed efficiently. For all potential middlebox locations $u \in U$, we introduce the binary variable $x_u \in \{0, 1\}$. The variable x_u indicates that u is selected as a middlebox node in the optimal solution M, i.e. $x_u = 1 \Leftrightarrow u \in M$. For all $u \in U$ and $p \in P$, we introduce the binary variable $x_{up} \in \{0, 1\}$. The variable x_{up} indicates that the pair $p = (s, t) \in P$ is assigned to the node $u \in U$, s.t. the path stretch from s to t through u is at most ρ.

The Objective Function (1) requires that a minimum cardinality middlebox set must be selected. Constraints (2) declare that each pair $p = (s, t) \in P$ is assigned to exactly one node $u \in U$. Constraints (3) state that each pair $p = (s, t) \in P$ can only be assigned to a node $u \in U$ with

Integer Program 1: Optimal Middlebox Deployment

$$\text{minimize} \quad \sum_{u \in U} x_u \tag{1}$$

$$\text{s.t.} \quad \sum_{u \in U} x_{up} = 1 \qquad \forall\, p \in P \tag{2}$$

$$\sum_{p \notin S_u} x_{up} = 0 \qquad \forall\, u \in U \tag{3}$$

$$\sum_{p \in P} x_{up} \leq \kappa \cdot x_u \qquad \forall\, u \in U \tag{4}$$

$$x_u, x_{up} \in \{0,1\} \qquad \forall\, u \in U, p \in P \tag{5}$$

$p \in S_u$. By the definition of S_u, the pair p can be routed through u via a path of stretch at most ρ. Constraints (4) describe that the capacity limit κ must not be exceeded at any node, and nodes $u \in U$ which are not selected in the solution M (where the corresponding variable x_u becomes 0 in the solution) are not assigned to any pair $p \in P$.

5. EVALUATION

In order to complement our formal analysis, we conducted a simulation study. In particular, we compare the performance of the greedy algorithm (Algorithm 1) and the Integer Program (IP) presented in Section 4 along the following metrics: (1) runtime, (2) the number of installed middleboxes, and (3) the number of relocations of middleboxes or reassignment of communication pairs in the incremental deployment setting. Our evaluation is based on seven real-world wide-are topologies obtained from the topology zoo collection [10]. As shown in Table 1, the topologies are either country-wide or continent-wide ISP, backbone, or – in case of Geant – research networks. The topologies were selected so that the number of nodes is spread more or less uniformly across the range 20 to 82. Furthermore, all selected topologies provide detailed geographical information for nodes, such that the latency can be estimated based on geographical distance (speed of light). We first consider the offline setting, in which all communication pairs are given in advance, and then consider the incremental setting.

Runtime. In the first set of experiments, we generate scenarios according to the following parameters. Each of the $|V| \cdot (|V|-1)/2$ potential communication pairs is selected with probability p, set to either 0.2, 0.3, or 0.4. To ensure comparability across topologies, all nodes are allowed to host middlebox functionality. The capacity κ is set to $\lceil 2 \cdot (|V|-1) \cdot p \rceil$: Hence, the number of to be installed middleboxes is likely to be higher than $|V|/4$. For

Name	Year	Size	$\lvert V\rvert$	$\lvert E\rvert$
Quest	1998	Continent	20	62
GtsHungary	2009	Country	30	62
Geant	2012	Continent	40	122
Surfnet	2010	Country	50	136
Forthnet	2010	Country	62	124
Telcove	2004	Country	71	140
Ulaknet	2010	Country	82	164

Table 1: Topology Zoo Instances used

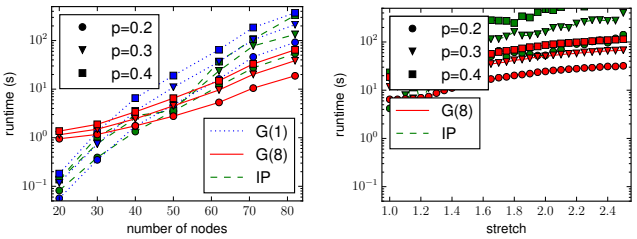

Figure 1: Runtime of the algorithms as a function of the different topologies (left) and the stretch for Ulaknet (right).

each topology and each probability, we generate 15 scenarios. All communication pairs are allowed the same maximal (latency) stretch. Concretely, we consider stretches of $\{1.00, 1.05, 1.10, \ldots 2.50\}$, which, when multiplied with the shortest path latency, defines the maximal allowed latency. A stretch of 1.00 implies that middleboxes must lie on (one of the possible) the respective shortest path.

The greedy approximation algorithm and the IP were implemented in Python. The IP uses Gurobi, a commercial solver, to compute optimal solutions. The computation of the middlebox to add in Algorithm 1 was parallelized.

In Figure 1 the average runtime of the greedy (G) algorithm with 1 and 8 threads and of the IP is depicted. In the left plot the averaged runtimes are shown. Each data point represents the aggregate of $31 \cdot 15 = 465$ experiments. While the average runtime of the greedy algorithm with one thread lies below the one of the IP for 20 and 30 nodes, the greedy algorithm with 8 threads clearly outperforms the IP on the topologies with 62, 71 and 82 nodes. On the largest topology the computation of the greedy algorithm can be sped up by a factor of around 5, by using 8 threads. Furthermore, the runtime of the IP on the largest topology Ulaknet is one magnitude higher than the one of the 8-threaded greedy algorithm. The right plot depicts the runtime on this topology as a function of the stretch. Starting at a stretch of 1.3, the runtime of the IP increases dramatically with the number of potential middleboxes for serving a communication pair.

Impact on the number of required middleboxes. Regarding the number of installed middleboxes, the left box plot of Figure 2 shows the approximation ratio, i.e., the number of middleboxes opened by the greedy algorithm divided by the optimal number of middleboxes computed by the IP. The median lies below 1.4 and the maximum is reached close to 1.8. Furthermore, the right plot of Figure 2 depicts how the number of used middleboxes decreases when the stretch grows. Concretely, the ratio of used middleboxes is divided by the number when the stretch equals 1.0. Interestingly, both algorithms can – on average – reduce the number of necessary middleboxes by 4% when considering a path stretch of only 5%. While the IP achieves a reduction of up to 15%, the greedy algorithm performs worse. This indicates that the approximation factor of the greedy algorithm worsens with increased stretch.

Incremental deployment and reconfigurations. We also study the opportunities arising when deploying middleboxes incrementally. Concretely, we consider the change in middlebox locations and the number of (re-)assignments of communication pairs under both algorithms that arise when a single new communication pair must be served. Our experimental setup is as follows. We first generate a set of

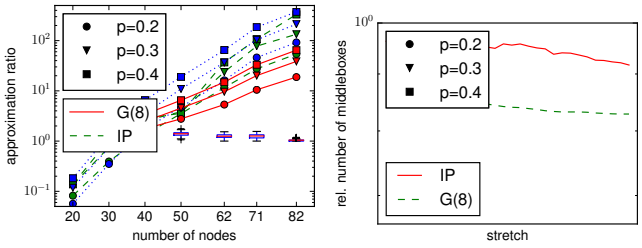

Figure 2: Empirical approximation ratio (*left*) and relative reduction of middleboxes with increasing stretch (*right*).

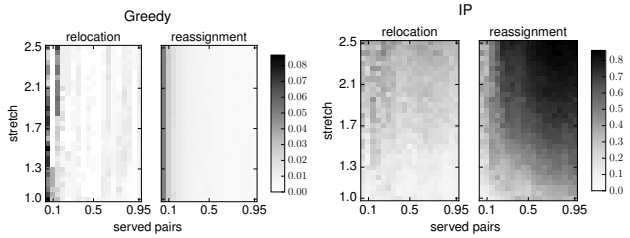

Figure 3: Relative relocation and reassignment of the greedy algorithm and the IP. Note the different respective scales.

communication pairs as above with a probability $p = 0.3$ on the medium-sized Surfnet topology. We consider the optimal solution obtained by the IP for assigning the first $\{0.05, 0.10, \cdots, 0.95\}$ percent of the generated communication pairs and obtain a single data point by adding the single next pair to the scenario. For these experiments, the optimal IP solution is read into the matching graph used by the greedy algorithm to warm-start it. The IP is executed as is. We consider 25 different experiment runs.

The metrics considered are the relative number of relocations and the relative reassignments. We define the relative relocation as $|M' \setminus M|/|M'|$, where M is the set of middleboxes installed before adding the new commmunication pair and M' denotes middlebox locations outputted by the respective algorithm. Similarly, we define the relative reassignment by $|A(M') \setminus A(M)|/|A(M')|$, where $A(M')$ denotes the assignments including the novel communication pair and $A(M)$ represents the previous assignments.

Our results are depicted in Figure 3. We first note the strong correlation between relocations and reassignments for both algorithms. The greedy algorithm relocates and reconfigures less than 8% of the middleboxes and assignments respectively. In fact, as soon as 10% of the communication pairs are served, the number of relative relocations and reassignments barely exceeds 2%. The IP performs much worse. Even when 50% of the communication pairs are already connected, the produced solutions vary by approximately 25% in terms of the chosen middleboxes and up to 70% of the assignments respectively. While this performance gap is due to the fact that the greedy algorithm is warm-started, it shows that the greedy algorithm yields highly stable solutions.

6. SUMMARY

This paper presented a deterministic and greedy approximation algorithm for deploying middleboxes as e.g. firewalls or proxies. The found algorithm performs particularly well in incremental deployment scenarios and also achieves near-optimal performance in the offline scenario.

7. REFERENCES

[1] A. Gember-Jacobson et al. OpenNF: Enabling innovation in network function control. In *Proc. ACM SIGCOMM*, 2014.

[2] M. Bagaa, T. Taleb, and A. Ksentini. Service-aware network function placement for efficient traffic handling in carrier cloud. In *Proc. IEEE WCNC*, 2014.

[3] J. Chuzhoy and J. Naor. Covering problems with hard capacities. *SIAM J. Comput.*, 2006.

[4] V. Chvatal. A greedy heuristic for the set-covering problem. *Math. of Operations Research*, 4(3), 1979.

[5] D. Levin et al. Panopticon: Reaping the benefits of incremental sdn deployment in enterprise network. In *Proc. USENIX ATC*, 2014.

[6] N. Feamster, J. Rexford, and E. Zegura. The road to sdn. *Queue*, 11(12):20:20–20:40, Dec. 2013.

[7] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[8] J. Sherry et al. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proc. ACM SIGCOMM*, 2012.

[9] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. 1972.

[10] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29(9):1765 –1775, october 2011.

[11] L. Lovász. Submodular functions and convexity. *Mathematical Programming The State of the Art*, 1983.

[12] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[13] T. Lukovszki and S. Schmid Online admission control and embedding of service chains. In *Proc. SIROCCO*, 2015.

[14] M. Markovitch and S. Schmid. Shear: A highly available and flexible network architecture: Marrying distributed and logically centralized control planes. In *Proc. 23rd IEEE ICNP*, 2015.

[15] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo. Towards making network function virtualization a cloud computing service. In *IFIP/IEEE IM*, 2015.

[16] S. Raza, G. Huang, C.-N. Chuah, S. Seetharaman, and J. P. Singh. Measurouting: A framework for routing assisted traffic monitoring. *IEEE/ACM Trans. Netw.*, 20(1):45–56, Feb. 2012.

[17] T. Ritter. Network-based firewall services: Extending the firewall into the cloud. In *Nemertes White Paper N0496*, 2009.

[18] S. Fayazbakhsh et al. Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions. In *Proc. ACM HotSDN*, 2013.

[19] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability pcp characterization of np. In *Proc. ACM STOC*, 1997.

[20] A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer-Verlag, 2003.

[21] S. Vissicchio, L. Vanbever, and O. Bonaventure. Opportunities and Research Challenges of Hybrid Software Defined Networks. *ACM CCR*, 2014.

[22] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[23] Z. Qazi et al. Simple-fying middlebox policy enforcement using sdn. In *Proc. ACM SIGCOMM*, 2013.