An Approximation Algorithm for Path Computation and Function Placement in SDNs

Guy Even[1], Matthias Rost[2], and Stefan Schmid[3]

1 **School of Electrical Engineering**
Tel Aviv University
Tel Aviv 6997801
Israel
guy@eng.tau.ac.il

2 **Technische Universität Berlin**
10587 Berlin
Germany
mrost@inet.tu-berlin.de

3 **Department of Computer Science**
Aalborg University
DK-9220 Aalborg
Denmark
schmiste@cs.aau.dk

Abstract

We consider the task of embedding multiple service requests in Software-Defined Networks (SDNs), i.e. computing (combined) mappings of network functions on physical nodes and finding routes to connect the mapped network functions. A single service request may either be fully embedded or be rejected. The objective is to maximize the sum of benefits of the served requests, while the solution must abide node and edge capacities.

We follow the framework suggested by Even *et al.* [5] for the specification of the network functions and routing of requests via processing-and-routing graphs (PR-graphs): a request is represented as a directed acyclic graph with the nodes representing network functions. Additionally, a unique source and a unique sink node are given for each request, such that any source-sink path represents a feasible chain of network functions to realize the service. This allows for example to choose between different realizations of the same network function. Requests are attributed with a global demand (e.g. specified in terms of bandwidth) and a benefit.

Our main result is a randomized approximation algorithm for path computation and function placement with the following guarantee. Let m denote the number of links in the substrate network, ε denote a parameter such that $0 < \varepsilon < 1$, and opt^* denote the maximum benefit that can be attained by a fractional solution (one in which requests may be partly served and flow may be split along multiple paths). Let $c_{\min}$ denote the minimum edge capacity, let $d_{\max}$ denote the maximum demand, and let $b_{\max}$ denote the maximum benefit of a request. Let $\Delta_{\max}$ denote an upper bound on the number of processing stages a request undergoes. If $c_{\min}/(\Delta_{\max} \cdot d_{\max}) = \Omega((\log m)/\varepsilon^2)$, then with probability at least $1 - \frac{1}{m} - exp(-\Omega(\varepsilon^2 \cdot \mathsf{opt}^*/(b_{\max} \cdot d_{\max})))$, the algorithm computes a $(1-\varepsilon)$-approximate solution.

1998 ACM Subject Classification F.2 ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

Keywords and phrases Approximation algorithms, linear programming, randomized rounding, software defined networks, routing, throughput maximization.

1 Introduction

Software Defined Networks (SDNs) and Network Function Virtualization (NFV) have been reinventing key issues in networking [7]. The key characteristics of these developments are: (i) separation between the data plane and the control plane, (ii) specification of the network control from a global view, (iii) introduction of network abstractions that provide a simple networking model, and (iv) programmability and virtualization of network components.

In this paper we focus on an algorithmic problem that an orchestrator needs to solve in an SDN+NFV setting, namely jointly optimizing the *path computation and function placement* (PCFP) [5]: In modern networks, networking is not limited to forwarding packets from sources to destinations. Requests can come in the form of flows (i.e., streams of packets from a source node to a destination node with a specified packet rate) that must undergo processing stages on their way to their destination. Examples of processing steps include: compression, encryption, firewall validation, deep packet inspection, etc. The crystal ball of SDN+NFV is the introduction of abstractions that allow one to specify, per request, requirements such as processing stages, valid locations for each processing stage, and allowable sets of links along which packets can be sent between processing stages. An important application for such goals is supporting security requirements that stipulate that unencrypted packets do not traverse untrusted links or reach untrusted nodes.

From an algorithmic point of view, the path computation and function placement problem combines two different optimization problems. Path computation alone (i.e., the case of pure packet forwarding without processing of packets) is an integral path packing problem. Function mapping alone (i.e., the case in which packets only need to be processed but not routed) is a load balancing problem.

To give a feeling of the problem, consider a special case of requests for streams, each of which needs to undergo the same sequence of k processing stages denoted by $w_1, w_2, \ldots, w_k$. This means that service of a request from s_i to t_i is realized by a concatenation of $k+1$ paths: $s_i \overset{p_0}{\rightsquigarrow} v_1 \overset{p_1}{\rightsquigarrow} v_2 \overset{p_2}{\rightsquigarrow} \cdots \overset{p_{k-1}}{\rightsquigarrow} v_k \overset{p_k}{\rightsquigarrow} t_i$, where processing stage w_i takes place in node v_i. Note that the nodes $v_1, \ldots, v_k$ need not be distinct and the concatenated path $p_0 \circ p_1 \circ \cdots \circ p_k$ need not be simple. A collection of allocations that serve a set of requests not only incurs a forwarding load on the network elements, it also incurs a computational load on the nodes. The computational load is induced by the need to perform the respective processing stages for the requests.

Previous works. The opportunities introduced by the SDN/NFV paradigm, in terms of novel services which can be deployed quickly and on-demand, has inspired much research over the last years [3, 4, 11, 12, 17]. The main focus of these works is usually on the system aspects, while less attention has been given to the algorithmic challenges. Moreover, the existing papers which do deal with the algorithmic challenges, often resort to heuristics or non-polynomial algorithms. For example, in the seminal work on service chaining [17] as well as in [9, 16, 18], mixed-integer programming is employed (and heuristics are sketched), Hartert et al. [12] use constrained programming, and others propose fast heuristics without approximation guarantees [1, 2], or ignore important aspects of the problem such as link capacity constraints [8]. The online version is studied in [5] in which also a new standby/accept service model is introduced, and in [8]. More generally, the problem of combined path computation and function placement is closely related to the virtual network embedding problem, for which many exponential-time and heuristic algorithms have been developed over the last years [6]. Indeed, only recently a first approximation scheme based on randomized rounding for the virtual network embedding problem was proposed in [15]. While the model

is more general by allowing for cyclic request graphs, the proposed algorithms might violate node and edge capacities by a logarithmic factor and is only applicable on a limited class of request graphs.

Our starting point is the model of SDN requests presented in [5]. In this model, each request is represented by a special graph, called a processing-and-route graph (PR-graph, in short). The PR-graph represents both the routing requirement and the processing requirements that the packets of the stream must undergo. We also build on the technique of graph products for representing valid realizations of requests [5].

Raghavan [13] initiated the study of randomized rounding techniques for multi-commodity flows, where the LP has two types of constraints: capacity constraints and demand constraints. The joint capacity constraints are common to all the flows, while the demand constraints are per request. Raghavan proves that randomized rounding succeeds with high probability if the ratio of the minimum capacity to maximum demand is logarithmic.

Contribution and Techniques. To the best of our knowledge, we present the first polynomial-time algorithm which comes with provable approximation guarantees for the PCFP-problem, under reasonable assumptions (i.e., logarithmic capacity-to-demand ratio, few processing stages per request, and sufficiently large optimal benefit). We begin by formulating a fractional relaxation of the problem. The fractional relaxation consists of a set of fractional single commodity flows, each over a *different* product graph. Each flow is fractional in the sense that it may serve only part of a request and may split the flow among multiple paths. We emphasize that the fractional flows do not constitute a multi-commodity flow because they are over different graphs. The fractional problem is a general packing LP [13]. Namely, the LP can be formulated in the form $\max\{\mathbf{b}^T \cdot \mathbf{x} \mid \mathbf{A} \cdot \mathbf{x} \leq \mathbf{c}, \mathbf{x} \geq \mathbf{0}\}$, where all the components of the vectors $\mathbf{b}, \mathbf{c}$ and the matrix A are nonnegative. However, this LP does not satisfy the logarithmic ratio required in Raghavan's analysis of general packing problems (due to demand constraints).

Although randomized rounding is very well known and appears in many textbooks and papers, the version for the general packing problem appears only in half a page in the thesis by Raghavan [13, p. 41]. A special case with unit demands and unit benefits appears in [10]. One of the contributions of this paper is a full description of the analysis of randomized rounding for the case of multiple-commodity flows over different graphs with joint capacity constraints.

2 Modeling Requests in SDN

We model SDN/NFV requests as process-and-route graphs (PR-graphs) [5]. The model is quite general, and allows each request to have multiple sources and destinations, varying bandwidth demands based on processing stages, task specific capacities, prohibited locations of processing, and prohibited links for routing between processing stages, etc. We overview a simplified version of this model to concisely define the problem of path computation and function placement (PCFP).

2.1 The Substrate Network

The substrate network is a fixed network of servers and communication links. The network is represented by a graph $N = (V, E)$, where V is the set of *nodes* and E is the set of *edges*. Nodes and edges have *capacities*. The capacity of an edge e is denoted by $c(e)$, and the capacity of a node $v \in V$ is denoted by $c(v)$. Let $c_{\min}$ denote the minimum capacity. We

note that the network is static and undirected (namely each edge represents a bidirectional communication link), but may contain parallel edges.

2.2 Requests and PR-Graphs

Each request is specified by a tuple $r_i = (G_i, d_i, b_i, U_i, s_i, t_i)$, where the components are as follows:

1. $G_i = (X_i, Y_i)$ is a directed (acyclic) graph called the process-and-route graph (PR-graph). There is a single source (respectively, sink) that corresponds to the source (resp. destination) of the request. We denote the source and sink nodes in G_i by s_i and t_i, respectively. The other vertices correspond to services or processing stages of a request. The edges of the PR-graph are directed and indicate precedence relations between PR-vertices. Any s_i-t_i path in G_i represents a valid realization of request i.
2. The demand of r_i is d_i and its benefit is b_i. By scaling, we may assume that $\min_i b_i = 1$.
3. $U_i : X_i \cup Y_i \to 2^V \cup 2^E$ where (1) $U_i(x) \subseteq V$ denotes a set of "allowed" nodes in the substrate N that can perform service x, and (2) $U_i(y) \subseteq E$ denotes the set of "allowed" edges of the substrate N along which the routing segment that corresponds to y may be routed.

Note that in the above definition the function $U_i(x)$ returns a set of substrate locations on which the function $x \in X_i$ can be executed. This allows to model network function *types*: If a substrate node $v \in V$ represents a specific hardware appliance (e.g. a firewall), then this node can only host this specific type of network function. Hence, if a virtualized network function $x \in X_i$ has the same type as $v \in V$, we include v in $U_i(x)$ and exclude v from $U_i(x)$ otherwise.

Given this understanding of the restriction function U_i, PR-graphs allow to model the selection of specific implementations of network functions. Assume e.g. that a request i is given that shall connect $v \in V$ to $u \in V$ such that the traffic passes through a firewall. The substrate network may offer two types of firewall implementations: a hardware-based (as hardware appliance) and a software-based (as virtual machine). Using the definition of PR-graphs, the selection of either of the choices can be modeled by setting $X_i = \{s_i, x_{\text{hw}}, x_{\text{sw}}, t_i\}$ and $Y_i = \{(s_i, x_{\text{hw}}), (s_i, x_{\text{sw}}), (x_{\text{hw}}, t_i), (x_{\text{sw}}, t_i)\}$ and restricting $U_i(x_{\text{hw}})$ to all the hardware firewalls and $U_i(x_{\text{sw}})$ to the set of all nodes that may host software firewalls. As any s_i-t_i path in G_i represents a valid realization of request i, a mapping of request i must select any of the options to realize the request (cf. [16] for a general discussion on decomposition opportunities).

We denote the maximum demand by any request as $d_{\max}$ and the maximum benefit of any request as $b_{\max}$.

2.3 The Product Network

In [5] the concept of product graphs was introduced and we shortly revisit the definition. For each request r_i, the product network $\text{pn}(N, r_i)$ is defined as follows. The node set of $\text{pn}(N, r_i)$, denoted V_i, is defined as $V_i \triangleq \cup_{y \in Y_i} (V \times \{y\})$. We refer to the subset $V \times \{y\}$ as the y-layer in the product graph. Note that there is a layer for every edge y in the PR-graph. The edge set of $\text{pn}(N, r_i)$, denoted E_i, consists of two types of edges $E_i = E_{i,1} \cup E_{i,2}$ defined as follows.

1. *Routing edges* connect vertices in the same layer.

$$E_{i,1} = \big\{ \big((u, y), (v, y)\big) \mid y \in Y_i,\ (u, v) \in U_i(y) \big\} \ .$$

2. Directed *processing edges* connect two copies of the same network vertex in different layers.

$$E_{i,2} = \{((v,y),(v,y')) \mid y,y' \in Y_i \text{ with } y = (\cdot,x), y' = (x,\cdot) \text{ and } v \in U_i(x)\}\,.$$

To simplify the description of valid realizations, we add a super source $\hat{s}_i$ and a super sink $\hat{t}_i$ to the respective product networks. The super source $\hat{s}_i$ is connected to all vertices (v,y) such that $v \in U_i(s_i)$ and y emanates from s_i. Similarly, there is an edge to the super sink $\hat{t}_i$ from all vertices (v,y) such that $v \in U_i(t_i)$ and y enters t_i.

Remarks. The following remarks may help clarify the definition of the product network.

1. Consider an edge $y = (x_1, x_2)$ of a request. The y-layer in the product graph contains a copy of the substrate to compute a route from the vertex that performs the x_1 processing to the vertex that performs the x_2 processing.
2. Consider two edges $y_1 = (x_1, x_2)$ and $y_2 = (x_2, x_3)$ in the PR-graph. The only processing edges between the y_1-layer and the y_2-layer are edges of the form $(v, y_1) \to (v, y_2)$, where $v \in U_i(x_2)$.
3. If we coalesce each layer of the PR-graph to a single vertex, then the resulting graph is the line graph of the PR-graph.

2.4 Valid Realizations of SDN Requests

We use product graphs to define valid realizations of SDN requests. Consider a path $\tilde{p}_i$ in the product graph $\mathrm{pn}(N, r_i)$ that starts in the super source $\hat{s}_i$ and ends in the super sink $\hat{s}_i$. Such a path $\tilde{p}_i$ represents the routing of request r_i from its origin to its destination and the processing stages that it undergoes. The processing edges along $\tilde{p}_i$ represent nodes in which processing stages of r_i take place. The routing edges within each layer represent paths along which the request is routed between processing stages. (The edges incident to the super source and super sink are not important).

▶ **Definition 1.** A path $\tilde{p}$ in the product network $\mathrm{pn}(N, r_i)$ that starts in the super source and ends in the super sink is a *valid realization* of request r_i.

2.5 The Path Computation and Function Placement Problem (PCFP)

Modeling SDN requests by product graphs helps in reducing SDN requests to path requests. The translation of paths in the product graph back to paths in the substrate network is called projection. This translation involves a loss due to multiple occurrences of the same substrate resource along a path in the product graph. We define projection and multiplicity before we present the formal definition of the PCFP-problem.

Projection of paths. Let $\tilde{p}_i$ denote a path in the product graph $\mathrm{pn}(N, r_i)$ from the super source to the super sink. The projection of $\tilde{p}_i$ to a path $p_i = \pi(\tilde{p}_i)$ in the substrate network N is simply the projection onto routing edges of $\tilde{p}_i$. Namely, each routing edge $((u,y),(v,y))$ in $\tilde{p}_i$ is projected to the edge (u,v) in the substrate. Hence, when projecting a path, we ignore the processing edges and the edges incident to the super source and super sink. Note that $p = \pi(\tilde{p}_i)$ may not be a simple path even if $\tilde{p}_i$ is simple.

Notation. The *multiplicity* of an edge or a vertex z in a path p is the number of times z appears in the path. We denote the multiplicity of z in p by $\mathsf{multiplicity}(z,p)$.

Capacity Constraints. Let $\tilde{P} = \{\tilde{p}_i\}_{i \in I'}$ denote a set of valid realizations for a subset of requests $\{r_i\}_{i \in I'}$ with $I' \subseteq I$. The set $\tilde{P}$ *satisfies the capacity constraints* if

$$\sum_{i \in I} d_i \cdot \mathsf{multiplicity}(e, \pi(\tilde{p}_i)) \leq c(e), \quad \text{for every edge } e \in E$$
$$\sum_{i \in I} d_i \cdot \mathsf{multiplicity}(v, \pi(\tilde{p}_i)) \leq c(v), \quad \text{for every vertex } v \in V$$

Definition of the PCFP-problem. The input in the PCFP-problem consists of (1) a substrate network $N = (V, E)$ with vertex and edge capacities, and (2) a set of requests $\{r_i\}_{i \in I}$. The goal is to compute valid realizations $\tilde{P} = \{\tilde{p}_i\}_{i \in I'}$ for a subset $I' \subseteq I$ such that: (1) $\tilde{P}$ satisfies the capacity constraints, and (2) the benefit $\sum_{i \in I'} b_i$ is maximum. We refer to the requests r_i such that $i \in I'$ as the *accepted* requests; requests r_i such that $i \in I \setminus I'$ are referred to as *rejected* requests.

3 The Approximation Algorithm for PCFP

The approximation algorithm for the PCFP-problem is described in this section. It is a variation of Raghavan's randomized rounding algorithm for general packing problems [13, Thm 4.7, p. 41] (in which the approximation ratio is $\frac{1}{e} - \sqrt{\frac{2 \ln n}{\varepsilon \cdot e \cdot \mathsf{opt}^*}}$ provided that $\frac{c_{\min}}{d_{\max}} \geq \frac{\ln n}{\varepsilon}$).

3.1 Fractional Relaxation of the PCFP-problem

We now define the fractional relaxation of the PCFP-problem. Instead of assigning a valid realization $\tilde{p}_i$ per accepted request r_i, we assign a fractional single commodity flow $\tilde{f}_i$ in the product graph $\mathrm{pn}(N, r_i)$. The source of the flow $\tilde{f}_i$ is the super source $\hat{s}_i$. Similarly, the destination of $\tilde{f}_i$ is the super sink $\hat{t}_i$. The demand of $\tilde{f}_i$ is d_i. Hence the demand constraint is $|\tilde{f}_i| \leq d_i$.

The capacity constraints are accumulated across all requests' flows. Formally,

$$\sum_{i,y} \tilde{f}_i((u,y),(v,y)) \leq c(u,v)$$
$$\sum_{i,y,y'} \tilde{f}_i((v,y),(v,y')) \leq c(v).$$

Hence, the cumulative load on the copies of substrate edge $(u, v) \in E$ in the respective PR-graphs is upper bounded by the original edge capacity $c(u, v)$. Similarly, as the usage of processing edges $((v, y), (v, y'))$ in the PR-graphs denotes the processing on substrate node v, the cumulative load is bounded by $c(v)$.

The objective function of the LP relaxation is to maximize $\sum_i b_i \cdot |\tilde{f}_i|/d_i$.

We emphasize that this fractional relaxation is not a classic multi-commodity flow. The reason is that each flow $\tilde{f}_i$ is defined over a different product graph. However, the fractional relaxation is a general packing LP.

3.2 The Algorithm

The algorithm uses a parameter $1 > \epsilon > 0$. The algorithm proceeds as follows.

1. Divide all the capacities by $(1 + \varepsilon)$. Namely, $\tilde{c}(e) = c(e)/(1 + \varepsilon)$ and $\tilde{c}(v) = c(v)/(1 + \varepsilon)$.
2. Compute a maximum benefit fractional PCFP solution $\{\tilde{f}_i\}_i$.

3. Apply the randomized rounding procedure independently to each flow $\tilde{f}_i$ over the product network $\mathrm{pn}(N, r_i)$ (See Appendix B for a description of the procedure). Let $\tilde{p}_i$ denote the path in $\mathrm{pn}(N, r_i)$ (if any) that is assigned to request r_i by the randomized rounding procedure. Let f_i denote a flow of amount d_i along the projection $\pi(\tilde{p}_i)$. Note that each f_i is an unsplittable all-or-nothing flow. The projection of p_i might not be a simple path in the substrate, hence the flow $f_i(e)$ along the edge e can be a multiple of the demand d_i.

3.3 Analysis of the algorithm

▶ **Definition 2.** The *diameter* of G_i is the length of a longest path in G_i from the source s_i to the destination t_i. We denote the diameter of G_i by $\Delta(G_i)$.

The diameter of G_i is well defined because G_i is acyclic for every request r_i. In all applications we are aware of, the diameter $\Delta(G_i)$ is bounded by a constant (i.e., e.g. less than 10).

Notation. Let $\Delta_{\max} \triangleq \max_{i \in I} \Delta(G_i)$ denote the maximum diameter of a request. Let $c_{\min}$ denote the minimum edge capacity, and let $d_{\max}$ denote the maximum demand. Let opt^* denote the maximum benefit achievable by a fractional PCFP solution (with respect to the original capacities $c(e)$ and $c(v)$). Let ALG denote the solution computed by the algorithm. Let $B(S)$ denote the benefit of a solutions S. Define $\beta(\varepsilon) \triangleq (1+\varepsilon)\ln(1+\varepsilon) - \varepsilon$.

Our goal is to prove the following theorem.[1]

▶ **Theorem 3.** *Assume that* $\frac{c_{\min}}{\Delta_{\max} \cdot d_{\max}} \geq \frac{4.2+\varepsilon}{\varepsilon^2} \cdot (1+\varepsilon) \cdot \ln |E|$ *and* $\varepsilon \in (0,1)$. *Then,*

$$\mathbf{Pr}\left[\text{ALG } \textit{does not satisfy the capacity constraints}\right] \leq \frac{1}{|E|} \tag{1}$$

$$\mathbf{Pr}\left[B(\text{ALG}) < \frac{1-\varepsilon}{1+\varepsilon} \cdot B(\mathsf{opt}^*)\right] \leq e^{-\beta(-\varepsilon) \cdot B(\mathsf{opt}^*)/((1+\varepsilon) \cdot b_{\max} \cdot d_{\max})}. \tag{2}$$

We remark in asymptotic terms, the theorem states that if $\frac{c_{\min}}{\Delta_{\max} \cdot d_{\max}} = \Omega(\frac{\log |E|}{\varepsilon^2})$, then ALG satisfies the capacity constraints with probability $1 - O(1/|E|)$ and attains a benefit of $(1 - O(\varepsilon)) \cdot B(\mathsf{opt}^*)$ with probability $1 - \exp(-\Omega(\varepsilon^2) \cdot B(\mathsf{opt}^*)/(b_{\max} \cdot d_{\max}))$.

Proof. The proof is based on the fact that randomized rounding is applied to each flow $\tilde{f}_i$ independently. Thus the congestion of an edge in ALG is the sum of independent random variables. The same holds for the $B(\text{ALG})$. The proof proceeds by applying Chernoff bounds.

Proof of Eq. 1. For the sake of simplicity we assume that there are no vertex capacities (i.e., $c(v) = \infty$). The proof is based on the Chernoff bound in Theorem 6. To apply the bound, fix a substrate edge $e \in E$, where $e = (u, v)$. Recall that the randomized rounding procedure decides which requests are supplied. A supplied request r_i is assigned a path $\tilde{p}_i$ in the product network $\mathrm{pn}(N, r_i)$. The path $\tilde{p}_i$ is the support of a single commodity flow f'_i with flow amount $|f'_i| = d_i$. The projection of f'_i to the substrate network is denoted by f_i and its support is the projected path $\pi(\tilde{p}_i)$. The multiplicity of every edge in $\pi(\tilde{p}_i)$ is at most $\Delta_{\max}$. Hence, for every edge e, $f_i(e)$ is a multiple of d_i between 0 and $\Delta_{\max} \cdot d_i$.

[1] We believe there is a typo in the analogous theorem for integral MCFs with unit demands and unit benefits in [10, Thm 11.2, p. 452] and that a factor of ε^{-2} is missing in their lower bound on the capacities.

Define the random variables X_i and the upper bounds μ_i on their expectation as follows (recall that $e = (u, v)$).

$$X_i \triangleq \frac{f_i(e)}{\Delta_{\max} \cdot d_{\max}}$$
$$\mu_i \triangleq \frac{\tilde{c}(e)}{\Delta_{\max} \cdot d_{\max}} \cdot \frac{\sum_y \tilde{f}_i((u,y),(v,y))}{\sum_{j,y} \tilde{f}_j((u,y),(v,y))}$$

The conditions of Theorem 6 are satisfied for the following reasons. The random variables X_i are independent and $0 \leq X_i \leq 1$ because $f_i(e) \in \{0, d_i, \ldots, \Delta_{\max} \cdot d_i\}$. Also, by Claim 1 (see Page 12) and linearity of expectation,

$$\mathbf{E}[X_i] = \frac{\sum_y \tilde{f}_i((u,y),(v,y))}{\Delta_{\max} \cdot d_{\max}}.$$

Since $\sum_{j,y} \tilde{f}_j((u,y),(v,y)) \leq \tilde{c}(e)$, it follows that $\mathbf{E}[X_i] \leq \mu_i$. Finally, $\mu \triangleq \sum_{i \in I} \mu_i = \tilde{c}(e)/(\Delta_{\max} \cdot d_{\max})$.

Let $\textsc{alg}(e)$ denote the load incurred on the edge e by ALG. Namely $\textsc{alg}(e) \triangleq \sum_{i \in I} f_i(e)$. Note that $\textsc{alg}(e) \geq (1+\varepsilon) \cdot \tilde{c}(e)$ iff

$$\sum_{i \in I} X_i \geq (1+\varepsilon) \cdot \frac{\tilde{c}(e)}{\Delta_{\max} \cdot d_{\max}} = (1+\varepsilon) \cdot \mu.$$

From Theorem 6 we conclude that:

$$\begin{aligned}\mathbf{Pr}[\textsc{alg}(e) \geq (1+\varepsilon) \cdot \tilde{c}(e)] &= \mathbf{Pr}\left[\sum_{i \in I} X_i \geq (1+\varepsilon) \cdot \mu\right] \\ &\leq e^{-\beta(\varepsilon) \cdot \mu} \\ &= e^{-\beta(\varepsilon) \cdot \tilde{c}(e)/(\Delta_{\max} \cdot d_{\max})}\end{aligned}$$

By scaling of capacities, we have $c(e) = (1+\varepsilon) \cdot \tilde{c}(e)$. By Fact 4, $\beta(\varepsilon) \geq \frac{2\varepsilon^2}{4.2+\varepsilon}$. By the assumption $\frac{\tilde{c}(e)}{\Delta_{\max} d_{\max}} \geq \frac{4.2+\varepsilon}{\varepsilon^2} \cdot \ln |E|$. We conclude that

$$\mathbf{Pr}[\textsc{alg}(e) \geq c(e)] \leq \frac{1}{|E|^2}.$$

Eq. 1 follows by applying a union bound over all the edges.

Proof of Eq. 2. The proof is based on the Chernoff bound stated in Theorem 7. To apply the bound, let

$$X_i \triangleq \frac{b_i \cdot |f_i|}{b_{\max} \cdot d_{\max}}$$
$$\mu_i \triangleq \frac{b_i \cdot |\tilde{f}_i|}{b_{\max} \cdot d_{\max}}.$$

The conditions of Theorem 7 are satisfied for the following reasons. Since $b_i \leq b_{\max}$ and $|f_i| \leq d_{\max}$, it follows that $0 \leq X_i \leq 1$. Note that $\sum_i X_i = B(\textsc{alg})/(b_{\max} \cdot d_{\max})$. By Corollary 1, $\mathbf{E}[X_i] = \mu_i$. By linearity, $\sum_i b_i \cdot |\tilde{f}_i| = \mathsf{opt}^*/(1+\varepsilon)$ and $\mu \triangleq \sum_i \mu_i = \frac{B(\mathsf{opt}^*)}{(1+\varepsilon) b_{\max} \cdot d_{\max}}$. Hence,

$$\begin{aligned}\mathbf{Pr}\left[B(\textsc{alg}) < \frac{1-\varepsilon}{1+\varepsilon} \cdot B(\mathsf{opt}^*)\right] &= \mathbf{Pr}\left[\sum_i X_i < (1-\varepsilon) \cdot \mu\right] \\ &\leq e^{-\beta(-\varepsilon) \cdot \mu} \\ &\leq e^{-\beta(-\varepsilon) \cdot B(\mathsf{opt}^*)/((1+\varepsilon) b_{\max} \cdot d_{\max})},\end{aligned}$$

and the theorem holds.

◀

3.4 Unit Benefits

In the case of unit benefits (i.e., all the benefits equal one and hence $b_{\max} = 1$), Theorem 3 gives a fully polynomial randomized approximation scheme.

▶ **Corollary 4.** *Suppose that $b_{\max} = 1$. Under the premises of Theorem 3, with probability $1 - O(1/Poly(|E|))$, the algorithm returns an all-or-nothing unsplittable multi-commodity flow whose benefit is at least $1 - O(\varepsilon)$ times the optimal benefit.*

Proof. If $B(\mathsf{opt}^*) > c_{\min}$, then the large capacities assumption implies that $B(\mathsf{opt}^*)/(d_{\max} \cdot b_{\max}) \geq c_{\min}/d_{\max} \geq \varepsilon^{-2} \cdot \ln |E|$. This implies that that $B(\text{ALG}) \geq (1 - O(\varepsilon)) \cdot B(\mathsf{opt}^*)$ with probability at least $1 - 1/poly(|E|)$. By adding the probabilities of the two possible failures (i.e., violation of capacities and small benefit) and taking into account the prescaling of capacities, we obtain that with probability at least $1 - O(1/poly(|E|))$, randomized rounding returns an all-or-nothing unsplittable multi-commodity flow whose benefit is at least $1 - O(\varepsilon)$ times the optimal benefit. ◀

4 Discussion

Theorem 3 provides an upper bound for the probability that ALG is not feasible and that $B(\text{ALG})$ is far from $B(\mathsf{opt}^*)$. These bounds imply that our algorithm can be viewed as version of an asymptotic PTAS in the following sense. Suppose that the parameters $b_{\max}$ and $d_{\max}$ are not a function of $|E|$. As the benefit of the optimal solution opt^* increases, the probability that $B(\text{ALG}) \geq (1 - O(\varepsilon)) \cdot B(\mathsf{opt}^*)$ increases. On the other hand, we need the capacity-to-demand ratio to be logarithmic, namely, $c_{\min} \geq \Omega((\Delta_{\max} \cdot d_{\max} \cdot \ln |E|)/\varepsilon^2)$. We believe that the capacity-to-demand ratio is indeed large in realistic networks.

Acknowledgment. This research was supported by the EU project UNIFY FP7-IP-619609 as well as by the German BMBF Software Campus grant 01IS1205.

References

1 A. Abujoda and P. Papadimitriou. Midas: Middlebox discovery and selection for on-path flow processing. In *Proc. COMSNETS Conference*, 2015.

2 D. Dietrich, A. Abujoda, and P. Papadimitriou. Network service embedding across multiple providers with nestor. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9, 2015.

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

4 GSNFV ETSI. Network functions virtualisation (NFV); use cases. *V1.1.1*, 2013.

5 Guy Even, Moti Medina, and Boaz Patt-Shamir. Competitive path computation and function placement in sdns. *CoRR*, abs/1602.06169, 2016.

6 A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, 2013.

7 D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

8 Tamas Lukovszki and Stefan Schmid. Online admission control and embedding of service chains. In *Proc. 22nd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2015.

9 Sevil Mehraghdam, Matthias Keller, and Holger Karl. Specifying and placing chains of virtual network functions. In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pages 7–13. IEEE, 2014.

10 Rajeev Motwani, Joseph Seffi Naor, and Prabhakar Raghavan. Randomized approximation algorithms in combinatorial optimization. In *Approximation algorithms for NP-hard problems*, pages 447–481. PWS Publishing Co., 1996.

11 P. Skoldstrom et al. Towards unified programmability of cloud and carrier infrastructure. In *Proc. European Workshop on Software Defined Networking (EWSDN)*, 2014.

12 R. Hartert et al. Declarative and expressive approach to control forwarding paths in carrier-grade networks. In *Proc. ACM SIGCOMM*, 2015.

13 Prabhakar Raghavan. Randomized rounding and discrete ham-sandwich theorems: provably good algorithms for routing and packing problems. In *Report UCB/CSD 87/312*. Computer Science Division, University of California Berkeley, 1986.

14 Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

15 Matthias Rost and Stefan Schmid. Service chain and virtual network embeddings: Approximations using randomized rounding. *CoRR*, abs/1604.02180, 2016.

16 Sahel Sahhaf, Wouter Tavernier, Matthias Rost, Stefan Schmid, Didier Colle, Mario Pickavet, and Piet Demeester. Network service chaining with optimized network function embedding supporting service decompositions. In *Journal Computer Networks (COMNET), Elsevier*, 2015.

17 Robert Soulé, Shrutarshi Basu, Parisa Jalili Marandi, Fernando Pedone, Robert Kleinberg, Emin Gun Sirer, and Nate Foster. Merlin: A language for provisioning network resources. In *Proc. 10th ACM International on Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 213–226, 2014.

18 M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs. Network function placement for NFV chaining in packet/optical datacenters. *Journal of Lightwave Technology*, 33(8):1565–1570, 2015.

19 Neal E Young. Randomized rounding without solving the linear program. In *SODA*, volume 95, pages 170–178, 1995.

A Multi-Commodity Flows

Consider a directed graph $G = (V, E)$. Assume that edges have non-negative capacities $c(e)$. For a vertex $u \in V$, let $\mathsf{out}(u)$ denote the outward neighbors, namely the set $\{y \in V \mid (u, y) \in E\}$. Similarly, $\mathsf{in}(u) \triangleq \{x \in V \mid (x, u) \in E\}$. Consider two vertices s and t in V (called the *source* and *destination* vertices, respectively). A *flow* from s to t is a function $f : E \to \mathbb{R}^{\geq 0}$ that satisfies the following conditions:

(i) Capacity constraints: for every edge $(u, v) \in E$, $0 \leq f(u, v) \leq c(u, v)$.

(ii) Flow conservation: for every vertex $u \in V \setminus \{s, t\}$

$$\sum_{x \in \mathsf{in}(u)} f(x, u) = \sum_{y \in \mathsf{out}(u)} f(u, y).$$

The *amount* of flow delivered by the flow f is defined by

$$|f| \triangleq \sum_{y \in \mathsf{out}(s)} f(s,y) - \sum_{x \in \mathsf{in}(s)} f(x,s).$$

Consider a set ordered pairs of vertices $\{(s_i,t_i)\}_{i\in I}$. An element $i \in I$ is called a *commodity* as it denotes a request to deliver flow from s_i to t_i. Let $F \triangleq \{f_i\}_{i\in I}$ denote a set of flows, where each flow f_i is a flow from the source vertex s_i to the destination vertex t_i. We abuse notation, and let F denote the sum of the flows, namely $F(e) \triangleq \sum_{i\in I} f_i(e)$, for every edge e. Such a sequence is a *multi-commodity flow* if, in addition it satisfies *cumulative capacity constraints* defined by:

$$\text{for every edge } (u,v) \in E: \quad F(u,v) \leq c(u,v).$$

Demands are used to limit the amount of flow per commodity. Formally, let $\{d_i\}_{i\in I}$ denote a sequence of positive real numbers. We say that d_i is the *demand* of flow f_i if we impose the constraint that $|f_i| \leq d_i$. Namely, one can deliver at most d_i amount of flow for commodity i.

The *maximum benefit optimization problem* associated with multi-commodity flow is formulated as follows. The input consists of a (directed) graph $G = (V,E)$, edge capacities $c(e)$, a sequence source-destination pairs for commodities $\{(s_i,t_i)\}_{i\in I}$. Each commodity has a nonnegative demand d_i and benefit b_i. The goal is to find a multi-commodity flow that maximizes the objective $\sum_{(u,v)\in E} b_i \cdot |f_i|$. We often refer to this objective as the *benefit* of the multi-commodity flow. When the demands are identical and the benefits are identical, the maximum benefit problem reduces to a maximum *throughput* problem.

A multi-commodity flow is *all-or-nothing* if $|f_i| \in \{0, d_i\}$, for every commodity $i \in I$. A multi-commodity flow is *unsplittable* if the support of each flow is a simple path. (The *support* of a flow f_i is the set of edges (u,v) such that $f_i(u,v) > 0$.) We often emphasize the fact that a multi-commodity flow is not all-or-nothing or not unsplittable by saying that it *fractional*.

B Randomized Rounding Procedure

In this section we overview the randomized rounding procedure. The presentation is based on [10]. Given an instance $F = \{f_i\}_{i\in I}$ of a fractional multi-commodity flow with demands and benefits, we are interested in finding an all-or-nothing unsplittable multi-commodity flow $F' = \{f'_i\}_{i\in I}$ such that the benefit of F' is as close to the benefit of F as possible.

▶ Observation 1. As flows along cycles are easy to eliminate, we assume that the support of every flow $f_i \in F$ is acyclic.

We employ a randomized procedure, called *randomized rounding*, to obtain F' from F. We emphasize that all the random variables used in the procedure are independent. The procedure is divided into two parts. First, we flip random independent coins to decide which commodities are supplied. Next, we perform a random walk along the support of the supplied commodities. Each such walk is a simple path along which the supplied commodity is delivered. We describe the two parts in detail below.

Deciding which commodities are supplied. For each commodity, we first decide if $|f'_i| = d_i$ or $|f'_i| = 0$. This decision is made by tossing a biased coin $bit_i \in \{0,1\}$ such that

$$\mathbf{Pr}\left[bit_i = 1\right] \triangleq \frac{|f_i|}{d_i}.$$

If $bit_i = 1$, then we decide that $|f'_i| = d_i$ (i.e., commodity i is fully supplied). Otherwise, if $bit_i = 0$, then we decide that $|f'_i| = 0$ (i.e., commodity i is not supplied at all).

Assigning paths to the supplied commodities. For each commodity i that we decided to fully supply (i.e., $bit_i = 1$), we assign a simple path P_i from its source s_i to its destination t_i by following a random walk along the support of f_i. At each node, the random walk proceeds by rolling a dice. The probabilities of the sides of the dice are proportional to the flow amounts. A detailed description of the computation of the path P_i is given in Algorithm 1.

Algorithm 1 Algorithm for assigning a path P_i to flow f_i.

$P_i \leftarrow \{s_i\}$.
$u \leftarrow s_i$
while $u \neq t_i$ **do** ▷ did not reach t_i yet
    $v \leftarrow$ *choose-next-vertex*(u).
    Append v to P_i
    $u \leftarrow v$
end while
return (P_i).
procedure *choose-next-vertex*(u, f_i) ▷ Assume that u is in the support of f_i
    Define a dice $C(u, f_i)$ with $|\mathsf{out}(u)|$ sides. The side corresponding to an edge (u, v) has probability $f_i(u, v)/(\sum_{(u,v')\in\mathsf{out}(u)} f_i(u, v'))$.
    Let v denote the outcome of a random roll of the dice $C(u, f_i)$.
    return (v)
end procedure

Definition of F'. Each flow $f'_i \in F'$ is defined as follows. If $bit_i = 0$, then f'_i is identically zero. If $bit_i = 1$, then f'_i is defined by

$$f'_i(u, v) \triangleq \begin{cases} d_i & \text{if } (u, v) \in P_i, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, F' is an all-or-nothing unsplittable flow, as required.

C Analysis of Randomized Rounding - Expected Flow Per Edge

The presentation in this section is based on [10].

▶ Claim 1. For every commodity i and every edge $(u, v) \in E$:

$$\mathbf{Pr}\left[(u, v) \in P_i\right] = \frac{f_i(u, v)}{d_i},$$
$$\mathbf{E}\left[f'_i(u, v)\right] = f_i(u, v).$$

Proof. Since

$$\mathbf{E}\left[f'_i(u, v)\right] = d_i \cdot \mathbf{Pr}\left[(u, v) \in P_i\right],$$

it suffices to prove the first part.

An edge (u, v) can belong to the path P_i only if $f_i(u, v) > 0$. We now focus on edges in the support of f_i. By Observation 1, the support is acyclic, hence we can sort the support in

topological ordering. The claim is proved by induction on the position of an edge in this topological ordering.

The induction basis, for edges $(s_i, y) \in \mathsf{out}(s_i)$, is proved as follows. Since the support of f_i is acyclic, it follows that $f_i(x, s_i) = 0$ for every $(x, s_i) \in \mathsf{in}(s_i)$. Hence $|f_i| = \sum_{y \in \mathsf{out}(s_i, f_i)} f_i(s_i, y)$. Hence,

$$\begin{aligned}\mathbf{Pr}\left[(s_i, y) \in P_i\right] &= \mathbf{Pr}\left[bit_i = 1\right] \cdot \mathbf{Pr}\left[\text{dice } C(s_i, f_i) \text{ selects } (s_i, y) \mid bit_i = 1\right] \\ &= \frac{|f_i|}{d_i} \cdot \frac{f_i(s_i, y)}{\sum_{y \in \mathsf{out}(s_i, f_i)} f_i(s_i, y)} \\ &= \frac{f_i(s_i, y)}{d_i},\end{aligned}$$

and the induction basis follows.

The induction step, for an edge (u, v) in the support of f_i such that $u \neq s_i$, is proved as follows. Vertex u is in P_i if and only if P_i contains an edge whose head is u. We apply the induction hypothesis to these incoming edges, and use flow conservation to obtain

$$\begin{aligned}\mathbf{Pr}\left[u \in P_i\right] &= \mathbf{Pr}\left[\bigcup_{x \in \mathsf{in}(u)} (x, u) \in P_i\right] \\ &= \frac{1}{d_i} \cdot \sum_{x \in \mathsf{in}(u)} f_i(x, u) \\ &= \frac{1}{d_i} \cdot \left(\sum_{y \in \mathsf{out}(u)} f_i(u, y)\right).\end{aligned}$$

Now,

$$\begin{aligned}\mathbf{Pr}\left[(u, v) \in P_i\right] &= \mathbf{Pr}\left[u \in P_i\right] \cdot \mathbf{Pr}\left[\text{dice } C(u, f_i) \text{ selects } (u, v) \mid u \in P_i\right] \\ &= \frac{1}{d_i} \cdot \left(\sum_{y \in \mathsf{out}(u)} f_i(u, y)\right) \cdot \frac{f_i(u, v)}{\sum_{y \in \mathsf{out}(u)} f_i(u, y)} \\ &= \frac{f_i(u, v)}{d_i},\end{aligned}$$

and the claim follows. ◀

By linearity of expectation, we obtain the following corollary.

▶ Corollary 1. $\mathbf{E}\left[|f_i'|\right] = |f_i|$.

D Chernoff Bounds

In this section we present material from Raghavan [14] and Young [19] about the Chernoff bounds used in the analysis of randomized rounding.

▶ Fact 1. $e^x \geq 1 + x$ and $x \geq \ln(1 + x)$ for $x > -1$.

▶ Fact 2. $(1 + \alpha)^x \leq 1 + \alpha \cdot x$, for $0 \leq x \leq 1$ and $\alpha \geq -1$.

▶ Fact 3 (Markov Inequality). For a non-negative random variable X and $\alpha > 0$, $\mathbf{Pr}\left[X \geq \alpha\right] \leq \frac{\mathbf{E}[X]}{\alpha}$.

▶ **Definition 5.** The function $\beta : (-1, \infty) \to \mathbb{R}$ is defined by $\beta(\varepsilon) \triangleq (1+\varepsilon)\ln(1+\varepsilon) - \varepsilon$.

▶ Fact 4. For ε such that $-1 < \varepsilon < 1$ we have $\beta(-\varepsilon) \geq \frac{\varepsilon^2}{2} \geq \beta(\varepsilon) \geq \frac{2\varepsilon^2}{4.2+\varepsilon}$. Hence, $\beta(-\varepsilon) = \Omega(\varepsilon^2)$ and $\beta(\varepsilon) = \Theta(\varepsilon^2)$.

▶ **Theorem 6** (Chernoff Bound). *Let $\{X_i\}_i$ denote a sequence of independent random variables attaining values in $[0,1]$. Assume that $\mathbf{E}[X_i] \leq \mu_i$. Let $X \triangleq \sum_i X_i$ and $\mu \triangleq \sum_i \mu_i$. Then, for $\varepsilon > 0$,*

$$\mathbf{Pr}[X \geq (1+\varepsilon)\cdot\mu] \leq e^{-\beta(\varepsilon)\cdot\mu}.$$

Proof. Let A denote the event that $X \geq (1+\varepsilon)\cdot\mu$. Let $f(x) \triangleq (1+\varepsilon)^x$. Let B denote the event that

$$\frac{f(X)}{f((1+\varepsilon)\cdot\mu)} \geq 1.$$

Because $f(x) > 0$ and $f(x)$ is monotonously increasing, it follows that $\mathbf{Pr}[A] = \mathbf{Pr}[B]$. By Markov's Inequality,

$$\mathbf{Pr}[B] \leq \frac{\mathbf{E}[f(X)]}{f((1+\varepsilon)\cdot\mu)}.$$

Since $X = \sum_i X_i$ is the sum of independent random variables,

$$\begin{aligned}
\mathbf{E}[f(X)] &= \prod_i \mathbf{E}\left[(1+\varepsilon)^{X_i}\right] \\
&\leq \prod_i \mathbf{E}[1+\varepsilon\cdot X_i] && (\textit{by Fact 2}) \\
&\leq \prod_i (1+\varepsilon\cdot\mu_i) \\
&\leq \prod_i e^{\varepsilon\cdot\mu_i} && (\textit{by Fact 1}) \\
&= e^{\varepsilon\cdot\mu}
\end{aligned}$$

We conclude that

$$\begin{aligned}
\mathbf{Pr}[A] &\leq \frac{e^{\varepsilon\cdot\mu}}{f((1+\varepsilon)\cdot\mu)} \\
&= e^{-\beta(\varepsilon)\cdot\mu},
\end{aligned}$$

and the theorem follows. ◀

We prove an analogue theorem for bounding the probability of the event that X is much smaller than μ.

▶ **Theorem 7** (Chernoff Bound). *Under the same premises as in Theorem 6 except that $\mathbf{E}[X_i] \geq \mu_i$, it holds that, for $1 > \varepsilon \geq 0$,*

$$\mathbf{Pr}[X \leq (1-\varepsilon)\cdot\mu] \leq e^{-\beta(-\varepsilon)\cdot\mu}.$$

Proof. We repeat the proof of Theorem 6 with the required modifications. Let A denote the event that $X \leq (1-\varepsilon)\cdot\mu$. Let $g(x) \triangleq (1-\varepsilon)^x$. Let B denote the event that

$$\frac{g(X)}{g((1-\varepsilon)\cdot\mu)} \geq 1.$$

Because $g(x) > 0$ and $g(x)$ is monotonously decreasing, it follows that $\mathbf{Pr}[A] = \mathbf{Pr}[B]$. By Markov's Inequality,

$$\mathbf{Pr}[B] \leq \frac{\mathbf{E}[g(X)]}{g((1-\varepsilon)\cdot\mu)}.$$

Since $X = \sum_i X_i$ is the sum of independent random variables,

$$\begin{aligned}
\mathbf{E}[g(X)] &= \prod_i \mathbf{E}\left[(1-\varepsilon)^{X_i}\right] \\
&\leq \prod_i \mathbf{E}[1-\varepsilon\cdot X_i] && (\textit{by Fact 2}) \\
&\leq \prod_i (1-\varepsilon\cdot\mu_i) \\
&\leq \prod_i e^{-\varepsilon\cdot\mu_i} && (\textit{by Fact 1}) \\
&= e^{-\varepsilon\cdot\mu}
\end{aligned}$$

We conclude that

$$\begin{aligned}
\mathbf{Pr}[A] &\leq \frac{e^{-\varepsilon\cdot\mu}}{g((1-\varepsilon)\cdot\mu)} \\
&= e^{-\beta(-\varepsilon)\cdot\mu},
\end{aligned}$$

and the theorem follows. ◀