Scalable Constraint-Based Virtual Data Center Allocation

Sam Bayless[✱] **Nodir Kodirov**[✱] **Ivan Beschastnikh**[✱] **Holger H. Hoos**[✱⋆] **Alan J. Hu**[✱]
[✱]University of British Columbia, Canada
[⋆]Universiteit Leiden, The Netherlands
{sbayless, knodir, bestchai, hoos, ajh}@cs.ubc.ca

Abstract

Constraint-based techniques can solve challenging problems arising from highly diverse applications. This paper considers the problem of virtual data center (VDC) allocation, an important, emerging challenge for modern data center operators. To solve this problem, we introduce NETSOLVER, which is based on the general-purpose constraint solver MONOSAT. NETSOLVER represents a major improvement over existing approaches: it is sound, complete, and scalable, providing support for end-to-end, multi-path bandwidth guarantees across all the layers of hosting infrastructure, from servers to top-of-rack switches to aggregation switches to access routers. NETSOLVER scales to realistic data center sizes and VDC topologies, typically requiring just seconds to allocate VDCs of 5–15 virtual machines to physical data centers with 1000+ servers, maintaining this efficiency even when the data center is nearly saturated. In many cases, NETSOLVER can allocate $150\% - 300\%$ as many total VDCs to the same physical data center as previous methods. Essential to our solution efficiency is our formulation of VDC allocation using monotonic theories, illustrating the practical value of the recently proposed *SAT modulo monotonic theories* approach.

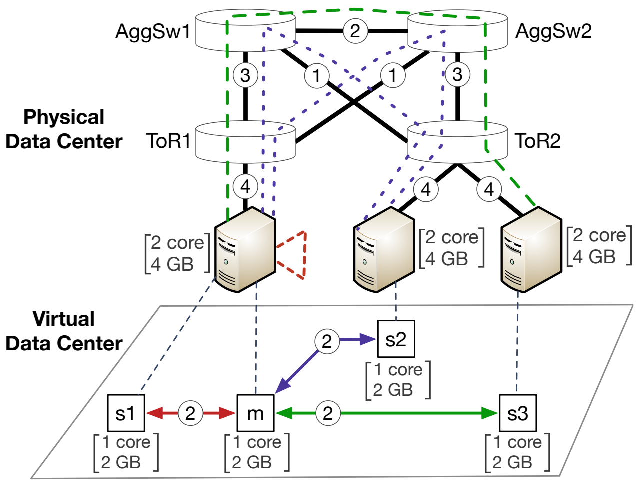


Figure 1: **(Top)** Example physical data center topology with three physical servers, two top-of-rack (ToR) switches, and two aggregation switches (AggSw). Circled numbers on links denote available bandwidth in Gbps. **(Bottom)** Example Hadoop VDC with one master (m) and three slave VMs (s1-s3) with a required throughput of 2 Gbps between each slave and the master (shown in circles). Each VM also requires a certain number of CPU cores and RAM. The problem is to find an allocation of the VDC to the physical data center, for example, as illustrated with the dashed lines. Note that s1 and m are mapped to the same physical server, while the virtual link m – s2 is allocated a multi-path route.

1 Introduction

Constraint-based techniques, such as SAT and SAT modulo theory (SMT) solvers, play a key role in state-of-the-art approaches for solving challenging problems across a wide range of applications (see, e.g., [Prasad *et al.*, 2005; Cadar *et al.*, 2008; Rintanen, 2011]). In this work, we demonstrate how virtual data center (VDC) allocation, a prominent and increasingly important problem arising in the operation of modern data centers, can be tackled using a high-performance SMT solver, MONOSAT [Bayless *et al.*, 2015]. Using a novel approach for handling multi-commodity flow constraints, we obtain substantial improvements in performance and functionality over previous VDC allocation techniques.

A VDC consists of multiple communicating virtual machines (VMs), each with individual server resource requirements (e.g., CPU or RAM), along with a virtual network of pair-wise bandwidth requirements between the VMs. The *VDC allocation problem* is to find a valid allocation of VMs to servers and links in the virtual network to links in the physical network. A valid allocation satisfies the compute, memory, and network bandwidth requirements of each VM across the entire data center infrastructure, including servers, top-of-rack (ToR) switches, and aggregation switches [Popa *et al.*, 2012; Lee *et al.*, 2014]. Figure 1 shows a simple instance of the VDC allocation problem and one solution.

The key insight to our approach is that VDC allocation can be formulated in terms of *monotonic theories*, a recently developed concept that lies at the heart of the SMT solver MONOSAT [Bayless *et al.*, 2015]. Exploiting this insight, we constructed NETSOLVER, a virtual data center allocation procedure that is scalable, sound, and complete, with support

Algorithm	Complete	Multi-path	VDC Topology	Data Center Topology
SecondNet [Guo *et al.*, 2010]			All	All
Importance Sampling [Tantawi, 2012]			All	Tree
Oktopus [Ballani *et al.*, 2011]			Star	All
VDCPlanner [Zhani *et al.*, 2013]			All	All
HVC-ACE [Rost *et al.*, 2015]		✓	Hose	All
VirtualRack [Huang *et al.*, 2014]	✓		Hose	All
Z3-AR [Yuan *et al.*, 2013]	✓		All	Tree
NETSOLVER (this work)	✓	✓	All	All

Table 1: Characteristics of sound VDC allocation algorithms from the literature and the NETSOLVER approach introduced in this work.

for end-to-end, multi-path bandwidth guarantees across all the layers of the networking infrastructure, from servers to top-of-racks to aggregation switches to access routers.

NETSOLVER harnesses MONOSAT to solve our novel formulation of the VDC allocation problem and efficiently allocates VDCs with a dozen or more VMs to full-size physical data centers (with 1000+ servers), typically in seconds per allocation. In many cases, NETSOLVER can allocate $150\%-300\%$ as many total VDCs to the same physical data center as state-of-the-art heuristic methods, such as SecondNet's VDCAlloc algorithm [Guo *et al.*, 2010], while offering the flexibility and extensibility characteristics of a constraint-based approach.

2 Related Work

We now survey prior work with respect to features of VDC allocation that are relevant to modern data centers. As can be seen from Table 1, all prior approaches have important limitations relative to the one we present here.

1. Soundness. Sound VDC allocation tools respect end-to-end bandwidth guarantees, while unsound tools only attempt to minimize data center network traffic without a guarantee that VMs will have sufficient dedicated bandwidth. Examples of unsound approaches to VDC allocation include [Meng *et al.*, 2010; Kakadia *et al.*, 2013], which dynamically identify VM communication patterns through network traffic analysis.

This prior work is in contrast to the approaches discussed in this paper, all of which, including our contribution, NETSOLVER, are sound and assume that VDCs and their communication requirements are explicitly known to the allocator.

2. Completeness. Most VDC allocation tools that respect bandwidth guarantees are *incomplete*: they can fail to find feasible VDC allocations in cases where such allocations exist (even when given unlimited runtime). Oktopus [Ballani *et al.*, 2011], VDCPlanner [Zhani *et al.*, 2013], HVC-ACE [Rost *et al.*, 2015], and SecondNet [Guo *et al.*, 2010] are examples of incomplete allocation algorithms. For example, SecondNet's algorithm is greedy in that it maps VMs to servers before checking for available paths, and allocates bandwidth one path at a time; if either of these steps fail, it will fail to allocate the VDC[1].

In contrast, the constraint-solver-based approaches described in [Yuan *et al.*, 2013] and NETSOLVER are both complete: they are guaranteed to (eventually) find a feasible allocation if one exists. We will show in our experiments that completeness does not merely represent a theoretical benefit, but can translate into substantial gains in practical allocation capability. NETSOLVER is the first sound and complete VDC allocator that can be applied to any VDC and data center topology without simplifying abstractions.

3. Multi-path allocations. Many data centers use multi-path allocations to maximize bandwidth and to provide fault-tolerance and load-balancing [Raiciu *et al.*, 2011; Alizadeh *et al.*, 2014]. Lack of multi-path support in traditional L2/L3-based networks was a primary motive for data center operators to develop networking stacks with multi-path support [Vahdat, 2015]. There are now multiple efforts underway to eliminate this restriction, which include using architectures specifically designed for multi-path routing, e.g., BCube [Guo *et al.*, 2009], VL2 [Greenberg *et al.*, 2009], and making the data center networking fabric itself multi-path [IETF, 2016].

Despite the increasing importance of multi-path routing, to the best of our knowledge, there is only one previous VDC allocator that supports multi-path communication between VMs: HVC-ACE [Rost *et al.*, 2015], a sound but incomplete allocator that uses a hose-model for VDCs. There are also several incomplete algorithms for virtual network embedding that have support for multi-path allocation for smaller physical networks with 50-150 servers [Yu *et al.*, 2008; Chowdhury *et al.*, 2009; Cheng *et al.*, 2011]. NETSOLVER is the *first* sound and complete multi-path tool for VDC allocation.

4. Unrestricted topologies. Many VDC allocators simplify the problem, either by abstracting VDC topologies into simpler ones that are easier to allocate, or by restricting the physical data center to simpler topologies. For example, the abstraction-refinement encodings from [Yuan *et al.*, 2013] only apply to tree-topology data centers. Oktopus [Ballani *et al.*, 2011] abstracts VDCs into virtual clusters, which are VMs connected to central virtual switch in a star topology. VirtualRack [Huang *et al.*, 2014] and HVC-ACE [Rost *et al.*, 2015] use a less-restricted *hose-model* [Duffield *et al.*, 1999] abstraction for VDCs. NETSOLVER is the first sound and complete VDC allocation approach that supports arbitrary VDC and data center topologies.

3 The Multi-path VDC Allocation Problem

The problem we consider in this work is defined as follows. We are given the description of a physical network (PN) and a virtual data center (VDC). The PN is specified through a set of servers S, switches N, and a directed (or undirected) graph $(S \cup N, L)$, with capacities $c(u, v)$ for each link in L. The VDC consists of a set of virtual machines *VM* and a set $R \subseteq \mathit{VM} \times \mathit{VM} \times \mathbb{Z}^+$ of directed (or undirected) bandwidth requirements between those machines. For each server $s \in S$, we have CPU core, RAM, and storage capacity specifications, $cpu(s), ram(s), storage(s)$, and for each virtual machine $v \in \mathit{VM}$, we are given CPU core, RAM, and storage requirements $cpu(v), ram(v), storage(v)$.

The objective in the multi-path VDC allocation problem is to find an assignment $A : \mathit{VM} \mapsto S$ of virtual machines $v \in \mathit{VM}$ to servers $s \in S$ along with an assignment of non-negative bandwidth $B_{u,v}(l)$ to links $l \in L$ for each bandwidth requirement $(u, v, b) \in R$, satisfying the following constraints:

[1]In fact, SecondNet will try this process several times on different sub-sets of the data center before giving up.

- **Local VM allocation constraints** ($\mathbf{L}$) ensure that each virtual machine is assigned to exactly one server, and that each server provides sufficient CPU core, RAM, and storage resources to accommodate the requirements of all VMs allocated to it: $\forall s \in S : \sum_{V(s)} cpu(v) \leq cpu(s) \wedge \sum_{V(s)} ram(v) \leq ram(s) \wedge \sum_{V(s)} storage(v) \leq storage(s)$, where $V(s) = \{v \in VM \mid A(v) = s\}$. Resource requirements are modelled using integer values, and VMs do not share resources.
- **Global bandwidth allocation constraints** ($\mathbf{G}$) ensure that sufficient bandwidth is available in the physical network to satisfy all bandwidth requirements between pairs of VMs. We formalise this by requiring that for all $(u, v, b) \in R$, the assignments $B_{u,v}(l)$ form a valid $A(u) - A(v)$ network flow no smaller than b, and that none of the link capacities l in the physical network is exceeded: $\forall l \in L : \sum_{(u,v,b) \in R} B_{u,v}(l) \leq c(l)$. Bandwidths are represented by integer values; bandwidth between VMs allocated on the same server is unlimited.

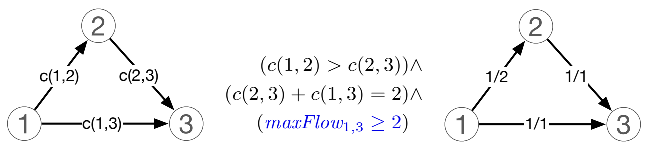
It has been previously observed [Gupta *et al.*, 2001; Szeto *et al.*, 2003; Yu *et al.*, 2008; Chowdhury *et al.*, 2009] that when allowing path-splitting, the global bandwidth allocation constraints give rise to a multi-commodity flow problem, which is strongly NP-complete even for undirected integral flows [Even *et al.*, 1975]. Conversely, any multi-commodity flow problem maps directly into bandwidth constraints, establishing the NP-hardness of the multi-path VDC allocation problem [Chowdhury *et al.*, 2009].

4 NETSOLVER

We will now show how multi-commodity integral flow problems can be encoded as a conjunction of maximum flow constraints over graphs with symbolic edge weights. By combining this encoding for the global constraints $\mathbf{G}$ with a pseudo-Boolean encoding of the local constraints $\mathbf{L}$, we are able to tackle the full multi-path VDC allocation problem using MONOSAT, a SAT modulo theory (SMT) solver that extends quantifier-free first-order Boolean logic with highly efficient, built-in support for a wide set of *finite monotonic predicates* [Bayless *et al.*, 2015].

Intuitively, a finite monotonic predicate is a predicate for which increasing the value of its arguments can never change the value of the predicate from true to false, e.g., adding links to a network can only increase the connectedness of the network. MONOSAT supports many common graph constraints, such as reachability, shortest paths, minimum spanning trees, and maximum flows. MONOSAT also supports a subset of the theory of fixed-width bitvectors.

MONOSAT accepts formulas with one or more directed *symbolic graphs*, each of which comprises a fixed set of nodes and symbolic edges (u, v). Each edge has an integer capacity, $c(u, v)$, which may be either a constant or a variable (a fixed-width bitvector). Finally, MONOSAT supports a number of common graph predicates, of which only one is relevant here: $maxFlow_{s,t,G} \geq f$, where G is a directed graph, s and t are nodes in G, and f is a constant integer or a bitvector term. This predicate is TRUE iff the maximum s-t flow in G, *under assignment to the edge capacities associated with* G, is greater or equal to f.



(a) Directed graph G **(b)** Formula to satisfy **(c)** Satisfying flow

Figure 2: **(a)** Example symbolic graph, with variable capacities $c(u, v)$ on each edge. includes the capacity assigned to each edge, as well as the flow along that edge. **(b)** A formula constraining the graph. **(c)** A solution, assigning a flow and a capacity (f/c) to each edge.

As an example, consider the directed graph G shown in Figure 2a, with variable integer capacities $c(u, v)$, and the formula in Figure 2b. In this example, MONOSAT finds edge capacities that satisfy the constraints and also produces a flow satisfying the maximum flow predicate in Figure 2c.

In the remainder of this section, we will first describe how we model integer-value multi-commodity flow in terms of the built-in maximum flow predicates supported by MONOSAT; then we will show how to use these multi-commodity flow constraints to express VDC allocation.

4.1 Multi-Commodity Flow in MONOSAT

SMT solvers have not traditionally been applied to large multi-commodity flow problems; instead, such problems are usually solved using integer-arithmetic solvers, or are approximated using linear programming. MONOSAT is different from other SMT solvers in that it has built-in predicates for (single-commodity) s-t maximum flow. While this does not directly provide support for multi-commodity flows, we will show that by expressing multi-commodity flows as a combination of single-commodity maximum flow predicates, we can use MONOSAT to solve large multi-commodity flow problems – a first for SMT solvers.

We consider this formulation of integer-value multi-commodity flows in terms of maximum flow predicates to be a key contribution of our work. While there are many obvious ways to encode multi-commodity flows in SMT solvers, the one we present here is, to the best of our knowledge, the only SMT encoding to scale to multi-commodity flow problems with thousands of nodes. As there are many applications to which SMT solvers are better suited than ILP solvers (and vice-versa), this SMT formulation has many potential applications beyond VDC allocation.

Given a directed graph $G = (V, E)$, an integer capacity $c(u, v)$ for each edge $(u, v) \in E$, and a set of commodity demands K, where a commodity demand $i \in K$ is a tuple (s_i, t_i, d_i), representing an integer flow demand of d_i between source $s_i \in V$ and target $t_i \in V$. The integral multi-commodity flow problem is to find a feasible flow such that each demand d_i is satisfied, while for each edge (u, v) the total flow of all capacities (summed) is at most $c(u,v)$:

$$f_i(u,v) \geq 0, \quad \forall (u,v) \in E, i \in K$$

$$\sum_{i \in K} f_i(u,v) \leq c(u,v), \quad \forall (u,v) \in E$$

$$\sum_{v \in V} f_i(u,v) - \sum_{v \in V} f_i(v,u) = \begin{cases} 0, \text{if } u \notin \{s_i, t_i\} \\ d, \text{if } u = s_i \\ -d, \text{if } u = t_i \end{cases} , \forall i \in K$$

We instantiate symbolic graphs $G_{1..|K|}$ with the same topology as G. We set the capacities of each edge $(u,v)_i \in G_i$ to a new integer variable, $c(u,v)_i$, with constraint $0 \leq c(u,v)_i \leq c(u,v)$. Next, we assert that the capacities in each graph sum to no more than the original edge capacity: $\sum_i c(u,v)_i \leq c(u,v)$. Together, these constraints partition the original capacity graph into K separate graphs, one for each demand. To complete the encoding, for each commodity demand (s_i, t_i, d_i), we use MONOSAT's built-in maximum flow constraints to assert that the maximum s_i–t_i flow in G_i is at least d_i.

4.2 Encoding Multi-path VDC Allocation

The global constraints $\mathbf{G}$ (Section 3) can be encoded as a multi-commodity flow as described above, with up to $|VM|^2$ commodity demands (one for each bandwidth tuple $(u, v, bandwidth) \in R$).[2] However, we can greatly improve on this by merging bandwidth constraints that share a common source into a single commodity demand: Given a set of bandwidth constraints $(u, v_i, bandwidth_i) \in R$ with the same source u, we can convert these into a single commodity demand, by adding an extra node $w \notin VM$, along with edges (v_i, w) with capacity $bandwidth_i$. The commodity demands $(u, v_i, bandwidth_i)$ can then be replaced by a single commodity demand $(u, w, \sum_i bandwidth_i)$. As there are at most $|VM|$ distinct sources in R, this reduces the number of demands from $|VM|^2$ in the worst case to $|VM|$ demands.[3]

In cases where the VDC is undirected, we improve on this further by swapping sources and sinks in communication requirements so as to maximize the number of requirements with common sources. This can be done efficiently even for large networks by finding an approximate minimum-cost vertex cover of R (*e.g.,* using the 2-opt approximation from [Bar-Yehuda and Even, 1985].

We first construct an undirected graph of communication requirements, with an undirected edge of weight $(u,v) = bandwidth$ for each bandwidth requirement and find an approximate minimum-cost vertex cover. Necessarily, each edge, and hence each communication requirement, will have at least one covering vertex. For each requirement $(u, v, bandwidth)$, if v is a covering vertex and u is not, we replace the requirement with $(v, u, bandwidth)$, swapping u and v. After swapping all uncovered source vertices in this way, we then proceed to merge requirements with common sources as above. For cases where the VDC is directed, we skip this vertex-cover optimization and only merge together connection requirements with the same (directed) source in the input description.

Given this set of commodity demands, we construct an undirected (or directed) graph G consisting of the physical network $(S \cup N, L)$, and one node for each virtual machine in *VM*. If any VDC communication requirements $(u, v_i, bandwidth_i)$ have been merged into combined requirements $(u, w, \sum bandwidth_i)$ as above, we add additional, directed edges (v_i, w) with capacity $bandwidth_i$ to G.

For each $v \in VM$ and each server $s \in S$, we add a directed symbolic edge e_{vs} from v to s with unlimited capacity to G; this edge controls the server to which each VM is allocated. Next, we assert (using a cardinality constraint) that for each VM v, exactly one edge e_{vs} is enabled, so that the VM is allocated to exactly one server: $\forall v \in VM : \sum_s e_{vs} = 1$. Using the multi-commodity flow encoding described above, we assert that the multi-commodity flow in G satisfies $(u, v, bandwidth)$ for each commodity requirement.

The above constraints together enforce global constraints $\mathbf{G}$; to enforce local constraints $\mathbf{L}$, we use pseudo-Boolean constraints (as described in [Eén and Sörensson, 2006]) to assert: $\sum_v cpu(v) \leq cpu(s) \wedge \sum_v ram(v) \leq ram(s) \wedge \sum_v storage(v) \leq storage(s)$. To allow multiple different VDC topologies to be allocated, the 'assumption' mechanism [Eén and Sörensson, 2003] found in many SAT solvers, including MONOSAT, can be used to dynamically restrict a generic graph.

Note that these encodings are novel contributions and critical to NETSOLVER's performance; however, they are empirically efficient only because MONOSAT (unlike other SMT solvers) has built-in support for graph constraints. As we show next, carefully crafted encodings alone, such as the one developed in [Yuan *et al.*, 2013], will not be competitive. Instead, fundamental improvements in the constraint solver, such as the ones we use in MONOSAT, are necessary.

5 Evaluation

We now present results from an extensive empirical evaluation demonstrating that our approach offers substantial advantages compared to state-of-the-art methods for VDC allocation. Specifically, we compare the performance of NETSOLVER to that of SecondNet's VDCAlloc [Guo *et al.*, 2010] — a seminal, sound VDC allocation algorithm with end-to-end bandwidth guarantees — and the Z3-based abstraction-refinement procedure from [Yuan *et al.*, 2013], which resembles our approach in that it makes use of an SMT solver. In each experiment, the algorithms repeatedly allocate VDCs to the data center until they are unable to make further allocations (or until a 1 CPU hour timeout is reached). This metric was also used in prior work [Guo *et al.*, 2010; Yuan *et al.*, 2013] and is important in practice, as it captures data center utilization.

[2]Note that in our approach, bandwidth values are required to be integers, as we are restricted to finding integer maximum flows. In practice, this is not a limitation, as data centers typically only offer a small number of (integer-valued) bandwidth/CPU choices to clients.

[3]Converting a single-source, multi-destination flow problem into a single-source, single-destination maximum flow problem is a well-known transformation, and safely preserves the maximum possible flow to each destination.

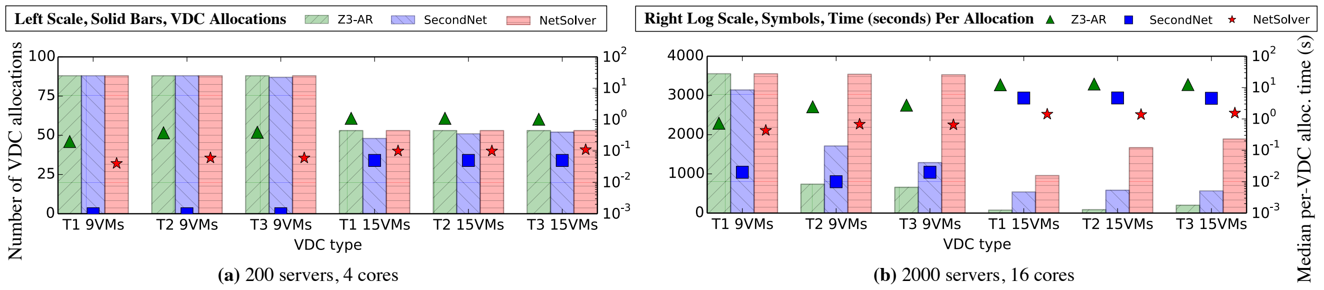


Figure 3: Total number of consecutive VDCs allocated by different algorithms and time required per allocation on various tree topologies from [Yuan *et al.*, 2013]. We report the median running time for allocating individual VDCs.

SecondNet's VDCAlloc algorithm ('SecondNet', except where ambiguous) is an incomplete, heuristic-driven algorithm that can find VDC allocations for physical networks with hundreds of thousands of servers. As SecondNet is based on bipartite matching, it fundamentally cannot allocate more than one VM in each VDC to any given server. Furthermore, because it performs allocation in an incomplete, greedy fashion, especially in heavily utilized networks, it can fail to find an existing feasible allocation. As we will demonstrate, under many realistic circumstances, this happens quite frequently, leading to substantially lower data center utilization than can be achieved with a complete method, such as NETSOLVER.

[Yuan *et al.*, 2013] introduced two approaches for performing single-path VDC allocation with bandwidth guarantees that use the general-purpose SMT solver Z3 [De Moura and Bjørner, 2008]. Unlike the MONOSAT solver at the core of NETSOLVER, Z3 has no built-in support for graph predicates. Therefore, in order to use Z3 for VDC allocation, the global bandwidth and connectivity constraints have to be expressed using a lower-level logical formulation. The first such encoding presented by [Yuan *et al.*, 2013] (which we call Z3-generic) can handle any data center topology but scales extremely poorly [Yuan *et al.*, 2013]. The second approach (which we call Z3-AR) makes use of an optimized abstraction-refinement technique; while substantially more scalable than the generic encoding, it is restricted to data centers with tree topologies. In preliminary experiments (not reported here), we confirmed that Z3-generic performed poorly, often failing to find any allocations within a 1-hour timeout on the benchmarks used in our experiments.

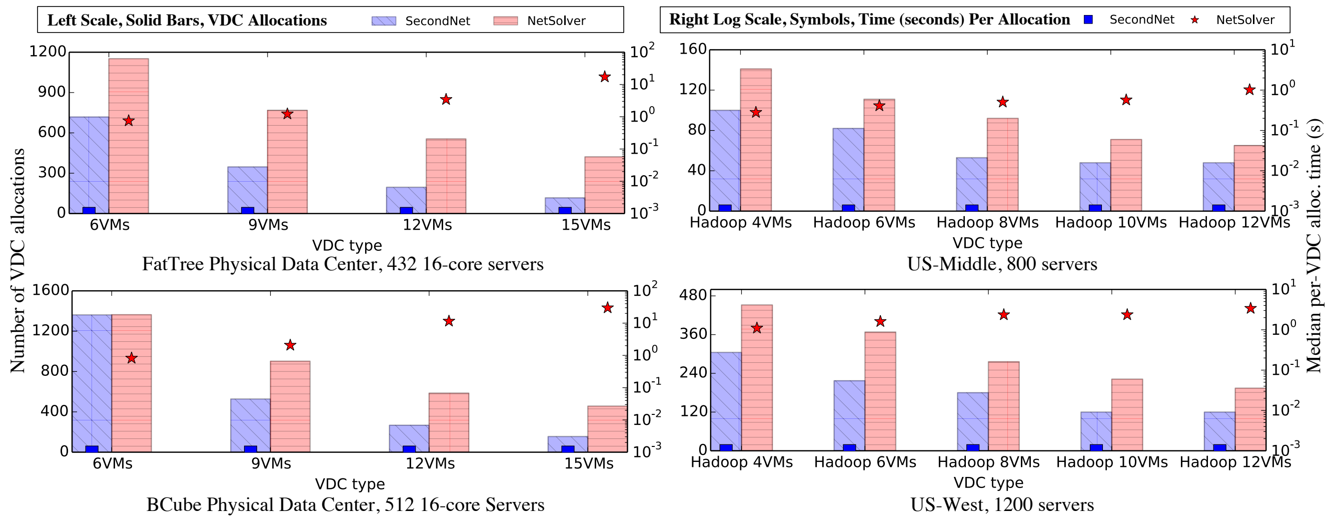
Comparison on Trees from [Yuan *et al.*, 2013] Our first experiment reproduces and extends an experiment from [Yuan *et al.*, 2013], in which a series of identical VDCs is allocated one-by-one to tree-structured data centers, until the solver is unable to make further allocations (or a timeout of 1 CPU hour is reached). We obtained the original implementation of Z3-AR from the authors for this experiment, along with a version of SecondNet they implemented with support for the tree-structured data centers considered here. In this experiment, the VDCs always have identical structure; this is a limitation introduced here for compatibility with the solvers from [Yuan *et al.*, 2013]. In our subsequent experiments, below, we lift this constraint. Except where noted, all experiments were run on a machine with a 2.66GHz (12MB L3 cache) Intel x5650 processor, running Ubuntu 12.04 and limited to 16GB RAM.

Figure 3 summarizes our results, showing the total number of consecutive VDCs allocated within 1 CPU hour. On the left, we used the 200-server/4-cores-per-server physical data center from [Yuan *et al.*, 2013]. On the right, we considered a larger data center with 2000 16-core servers. We note that, although all three solvers perform similarly on the small tree-structured data centers (with SecondNet, being heuristic and incomplete, faster than NETSOLVER, and NETSOLVER faster than Z3-AR), on the larger data center, NETSOLVER greatly outperforms SecondNet and Z3-AR, often allocating two or even three times as many VDCs on the same infrastructure. Importantly, we see that NETSOLVER scales to thousands of servers, with median per-instance allocation times of a few seconds or less per VDC.

Comparison on BCube and FatTree from [Guo *et al.*, 2010] The second experiment we conducted is a direct comparison against the original SecondNet implementation (which we also used for all comparisons reported later). Note that the implementation of Z3-generic, and both the theory and implementation of Z3-AR, are restricted to tree topologies, so they could not be included in these experiments.

The SecondNet benchmark instances are extremely large — in one case exceeding 100 000 servers — but also extremely easy to allocate: the available bandwidth per link is typically $\geq 50\times$ the requested communication bandwidths in the VDC, so with only 16 cores per server, the bandwidth constraints are mostly irrelevant. For such easy allocations, the fast, incomplete approach that SecondNet uses is the better solution. Accordingly, we scaled the SecondNet instances down to 432–1024 servers, a realistic size for many real-world data centers. For these experiments, we generated sets of 10 VDCs each of several sizes (6, 9, 12 and 15 VMs), following the methodology described in [Yuan *et al.*, 2013]. These VDCs have proportionally greater bandwidth requirements than those originally considered by SecondNet, requiring 5–10% of the smallest link-level capacities. The resulting VDC instances are large enough to be representative of many real-world use cases, while also exhibiting non-trivial bandwidth constraints. For each of these sets of VDCs, we then repeatedly allocated instances (in random order) until the data center is saturated.

Figure 4a shows allocations made by SecondNet and NET-



(a) Allocations on sample real topologies

(b) Allocations on real commercial topologies

Figure 4: Total number of consecutive VDCs allocated by different algorithms and time required per allocation on two real sets of benchmarks: (a) FatTree and BCube topologies from [Guo *et al.*, 2010], and (b) commercial Data Center topologies. We report median running times for allocating individual VDCs.

SOLVER on two data centers, one with a FatTree topology with 432 servers (top), and one with a BCube topology with 512 servers (bottom). As in our previous experiment, SecondNet was much faster than NETSOLVER, but NETSOLVER ran fast enough to be practical for data centers with hundreds of servers, with typical allocation times of a few seconds per VDC (however, in a minority of cases, NETSOLVER did require tens or even hundreds of seconds for individual allocations). In many cases, NETSOLVER was able to allocate more than twice as many VDCs as SecondNet on these data centers — a substantial improvement in data center utilization.

Comparison on commercial networks The above comparisons consider how NETSOLVER compares to existing VDC allocation tools on several artificial (but representative) network topologies from the VDC literature. To address whether there are real-world VDC applications where NETSOLVER performs not only better than existing tools, but is also fast enough to be used in practice, we also considered a deployment of a standard Hadoop virtual cluster, on a set of actual data center topologies. We collaborated with the private cloud provider ZeroStack Inc. to devise an optimal virtual Hadoop cluster to run Terasort[4]. Each Hadoop virtual network consists of a single master VM connected to 3–11 slave VMs.[5] We considered 5 different VM sizes, ranging from 1 CPU and 1GB RAM, to 8 CPUs and 16GB of RAM; for our experiments, the slave VMs were selected at random from this set, with the master VM also randomized but always at least as large as the largest slave VM. The Hadoop master has tree connectivity with all slaves, with either 1 or 2 Gbps links connecting the master to each slave (like the VDC in Figure 1).

The physical data center topology was provided by another company, which requested to remain anonymous. This company uses a private cloud deployed across four data centers in two geographic availability zones (AZs): *us-west* and *us-middle*. Each data center contains between 280 and 1200 servers, spread across 1 to 4 clusters with 14 and 40 racks. Each server has 16 cores, 32 GB RAM, 20 Gbps network bandwidth (via two 10 Gbps links). The network in each data center has a leaf-spine topology, where all ToR switches connect to two distinct *aggregation switch*es over 40 Gbps links each (a total of 2 links with 80 Gbps; one on each aggregation switch) and aggregation switches are interconnected with four 40 Gbps links each. For each cluster, there is a *gateway switch* with a 240 Gbps link connected to each aggregation switch. All data centers use equal-cost multi-path (ECMP) to take advantage of multiple paths.

A VDC is allocated inside one AZ: VMs in one VDC can be split across two clusters in an AZ, but not across two AZs. Figure 4b shows VDC allocation results per AZ.[6] More generally, executing NETSOLVER on distinct physical network units, such as an AZ, improves its scalability. This also works well in practice, as modern data centers are modular by design. For example, one of the largest data center operators in the world, Facebook, designed its Altoona data center with over 100 000 servers using *pods*, with each pod containing fewer than 1000 servers [Andreyev, 2014].

We applied SecondNet and NETSOLVER in this setting, consecutively allocating Hadoop master-slave VDCs of several sizes, ranging from 4 to 12 VMs, until no further allocations

[4]The Sort Benchmark Committee: http://sortbenchmark.org/

[5]Many industrial VDCs have fewer than 15 VMs; e.g., [Bodík *et al.*, 2012] states that 80% of Bing services use fewer than 10 VMs. NETSOLVER performs well with up to 30 VMs.

[6]NETSOLVER is not limited to allocating to a single AZ and can support multi-AZ allocation, assuming it is aware of the capacity of each AZ, including inter-AZ network bandwidth.

could be made. Note that, in addition to using a realistic physical topology, the CPU/memory, bandwidth values, and the VDCs being allocated are all real-world VDCs derived from real Hadoop jobs. By contrast, previous experiments used artificial VDCs from the Z3-AR paper [Yuan *et al.*, 2013]. Again, we could not run Z3-AR in this setting, as it is restricted to tree-topology data centers.

In Figure 4b, we show the results for the largest of these data centers (results for the smaller data centers were similar). As observed in our previous experiments, although SecondNet was much faster than NETSOLVER, NETSOLVER's per-instance allocation time was typically just a few seconds, which is reasonable for long-running applications, such as the Hadoop jobs considered here. Again, NETSOLVER was able to allocate many more VDCs than SecondNet (here, 1.5–2 times as many), across a range of data center and VDC sizes, including a commercial data center with more than 1000 servers. Moreover, with increasing virtual network size, NETSOLVER was able to allocate many more virtual machines, while respecting end-to-end bandwidth constraints. Often NETSOLVER allocated several times as many VDCs as SecondNet, and in extreme cases, it found hundreds of allocations, while SecondNet was unable to make any allocations (not shown for brevity). Similarly, keeping the virtual network the same size, but doubling the bandwidth requirements of each virtual machine greatly decreased the number of allocations made by SecondNet, while NETSOLVER showed considerably more robust performance in these more congested settings.

Overall, NETSOLVER was not only able to find many more allocations than SecondNet in this realistic setting, but NETSOLVER's median allocation time, 1–30 CPU seconds, shows that it can be practically useful in a real, commercial setting, for data centers and VDCs of this size. This provides strong evidence that NETSOLVER can find practical use in realistic settings where large or bandwidth-hungry VDCs need to be allocated. It also demonstrates the practical advantage of a (fast) complete algorithm like NETSOLVER over a much faster but incomplete algorithm like SecondNet: for bandwidth-heavy VDCs, even with arbitrary running time, SecondNet's VDCAlloc was unable to find the majority of the feasible allocations.

This generally reinforces our observations from our earlier experiments with artificial topologies: NETSOLVER improves greatly on state-of-the-art VDC allocation, for bandwidth-constrained data centers with as many as 1000 servers.

Allocation Robustness In the above experiments, we showed that across many conditions, NETSOLVER was able to make many (often hundreds) more allocations than SecondNet or Z3-AR. One may wonder whether these additional allocations are the result of NETSOLVER having a better ability to solve challenging allocations quickly, or if NETSOLVER is somehow making good allocation decisions early on that leave more space for later VDC instances.

In the experiments where Z3-AR makes many fewer allocations (Figure 3b), Z3-AR's problem is excessively slow run times, allocating only a handful of VDCs in data centers with room for hundreds or thousands. In those cases, both NETSOLVER and SecondNet can both make hundreds of further allocations starting from where Z3-AR was cut off.

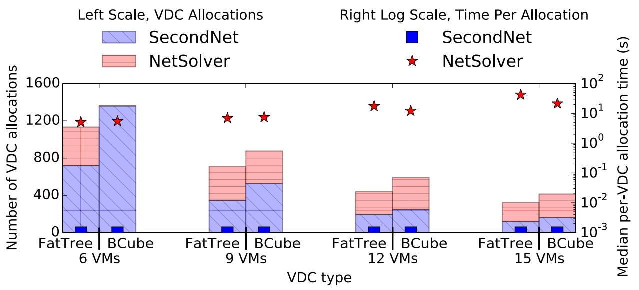


Figure 5: Additional VDC allocations made by NETSOLVER (red), after SecondNet (blue) has allocated its maximum number of VDCs. These experiments used the same VDCs and physical topologies as in Figure 4a. In many cases, NETSOLVER allocated hundreds of additional VDCs after SecondNet could not make further allocations.

The robustness question is more apropos versus SecondNet. We found conclusive evidence that good early allocations cannot be entirely responsible for NETSOLVER's performance, by observing that NETSOLVER can continue to allocate VDCs in cases where SecondNet can no longer make any further allocations. We repeated the experiments from Figure 4a by first using SecondNet to allocate as many VDCs as it can into the data center. Then, starting from that already partially utilized data center, we used NETSOLVER to allocate further VDCs. The results of this experiment are shown in Figure 5. Similarly to the earlier experiment, NETSOLVER can still allocate hundreds of additional VDCs starting from SecondNet's final allocation.

6 Conclusions and Future Work

We introduced a new, SMT-based VDC allocation method, NETSOLVER, for multi-path VDC allocation with end-to-end bandwidth guarantees. NETSOLVER scales well to data centers with 1000 or more servers, while substantially improving data center utilization as compared to current methods. Notably, we have demonstrated that in several realistic settings, NETSOLVER allocates *3 times* as many virtual data centers as previous approaches, with a runtime that is fast enough for practical use.

NETSOLVER overcomes major limitations of current state-of-the-art approaches for VDC allocation with hard bandwidth guarantees: Unlike SecondNet, our approach is complete and, as a result, is able to continue making allocations in bandwidth-constrained networks; unlike the abstraction-refinement techniques from [Yuan *et al.*, 2013], NETSOLVER supports arbitrary data center topologies (as well as being much faster). Our constraint-based approach represents the first complete VDC allocation algorithm supporting multi-path bandwidth allocation for arbitrary network topologies – an important capability in modern data centers.

Because NETSOLVER is built on-top of MONOSAT– a general purpose SMT-solver with high-performance support for graph constraints – it is easily extensible. For example, it can handle additional VDC allocation constraints not supported by other approaches, yet relevant in real-world scenarios, such as maximization of data locality (VM affinity), minimization of the total number of utilized servers and load balancing (hotspot avoidance). While these constraints make VDC allocation substantially more challenging, preliminary results indicate that our approach can still efficiently allocate VDCs with up to 10 VMs on real-world data centers (typical running times per allocation are less than 1 CPU sec).

In future work, we will explore extensions to NETSOLVER with these and other constraints. We will also apply NETSOLVER to the related problems of virtual network embedding (VNE) [Belbekkouche *et al.*, 2012; Fischer *et al.*, 2013] and network function virtualization (NFV) [Palkar *et al.*, 2015; Heorhiadi *et al.*, 2016].

More broadly, we believe that the combination of network flow constraints with state-of-the-art SMT solvers underlying NETSOLVER is a promising approach for a variety of problem types. This combination may enable fast and flexible algorithms that can benefit a broad range of applications.

Acknowledgements

We thank the authors of [Guo *et al.*, 2010] and [Yuan *et al.*, 2013] for making their implementations available to us, and especially Yifei Yuan for his extensive assistance. This research was funded by NSERC discovery grants to Ivan Beschastnikh, Holger H. Hoos and Alan J. Hu. Nodir Kodirov is supported by a UBC four-year doctoral fellowship (4YF).

References

[Alizadeh *et al.*, 2014] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. Conga: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the ACM Conference on SIGCOMM*, 2014.

[Andreyev, 2014] Alexey Andreyev. Introducing data center fabric, the next-generation facebook data center network, 2014. [Online at code.facebook.com/posts/360346274145943; accessed 05-23-2017].

[Ballani *et al.*, 2011] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. *ACM SIGCOMM Computer Communication Review*, 2011.

[Bar-Yehuda and Even, 1985] Reuven Bar-Yehuda and Shimon Even. A local-ratio theorem for approximating the weighted vertex cover problem. *North-Holland Mathematics Studies*, 1985.

[Bayless *et al.*, 2015] Sam Bayless, Noah Bayless, Holger Hoos, and Alan Hu. SAT Modulo Monotonic Theories. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Belbekkouche *et al.*, 2012] Abdeltouab Belbekkouche, Md. Mahmud Hasan, and Ahmed Karmouch. Resource discovery and allocation in network virtualization. *IEEE Communications Surveys & Tutorials*, 2012.

[Bodík *et al.*, 2012] Peter Bodík, Ishai Menache, Mosharaf Chowdhury, Pradeepkumar Mani, David A. Maltz, and Ion Stoica. Surviving failures in bandwidth-constrained datacenters. In *Proceedings of the ACM Conference on SIGCOMM*, 2012.

[Cadar *et al.*, 2008] Cristian Cadar, Daniel Dunbar, and Dawson Engler. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, 2008.

[Cheng *et al.*, 2011] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, 2011.

[Chowdhury *et al.*, 2009] N. M. Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual network embedding with coordinated node and link mapping. In *IEEE International Conference on Computer Communications*, 2009.

[De Moura and Bjørner, 2008] Leonardo De Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2008.

[Duffield *et al.*, 1999] N. G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merive. A flexible model for resource management in virtual private networks. In *Proceedings of the ACM Conference on SIGCOMM*, 1999.

[Eén and Sörensson, 2003] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing*, 2003.

[Eén and Sörensson, 2006] Niklas Eén and Niklas Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2006.

[Even *et al.*, 1975] Shimon Even, Alon Itai, and Adi Shamir. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science*, 1975.

[Fischer *et al.*, 2013] Anath Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 2013.

[Greenberg *et al.*, 2009] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. VL2: A Scalable and Flexible Data Center Network. In *Proceedings of the ACM Conference on SIGCOMM*, 2009.

[Guo *et al.*, 2009] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. BCube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 2009.

[Guo *et al.*, 2010] Chuanxiong Guo, Guohan Lu, Helen J. Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *The International Conference on emerging Networking Experiments and Technologies*, 2010.

[Gupta *et al.*, 2001] Anupam Gupta, Jon Kleinberg, Amit Kumar, Rajeev Rastogi, and Bulent Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, 2001.

[Heorhiadi *et al.*, 2016] Victor Heorhiadi, Michael K. Reiter, and Vyas Sekar. Simplifying software-defined network optimization using sol. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, 2016.

[Huang *et al.*, 2014] Tianlin Huang, Chao Rong, Yazhe Tang, Chengchen Hu, Jinming Li, and Peng Zhang. Virtual-Rack: Bandwidth-aware virtual network allocation for multi-tenant datacenters. In *IEEE International Conference on Communications*, 2014.

[IETF, 2016] IETF. TRILL: Transparent interconnection of lots of links, 2016. [Online at en.wikipedia.org/wiki /TRILL; accessed 05-23-2017].

[Kakadia *et al.*, 2013] Dharmesh Kakadia, Nandish Kopri, and Vasudeva Varma. Network-aware virtual machine consolidation for large data centers. In *Proceedings of the Third International Workshop on Network-Aware Data Management*, 2013.

[Lee *et al.*, 2014] Jeongkeun Lee, Yoshio Turner, Myungjin Lee, Lucian Popa, Sujata Banerjee, Joon-Myung Kang, and Puneet Sharma. Application-driven bandwidth guarantees in datacenters. In *Proceedings of the ACM Conference on SIGCOMM*, 2014.

[Meng *et al.*, 2010] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *IEEE International Conference on Computer Communications*, 2010.

[Palkar *et al.*, 2015] Shoumik Palkar, Chang Lan, Sangjin Han, Keon Jang, Aurojit Panda, Sylvia Ratnasamy, Luigi Rizzo, and Scott Shenker. E2: A framework for NFV applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[Popa *et al.*, 2012] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: Sharing the network in cloud computing. In *Proceedings of the ACM Conference on SIGCOMM*, 2012.

[Prasad *et al.*, 2005] Mukul R Prasad, Armin Biere, and Aarti Gupta. A survey of recent advances in SAT-based formal verification. *International Journal on Software Tools for Technology Transfer*, 2005.

[Raiciu *et al.*, 2011] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving datacenter performance and robustness with multipath TCP. In *Proceedings of the ACM Conference on SIGCOMM*, 2011.

[Rintanen, 2011] Jussi Rintanen. Madagascar: Efficient planning with SAT. *International Planning Competition*, 2011.

[Rost *et al.*, 2015] Matthias Rost, Carlo Fuerst, and Stefan Schmid. Beyond the stars: Revisiting virtual cluster embeddings. *ACM SIGCOMM Computer Communication Review*, 2015.

[Szeto *et al.*, 2003] Wayne Szeto, Youssef Iraqi, and Raouf Boutaba. A multi-commodity flow based approach to virtual network resource allocation. In *IEEE Global Telecommunications Conference*, 2003.

[Tantawi, 2012] Asser N Tantawi. A scalable algorithm for placement of virtual clusters in large data centers. In *20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2012.

[Vahdat, 2015] Amin Vahdat. A look inside Google's data center network, 2015. [Online at youtube.com/watch?v= FaAZAII2x0w; accessed 05-23-2017].

[Yu *et al.*, 2008] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 2008.

[Yuan *et al.*, 2013] Yifei Yuan, Anduo Wang, Rajeev Alur, and Boon Thau Loo. On the feasibility of automation for bandwidth allocation problems in data centers. In *Formal Methods in Computer-Aided Design*, 2013.

[Zhani *et al.*, 2013] Mohamed Faten Zhani, Qi Zhang, Gael Simon, and Raouf Boutaba. VDC Planner: Dynamic migration-aware virtual data center embedding for clouds. In *IFIP/IEEE International Symposium on Integrated Network Management*, 2013.