Virtual Network Embedding on Interconnection Networks

Rob Ballantyne, Soroush Haeri, and Ljiljana Trajković

***Abstract*—In this paper, we consider performance of virtual network embedding algorithms and their applicability to interconnection topologies used for inter-processor communication networks. Virtual Network Embedding (VNE) is a process of assigning virtual network requests to a substrate network (physical infrastructure) in data centers. VNE algorithms are applied to interconnection networks employing Butterfly and Hypercube topologies that are distinct from network topologies used in data center networks. Simulation results indicate that these topologies lead to higher acceptance ratios than topologies used in data center networks. The Butterfly topology leads to the highest acceptance ratio while having lower revenue to cost ratio. In contrast, the Hypercube topology offers similar acceptance ratio to Butterfly topology while having a higher revenue to cost ratio.**

I. Introduction

Computer virtualization permits a server to be shared by providing virtual machines. It has several advantages: efficient sharing of physical computing infrastructure and enabling administrative control of virtual machines while maintaining control of the physical infrastructure. Network virtualization has similar advantages as computer virtualization. Infrastructure providers operate large-scale cloud computing infrastructure (Google Cloud, Amazon Web Services, Microsoft Azure) and offer virtualized servers and networks. Infrastructure providers allocate resources based on virtual network requests (VNRs) and provide consumer access to virtual machines and virtual networks. The Virtual Network Embedding (VNE) process deals with assigning virtual networks to the physical infrastructure or substrate network.

Interconnection networks are used for high-bandwidth, low-latency connections within or among closely coupled computing systems. They are typically implemented between the CPUs of a multi-core integrated circuit (chips), multiple chips of a multi-chip motherboard (printed circuit boards), or systems constructed of such chips and motherboards. Cloud providers employ multi-chip multi-core systems that rely on interconnect networks and may also utilize various interconnect instantiation technologies. The communication requirements of interconnection networks call for specific network topologies that differ from other data communications networks employed in data centers.

Performance of VNE algorithms intended for various data center network topologies have been reported [1], [2]. We evaluate the performance of VNE algorithms on interconnection networks to determine topologies that are better suited for substrate networks and to identify VNE algorithms that offer the best performance.

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant R31-611284 and enabled by British Columbia Digital Research Infrastructure Group, Compute Ontario, Calcul Quebec, and the Digital Research Alliance of Canada.

Rob Ballantyne, Soroush Haeri, and Ljiljana Trajković are with Simon Fraser University, Vancouver, British Columbia, Canada (email: ballanty@sfu.ca, soroosh.haeri@gmail.com, ljilja@sfu.ca)

The remainder of this manuscript is organized as follows: The VNE problem and its objective function are addressed in Section I. The simulated DCN and interconnection network topologies are described in Section II. The experimental results generated by the VNE-Sim simulation tool, various simulation scenarios, and performance results are reported in Section III. We conclude with Section IV.

A. Virtual Network Embedding

VNE is a mapping of a virtual network consisting of virtual nodes and links onto a substrate physical shared network defined by physical nodes and links. Network virtualization enables coexistence of multiple virtual networks on a shared infrastructure. The goal is to achieve an computational efficient way resource-optimal mappings while considering constraints such as topology, CPU capacity, and link bandwidth. Software-Defined Networking (SDN) centralizes network control and provides a programmable, flexible, and dynamic way to manage network resources. It is used to design, manage, and operate computer networks by decoupling the control plane (decision-making) from the data plane (responsible for forwarding traffic). It is employed by cloud service providers to offer virtualized network services that embed virtual network requests in data centers.

SDN is often implemented together with Network Function Virtualization (NFV) to achieve flexible and agile network infrastructures. NFV is a network architecture approach that virtualizes network functions enabling flexibility, scalability, and cost savings for service providers. Virtual Network Function (VNFs) are software-based implementations of network functions such as firewalls or routers that run on virtualized infrastructure instead of dedicated hardware. SDN focuses on the centralized control and management of network resources while NFV virtualizes network functions allowing them to be deployed and managed as software rather than hardware.

A substrate network consisting of j substrate nodes (vertices) and k substrate links (edges) is represented as a graph $G^s(N^s, E^s)$, where $N^s = \{n_1^s, n_2^s, \dots, n_j^s\}$ and $E^s = \{e_1^s, e_2^s, \dots, e_k^s\}$. The i^{th} VNR is a triplet $\Psi_i(G^{\Psi_i}, \omega^{\Psi_i}, \xi^{\Psi_i})$, where $G^{\Psi_i}(N^{\Psi_i}, E^{\Psi_i})$ is the virtual network (VN) graph with l virtual nodes $N^{\Psi_i} = \{n_1^{\Psi_i}, n_2^{\Psi_i}, \dots, n_\ell^{\Psi_i}\}$ and m virtual edges $E^{\Psi_i} = \{e_1^{\Psi_i}, e_2^{\Psi_i}, \dots, e_m^{\Psi_i}\}$, ω^{Ψ_i} is the VNR arrival time, and ξ^{Ψ_i} is the VNR lifetime [3].

Performance of VNE algorithms is evaluated based on: acceptance ratio, generated revenue, incurred cost, and substrate resource (node and link) utilization [3]. The goal of VNE algorithms is to increase revenue by minimizing the embedding cost while increasing node and link utilization.

One performance metric is acceptance ratio:

$$\mathbf{p}_a^\tau = \frac{|\Psi^a(\tau)|}{|\Psi(\tau)|}, \tag{1}$$

where $\Psi^a(\tau)$ is the number of accepted VNRs and $\Psi(\tau)$ is the number of VNRs that arrive over a time interval τ.

The revenue generated from embedding a VNR graph G^{Ψ_i} is:

$$\mathbf{R}(G^{\Psi_i}) = w_c \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + w_b \sum_{e^{\Psi_i} \in E^{\Psi_i}} \mathcal{B}(e^{\Psi_i}), \quad (2)$$

where w_c and w_b are the weights for the CPU capacity $\mathcal{C}(n^{\Psi_i})$ and bandwidth $\mathcal{B}(e^{\Psi_i})$ requirements, respectively.

The cost of embedding a VNR graph G^{Ψ_i} is:

$$\mathbf{C}(G^{\Psi_i}) = \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + \sum_{e^{\Psi_i} \in E^{\Psi_i}} \sum_{e^s \in E^s} f_{e^s}^{e^{\Psi_i}}, \quad (3)$$

where $f_{e^s}^{e^{\Psi_i}}$ is the bandwidth of substrate link e^s allocated to the virtual link e^{Ψ_i}.

The substrate node and link utilization depends on the available cost and bandwidth resources:

$$\mathbf{U}(N^s) = 1 - \frac{\sum\limits_{n^s \in N^s} \mathcal{C}(n^s)}{\sum\limits_{n^s \in N^s} \mathcal{C}_{max}(n^s)} \quad (4)$$

$$\mathbf{U}(E^s) = 1 - \frac{\sum\limits_{e^s \in E^s} \mathcal{B}(e^s)}{\sum\limits_{e^s \in E^s} \mathcal{B}_{max}(e^s)}, \quad (5)$$

where $\mathcal{C}(n^s)$ and $\mathcal{C}_{max}(n^s)$ are the already assigned and maximum available substrate node CPUs capacities while $\mathcal{B}(n^s)$ and $\mathcal{B}_{max}(n^s)$ are the already assigned and maximum available substrate link bandwidths. Revenue increases with higher acceptance ratios and improves by increasing node and link utilization.

B. *Virtual Network Embedding Algorithms*

Comprehensive surveys of the VNE field included a prior [4] and a more recent contribution [5].

The VNE problem was described [4] and VNE algorithms categorized as: static vs. dynamic, centralized vs. distributed, and concise vs. redundant. Optimization metrics for several proposed algorithms to the VNE problem were discussed. A taxonomy of current approaches to solving the VNE problem was based on a novel classification scheme for the considered VNE algorithms. The VNE problem involved solving two sub-problems: Virtual Node Mapping (VNoM) and Virtual Link Mapping (VLiM). VNoM assigns virtual nodes to substrate nodes while VLiM assigns virtual links to paths in the substrate network that connect the corresponding substrate nodes.

A recent survey [5] reviews the state-of the-art VNE strategies: hardware and software support for VNE; a primer of VNE modeling; evaluation frameworks; classification, review, and comparative study; machine learning-based embedding solutions; and future directions, challenges, and open opportunities for research. This survey updates a taxonomy for VNE solutions that recognizes the different optimization objectives of service-providers and users by classifying solutions as service-provider-centric or user-centric. Service-provider-centric metrics include: embedding cost, revenue, revenue-to-cost ratio, and acceptance ratio while user-centric metrics include: delay, throughput, and path length. The survey reviews the existing simulation framework including various publicly available software tools.

The VNE problem [6] was solved using mixed-integer programming. The ViNEYard (R-ViNE and D-ViNE) algorithms coordinate between the VNoM and VLiM embedding phases to minimize the cost of embedding Virtual Network Requests (VNRs). The ViNEYard algorithms include Deterministic VNE (D-ViNE) and Randomized VNE (R-ViNE). Both achieve better correlation between the node mapping and the link mapping phases. Virtual nodes are mapped onto substrate nodes followed by mapping of virtual links to physical paths in the subsequent phase. A greedy load-balance approach was based on the Global Resource Capacity (GRC) metric to embed sequentially each virtual node and then used the shortest path routing to embed each virtual link [7].

The computational complexity of the VNE problem has been analyzed considering restrictions including: node and link bandwidth, node mapping, edge routing, and latency. The complexity of the decision version of the problem has been formally proved $\mathcal{NP}$-complete [8]. Provided that the VNoM assignment sub-problem is solved, admitting flow path splitting enables the virtual link (VLiM) assignment sub-problem to be solved in polynomial time [9].

Power consumption in data center networks (DCNs) was also considered [10], [11] and compared based on various network architectures. It was observed that the size of a data center network impacts the power consumption more than its topology (latency and bisection bandwidth) and that types of interconnections of hosts and switches do not impact their consumption. Furthermore, the power consumption in larger DCNs becomes independent of their layout.

Using the VNE-Sim platform, performance of various VNE algorithms on substrate networks of switch-centric and server-centric data center topologies has been considered [1]–[3], [12]–[15].

The VNE problem was modeled as a Markov Decision Process (MDP) and two VNE algorithms (MaVEn-M and MaVEn-S) that utilize the Monte Carlo Tree Search were proposed [3]. Various VNE algorithms (GRC, GRC-M, D-ViNE, R-ViNE, MaVEn-M, and MaVEn-S) were evaluated and compared using a developed discrete event simulator VNE-Sim. The proposed algorithms exhibited promising performance and were able to search for more profitable embeddings compared to the available algorithms. Global Resource Capacity (GRC), D-ViNE, and R-ViNE VNE algorithms were compared using BCube$(2,4)$ and Fat-Tree$_6$ data center network topologies as substrate networks [12]. Fat-Tree networks achieved higher substrate node and link utilization using the proposed algorithms (GRC-M, MaVeN-M, and MaVeN-S).

The GRC-M [13] algorithm was developed by replacing the shortest path algorithm in GRC with a Multicommodity Flow algorithm to solve the VLiM sub-problem. GRC-M achieves revenue to cost ratios comparable to the D-ViNE and R-ViNE algorithms, it enables embedding additional virtual network requests onto a substrate network.

Data center network topologies are categorized as switch-centric and server-centric topologies. VNE algorithms were simulated and compared using a variety of switch-centric and server-centric DCN topologies [16] to characterize the efficacy of VNE in a typical data center network.

Switch-centric networks: Performance of VNE algorithms using switch-centric DCN topologies such as Two-Tier, F^{2}Tree (an enhancement of Fat-Tree), and Diamond was compared and evaluated [15]. While simulation results illustrate that both GRC-M and R-ViNE algorithms performed well, GRC-M required less processing time. F^{2}Tree topologies were able to accept additional virtual network requests while Diamond topologies achieved higher node utilization. Spine-Leaf, Three-Tier, and Collapsed Core switch-centric DCNs were compared [1] using MaVEn-M and MaVEn-S VNE algorithms. They, in most cases, offered superior performance with respect to the acceptance ratio, revenue to cost ratio, resource (node and link) utilization, and average processing time. Simulation results illustrated that selecting an appropriate switch-centric topology is a trade-off between network implementation cost and performance of a VNE algorithm.

Server-centric networks: The MaVEn-M and MaVEn-S algorithms were also applied to DCell and BCube server-centric topologies [2]. The algorithms outperformed most other algorithms with respect to acceptance ratio, revenue to cost ratio, and resource (node and link) utilization. DCell topology having fewer network elements offered similar acceptance ratios with higher node and link utilization resulting in higher revenue to cost ratio.

II. Network Topologies

We compare DCN (Two-Tier and BCube [17]) with interconnection (Butterfly and Hypercube) network topologies. The interconnection networks differ from DCNs by their topology, routing, and flow control. Networks properties such as bisection bandwidth and graph diameter affect performance of VNE algorithms. Individual link bandwidths are assumed to be identical. Bisection bandwidth is the maximum bandwidth of between two halves of a network topology. The diameter of a network topology is the longest shorted path between any pair of hosts and is indicative of the worst case communications latency in a topology. Hypercube networks have 2^k hosts, for some integer k. All simulated networks have the same number of hosts while the number of switches depends on the specific network topology.

A Two-Tier Spine-Leaf (Leaf-Spine) [18] switch-centric DCN topology consists of a layer of n_c core network switches and a layer of n_e edge switches. Hosts are connected only to edge switches while core switches interconnect the edge switches forming a complete bipartite graph between the core and edge switches. Therefore, there are redundant links between core and edge switches that offer larger bandwidth and redundancy. Two-Tier networks are commonly employed as DCNs. If n_s is the number of hosts, there are: $n_e + n_c$ switches and $n_e n_c + n_s$ links in a Two-Tier topology. The bisection bandwidth of the Two-Tier topology is $(n_c n_e)/2$ and the diameter of a Two-Tier network is 4. An example of a Two-Tier topology with 4 core switches, 8 edge switches, 24 hosts, and 40 links is shown in Fig. 1.

The BCube network topology [19] is a server-centric DCN designed to facilitate the construction of a modular data center,

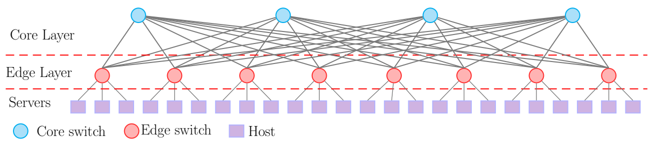

Fig. 1. Switch-centric DCN: Two-Tier topology with only core and edge layers. It consists of 12 switches, 24 hosts, and 56 links [17].

where modules of hosts and switches are interconnected as units and where, over time, additional modules may be easily added while maintaining the network topology. A BCube(n,k) network is a recursive structure that has k levels, level zero being a "cube" of n hosts and a single switch. Each higher level is an array of n lower level structures where n higher level switches connect a host in each lower level structures. A BCube(n,k) network has n^{k+1} hosts and $(k+1)n^k$ switches. The diameter of the network is $k+1$. The bisection bandwidth is not straightforward [20]. When n is even (as it is in our subsequent analysis) and the switches themselves do not present a bottle neck in communications a lower bound is $\frac{n^{k+1}}{4(n-1)}$. An example of BCube network topology (BCube(1,4)) with 4 BCube$(0,4)$s having 4 hosts, resulting in 8 switches and 16 hosts, is shown in Fig. 2.

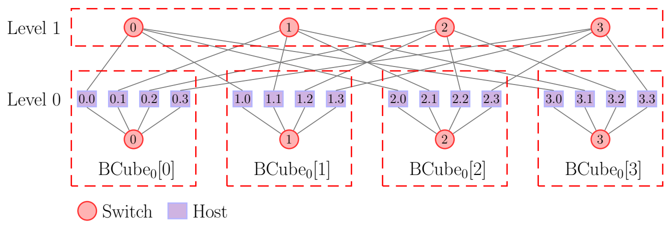

Fig. 2. Server-centric DCN: BCube(1,4) with 4 BCube$_0$s and 4 hosts in each BCube$_0$ [19].

The Butterfly topology is a switched interconnection network topology employed in the BBN Butterfly massively parallel computer [21], [22]. A Butterfly topology with 2^k hosts has $k+1$ switches associated with each host. Hence, there are $(k+1)2^k$ switches and $(k+2)2^k$ nodes. The network bisection bandwidth is 2^{k-1}. An example with 8 hosts is shown in Fig. 3 where each host is connected to two switches. Note that host no. 8 is connected to switches $0,7$ and to $3,7$. The second link is not shown to simplify the diagram.

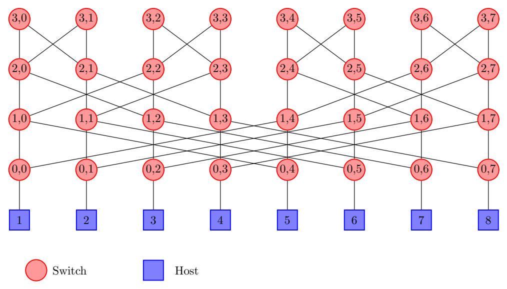

Fig. 3. Butterfly interconnection network with 8 hosts.

Hypercube topologies are employed in high-end, high-performance architectures such as the HP Superdome system that employs a NUMAlink [23] interconnect network. The well-known Hypercube interconnection network consists of 2^k hosts where each host has k links. Hypercubes with $k = 2$ and $k = 3$ are squares of 4 hosts each linked to 2 hosts and cubes of 8 hosts each linked to 3 hosts, respectively. Higher-order Hypercubes with $k > 3$ correspond to cubes in a k-dimensional space. A Hypercube of size k has bisection bandwidth of k.

III. Experimental Results

Performance evaluation of VNE algorithms relies on discrete event simulations that benefit from modular and scalable simulators. Reported simulation results are generated using the VNE-Sim platform [14] that enables: simulation of various network topologies, implementation of VNE algorithms of interest, and generation of random sequences of VNRs.

A. Simulation Platform

VNE-Sim [14] was developed using the ADEVS [24] discrete event simulator and has a modular and scalable architecture that permits implementation of new substrate network topologies and new VNE algorithms. VNE-Sim integrates the BRITE [25], [26] and FNSS [27], [28] network generation tools to produce substrate networks and VNRs. Users may choose VNR parameters: number of network nodes, network topology, as well as frequency and distribution of requests. VNE-Sim users may implement new VNE algorithms and perform simulations using various substrate network topologies and parameters. Butterfly and Hypercube topologies were implemented in VNE-Sim to complete this study.

B. Simulation Scenarios

In this study, we compare network topologies and, therefore, we choose a fixed number of hosts (64) for each considered network topology. The selected network topology determines the numbers of network components (switches and links). Topological properties of networks are listed in Table I.

TABLE I
Properties of Network Topologies

Network topology	No. hosts	No. switches	No. links	Bisection bandwidth	Diameter
Two-Tier	$n_e n_s$	$n_e + n_c$	$n_e n_c + n_s$	$n_e n_c/2$	4
BCube(k, n)	n^{k+1}	$(k+1)n^k$	$(k+1)n^{k+1}$	$\frac{n^{k+1}}{4(n-1)}$	$k+1$
Hypercube	2^k	0	$k2^{k-1}$	k	k
Butterfly	2^k	$(k+1)2^k$	$(k+1)2^{k+1}$	2^{k-1}	$k+1$

We simulated a switch-centric Two-Tier network and a server-centric BCube(2,4) topologies. Examples of each topology are shown in Fig. 1 and Fig. 2. For comparison, we also simulated two interconnection network topologies: a Butterfly topology and a Hypercube topology. An example of a Butterfly topology is shown in Fig. 3. Parameters of the simulated network topologies are shown in Table II.

TABLE II
Parameters of the Simulated Networks

Network Topology	Hosts	Switches	Links	Bisection Bandwidth	Diameter
Two-Tier	64	16	128	32	4
BCube(2,4)	64	48	192	5.33	7
Hypercube	64	0	192	6	6
Butterfly	64	448	896	32	7

C. Performance Results

Simulations are performed using the Digital Research Alliance of Canada's Linux clusters [29] with heterogeneous CPU types: Intel Silver and Gold as well as AMD EPYC Zen2 and Zen3 processors. Simulations require less than 16 GB of memory. Simulated VNR processing times are shown in Table III. The large number of network components (nodes and links) of the Butterfly network caused lengthy simulations. In order to complete the simulations using D-ViNE and R-ViNE algorithms within a reasonable time, the execution time of each network embedding request was limited to 5 minutes.

TABLE III
Average VNR Processing Times

VNE Algorithm	Two-Tier		BCube(2,4)		Hypercube		Butterfly	
	Min (s)	Max (s)	Min (s)	Max (s)	Min (s)	Max (s)	Min (s)	Max (s)
GRC	0.019	0.021	0.046	0.049	0.015	0.018	0.801	0.882
GRC-M	0.377	0.430	0.765	0.959	0.300	0.349	23.388	28.510
MaVEn-M	0.683	1.211	1.316	1.719	0.788	0.922	8.489	10.915
MaVEn-S	1.060	1.542	1.948	2.945	0.921	1.405	34.057	41.713
D-ViNE	1.187	1.615	2.397	3.247	1.008	1.914	35.298	42.698
R-ViNE	1.700	1.770	2.290	3.518	0.962	1.766	36.648	50.526

The comparison of Butterfly, Hypercube, Two-Tier, and BCube topologies is evaluated based on: acceptance ratio, revenue to cost ratio, and substrate resource (node and link) utilization as functions of VNR traffic load. The goal of VNE algorithms is to increase the revenue of infrastructure providers by minimizing the embedding cost [3]. Simulation results are organized based on employed algorithms GRC and GRC-M, D-ViNE and R-ViNE, and MaVEn-M and MeVEn-S and shown in Figs. 4, 5, and 6, respectively.

Plots illustrate the evaluation metrics and compare the four topologies under consideration. The goal of the study is to compare performance of interconnect networks (Butterfly and Hypercube) to data center networks (Two-Tier and BCube). Simulation results indicate that the Butterfly topology leads to: the preferred highest acceptance ratio, the undesirable lowest revenue to cost ratio, and the lowest node and link utilization. In contrast, the Hypercube topology offers: similar acceptance ratio, the desirable highest revenue to cost ratio, and the highest node utilization of all simulation scenarios, thus, illustrating its advantage. Both Butterfly and Hypercube topologies achieve higher acceptance ratios compared to DCN topologies for all considered algorithms. Compared to DCNs, the Butterfly and Hypercube topologies have lower and higher revenue to cost ratio and node utilization, respectively.

An important observation is that the Butterfly topology has nearly an order of magnitude more switches and links resulting in lower revenue to cost ratio and node and link utilization. Hence, the highest acceptance ratio is achieved at a cost of significantly more network elements. The Hypercube topology, using approximately one quarter of links compared to the Butterfly topology, achieves nearly the same acceptance ratio. It is expected that increasing acceptance ratios leads to increased node and link utilization, thus, implying "better" performance. However, if acceptance ratios remain comparable, lower node or link utilization may imply more efficient use of network resources.

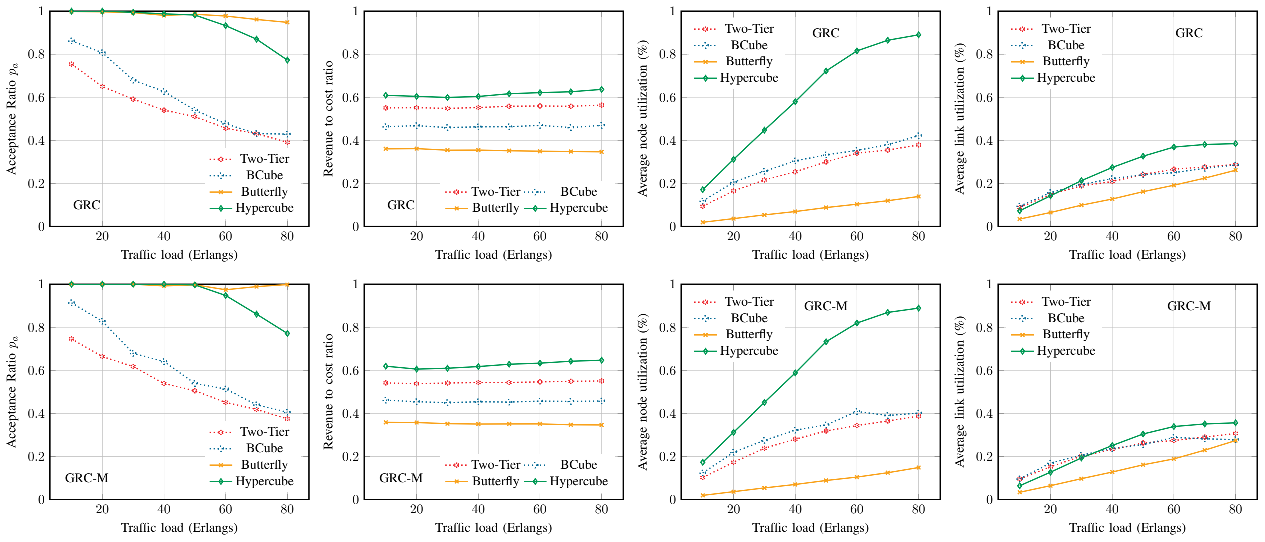
Fig. 4. Comparison of GRC and GRC-M for DCN (Two-Tier and BCube) and interconnection (Butterfly and Hypercube) topologies.

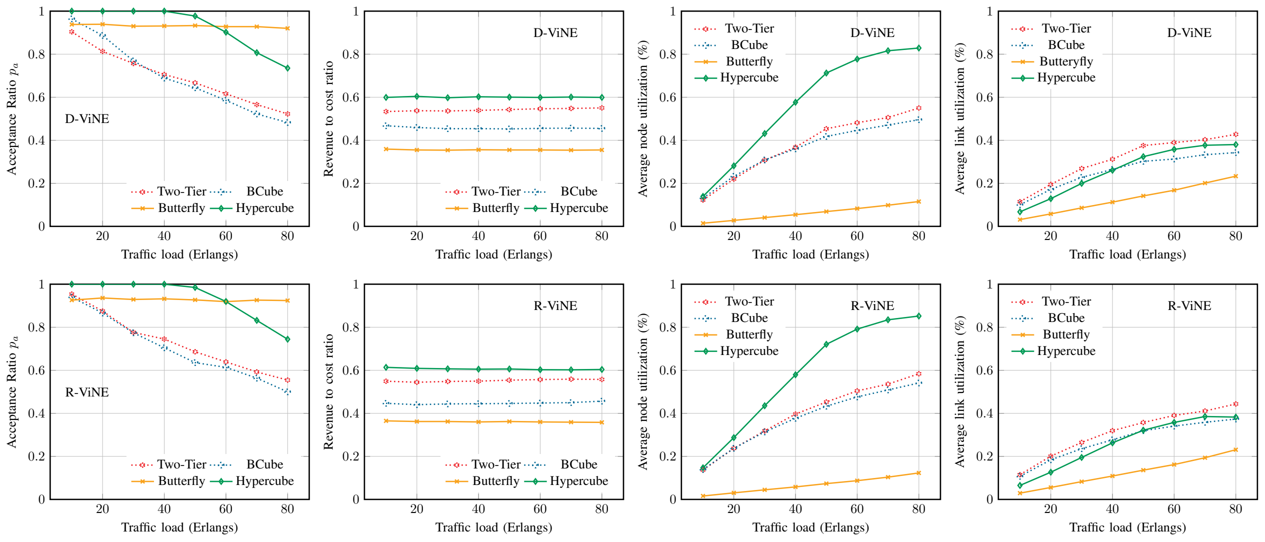
Fig. 5. Comparison of D-ViNE and R-ViNE algorithms for DCN (Two-Tier and BCube) and interconnection (Butterfly and Hypercube) topologies.

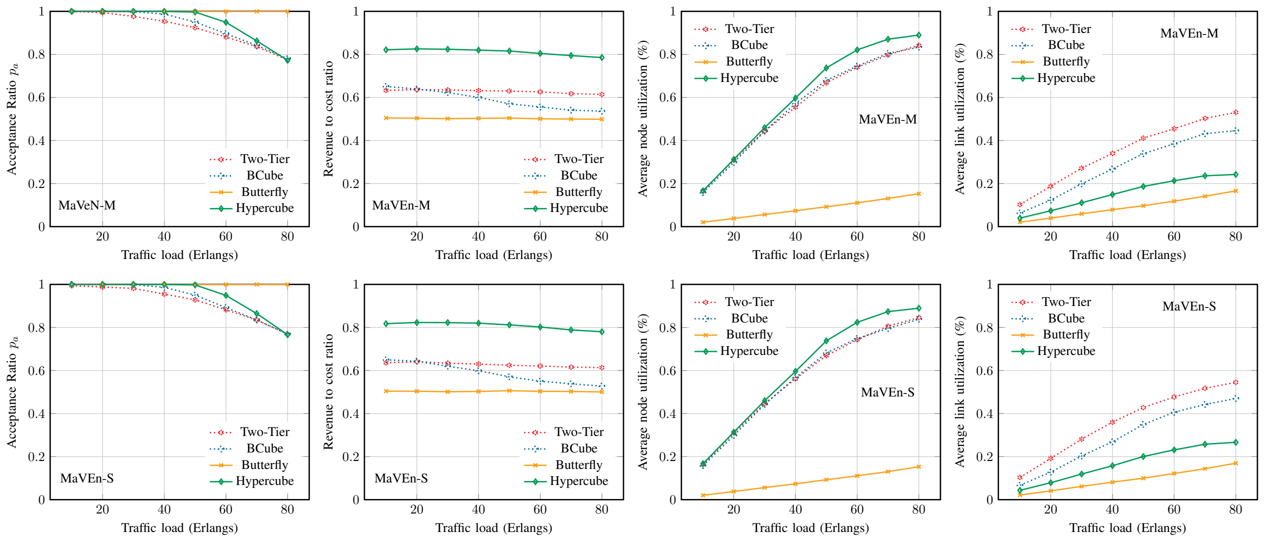
Fig. 6. Comparison of MaVEn-M and MaVEn-S algorithms for DCN (Two-Tier and BCube) and interconnection (Butterfly and Hypercube) topologies.

Simulation execution times for the Butterfly network proved rather lengthy in case of the D-ViNE and R-ViNE algorithms, which rely on solutions of the linear programming (LP) problem. The number of LP variables and equations to be solved are proportional to number of network elements (hosts, switches, links). The VNE-Sim tool employs the GLPK [30] library that solves the LP problem using the simplex method [31] that improves the objective function by iterating through feasible solutions. Hence, the simplex method contributed to lengthy simulation runs. In the case of Butterfly network having large number of network elements, the LP solver search time was limited to 5 minutes of computing time. In cases when the LP solver times-out, the partial solution is discarded and the VNR is rejected. Therefore, the reported D-ViNE and R-ViNE acceptance ratios represent lower bounds. In the case of the Butterfly topology, 6% – 7% of VNRs were discarded due to time-out. Therefore, the upper bound of the acceptance ratio is 6% – 7% higher than the reported results. This accounts for the slightly lower acceptance ratio for the Butterfly topology observable in Fig. 5.

IV. Conclusions

We considered the applicability of Butterfly and Hypercube interconnection topologies to be used as VNE substrate networks. We compared acceptance ratio, revenue to cost ratio, and node and link utilization of interconnection and DCN topologies across various VNE algorithms. Simulations results demonstrate that interconnection networks have the advantage of higher VNR acceptance ratios and, depending on topology, higher revenue to cost ratios than Two-Tier and BCube DCN topologies irrespective of the employed VNE algorithms. While potentially offering improved VNE performance over DCNs, interconnection topologies may require additional network elements. The Butterfly topology considered in this study, offered the highest performance while having the largest number of network elements. The Hypercube topology also offered high performance while requiring substantially fewer network elements. Infrastructure providers offering virtualization services may consider interconnection topologies as the foundation of their designs as they offer distinct advantages over DCN topologies.

References

[1] A. L. G. Rios, K. Bekshentayeva, M. Singh, S. Haeri, and Lj. Trajković, "Virtual network embedding for switch-centric data center networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, Daegu, Korea, May 2021, pp. 1–5.

[2] H. K. Takhar, A. L. G. Rios, and Lj. Trajković, "Comparison of virtual network embedding algorithms for data center networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, Austin, TX, USA, May 2022, pp. 1660–1664.

[3] S. Haeri and Lj. Trajković, "Virtual network embedding via Monte Carlo tree search," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 510–521, Feb. 2018.

[4] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[5] A. Satpathy, M. N. Sahoo, C. Swain, P. Bellavista, M. Guizani, K. Muhammad, and S. Bakshi, "Virtual network embedding: Literature assessment, recent advancements, opportunities, and challenges," *IEEE Commun. Surveys Tuts.*, pp. 1–1, 2025.

[6] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[7] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1–9.

[8] M. Rost and S. Schmid, "On the hardness and inapproximability of virtual network embeddings," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 791–803, 2020.

[9] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *Comput. Commun. Rev.*, vol. 38, no. 2, pp. 19–29, Mar. 2008.

[10] P. Ruiu, A. Bianco, C. Fiandrino, P. Giaccone, and D. Kliazovich, "Power comparison of cloud data center architectures," in *Proc. IEEE Int. Conf. on Commun.*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[11] P. Ruiu, C. Fiandrino, P. Giaccone, A. Bianco, D. Kliazovich, and P. Bouvry, "On the energy-proportionality of data center networks," *IEEE Trans. on Sustain. Comput.*, vol. 2, no. 2, pp. 197–210, 2017.

[12] S. Haeri and Lj. Trajković, "Virtual network embeddings in data center networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, Montreal, QC, Canada, May 2016, pp. 874–877.

[13] S. Haeri, Q. Ding, Z. Li, and Lj. Trajković, "Global resource capacity algorithm with path splitting for virtual network embedding," in *Proc. IEEE Int. Symp. Circuits and Systems*, Montreal, QC, Canada, May 2016, pp. 666–669.

[14] S. Haeri and Lj. Trajković, "VNE-Sim: a virtual network embedding simulator," in *Proc. 9th EAI Int. Conf. Simulation Tools Techniques*, Prague, Czech Republic, Aug. 2016, pp. 112–117.

[15] H. B. Yedder, Q. Ding, U. Zakia, Z. Li, S. Haeri, and Lj. Trajković, "Comparison of virtualization algorithms and topologies for data center networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. 2nd Workshop Netw. Secur. Analytics and Automat.*, Vancouver, Canada, Aug. 2017, pp. 1–6.

[16] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: a survey," *J. Parallel Distrib. Comput.*, vol. 96, pp. 45–74, 2016.

[17] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): infrastructure and operations," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 640–656, 2016.

[18] M. Alizadeh and T. Edsall, "On the data path performance of leaf-spine datacenter fabrics," in *Proc. IEEE 21st Annu. Symp. High-Perform. Interconnects*, San Jose, Calfiornia, USA, Aug. 2013, pp. 71–74.

[19] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, Oct. 2009.

[20] J. Arjona Aroca and A. Fernández Anta, "Bisection (band)width of product networks with application to data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, 02 2012.

[21] T. J. LeBlanc, M. L. Scott, and C. M. Brown, "Large-scale parallel programming: experience with BBN butterfly parallel processor," in *Proc. ACM/SIGPLAN Conf. Parallel Programming: Experience with Applications, Languages and Systems*, Sep. 1988, pp. 161–172.

[22] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. New York, NY, USA: Elsevier, 2004.

[23] (2025, Aug.) SGI NUMALink, Industry Leading Interconnect Technology. [Online]. Available: https://web.archive.org/web/20060328173509/http://www.sgi.com/pdfs/3771.pdf.

[24] J. J. Nutaro, *Building Software for Simulation: Theory and Algorithms, With Applications in C++*. Hoboken, NJ, USA: John Wiley & Sons, 2011.

[25] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: an approach to universal topology generation," in *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2001, pp. 346–353.

[26] (2025, May) Boston University Representative Internet Topology Generator. [Online]. Available: http://www.cs.bu.edu/brite/.

[27] L. Saino, C. Cocora, and G. Pavlou, "A toolchain for simplifying network simulation setup," in *Proc. 6th Int. ICST Conf. Simulation Tools and Techniques (SIMUTools 2013)*, Cannes, France, Mar. 2013, pp. 82–91.

[28] (2025, Aug.) Fast Network Simulation Setup. [Online]. Available: http://fnss.github.io/.

[29] (2025, May) Digitial Research Alliance of Canada National Host Sites. [Online]. Available: https://alliancecan.ca/en/services/advanced-research-computing/federation/national-host-sites

[30] (2025, Aug) GLPK–GNU Linear Programming Kit. [Online]. Available: http://www.gnu.org/software/glpk/.

[31] V. Chvátal, *Linear Programming*. Hoboken, NJ, USA: John Wiley & Sons, 1983.