



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA
DISSERTATION

**EFFICIENT ORCHESTRATION OF DATA
CENTERS VIA COMPREHENSIVE AND
APPLICATION-AWARE TRADE-OFF
EXPLORATION**

by

Alan M. Bairley

December 2016

Dissertation Supervisor:

Geoffrey G. Xie

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE 12-16-2016		3. REPORT TYPE AND DATES COVERED Dissertation 2014-01-06 to 2016-12-16
4. TITLE AND SUBTITLE EFFICIENT ORCHESTRATION OF DATA CENTERS VIA COMPREHENSIVE AND APPLICATION-AWARE TRADE-OFF EXPLORATION			5. FUNDING NUMBERS	
6. AUTHOR(S) Alan M. Bairley				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of the Department of Defense, Chief Information Officer 6000 Defense Pentagon, Washington, D.C. 20301-6000			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) Software-defined network (SDN) orchestration, the problem of integrating and deploying multiple network control functions (NCFs) while minimizing suboptimal network states that can result from competing NCF proposals, is a challenging open problem. In this work, we formulate SDN orchestration as a multiobjective optimization problem, present an evolutionary algorithm designed to explore the NCF tradeoff space comprehensively and avoid local optima, and propose a new application-aware approach that explicitly models resource preferences of individual application workloads. Further, we propose a new logical application workload (LAW) abstraction to enable precomputation of the required relative positioning of an application's virtual machines (VMs) and allocation of these VMs in a single atomic step, leading to online algorithms that are one order of magnitude faster than existing solutions for placing data center workloads. For an instance of the SDN orchestration problem subject to four independent NCFs attempting to optimize network survivability, bandwidth efficiency, power conservation, and computational contention, we demonstrate that our approach enumerates a wider range of, and potentially better, solutions than current orchestrators, for data centers with hundreds of switches, thousands of servers, and tens of thousands of VM slots.				
14. SUBJECT TERMS software-defined networking, network state, data center network, genetic algorithms, virtual machine placement			15. NUMBER OF PAGES 107	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**EFFICIENT ORCHESTRATION OF DATA CENTERS VIA
COMPREHENSIVE AND APPLICATION-AWARE TRADE-OFF
EXPLORATION**

Alan M. Bairley
Major, United States Army
B.S., United States Military Academy, 2004
M.S., Nova Southeastern University, 2012

Submitted in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE
from the
NAVAL POSTGRADUATE SCHOOL
December 2016

Approved by:	Geoffrey G. Xie Professor of Computer Science Dissertation Supervisor	Robert Beverly Associate Professor of Computer Science
	W. Matthew Carlyle Professor of Operations Research	Raluca Gera Associate Professor of Applied Mathematics
	Mathias Kolsch Associate Professor of Computer Science	Dennis Volpano Associate Professor of Computer Science
Approved by:	Peter J. Denning Chair, Department of Computer Science	
Approved by:	O. Douglas Moses Associate Provost for Academic Affairs	

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Software-defined network (SDN) orchestration, the problem of integrating and deploying multiple network control functions (NCFs) while minimizing suboptimal network states that can result from competing NCF proposals, is a challenging open problem. In this work, we formulate SDN orchestration as a multiobjective optimization problem, present an evolutionary algorithm designed to explore the NCF tradeoff space comprehensively and avoid local optima, and propose a new application-aware approach that explicitly models resource preferences of individual application workloads. Further, we propose a new logical application workload (LAW) abstraction to enable precomputation of the required relative positioning of an application’s virtual machines (VMs) and allocation of these VMs in a single atomic step, leading to online algorithms that are one order of magnitude faster than existing solutions for placing data center workloads. For an instance of the SDN orchestration problem subject to four independent NCFs attempting to optimize network survivability, bandwidth efficiency, power conservation, and computational contention, we demonstrate that our approach enumerates a wider range of, and potentially better, solutions than current orchestrators, for data centers with hundreds of switches, thousands of servers, and tens of thousands of VM slots.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction	1
1.1	Orchestration Challenges in Data Centers	3
1.2	Current Data Center Orchestration Approaches	6
1.3	Emerging Trends For Rethinking Data Center Orchestration	10
1.4	Key Design Principles	14
1.5	A Scalable Approach To Comprehensive and Application-Aware Data Center Orchestration	16
2	Orchestrating Network Control Functions via Comprehensive Trade-off Exploration	19
2.1	Introduction	19
2.2	Rethinking Orchestration	20
2.3	Multi-Objective Optimization Formulation	25
2.4	Evolutionary Approach	27
2.5	Evaluation.	30
2.6	Conclusion	36
3	An Application-Aware Approach to Scalable Online Placement of Data Center Workloads	37
3.1	Introduction	38
3.2	Application-Aware Placement	40
3.3	LAWs for Application Efficiency	44
3.4	Evaluation.	56
3.5	Statistical LAWs	64
3.6	LAWs for Workload Prioritization.	68
3.7	Related Work	72
3.8	Conclusion	72

4 Conclusion	73
4.1 Summary of Contributions	73
4.2 EASO-LAW Synthesis: Better Than the Sum of Its Parts	74
4.3 Future Work	78
References	83
Initial Distribution List	89

List of Figures

Figure 1.1	NCF-B proposes to reduce link congestion by load balancing traffic across all network links, whereas NCF-P proposes to save power by concentrating traffic over a small subset of links. . .	4
Figure 1.2	An example of inter-NCF conflict. After implementing a proposal from NCF-B (“balance”) to reduce link congestion by load balancing an active flow from Switch A to Switch C via Switch B (perhaps to avoid congestion at Switch D), NCF-P (“power”) makes a counter-proposal to conserve power by proposing to power down Switch B. If NCF-P’s proposal is implemented while flows through Switch B are still active, a loss of traffic will occur at Switch B.	7
Figure 2.1	Example proposals from NCF-S, NCF-B, and NCF-P. Green slots are for VMs of R1, red R2, and gold R3. We assume that each server, ToR switch, aggregation switch, or core switch uses 1, 2, 2.5, and 3 units of power, respectively.	21
Figure 2.2	Mutated power conservation proposal and recombined BW conservation and survivability proposal.	23
Figure 2.3	Two compromises from successive rounds of recombination and mutation of nondominated candidates.	25
Figure 2.4	EASO Nondominated Front vs. Outer Approximation (OA) of Y_p , for a large-scale data center scenario.	36
Figure 3.1	An example contrasting application-aware vs. application agnostic VM allocations. Green slots are for VMs of R1, red R2, brown R3, and purple R4. “C” and “D” denote a CI and DI workload, respectively. The allocation on the right considers application type (CI or DI) of each workload, and thus attempts to spread VMs of R1 and R3 (CI) to minimize CPU contention, while consolidating VMs of R2 and R4 (DI) to minimize communication overhead.	41

Figure 3.2	LAWs for workloads R1-R4 used in the example scenario of Fig. 3.1. The r_slots and r_bw annotations are omitted for simplicity.	53
Figure 3.3	LAW allocations using “Min Power,” “Min BW,” and “Min CRC” heuristics for example scenario.	54
Figure 3.4	Power usage vs. capacity allocated.	58
Figure 3.5	Mean BW usage vs. capacity allocated.	59
Figure 3.6	Max. link BW usage vs. capacity allocated.	59
Figure 3.7	CRC vs. capacity allocated.	60
Figure 3.8	Mean RCI vs. capacity allocated.	60
Figure 3.9	Execution time vs. capacity allocated.	61
Figure 3.10	Statistical LAW (70%) for workload R4 in example scenario.	65
Figure 3.11	“Min CRC” LAW allocation using a 70% Statistical LAW to allocate previously infeasible workload R4 for the example scenario.	66
Figure 3.12	Effect of Statistical LAW on CRC.	67
Figure 3.13	Effect of Statistical LAW on Mean RCI.	67
Figure 3.14	Effect of Statistical LAW on execution time.	69
Figure 3.15	RCI vs. feasibility tradeoff introduced by Early Statistical LAW (using Min CRC heuristic).	71
Figure 4.1	System flow diagrams for (a) EASO, (b) LAW, and (c) EASO-LAW Synthesis. Steps highlighted in red are computationally intensive (order of tens of seconds or more), those contained in blue are part of the typical process flow, and those in green provide tradeoff exploration. EASO-LAW Synthesis (c) provides tradeoff exploration while avoiding time intensive computation in the typical process flow.	75

Figure 4.2 Heterogeneous multi-tier LAW constructed from a proposal generated by EASO for allocating the 200 VM, 5-tier (40 VMs per tier) tenant workload described in Section 2.5.3. VMs of tier T1 are represented by green ovals, tier T2 red ovals, tier T3 brown ovals, tier T4 purple ovals, and tier T5 blue ovals. For brevity, logical host allocations for each tier are only shown for those under the first logical ToR (vT1). For the other logical ToRs (vT2-vT4), the number of VMs of each respective tier allocated to logical hosts under underneath them is represented by the numbers in the corresponding ovals. 80

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 1.1	Current network orchestration solutions in the context of three design considerations, including the key design principle of comprehensive tradeoff exploration, which none of these solutions offer.	15
Table 2.1	X_s^{EASO} and X_s^{GASO} nondominated solutions. The “Allocation” column represents the allocation of VMs to servers on the four different racks, e.g., [(3,5,0), (2,0,5), (0,0,0), (0,0,0)] represents the assignment of 3 VMs of R1 and 5 VMs of R2 to Rack 1, 2 VMs of R1 and 5 VMs of R3 to Rack 2, and none to Racks 3 and 4.	33
Table 2.2	EASO vs. GASO in distance and coverage of their solution sets w.r.t. Y_p , and in avg. execution time.	33
Table 2.3	Performance of three runs of EASO vs. GASO for large scenario. Best, moderate, and worst results are shaded green, yellow, and red, respectively.	35
Table 3.1	LAW feasibility results for “Min Power,” “Min BW,” and “Min CRC” for different 10240 VM slot infrastructures of varying dimensions. The values denote the average physical infrastructure utilization (fraction of VM capacity allocated) when the corresponding allocation heuristic and LAW type first encounter infeasibility. Poor results (less than 50%) are shaded red, moderate (at least 50% but less than 80%) yellow, and good (at least 80%) green. * reference dimensions	63
Table 3.2	Statistical LAW feasibility results for “Min Power,” “Min BW,” and “Min CRC” for the large-scale evaluation. The values denote the average physical infrastructure utilization (fraction of VM capacity allocated) when the corresponding allocation heuristic and LAW type first encounter infeasibility.	68

Table 3.3	Early Statistical LAW feasibility results for high priority workloads in the large-scale evaluation. The raw values denote the average physical infrastructure utilization when the corresponding allocation heuristic and LAW type first encounter high priority workload infeasibility. The values in parentheses denote the percentage of feasibility increase over default Statistical LAW allocation.	70
-----------	--	----

Acknowledgements

I would like to thank my advisor Geoffrey Xie for teaching me the tactics, techniques, and procedures necessary to become a successful researcher and problem solver in the field of computer networking, and also for providing an infinite supply of positive guidance, encouragement, and support. He has taught me how to find open research problems within the current state-of-the-art, explore and evaluate solutions to these problems, and present research results to different audiences. He has also taught me how to successfully persevere when faced with research obstacles or negative research results. It is both a privilege and an honor to be his student.

I would like to thank Dennis Volpano for his initial and continued guidance in conducting computer science research more generally, and for teaching me how to leverage formal models of computation to concisely formulate real-world problems and determine their complexity. His teachings were instrumental in my development as a student of computer science.

Matthew Carlyle has been another excellent mentor and colleague to me during the course of this dissertation research. His strategy for conducting research, his approach toward finding ways to make contributions, and his passion for solving real problems served as a good example for me to follow.

I also want to thank those people whose collaborations made this dissertation possible. I especially thank Dennis Volpano for pointing me to data center orchestration as a fertile ground for making contributions; Geoffrey Xie for insisting, on multiple occasions, that I “stay the course” to explore each research direction to the fullest extent possible, and for suggesting the use of a bipartite graph matching algorithm to check LAW feasibility; Matthew Carlyle for his insights on multiobjective optimization, especially the use of an outer approximation as an evaluation method, and for his insights on logical application workload (LAW) characteristics and applications; Robert Beverly for his knowledge and insight from a real-world systems research perspective, for his extensive feedback and discussion on the technical aspects of this dissertation, and for connecting me with other researchers in the field; Raluca Gera for initial discussions regarding the general mathematical complexity of the VM place-

ment problem for a set of distinct tenant workloads, and for providing unwavering support and encouragement; and Mathias Kolsch for introducing me to large-scale workloads via Hadoop, and for always reminding me to think about how we can better find and solve real problems in industry.

Another round of thanks goes out to all of the other people who provided constructive comments to help strengthen and improve this dissertation. Specifically, I would like to thank Justin Rohrer, Franck Le, Xin Sun, and the anonymous reviewers of the IEEE NFV-SDN Conference for their helpful comments.

I would also like to thank the other Ph.D. students during my time at NPS, namely Ryan Craven and Brian Kropa, for helping me to prepare for the qualifying exam and for providing feedback on my initial research ideas.

I would like to thank all of the other NPS faculty who provided support, assistance, and guidance to me. I especially thank Man-Tak Shing and Ted Huffmire for providing independent study instruction to assist me in preparing for the qualifying exam; Peter Denning for providing general support, and for encouraging me to pursue my research interests in studying software-defined networks; Al Shaffer for general support and guidance; Cynthia Irvine and Mark Gondree for general support and for helping me connect with other researchers early on; Brittany Ramsey for assisting me in obtaining approval and funding to present at the IEEE NFV-SDN Conference; and Simson Garfinkel, Michael McCarrin, Bruce Allen, and Riqui Schwamm for providing lab resources and general assistance during the initial portion of my candidacy.

I thank Department of Defense Information Assurance Scholarship Program for providing funding for my tuition and general support towards this research.

Finally, I thank my wife, Dr. Robin Bairley, for her unequivocal faith and confidence in my ability to succeed, even when mine was lacking. She also provided helpful feedback on countless presentation rehearsals and paper drafts, and additionally provided insight into certain situations that otherwise would not have been handled so gracefully. Most importantly, she took care of our three children, Braden, Lillian, and Sienna, so that I could pursue this opportunity. Thank you, Robin.

CHAPTER 1:

Introduction

Cloud computing [1], the practice of using warehouse-scale networks of computer servers called data centers to remotely store, manage, and process user data, is becoming increasingly popular as a cost-effective and on-demand solution for both corporate (e.g., Google, Amazon, Microsoft) and government (e.g., Defense Information Systems Agency) service providers to provide mission-critical applications such as email, video teleconferencing, and mission planning, to cloud tenants. A cloud tenant, or simply “tenant,” is a consumer of a provided cloud service. Such consumers may include government agencies, corporate enterprises, or even individual users. The relationship between cloud provider and tenant is typically defined in terms of special contract called a service-level agreement (SLA), which defines the amount of physical data center resources (e.g., CPU, memory, storage, link bandwidth, etc.) to be allocated to some tenant application requirement, or “workload”. A workload is typically represented by the number and size of application components (i.e., virtual machines) required to support the tenant application. It is the task of the data center operator to allocate these application workload components to resources of the physical infrastructure that can support them (e.g., physical host servers, switches, middleboxes). Quality-of-service (QoS) requirements, such as latency, response time, throughput, and application availability required by the tenant workload may also be stated in the SLA. Hence, although cloud tenants are typically only concerned with a single objective, namely the cloud provider’s ability fulfill their stated SLA, the goals of the data center operators providing cloud services are more complex: their aim is to fulfill the SLA of all tenants while jointly achieving a multitude of “operator objectives”. Such objectives include fault tolerance, typically measured by the fraction of application components (e.g., virtual machines) that survive a single worst case physical component (e.g., network device) failure; network communication efficiency, which may be measured by the average (or maximum) amount of bandwidth allocated (or latency incurred) over a set of network links, and power conservation, measured by the amount electrical power used.

A recent Cisco report [2] provides clear evidence that “enterprise and government organizations are moving from test environments to placing more of their mission-critical application workloads into the cloud.” According to this report, the number of application workloads processed by cloud data centers and the total amount of network traffic generated by these workloads are each expected to increase by over 300% over the next five years. This unprecedented demand for cloud services has led to the development of complex large-scale data centers to host them while creating a multitude of new challenges for the data center operators charged with orchestrating their deployment.

First, certain operator objectives like fault tolerance, network communication efficiency, and power conservation may be in conflict with each other, and hence require tradeoff considerations. Second, different types of applications may bottleneck different data center resources. For instance, placing components of computationally intensive workloads on the same physical host server may cause a throughput bottleneck from CPU overcontention, whereas data intensive workload components may cause a network bottleneck if spread across several different physical hosts due to the high data throughput required of such tasks. Thus, different types of applications require different resource allocation considerations. Third, the data center networks that serve as the backbone of cloud service providers are large, on the order of hundreds of switches, thousands of servers, and tens of thousands of virtual machines (VMs), or larger [2]; thus, for automated orchestration solutions to be effective, they must be scalable to large network topologies (e.g., thousands of devices) at reasonable time scales (e.g., order of seconds).

The key question that this dissertation answers is how to orchestrate the allocation of tenant workloads in data center networks, in a scalable way, to comprehensively explore the tradeoff space between competing operator objectives while minimizing resource contention among individual applications.

1.1 Orchestration Challenges in Data Centers

Software-defined networking (SDN) technology provides an open platform for developing specialized network control functions (NCFs) to achieve separate operator goals, such as bandwidth efficiency [3, 4], fault tolerance optimization [5], power conservation [6], quality-of-service (QoS) control [7], and security enforcement [8]. In this work, our model for network orchestration is similar to Statesman [9], Corybantic [10], and Athens [11], where each NCF is a specialized SDN control program that proposes changes to the current network state for consideration by an orchestration program in order to accomplish some specific objective. The orchestration program is charged with evaluating the proposals of each NCF, and subsequently implementing any changes to the physical network. Proposed changes made by NCFs may include allocating or migrating tenant VMs to host servers, adding or removing forwarding rules from routing tables, updating access control lists on network devices, powering on or off devices or device components, etc. Clearly, two separate NCFs may make conflicting proposals. For instance, one NCF may want to use a specific resource (e.g., server, switch, network link), while another may want to power it off, as illustrated in Figures 1.1 and 1.2. Furthermore, we assume that each NCF has a corresponding utility or objective function that its proposed changes seek to maximize.

Ideally, data center operators seek to achieve network states that jointly optimize their objectives given customer (i.e., tenant) requirements. Conflicting objectives, diverse application workloads, varying application demand, and the size of data center networks make achieving ideal network orchestration challenging, especially within large-scale multi-tenant data centers.

1.1.1 Objectives May Conflict

Data center orchestration is inherently multi-objective in nature. Even if a data center operator is primarily concerned with a single objective such as minimizing link congestion, there are natural tradeoffs in other objectives, such as power conservation, that must be accepted for such optimization.

To illustrate such tradeoffs, consider two hypothetical NCF proposals for a fat-tree

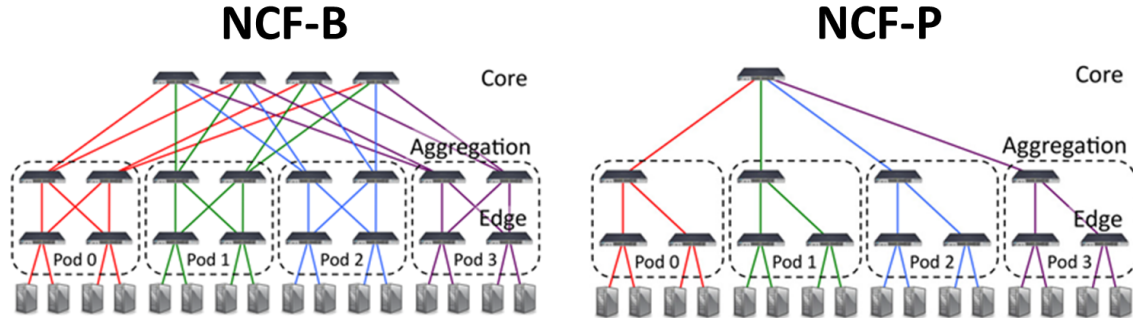


Figure 1.1: NCF-B proposes to reduce link congestion by load balancing traffic across all network links, whereas NCF-P proposes to save power by concentrating traffic over a small subset of links.

data center architecture [12], as depicted in Figure 1.1. The fat-tree architecture is a physical network topology commonly used in data networks representing a hierarchical multi-rooted tree consisting of four levels: core switches (root, level = 0), aggregate switches (level = 1), edge or top-of-rack (ToR) switches (level = 2), and host servers (leaves, level = 3), with redundant links connecting devices at each level. A data center operator wishing to minimize link congestion may choose to implement NCF-B’s proposal to load balance traffic over the maximum number of links, but in doing so achieves a network state that may be suboptimal when the additional objective of power conservation is considered. Conversely, if a data center operator is solely concerned with power conservation, then he/she may opt to implement NCF-P’s proposal, which concentrates traffic over a minimum subset of network links and switches, allowing unused resources to be powered down. But this choice comes with the tradeoff of higher link congestion. Clearly, a comprehensive approach for exploring the tradeoff space of multiple NCF proposals is desirable.

1.1.2 Application Workloads Are Diverse

Application workloads hosted by data center networks stress a range of different resources. Generally, we observe that these workloads are either 1) CPU or memory intensive workloads (e.g., meteorological, geological, and particle physics simulations, or other high performance computing tasks), commonly referred to as compute-intensive

(CI) workloads, 2) network or storage intensive workloads (e.g., client-server applications or Hadoop and other big data applications), commonly referred to as data-intensive (DI) workloads, or 3) both (e.g., large big data processing tasks). Recent research efforts [13, 14] confirm our intuition that CI workloads perform better when their subcomponent processes and VMs are spread across *separate* physical CPU cores and host servers, respectively, while DI workloads perform better when the VMs (or processes) are placed on the *same* host server (or CPU core). For example, by placing all VMs of a client-server (DI) application on the same physical host, tenfold increases in throughput have been observed [13]. In contrast, over-contention of CPU resources by VMs of CI applications have been shown to increase job completion time by as much as 260% [13].

Therefore, we argue that an ideal data center resource allocation strategy should consider the characteristics of individual application workloads in addition to the tradeoffs between the various competing global objectives considered by the data center operator.

1.1.3 Data Center Networks Are Large

The devices composing a data center network is typically in the order of thousands [5], making the manual management of such a network tedious and prone to error and inefficiency. Software-defined networking (SDN) technology offers network administrators the promise of convenient, efficient, and accurate network management by enabling the development and deployment of automated NCFs, i.e., SDN control programs, that automatically perform some set of resource allocation or network configuration actions to achieve operator objectives. However, even automated network management and orchestration strategies that leverage SDN technology must be scalable to handle the large application workloads and physical topologies characteristic of today’s data centers.

1.2 Current Data Center Orchestration Approaches

Prior work in data center orchestration can be distinctly categorized as either 1) NCF synchronization approaches [9], or 2) NCF resource allocation approaches [10, 11]. Synchronization approaches like Statesman [9] view the underlying network as a shared resource contested for by several NCFs, and seek to arbitrate control of devices (e.g., switches, servers) or device-components (e.g., switchports, CPU cores) by detecting and resolving conflict among NCF proposals, whereas existing resource allocation approaches like Corybantic [10] and Athens [11] evaluate the utility of various NCF proposals in terms of the allocation of physical network resources (e.g., hosts, switches, network links, middleboxes) to support tenant requirements (e.g., VMs, application SLA, flow rules, network services) in order to maximize the utility afforded to the network operator. So while NCF synchronization approaches detect and resolve conflict among conflicting NCF proposals, and perhaps allow multiple non-conflicting NCF proposals to be implemented simultaneously, NCF resource allocation approaches assume that NCF proposals conflict, and aim to select the proposal that best achieves the objectives of the operator.

1.2.1 NCF Synchronization Approaches

Synchronization approaches to data center orchestration, and network orchestration more generally, essentially seek to detect and resolve inter-NCF conflict (Figure 1.2) at a low level by ensuring mutual exclusion of device or device-component control among competing NCF proposals.

In the Figure 1.2 conflict scenario, NCF-B wants to load balance traffic to minimize link congestion, while NCF-P wants to power down one of these switches to conserve power. Clearly, shutting down some switch forwarding an active flow may result in traffic loss. If the orchestration program allows both NCFs to control the same resource by attempting to implement both proposals simultaneously or by cycling between them, then network instability (oscillation) may result [15], e.g., the NCFs alternate by cycling the power on the resource.

Synchronization approaches to network orchestration have several benefits, but a few

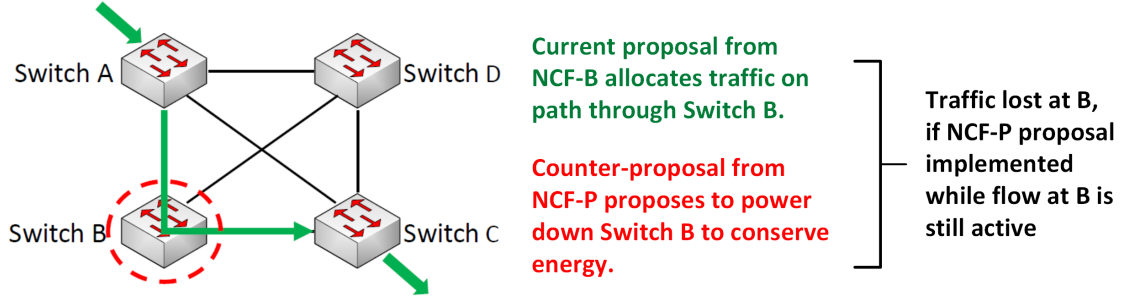


Figure 1.2: An example of inter-NCF conflict. After implementing a proposal from NCF-B (“balance”) to reduce link congestion by load balancing an active flow from Switch A to Switch C via Switch B (perhaps to avoid congestion at Switch D), NCF-P (“power”) makes a counter-proposal to conserve power by proposing to power down Switch B. If NCF-P’s proposal is implemented while flows through Switch B are still active, a loss of traffic will occur at Switch B.

significant drawbacks. One benefit is the speed and scalability by which such an approach may be implemented. Statesman effectively detects and resolves conflict among NCF proposals, and further implements a policy checker to ensure that the resultant network state complies with high-level network policy, all within the order of seconds for large-scale data centers containing thousands of devices. Synchronization solutions are also well suited to implement proposals from multiple NCFs simultaneously, as long as the proposals are non-conflicting.

Real-time NCF Synchronization. NCF synchronization solutions equipped to orchestrate real-time NCFs (RT-NCFs), or NCFs that operate with hard time constraints (e.g., at line speed), comprise a subset of NCF synchronization that is gaining increasing attention from the research community. One recent work by Volpano et al. [15] offers a unique approach in addressing the SDN orchestration problem for RT-NCFs by representing NCFs as deterministic finite transducers (DFTs). Where Statesman, Corybantic, and Athens may be considered “black box” or “grey box” approaches, the work by Volpano et al. employs a distinctive “white box” approach, by exposing internal NCF control logic as DFTs. Unlike prior work in SDN orchestration that uses an orchestration program to act as shim layer between NCF proposals and the physical network infrastructure, the DFT approach used in [15] enables NCFs to impart their proposals directly on the physical devices and device components of the

infrastructure in a way that is free of both conflict and oscillation. This approach performs direct low-level NCF orchestration without the need to consult an external orchestration program, making it suitable for the orchestration of RT-NCFs.

However, none of these synchronization approaches consider the exploration of trade-offs with respect to the *utility* (e.g., operator-defined objective metrics) offered by a range different NCF proposals. Thus, we view these approaches as largely orthogonal to our work, as we are primarily interested in exploring the utility of various feasible NCF proposals within the tradeoff space with respect to competing operator objectives.

1.2.2 NCF Resource Allocation Approaches

NCF resource allocation approaches evaluate competing NCF proposals in terms of the joint utility offered to the network operator. In contrast to NCF synchronization approaches, which may implement multiple non-conflicting proposals simultaneously, NCF resource allocation approaches rank each proposal in terms of its utility offered, and strive to discover the one that offers the best value to the operator. Current resource allocation solutions [3, 4, 10, 11, 16] overwhelmingly attempt to reduce the multi-objective nature of orchestration to a single-objective problem (SOP), either by optimizing a single objective function subject to the others cast as constraints, as done in [3, 4]; or by combining multiple objectives into a single global cost function, as done in [10, 11, 16].

Single Objective Optimization Approaches

Pure SOP optimization approaches such as [3, 4] are limiting in that only a single objective metric, such as bandwidth allocation, is optimized; all other objectives (e.g., fault tolerance, communication latency, power consumption, etc.) are cast as constraints. Although using such approaches permit the use of specialized heuristics to quickly and scalably achieve resource allocations that optimize the selected metric, they are limiting due to the fact that only a single metric is considered.

Global Cost Minimization Approaches (MOP-to-SOP Reduction)

Global cost minimization approaches like [10, 11, 16] recognize the inherently multi-objective nature of data center orchestration, but instead of using a pure multi-objective optimization problem (MOP) formulation for orchestrating multiple NCFs, they instead attempt to combine the individual NCF utility metrics representing distinct operator objectives, into a single, consolidated global objective metric, that is intended to represent the net “cost” of implementing a proposal to the data center operator. The problem with these types of approaches mainly concerns the difficulty in effective implementation. Because all of the NCF utility metrics are combined into a single global cost function, some method of NCF weighting must be used to ensure that the proposal selected (i.e., the discovered proposal that yields the lowest value of the global cost function) actually achieves the goals of the operator. But the operator may not have a particular set of NCF weights a priori, and it may take several successive runs using different weightings in order to find weights that produce a desirable proposal. Furthermore, certain objectives, such as bandwidth allocation and fault tolerance, may be disparate or orthogonal with respect to one another. Thus, it may not be possible to combine such objectives into a single global objective function in a way that preserves operator intent.

1.2.3 Summary

Because we are primarily interested in comprehensively exploring the utility of various feasible proposals within the tradeoff space with respect to competing NCFs, we choose to focus on the formulation of the data center orchestration problem as a NCF resource allocation problem. Hence, our work is motivated by existing resource allocation approaches, which largely attempt to optimize SOP formulations of the orchestration problem. However, although SOP formulations of the orchestration problem permit faster solutions, solving a SOP yields only a *single* solution within a potentially vast tradeoff space. Furthermore, many current approaches use search algorithms based on greedy heuristics [4, 10, 11], which may prematurely converge to suboptimal local maxima when applied to non-convex optimization problems.

Thus, we believe it prudent to explore an alternative formulation based on the classical

multi-objective optimization problem (MOP) literature [17–19], where the goal is to enumerate a diverse set of *Pareto-optimal* solutions [19] among competing NCFs, i.e., no solution can be improved in any objective without causing a degradation in at least one other objective.

1.3 Emerging Trends For Rethinking Data Center Orchestration

Most of the limitations with current techniques for data center orchestration stem from assumptions made about the types of application workloads to be hosted or the specific objective metrics to be optimized. As a result, the large body of work related to orchestration and resource allocation in data center networks mainly consists of disparate specialized solutions for specific problems rather than a cohesive body of general solutions applicable to a wide range of problems. However, due to the overwhelming industry migration from the smaller, traditional, task-specific data centers of the past, to the larger, cloud-based, multi-tenant data centers of the future, data center operators are hosting an increasingly wide range of diverse applications with different requirements and optimization preferences.

There have been several recent directions arising from the research community to address these new challenges: multi-objective orchestration, application-aware orchestration, and scalable orchestration.

1.3.1 Multi-Objective Orchestration

Corybantic [10], Athens [11], and NFC [16] are recent data center orchestration proposals that illustrate the continuing trend towards orchestration solutions that attempt to jointly optimize multiple objectives.

In Corybantic [10], the authors view SDN orchestration as a multi-objective problem, and attempt to reduce it to a SOP transforming the performance criteria of each NCF, or module as called by the authors, into a common representative currency interpretable by the orchestrator. Corybantic proceeds by soliciting each NCF for its proposed network state, evaluating the cost each proposal in terms of the common currency, and finally selecting the most cost-effective proposal for implementation.

Unfortunately, the Corybantic approach suffers from three significant pitfalls: 1) Expressing the performance criteria of disparate NCFs in terms of a singular, universal metric may not be feasible due to disparate objectives. 2) The range and diversity of candidate network states are limited to proposals made by specialized NCFs, and hence unlikely to achieve mutually beneficial inter-NCF compromises. Hence a large portion of the state space may be unexplored. 3) Corybantic uses an iterative selection process based upon a single greedy criterion, i.e. it selects the NCF proposal that yields the lowest “common currency,” a single global metric that each NCF utility metric is reduced to via an operator-specified weighting. Hence, it is essentially a hill climbing approach, and thus may converge suboptimally towards a local minima.

Another related work that uses a multi-objective approach towards orchestration, is Athens [11]. Athens builds on Corybantic by suggesting a family of voting procedures available to each NCF, better enabling them to reconcile disparate objectives by soliciting for each other's feedback in terms of “votes”. So while Athens partially addresses the first of the Corybantic pitfalls described above, it is still subject to pitfalls 2) and 3) of Corybantic as described in the previous paragraph.

Finally, the authors of the recent Network Function Center work [16], while using a similar MOP to SOP reduction as Corybantic and Athens in their problem formulation, recognize the pitfalls of solely using greedy NCF-specific heuristics to perform non-convex SOP optimization, and choose to use genetic algorithms to help overcome the issue of premature algorithm convergence to local optima. However, although the Network Function Center provides better solutions to non-convex SOPs, it is still limiting in its formulation as a SOP for reasons described previously. It is also unclear how well the Network Function Center approach enumerates the tradeoff space between multiple objectives, as all SOP component metrics and weightings are determined a priori and only a single resultant network state is produced as a solution.

In summary, the chronological evolution of related work in data center orchestration from Corybantic (2013) to Athens (2014) to Network Function Center (2015), represents a strong research direction towards multi-objective formulations for the orchestration problem and better approaches for finding good solutions. However, what's still missing from the state-of-the-art is a pure MOP formulation of the or-

chestration problem and a corresponding approach to comprehensively explore the tradeoffs between multiple NCFs, as proposed in this dissertation.

1.3.2 Application-Aware Orchestration

As opposed to focusing solely on optimizing the objectives of data center operators, such as cumulative resource usage, there is an increasing trend in the research community to leverage additional opportunities to intelligently orchestrate application workloads to improve per application performance of all hosted applications. Specifically, recent studies [13, 14] demonstrate that different types of workloads contend for different types of resources. Consequently how VMs of an application are relatively positioned can significantly impact the performance of an application. For example, it would be advisable not to co-locate VMs of computation intensive workloads to avoid unnecessary CPU contention while at the same time, position VMs of the same data intensive workload as close as possible to reduce both bandwidth contention and communication latency [14].

But although the concept of application-aware orchestration is not new, existing solutions that claim to support application-aware orchestration, such as [20–22], only do so with respect to certain types of applications. For instance, [20] only considers data intensive applications, [21] strives to minimize network traffic subject to physical server resource constraints, similar to [23], and [22] focuses primarily on computation and memory intensive high-performance computing applications. Reference [24] considers fair-ness in multi-tenant datacenter environments by searching for placements of tenant applications that minimize the sum of network diameter of all tenants, but does not consider different application types. And while NETMAP [25] considers tenant applications with different requirements, it only considers network traffic and fault tolerance requirements of applications, and thus does not speak to the placement of computationally intensive or storage intensive applications.

None of these provides a general solution for best allocating different types of individual application workloads (e.g., computation or memory intensive vs. data or network intensive) within a shared multi-tenant data center to jointly achieve tenant (cloud customer) and operator (cloud provider) objectives.

1.3.3 Scalable Orchestration

Maintaining fast speeds of execution (e.g., order of seconds) for online orchestration algorithms, i.e. algorithms that allocate data center resources to new tenant application workload requests as they arrive, is yet another clear trend in the research community, especially as the size of data center network topologies approaches large-scale (i.e., thousands of physical hosts, tens of thousands of VM slots). The reason for this trend is likely tied to the overwhelming increase in the number and size of application workloads hosted by cloud data centers, as well as the increases in size (i.e., number of devices) of the physical topologies that make up the networks of these data centers.

However, despite this increasing need for scalable orchestration approaches, recent work has been struggling to maintain fast resource allocation times as the size of workloads and physical topologies approach large-scale, especially when multiple objectives are considered. Specifically, the approach in [23] optimally minimizes link congestion in the order of minutes for 1024 VM slots, but may be slowed to the order of hours when other objectives are considered [16]. In [16] a genetic algorithm is used to approximate a minimal solution to a five-component weighted cost function in the order of tens of seconds, but only scales to a relatively small 64-server topology. Other online VM placement solutions, such as [26] and [27], consider large and distributed data center scenarios, respectively, but do not explicitly state execution times for these scenarios. Single-objective orchestration approaches such as [3, 4] achieve execution time in the order of seconds for 2000+ host, 40000+ VM slot topologies, but are limiting in that they only approximate a solution to a single objective function, namely bandwidth usage, given application survivability constraints. As such, while the approaches in [3, 4] have proven scalable, they are not generally extensible to multi-objective scenarios.

What's needed is a scalable orchestration solution extensible to a multitude of disparate objectives that still achieves fast execution times (e.g., order of seconds) for allocating large application workloads (e.g., hundreds of VMs) within large data center networks (e.g., thousands of servers).

1.4 Key Design Principles

This dissertation describes the design, implementation, and evaluation of new approaches that address two key challenges: 1) achieving visibility and consideration of potential tradeoffs among a multitude of NCFs to accommodate a wide range of operator objectives and network conditions, and 2) achieving efficient allocations of network resources to application workloads in a manner that jointly considers the objectives of the operator, and the preferences of each individual application. There are two key design principles that guide the design of these new approaches:

Comprehensive Tradeoff Exploration Principle. A data center orchestration program at minimum must solicit and rank proposals from all NCFs. If an operator knows *a priori* how to jointly model multiple objectives with a single ranking metric, the orchestrator may optimize the allocation based on the metric in order to find a “best compromise” solution for all the objectives. However, this approach places a heavy burden on the operator to create the right ranking model for his/her network. More importantly, it is an open question whether a search based on such joint models can cover the potentially vast tradeoff space between multiple objectives. Table 1.1 illustrates the shortcomings of current orchestrators with respect to three desirable design considerations. Note that none of the current solutions offer a diverse set of tradeoffs among separate NCFs.

Thus, we argue that formulating the orchestration problem as a MOP that explicitly represents each distinct operator objective, is a more effective way to discover globally optimal compromises when all NCFs and operator objectives are considered. By maintaining a wide range of solution candidates and applying the concepts of natural evolution, i.e., performing mutations and recombinations of high-fitness candidates, a diverse set of nondominated proposals may be generated and presented to the operator for consideration. This is better than proposing a single “best” allocation, since operator priorities are likely to be fluid to accommodate rapidly changing tenant demands and network conditions.

Logical Application Workload (LAW) Principle. Because different types of applications contend for different types of resources, for a given application workload,

Orchestration Scheme	Offer Diverse Set of Tradeoffs	Black Box NCFs	Disparate Objectives
Corybantic [Mogul et al. 2013]			
Statesman [Sun et al. 2014]			
DFT [Volpano et al. 2014]			
Athens [AuYoung et al. 2014]			

Table 1.1: Current network orchestration solutions in the context of three design considerations, including the key design principle of comprehensive tradeoff exploration, which none of these solutions offer.

some feasible resource allocations for it may be better than others. For instance, two distinct allocations for a given workload may use the same number of VM slots, but one allocation may be better than the other if the relative positioning of its VMs achieves strictly less resource contention while maintaining the same values of global operator objectives with respect to the other.

Moreover, we recognize that by independently simulating the desired placements of individual applications in a contention-free environment, e.g., an empty physical topology, we can precompute the ideal allocations of an application’s VMs for a wide range of workload profiles, and then subsequently treat the collection of individual “per VM” allocations for a workload as single atomic unit. We term each of these atomic units a logical application workload (LAW). LAWs are important because current per VM approaches for placing workloads into data centers consisting of thousands of servers does not scale very well. Specifically, execution times for such resource allocation approaches range from the order of hours at worst to the order of tens of seconds at best, and these times only get worse as the sizes of application workloads and physical topologies increase, which, according to current trends in data center growth, is already happening [2].

By allocating LAWs, which typically consist of tens or even hundreds of VMs, as opposed to individual VMs, algorithms for efficiently allocating large data center workloads become scalable, permitting sub-second placement times. And since LAWs are explicitly constructed to minimize contention from the perspective of the VMs belonging to the application, resultant allocations comprised of successive LAW placements come with some guarantees of application performance, which may assist operators in ensuring that they meet required SLAs.

1.5 A Scalable Approach To Comprehensive and Application-Aware Data Center Orchestration

Following the key design principles, we propose a scalable approach to comprehensive and application-aware data center orchestration consisting of two conceptually distinct but thematically coherent algorithms: 1) an Evolutionary Algorithm for SDN Orchestration (EASO) that comprehensively explores the tradeoff space among multiple objectives, and 2) a scalable algorithm for rapidly constructing and allocating Logical Application Workloads (LAWs) to jointly achieve operator objectives and application preferences. We have built prototypes for both algorithms and evaluated them with realistic workload traces in simulated large-scale data center environments.

EASO [28]: Orchestrating Network Control Functions via Comprehensive Trade-off Exploration. In this contribution, we formulate SDN orchestration as a multiobjective optimization problem, and present an evolutionary approach designed to explore the tradeoff space between multiple NCFs comprehensively and avoid local optima. For an instance of the VM allocation problem subject to three independent NCFs optimizing network survivability, bandwidth efficiency, and power consumption, respectively, we demonstrate that our approach can enumerate a wider range of, and potentially better, solutions than current orchestrators, for data centers with hundreds of switches, thousands of servers, and tens of thousands of VM slots.

LAW [29]: An Application-Aware Approach to Scalable Online Placement of Data Center Workloads. Currently, operators allocate workload VMs primarily in an application-agnostic fashion, focusing on minimizing total resource usage.

In this contribution, we first show that such allocations can be suboptimal when intra- and inter-application resource contention is present, and then present a new application-aware approach that explicitly models the resource preferences of individual workloads. Further, we propose a new logical application workload (LAW) abstraction to enable precomputation of the required relative positioning of an application’s VMs and allocation of these VMs in a single atomic step, leading to online algorithms that are one order of magnitude faster than existing per VM placement solutions. We then develop a statistical extension of LAW to add flexibility in characterizing application requirements and to support prioritization of workloads. Using realistic workload traces and physical topologies, we evaluate our algorithms in a simulated large-scale data center setting, and demonstrate their performance advantages and potential tradeoffs versus existing solutions.

The remainder of this dissertation is organized as follows: Chapter 2 presents EASO, implemented via an evolutionary algorithm for comprehensively exploring tradeoffs between multiple NCFs. Chapter 3 represents the LAW principle, implemented via scalable LAW construction and allocation algorithms to achieve specified operator objectives while minimizing resource contentions among individual applications. Chapter 4 discusses contributions of each separate principle when used independently, and also describes how the two principles may be combined to achieve the benefits of both. We then present ideas for future work and provide conclusions.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:

Orchestrating Network Control Functions via Comprehensive Trade-off Exploration

This chapter focuses on the goal of providing several workable, alternative solutions to the orchestration problem to a decision maker. We argue that this goal is best achieved by comprehensively exploring the tradeoff space among multiple network control functions (NCFs), each designed to achieve a specific operator objective. Simply striving to minimize a global cost function representative of cumulative resource usage is not sufficient, as disparate objectives, such as fault tolerance and security enforcement, exist and are important to tenants and data center operators alike. Instead, we choose to formulate the orchestration problem as a pure multi-objective problem that explicitly represents each operator objective individually, and subsequently propose EASO, an evolutionary approach that provides a set of desirable tradeoffs to the data center operator, given a set of tenant application workload requirements.

For an instance of the orchestration problem subject to three independent NCFs attempting to optimize network survivability, bandwidth efficiency, and power conservation, respectively, we demonstrate that our approach can enumerate a wider range of, and potentially better, solutions than current orchestrators, for data centers with hundreds of switches, thousands of servers, and tens of thousands of VM slots.

2.1 Introduction

In this contribution, we defer the detection and resolution of NCF conflicts to other work, such as [9, 15], and choose to focus on exploring the *utility* of various feasible NCF proposals within the tradeoff space with respect to competing NCFs. We define this notion of utility with respect to the NCFs themselves, as done in [10, 11], under the assumption that each independent NCF has a corresponding utility function that it seeks to maximize.

Using this notion of NCF utility, we define the NCF orchestration problem as a multi-objective optimization problem (MOP) where each NCF utility function comprises a component of the optimization problem. Prior work [3, 4, 10, 11, 16], overwhelmingly attempts to reduce the multi-objective nature of orchestration to a single-objective problem (SOP), either by casting multiple objectives in terms of a single global utility function [10, 11, 16], or by optimizing one objective function subject to the others cast as constraints [3, 4].

Although SOP formulations of the orchestration problem permit faster solutions, solving a SOP yields only a *single* solution within a potentially vast tradeoff space. Furthermore, many current approaches use search algorithms based on greedy heuristics [4, 10, 11], which may prematurely converge to suboptimal local maxima when applied to non-convex optimization problems. Thus, we believe it prudent to explore an alternative, more basic formulation based on the classical multi-objective optimization problem (MOP) literature [17–19], where our goal is to enumerate a diverse set of *efficient* solutions among competing NCFs, i.e. each solution in the set cannot be improved in any objective without causing degradation in at least one other objective.

In the following sections, our contribution is three-fold. First, we present a novel MOP problem formulation for SDN orchestration. Second, we describe an evolutionary approach for enumerating a wide range of efficient NCF proposals, scalable to topologies of thousands of hosts and hundreds of switches. Third, we present new metrics and use them to evaluate our approach vs. current solutions in the context of the VM allocation problem.¹

2.2 Rethinking Orchestration

Orchestrating multiple NCFs to achieve and maintain stable and desirable network operating conditions face major technical challenges. To illustrate them, consider the following simplistic scenario where a data center must allocate virtual machines (VMs) that can be supported by its physical infrastructure to a set of tenant applications.

¹Although we focus on VM allocation NCFs in this work, we are confident that our approach is generally applicable for orchestrating NCFs of any virtual network function.

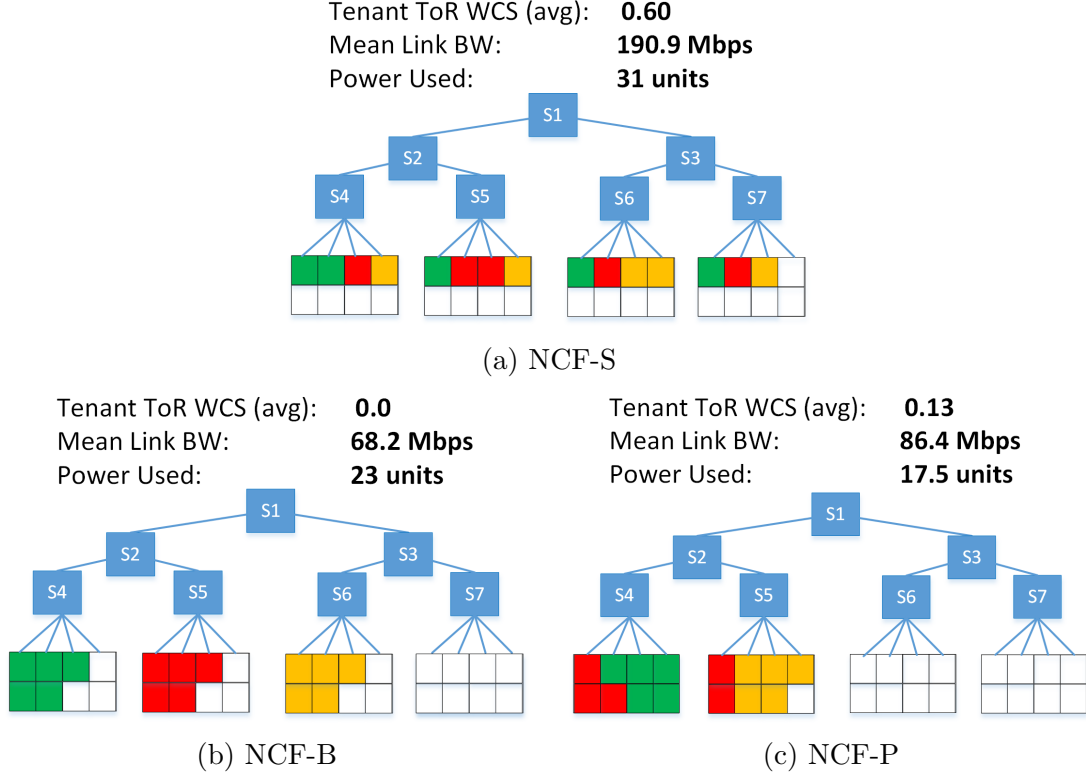


Figure 2.1: Example proposals from NCF-S, NCF-B, and NCF-P. Green slots are for VMs of R1, red R2, and gold R3. We assume that each server, ToR switch, aggregation switch, or core switch uses 1, 2, 2.5, and 3 units of power, respectively.

We have chosen to study VM allocation because 1) it is one of the important first steps of any data center operation, and 2) the problem has an extensive collection of prior work for us to compare to our contribution.

In order to specify requirements for instances of the SDN orchestration problem, we use $\langle \text{VM}, \text{BW} \rangle$ pairs, where the first value in the tuple represents the number of VMs required, and the second value represents the inter-VM bandwidth (BW) requirement. Suppose three independent tenant applications R1, R2, and R3 have requirements $\langle 5, 50 \text{ Mbps} \rangle$, $\langle 5, 100 \text{ Mbps} \rangle$, and $\langle 5, 150 \text{ Mbps} \rangle$ respectively. Suppose the underlying physical infrastructure has a simple tree topology, consisting of one core switch as root, two aggregation switches and four top of rack (ToR) switches, four host servers per ToR switch, and two VM slots per host server, where

a “VM slot” is defined as a standard physical resource unit (e.g., CPU, Memory) provisioned to a VM. Therefore, the VMs are to be allocated against $4 \times 4 \times 2 = 32$ possible slots.

Suppose the data center utilizes three different NCFs simultaneously: NCF-S, NCF-B and NCF-P. NCF-S is designed to maximize the applications’ worst case survivability (WCS) over failures of ToR switches and physical servers with a preference for spreading VMs of each application across separate racks and servers (Figure 2.1a). Specifically, an application’s ToR WCS is defined as the fraction of its VMs that survive a single worst case ToR switch failure [5]. In this scenario, we use the mean ToR WCS (across all tenants) as a representative metric for NCF-S. NCF-B is designed to minimize the mean link BW reservation, allocated using the hose model [30]. Using the hose model, each network link utilized by a tenant application divides the tenant application subtree into two components, and the bandwidth needed on this link for the tenant application is determined by multiplying the per-VM bandwidth required by the application and the number of VMs on the smaller of the two components, as done in [30]. Thus, NCF-B has a preference for consolidating VMs of the same application as close to one another as possible, e.g., placing on same server or rack (Figure 2.1b). NCF-P aims to minimize total power consumption by placing VMs on the fewest number of racks and servers, thus allowing unused resources to be powered down (Figure 2.1c). One candidate proposal of allocation can dominate, i.e., be strictly better than, another if it achieves better performance for at least one objective and no worse performance for each other objective. However, sometimes, two candidate proposals cannot be simply ranked against each other as each is better for a different objective; in this case, we say they are *nondominated* with respect to each other, which is the case for any pair of proposals in Figure 2.1.

An orchestrator at minimum must solicit and rank candidate proposals from all NCFs. Clearly, the network cannot execute all three proposals in Figure 2.1. Nor should it be asked to somehow accommodate each by time-slicing them. If an operator knows *a priori* how to jointly model the three objectives with a single ranking metric, the orchestrator may optimize the allocation based on the metric in order to find a “best compromise” solution for all the objectives. However, this approach places a heavy

burden on the operator to create the right ranking model for his/her network. More importantly, it is an open question whether a search based on such joint models can cover the potentially vast tradeoff space between the objectives. Compounding the problem is that the NCFs are supposed to come from third-party vendors, and thus are likely to be *black-box* solutions to the operator. The operator could conceivably collaborate with the NCF vendors to create “grey-box,” or even “white-box” solutions where he/she has access to the internal logic of the NCFs. While doing so may reduce the search space for finding an acceptable compromise, it remains a challenge for an orchestrator to adequately explore the multi-NCF tradeoff space for large network configurations.

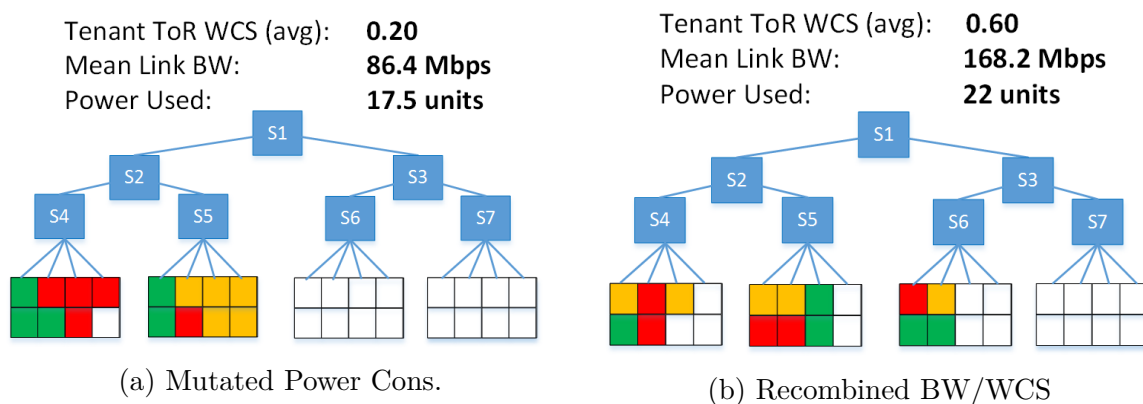


Figure 2.2: Mutated power conservation proposal and recombined BW conservation and survivability proposal.

In prior work, e.g., Athens [11] and Corybantic [10], all candidate solutions are exclusively generated by the NCFs. Assuming that each NCF generates one proposal, the initial population in the context of this example is limited to the three proposals depicted in Figure 2.1. It may be possible for each NCF to generate multiple proposals, but even so, it seems unrealistic to expect a specialized NCF to generate mutually beneficial compromises without knowledge or understanding of the performance criteria of the others. Corybantic and Athens then proceed by selecting the NCF proposal that maximizes some global utility function (e.g., votes, cost-effectiveness), and may iteratively solicit the NCFs for incremental counter-proposals to the previously selected allocation until the greedy criterion cannot further be improved by any NCF proposal.

The problem here is that even if a specialized NCF is able to make effective counter-proposals, which may itself be challenging, the region of the search space enumerated by such proposals is limited to what is reachable from the previously selected proposal. For instance, if the power conservation proposal depicted in Figure 2.1c is selected initially, then it may be the case that while the proposal in Figure 2.2a is reachable via a series of incremental counter-proposals, the proposal in Figure 2.2b may not be. As a result, the final proposal obtained may not be globally optimal, i.e., it may be dominated by another feasible, but unexplored allocation.

In contrast, we argue that mutation and recombination of proposals in an NCF-agnostic manner, *in addition* to NCF-specific heuristic mutations, and subsequent evaluation as a MOP comprised of distinct NCF performance criteria, is a more effective way to discover “globally optimal” compromises. A *mutation* is similar to a counter-proposal in [10, 11] in that it is a small-scale change to some previous allocation in an effort to guide the search towards a local optimum, whereas a *recombination* is large-scale change produced by combining desirable elements of two different states with the aim of exploring a new frontier of the state space to subsequently discover new local, and possibly global, optima. Furthermore, by maintaining a wide range of solution candidates and applying the concepts of natural evolution, i.e., performing mutations and recombinations of high-fitness candidates, a diverse set of nondominated allocation alternatives may be generated and presented to the network operator for consideration. If an operator requirements are fluid then this approach offers multiple prospective proposals for consideration as opposed to a single “best” one, which may be undesirable.

For instance, consider a potential mutation of the power conservation proposal, and a potential recombination of the survivability and BW conservation proposals, presented in Figure 2.2. Note that while both the mutated power conservation proposal (Figure 2.2a) and the recombined survivability and BW conservation proposal (Figure 2.2b) each dominate a predecessor state (Figures 2.1c and 2.1b, respectively), these proposals are nondominated with respect to one another. Although the proposal in Figure 2.2b offers better ToR WCS, it uses more power and BW than the proposal in Figure 2.2a. Since neither proposal dominates the other, *both should be maintained*

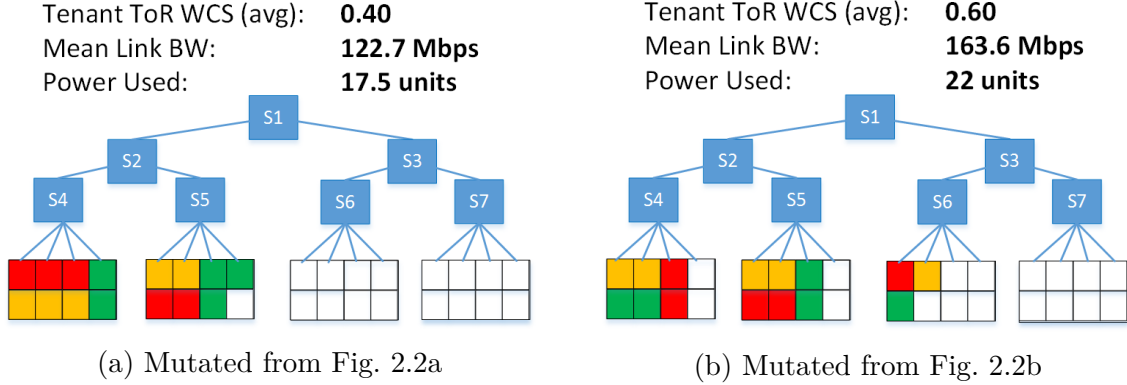


Figure 2.3: Two compromises from successive rounds of recombination and mutation of nondominated candidates.

as *desirable solution candidates*.

This is especially critical when the operator requirements are complex. The proposal in Figure 2.2b is more appropriate for mission critical applications that require maximum fault tolerance, while the proposal in Figure 2.2a is better suited towards general applications that do not require such a high level of availability. Depending on the criticality of the tenant applications R1, R2, and R3, the operator can easily implement either proposal. In contrast, [10, 11] would discard one of these desirable proposals, leaving the operator with only one proposal without presenting further desirable evolutions (as shown in Figure 2.3) to the operator.

2.3 Multi-Objective Optimization Formulation

In this section, we formally model VM allocation involving M tenant applications, N NCFs, and physical topology P , as a MOP. Let X represent the set of all possible allocations; each instance $x \in X$ captures a proposed allocation (e.g., for the Section II scenario, x is a 32-element vector describing which application, if any, occupies each of the VM slots of P). Let $X_f \subseteq X$, termed *feasible allocation set*, denote the subset of all allocations that can meet the requirements of all applications (e.g., quantity of VMs, intra-VM network bandwidth, etc).

Let $f_i()$ denote the utility function used to evaluate the goodness of an allocation

with respect to NCF i , $i = 1, \dots, N$ (e.g., ToR WCS for NCF-S, mean link BW for NCF-B, and total power usage for NCF-P). We believe this function will most likely be defined by the data center operator to account for local conditions, possibly with input from the NCF vendor. Therefore, for a proposed allocation $x \in X_f$, the *objective vector* $y = (f_1(x), f_2(x), \dots, f_N(x))$ precisely captures its operational merit from the perspectives of *all* objectives.

Instead of computing a weighted sum or using other techniques to reduce the objective vector to a single scalar metric and then finding one “best” allocation with respect to that metric, as done in prior work, we leverage the classical MOP literature [17] to look for a set of solutions that illuminates the entire trade-off space of the diverse objectives. First, we formally define when two allocations can be ordered in the N -dimension objective vector space, i.e., when one dominates the other, and when they cannot because they may be preferred for different objectives.

Def. 1. (Pareto Dominance [19]) For any two allocations $x_1, x_2 \in X_f$,

- (i) $x_1 \succ x_2$ (x_1 dominates x_2) **iff** $\exists i, f_i(x_1) > f_i(x_2)$
and $\forall j \neq i, f_j(x_1) \geq f_j(x_2)$.
- (ii) $x_1 \sim x_2$ (x_1, x_2 are nondominated w.r.t. each other)
iff $\exists i, j. i \neq j$ **and** $f_i(x_1) > f_i(x_2)$ **and** $f_j(x_2) > f_j(x_1)$.

This concept of Pareto dominance allows us to define the optimality criterion for the MOP formulation. If some allocation, x , is not dominated by any other allocation, then this means that x is optimal in the sense that it cannot be improved in any objective without causing degradation in at least one other objective. Such solutions are referred to as *Pareto-optimal* [19].

Def. 2. (Pareto Optimality [19]) An allocation x is *Pareto-optimal* regarding the feasible allocation set X_f **iff** $x \in X_f$ **and** for no other proposal $x' \in X_f$ is $x' \succ x$.

The entirety of all Pareto-optimal solutions is called the *Pareto-optimal set*, denoted by X_p ; the corresponding objective vectors form the *Pareto-optimal front* or *surface*,

denoted by Y_p . Now a MOP formulation of the VM allocation problem is simply as follows.

Def. 3. (MOP for SDN Orchestration)

***maximize** $y = (f_1(x), f_2(x), \dots, f_N(x))$, i.e., enumerate all Pareto-optimal solutions,*

***subject to** $x \in X_f$ **and** additional criteria such as lower bounds for $f_i(x)$'s.*

2.4 Evolutionary Approach

Prior MOP work [18, 31, 32] shows that an evolutionary approach (i.e., genetic algorithm), which keeps track of potential nondominated solutions and evolves (i.e., expands and improves) them via *mutation* and *recombination*, can ensure 1) suboptimal local maxima tend to be avoided, and 2) a wider range of solution candidates will be considered vs. a greedy approach. In this section, we present such an evolutionary algorithm, termed Evolutionary Algorithm for SDN Orchestration (EASO), to solve the MOP problem formulated in the previous section.

2.4.1 EASO Specification

A specification of the EASO algorithm is provided below. In addition to Pareto-dominance, fitness assignment in Step 2 is also based on crowding distance, which measures the uniqueness of a candidate solution with respect to other members in the set, as done in [31, 32].

Algorithm 2.1. *EASO*

Input: K : number of generations; L : external set size;
 P : physical topology tree; s : mutation size

Output: solution set X_s

Step 1: Initialization:

- a) Set initial population $A_0 = \emptyset$, $k = 0$
- b) Set initial external set $ES_0 = \emptyset$

- c) Solicit each of the N NCFs for its proposed allocation
 $x \in X_f$ and add x to A_0
- d) Add $(L - N)$ randomly generated allocations to A_0

Step 2: Fitness Assignment / Termination:

- a) Calculate fitness F of allocations in $A_k \cup ES_k$
- b) Derive nondominated feasible set X_s from $A_k \cup ES_k$
- c) If $k \geq K$ then return X_s

Step 3: Update of external set:

- a) If $|X_s| > L$ then
Remove $(|X_s| - L)$ worst fitness allocations from X_s
- b) Else if $|X_s| < L$ then
Add $(L - |X_s|)$ best fitness allocations of A_k to X_s
- c) Set $ES_{k+1} = X_s$

Step 4: Recombination:

- a) Set mating pool $MP = X_s$, child pool $CP = \emptyset$
- b) For each NCF i do
 - 1) Sort MP in ascending order of f_i
 - 2) For a in 1 to $\lfloor |MP|/2 \rfloor$ do
 - $b = (|MP| + 1) - a$
 - $x = \text{RECOMBINE}(MP[a], MP[b])$
 - Add x to CP

Step 5: Mutation:

- a) For each NCF i do
 - 1) For each allocation $x \in MP$ do
 - $u = \text{MUTATE}(x, s, i, \text{goal} = \text{"Improve"})$
 - $w = \text{MUTATE}(x, s, i, \text{goal} = \text{"Degrade"})$
 - Add u, w to CP

Step 6: Update Population:

- a) Set $A_{k+1} = CP$, $k = k + 1$; Goto Step 2

2.4.2 Evolutionary Primitives

In EASO, the **MUTATE** primitive procedure takes an NCF i as an input parameter, and uses an NCF-specific heuristic to attempt to relocate up to s VMs (mutation size) in order to improve (or degrade) the value of f_i . Although not strictly necessary, the degrade step is included in order to increase entropy and help to maximize the diversity of the candidate solution set. Because a tradeoff space is assumed, by intentionally degrading the utility of one NCF, another may benefit. For the example scenario described in Section 2.2, the following NCF-specific mutation heuristics are used within the **MUTATE** procedure. Here, we use the term *affinity* to refer to the number of VMs of a particular application residing in the same subtree.

- f_1 (ToR WCS) : 1) Identify the application m with the lowest value of ToR WCS. 2) Relocate up to s VMs of m from the highest affinity subtree of the physical topology to some number of lower affinity subtrees.
- f_2 (Bandwidth Conservation) : 1) Identify the application m with the highest BW usage. 2) Relocate up to s VMs of m from the lowest affinity subtree to higher affinity subtrees.
- f_3 (Power Conservation) : 1) Identify the application m using the highest number of racks (and servers in the case of a tie). 2) Remove up to s VMs of m from the lowest affinity subtree and replace them using a “first-fit” bin packing heuristic.

In contrast to **MUTATE**, the **RECOMBINE** primitive procedure is NCF-agnostic, and simply performs a merging of two input allocations by randomly selecting VM placements from each to form a new output allocation. To help encourage diversity during the recombination step, the mating pool MP is sorted in each dimension f_i , and for each sorting, each candidate solution is recombined with its counterpart at the opposite end of the f_i spectrum, i.e., first vs. last, second vs. second-to-last, etc.

2.4.3 Complexity Analysis

Ideally, the solution set X_s returned by EASO is equal to the Pareto-optimal set (denoted by X_p). However, the size of the feasible allocation set X_f , and hence the

time required to totally enumerate X_p , grows combinatorially with the number of switches and servers in the physical topology tree (denoted by $|P|$). For nontrivial values of $|P|$, e.g., the large topology used in Section 2.5.3, totally enumerating X_p may require prohibitively large EASO input parameters. In these cases, X_s is rather an inner approximation [18] of X_p , i.e., elements in X_s may be dominated by those in X_p .

Space Complexity: The maximum population size contains $|CP| = N * (L/2 + 2L) = \frac{5NL}{2}$ states, and each state contains $|P|$ elements, hence yielding a space complexity of $O(N \cdot L \cdot |P|)$.

Time Complexity: Each of the $\frac{5NL}{2}$ proposals are evaluated by N utility functions, and each utility function evaluates at most $|P|$ elements of each proposal, for a resultant complexity of $O(N^2 \cdot L \cdot |P|)$. Fitness assignment requires at most $O(N \cdot L^2)$ comparisons using the scheme presented in [32], **RECOMBINE** is called $\frac{NL}{2}$ times, and **MUTATE** is called $2NL$ times. Each call to **MUTATE** performs at most s VM reallocations, and the main algorithm loop runs K times. Hence, the total time complexity of this algorithm is $O(N^2 \cdot L^2 \cdot |P| \cdot K \cdot s)$.

2.5 Evaluation

In this section, we evaluate EASO using two topologies. First, the simplistic scenario described in Section II is used to illustrate that EASO can produce more diverse, and potentially better solutions than current orchestrators. Then, a relatively large topology [4] is used to illustrate that EASO scales to large data centers.

2.5.1 Performance Metrics

As discussed in Section 2.4.3, for nontrivial topologies, EASO may only (inner) approximate the Pareto-optimal set X_p . To evaluate the accuracy of this inner approximation (denoted by X_s), we propose two metrics: *distance* and *coverage* to compare X_s against X_p using their corresponding sets of objective vectors, i.e., images in the objective vector space. Specifically, let Y_s and Y_p denote the image sets of X_s and X_p , respectively. (When it is infeasible to obtain X_p and Y_p due to nontrivial values

of $|T|$, we use the “constraint method” well known in MOP literature [17] to first construct an outer approximation of Y_p , and then use it in place of Y_p in equations (2.1) and (2.2) to compute distance and coverage. See Section 2.5.3 for more details.) The distance of an objective vector $y \in Y_s$ is defined as follows.

Def. 4. (Distance)

$$distance(y, Y_p) = \min_{w \in Y_p} dist(y, w), \quad (2.1)$$

where $dist(y, w)$ represents the Euclidean distance between points y and w .

Once the distance of each point $y \in Y_s$ has been calculated, we calculate the *mean*, *min*, and *max* distances of points in Y_s to provide a set of distance measures representative of the solution set as a whole.

The current orchestrators, such as [10, 11] strive only to find a single solution that minimizes distance, without regard for the potentially vast tradeoff space. In contrast, a novel aspect of our approach is the enumeration of a set of nondominated tradeoffs. Hence, to evaluate the area of the tradeoff space covered by a solution set produced by EASO or similar algorithms, we propose the *coverage* metric (Def. 5), representing the fraction of points in the reference image set Y_p that are “covered,” i.e., nearest to objective vectors in Y_s . Hence, solution sets with higher coverage values are more desirable.

Def. 5. (Coverage)

$$coverage(Y_s, Y_p) = \frac{|\bigcup_{y \in Y_s} nearest(y, Y_p)|}{|Y_p|}, \quad (2.2)$$

where $nearest(y, Y_p) = \arg \min_{w \in Y_p} dist(y, w)$.

2.5.2 EASO vs. Current Orchestrators

In order to illustrate the unique merits of EASO vs. the current orchestrators, we developed GASO, a greedy version of EASO, to emulate methods proposed in [10, 11]. In [10, 11], the authors do not explicitly state the mutation heuristics used by independent NCFs to generate incremental counterproposals, but rather defer this issue

to future work, whereas the GASO mutation heuristics (identical to EASO, Section 2.4.2) explicitly specify how such counterproposals are suggested. Additionally, we enhance GASO to enumerate not just one solution, but a set of solutions, as described later in this section.

GASO has four notable differences vs. EASO: 1) the recombination step is omitted, 2) the Pareto-based fitness function F is replaced with the global objective function $F_{global}(x) = w_1(f_1(x)) + w_2(f_2(x)) + \dots + w_N(f_N(x))$ where $\vec{w}$ is an N -dimensional NCF weight vector and w_i represents the weighting value for NCF i , 3) the external set ES contains only a single member ($L = 1$): the solution candidate with the highest value of F_{global} , and 4) the algorithm terminates when no NCF-specific mutation of the external set member yields a higher F_{global} value.

To compare GASO to EASO, we generated a comparable set of GASO solutions for the Section 2.2 scenario by way of parametric analysis over a set of fixed aspiration levels (lower bounds) for f_1 (WCS), and different weightings for f_2 (BW) and f_3 (power). For each aspiration level of f_1 , $f_1 \geq 0.00, 0.066, \dots, 0.594$; we used two different weightings: $\vec{w} = (1, 4, 2)$, which clearly favors f_2 over f_3 , and $\vec{w} = (1, 2, 4)$, which conversely favors f_3 over f_2 . f_1 maintains a minimum weighting here, as the aspiration levels force an enumeration over the its range.

For EASO, we set the size of the external set $L = 25$, the number of generations $K = 25$, and the mutation size $s = 5$. EASO consistently enumerated all 14 Pareto-optimal solutions² for *each* of 100 simulation³ runs, represented by the X_s^{EASO} solution set (Table 2.1).

For GASO, we performed multiple runs via parametric analysis, across the range of *all* mutation sizes ($s = 1, 2, \dots, 15$). The resulting set of solutions, X_s^{GASO} , represents the best solutions produced by GASO throughout *all* 270 simulation runs. GASO was only able to enumerate six of the fourteen distinct Pareto-optimal states (Table 2.1).

²In this simplistic scenario, we were able to enumerate the entire Pareto-optimal front Y_p (14 solutions) via brute force enumeration, and hence used Y_p as a basis of comparison for EASO and GASO.

³The simulation consists of approximately 2500 lines of Java code, and was run on a Linux VM allocated 8GB of RAM and 2 x vCPUs. The host PC (laptop) was running 64-bit Windows on an Intel 2.4 GHz quad-core processor with 12 GB of RAM.

	X_s^{EASO}		X_s^{GASO}	
	Allocation	(WCS, BW, Power)	Allocation	(WCS, BW, Power)
1	[(5,0,0), (0,5,0), (0,0,5), (0,0,0)]	(0.000, 68 Mbps, 23 units)	[(5,0,0), (0,5,0), (0,0,5), (0,0,0)]	(0.000, 68 Mbps, 23 units)
2	[(4,0,0), (1,5,0), (0,0,5), (0,0,0)]	(0.067, 72 Mbps, 22 units)	[(3,4,0), (2,1,5), (0,0,0), (0,0,0)]	(0.200, 86 Mbps, 17.5 units)
3	[(3,5,0), (2,0,5), (0,0,0), (0,0,0)]	(0.133, 77 Mbps, 17.5 units)	[(3,4,1), (2,1,4), (0,0,0), (0,0,0)]	(0.267, 100 Mbps, 17.5 units)
4	[(2,5,0), (2,0,0), (1,0,5), (0,0,0)]	(0.200, 84 Mbps, 22 units)	[(3,3,1), (2,2,4), (0,0,0), (0,0,0)]	(0.333, 109 Mbps, 17.5 units)
5	[(3,4,0), (2,1,5), (0,0,0), (0,0,0)]	(0.200, 86 Mbps, 17.5 units)	[(3,3,2), (2,2,3), (0,0,0), (0,0,0)]	(0.400, 123 Mbps, 17.5 units)
6	[(2,4,0), (2,1,5), (1,0,0), (0,0,0)]	(0.267, 93 Mbps, 22 units)	[(2,2,3), (2,2,2), (0,0,0), (1,1,0)]	(0.533, 143 Mbps, 22 units)
7	[(3,4,1), (2,1,4), (0,0,0), (0,0,0)]	(0.267, 100 Mbps, 17.5 units)	[(2,2,1), (1,1,2), (1,1,1), (1,1,1)]	(0.600, 191 Mbps, 25 units)
8	[(2,3,0), (2,2,0), (1,0,5), (0,0,0)]	(0.333, 102 Mbps, 22 units)		
9	[(3,3,1), (2,2,4), (1,1,1), (0,0,0)]	(0.333, 109 Mbps, 17.5 units)		
10	[(3,3,2), (2,2,3), (0,0,0), (0,0,0)]	(0.400, 123 Mbps, 17.5 units)		
11	[(2,3,1), (2,2,4), (1,0,0), (0,0,0)]	(0.400, 116 Mbps, 22 units)		
12	[(3,5,0), (2,0,5), (0,0,0), (0,0,0)]	(0.467, 130 Mbps, 22 units)		
13	[(2,2,3), (2,2,2), (1,1,0), (0,0,0)]	(0.533, 143 Mbps, 22 units)		
14	[(2,2,2), (2,2,2), (1,1,1), (0,0,0)]	(0.600, 164 Mbps, 22 units)		

Table 2.1: X_s^{EASO} and X_s^{GASO} nondominated solutions. The “Allocation” column represents the allocation of VMs to servers on the four different racks, e.g., [(3,5,0), (2,0,5), (0,0,0), (0,0,0)] represents the assignment of 3 VMs of R1 and 5 VMs of R2 to Rack 1, 2 VMs of R1 and 5 VMs of R3 to Rack 2, and none to Racks 3 and 4.

	Y_s^{EASO}	Y_s^{GASO}
Distance (Mean, Min, Max)	(0, 0, 0)	(0.014, 0, 0.097)
Coverage	1	0.5
Execution Time (seconds)	3.73	0.45

Table 2.2: EASO vs. GASO in distance and coverage of their solution sets w.r.t. Y_p , and in avg. execution time.

Note that X_s^{GASO} alloc. #7, although nondominated with respect to X_s^{GASO} , is *not* Pareto-optimal, as it is dominated by X_s^{EASO} alloc. #14. Moreover, X_s^{GASO} contains four additional dominated solutions not displayed in Table 2.1. These suboptimal solutions show that GASO was often stuck in local maxima.

Table 2.2 presents a comparison between Y_s^{EASO} and Y_s^{GASO} in terms of the metrics presented at the beginning of this section, using Y_p as the reference set. The solu-

tion set produced by EASO has smaller distance and higher coverage ratio GASO. These results demonstrate that EASO yields a wider range of, and potentially better solutions than the SOP orchestrators in [10, 11]. Furthermore, and perhaps the most distinguishing feature of EASO, is how well it enumerates the tradeoff space.

To illustrate this point, again consider Table 2.1, representing the nondominated solutions returned by EASO and GASO. Now suppose a network operator using GASO decides that GASO alloc. #5 (equiv. to EASO alloc. #10) is most appropriate for his/her needs, because it offers the best compromise between BW and WCS. However, EASO alloc. #11 is a better compromise, as it offers the same level of WCS as GASO alloc. #5, but even better BW, at the expense of power. Moreover, the diverse EASO solution set allows an operator to program the orchestrator to automatically select an allocation based upon the prevailing network conditions. For example, run EASO alloc. #11 during peak hours to conserve BW, and EASO alloc. #10 during non-peak hours to save power.

2.5.3 Is EASO Scalable?

To evaluate its scalability, we simulated EASO on a large-scale, multi-tier application data center scenario similar to the one presented in [4], but with the additional third objective of power conservation (adjusted for various host/ToR power consumption ratios). Specifically, we ran EASO on a simulated physical infrastructure consisting of 40 aggregation switches, 160 ToR switches (4 x ToRs per aggregate), 2560 hosts (16 x hosts per ToR), and 40960 VM slots (16 x VM slots per host), for the following 5-tier application requirements: T1: <40 x 4, 10 Mbps>, T2: <40 x 1, 100 Mbps>, T3: <40 x 2, 50 Mbps>, T4: <40 x 1, 100 Mbps>, T5: <40 x 4, 10 Mbps>. Here the first element in each tuple represents the number of VMs and slots required per VM, e.g., <40 x 4, 10 Mbps> denotes 40 VMs requiring 4 slots and 10 Mbps BW each. The NCFs remain the same as presented in Section 2.2.

At this scale, totally enumerating Y_p is intractable [4]. Therefore, we constructed an outer approximation (OA) of Y_p based on the well known “constraint method” found in MOP literature [17]. Specifically, we formulate each ordered pair of NCF utility functions, (f_1, f_2) , (f_1, f_3) , (f_2, f_1) , (f_2, f_3) , (f_3, f_1) , (f_3, f_2) as a biobjective op-

	$\mathbf{Y}_s^{EASO}$			$\mathbf{Y}_s^{GASO}$
	$(L = 25, K = 25, s = 2)$	$(L = 50, K = 50, s = 4)$	$(L = 75, K = 75, s = 6)$	$(f_1 \geq .005, .010, \dots .975) (s = 1, 2, \dots, 40)$
Distance (Mean, Min, Max)	(0.122, 0.0, 0.241)	(0.094, 0.0, 0.248)	(0.047, 0.0, 0.174)	(0.197, 0.0, 0.244)
Coverage	0.030	0.059	0.089	0.033
# of Nondominated Solutions	83	109	197	43
Execution Time (seconds)	69	566	5059	1184

Table 2.3: Performance of three runs of EASO vs. GASO for large scenario. Best, moderate, and worst results are shaded green, yellow, and red, respectively.

timization problem (BOP) where the first utility function is cast as a discrete set of lower bounds (e.g., C_{f_1} for f_1 where $C_{f_1} = \{0.005, 0.010, \dots, 0.975\}$), and the second is maximized for each. The OA then consists of all points (c_{f_1}, f_2, f_3) where f_2 and f_3 are separately maximized for each $c_{f_1} \in C_{f_1}$, and so on for (f_1, c_{f_2}, f_3) and (f_1, f_2, c_{f_3}) . Note that tractably constructing a tight OA for nonlinear BOPs is a challenging problem in and of itself [33].

Table 2.3 illustrates the performance of EASO with respect to OA for different sets of input parameters⁴. Observe that there is a clear tradeoff between time and optimality. As the size of input parameters (L, K, s) increase, EASO produces better and more diverse⁵ solution sets at the cost of increased execution time. The “short-run” parameter set $(25, 25, 2)$, completes in just over a minute, hence most appropriate for network operators with rapidly changing tenant application requirements. In contrast, the “long-run” parameter set $(75, 75, 6)$ takes over an hour to complete, and thus may be warranted for steady state data center operations where network configurations are unlikely to change frequently. Finally, the “medium-run” parameter set $(50, 50, 4)$ finishes in under ten minutes, and represents a reasonable compromise between agility and quality. The increase of EASO execution time (last row of Table 2.3) corroborates the time complexity analysis in Section 2.4.3.

⁴For comparative purposes, we ran a fine-grained parametric analysis of GASO over f_1 using a range of mutation sizes. Note that GASO performed worse than the EASO “medium-run” parameter set in every category, including execution time.

⁵Because the size of OA is very large (843 solutions), coverage should be viewed as a relative metric, as obtaining high absolute coverage values is not possible for relatively small values of L .

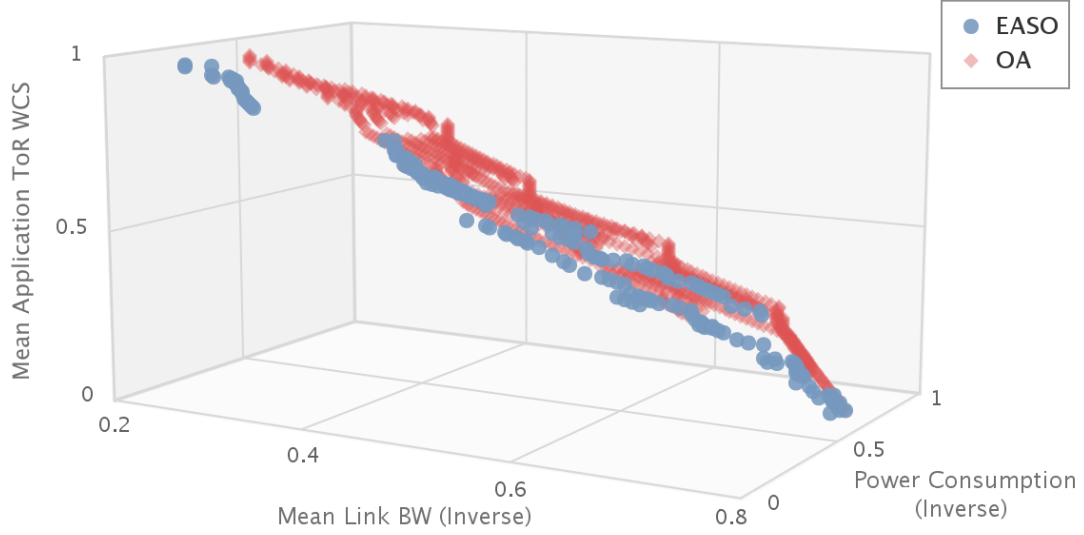


Figure 2.4: EASO Nondominated Front vs. Outer Approximation (OA) of Y_p , for a large-scale data center scenario.

Figure 2.4 depicts the EASO “long-run” solution set vs. OA for the large-scale data center scenario. From this figure, we can see that the EASO solution set is well spread and relatively close regarding OA. Also realize that the EASO solutions are *at least as close* to Y_p as OA, since points in OA are not necessarily feasible.

2.6 Conclusion

We have demonstrated that that our proposed evolutionary approach can enumerate a wider range of, and potentially better solutions than, current orchestrators for relatively large data center networks.

For future work, we find several areas intriguing. The mutation and recombination evolutionary primitives may be further refined and adapted for other orchestration tasks, such as traffic engineering, risk management, or cybersecurity. For example, in [16], one specialized mutation procedure is used to select alternate routing paths between network services. Fine-tuning the tradeoff space based on operational requirements and automated decision making with respect to the tradeoff space are other promising areas, e.g., how to enumerate a relevant subset of the tradeoff space in less time, or how to select the best EASO candidate solution given prevailing network conditions.

CHAPTER 3:

An Application-Aware Approach to Scalable Online Placement of Data Center Workloads

This chapter focuses on the challenges of achieving application-aware resource allocations that not only provide desirable outcomes in terms of operator objectives, but are also desirable for minimizing resource contentions both within and between application workloads. Moreover, we show that by precomputing ideal workload allocations in a contention-free environment, such as a simulated empty topology, the resultant logical application workload (LAW) may be used as a single atomic unit for allocating a workload. This is in contrast to the tens to hundreds of workload VMs that would typically require individual per VM allocation. As a result, not only are the corresponding LAW allocations largely free of application contentions vs. existing per VM methods, but also by using the larger-grained LAW allocation algorithms, workload placement can be accomplished one order of magnitude faster than current solutions, achieving average allocation times of less than a second for data center networks with hundreds of switches and thousands of VMs.

Data center operators strive for maximum resource utilization while satisfying tenant service level agreements; however, they face major challenges as application workload types are diverse and tenants add, remove, and update their workloads sporadically to meet changing user demands. Currently, operators allocate workload VMs primarily in an application agnostic fashion, focusing on minimizing total resource usage. In this work, we first show that such allocations can be suboptimal and then present a new application-aware approach that explicitly models resource preferences of individual workloads. Further, we propose a new LAW abstraction to enable precomputation of the required relative positioning of an application’s VMs and allocation of these VMs in a single atomic step, leading to online algorithms that are one order of magnitude faster than existing per VM placement solutions. We then develop a statistical extension of LAW to add flexibility in characterizing application requirements and to support prioritization of workloads. Using realistic workload traces and physical

topologies, we evaluate our algorithms in a simulated large-scale data center setting, and demonstrate their performance advantages and potential tradeoffs versus existing solutions.

3.1 Introduction

The data center operators of large enterprises and public cloud providers face major challenges in allocating virtual machines (VMs) to individual application workloads. First, the workloads are diverse in sizes (measured by the number of VMs required, the amount of link bandwidth required, running time, etc.), in resource consumption characteristics (e.g., computation intensive vs. data intensive), and in performance requirements such as bounds on response times and application availability [34]. Second, the increased flexibility and agility provided by software-defined data center networks allows enterprises and public cloud tenants to rapidly add, remove, update, and prioritize their workloads to meet sporadically changing user demands. There is an urgent need for online allocation algorithms that can deal with such dynamic behaviors without incurring too much latency [24,27]. Last, but not least, the operators must strive for efficient uses of their physical resources (link bandwidth, electrical power, etc.) in order to minimize cost while meeting the quality of service requirements of applications.

Maximizing resource utilization within data centers is a problem that many prior efforts have attempted to solve using various approaches, including integer linear programming [23], greedy heuristics [3,4,10,11], and genetic algorithms [16,28]. However, we observe that this body of work overwhelmingly strives to optimize *cumulative* resource usage among all hosted tenant applications while deferring the *per application* performance concerns to relatively low level mechanisms such as CPU scheduling and traffic engineering, despite recent studies demonstrating that *different types of workloads contend for different types of resources* and consequently how VMs of an application are *relatively positioned* can significantly impact the performance of an application [13,14]. Specifically, it would be advisable not to co-locate VMs of computation intensive workloads to avoid unnecessary CPU contention while at the same time, position VMs of the same data intensive workload as close as possible to reduce both bandwidth contention and communication latency [14].

In other words, while workload placement should concern about cumulative resource usage, additional opportunities exist to intelligently place the VMs of individual workloads to improve per application performance of all hosted applications. In this paper, we investigate a placement strategy, which we term “application-aware”, to leverage such opportunities. More specifically, we explicitly model the resource preference of each workload and develop a unified framework to characterize and minimize resource contentions introduced by a new workload, which can be between VMs of the same workload as well as with respect to existing workloads. Further, we enhance the application-aware approach by introducing a new logical application workload (LAW) abstraction. The LAW represents the most desirable relative positioning of the workload’s VM placement in a physical topology in terms of *both* meeting operator specific resource efficiency goals *and* minimizing resource contentions. We show that a data center controller can precompute LAWs and subsequently assign VMs of each workload in a single atomic step, leading to online algorithms that are one order of magnitude faster than current per VM placement solutions.

Our contribution is multi-fold as follows.

1. First, we show that application agnostic workload placement can introduce unnecessary resource contentions, and demonstrate potential performance gains from precisely modeling resource contentions that can be introduced by a workload. (Section 3.2)
2. Second, we formally define LAW and describe how to construct LAWs that minimize potential resource contentions. We then design a range of bin packing heuristics to place workloads on a per LAW basis and at the same time optimize different cumulative resource usage objectives. (Section 3.3)
3. Third, we observe that LAW based workload placement can be infeasible sooner than per VM placement. Consequently, we propose a statistical extension of the LAW abstraction to add flexibility in characterizing application requirements. We show that the extension permits a data center operator to increase LAW placement feasibility via graceful degradation of application performance. (Section 3.4)
4. Fourth, we show that the statistical LAW extension provides a natural mechanism to support prioritization of workloads, conceptually similar to the Random Early Detection (RED) queueing [35]. (Section 3.5)

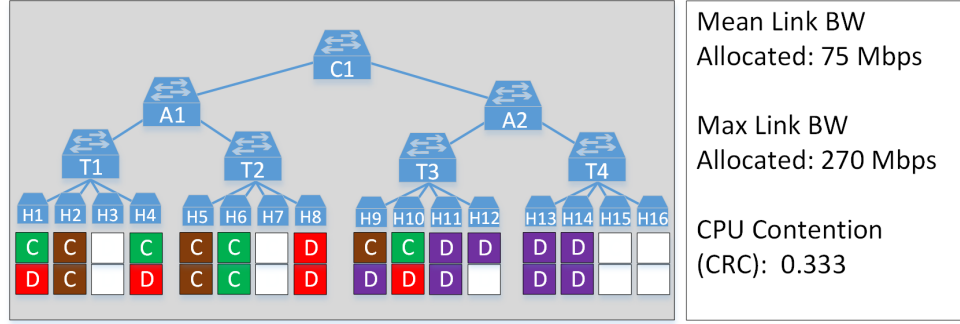
5. Fifth, using a simulated scenario with realistic workload traces and a relatively large data center topology, we evaluate the LAW based workload placement heuristics and demonstrate their performance advantages and potential tradeoffs versus existing solutions. (Sections 3.4, 3.5, and 3.6)

3.2 Application-Aware Placement

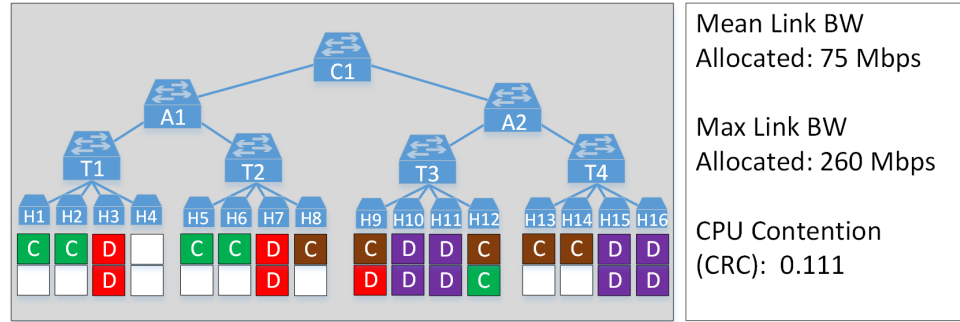
We motivate and illustrate the advantages of application-aware placement with a simplistic but telling scenario. We also describe how to model a workload’s resource preference and present metrics to characterize resource contentions introduced by a workload.

First, from prior work we observe that there are two basic types of application workloads: 1) CPU or memory intensive workloads (e.g., meteorological, geological, and particle physics simulations, or other high performance computing tasks), commonly referred to as compute-intensive (CI) workloads, and 2) network or storage intensive workloads (e.g., client-server or Hadoop and other big data applications), commonly referred to as data-intensive (DI) applications. Generally, state-of-the-art research [13, 14] concludes that CI workloads perform better when the processes and VMs are spread across *separate* CPU cores and hosts, respectively, while DI workloads perform better when the VMs (or processes) are placed on the *same* host (or CPU core). For example, by placing all VMs of a client-server (DI) application on the same physical host, tenfold increases in throughput have been observed [13]. In contrast, over-contention of CPU resources by VMs of CI applications have been shown to increase job completion time by as much as 260% [13].

Therefore, we propose to extend the current workload representations, which primarily specify the number of VMs and intra-VM bandwidth [10, 11, 30], to also include a designation of application type (CI or DI). Such a classification may be specified a priori by tenants, or determined post-allocation by the operator upon monitoring workload resource usage in practice. In addition, we believe it is advantageous for a data center to model the survivability requirement (i.e., resiliency against single top-of-rack (ToR) switch failures) on a per application basis because typically, differ-



(a) Application Agnostic Allocation



(b) Application Aware Allocation

Figure 3.1: An example contrasting application-aware vs. application agnostic VM allocations. Green slots are for VMs of R1, red R2, brown R3, and purple R4. “C” and “D” denote a CI and DI workload, respectively. The allocation on the right considers application type (CI or DI) of each workload, and thus attempts to spread VMs of R1 and R3 (CI) to minimize CPU contention, while consolidating VMs of R2 and R4 (DI) to minimize communication overhead.

ent applications have different levels of importance to the tenants. Commonly, the survivability requirement is represented by a metric called worst-case survivability (WCS), which is the maximal possible fraction of VMs taken offline due to a single ToR failure [5].

Now consider the following simplistic scenario. Suppose four independent tenant application workloads R1, R2, R3, R4, arrive in sequential order and have requirements of $\langle \text{CI}, 5, 10 \text{ Mbps}, 0.5 \rangle$, $\langle \text{DI}, 5, 50 \text{ Mbps}, 0.5 \rangle$, $\langle \text{CI}, 5, 10 \text{ Mbps}, 0.5 \rangle$, and $\langle \text{DI}, 8, 50 \text{ Mbps}, 0.5 \rangle$ respectively. The first value in the tuple represents the application type (appType), the second value represents the number of VMs (numVMs) required,

the third value represents the inter-VM bandwidth (BW) reservation requirement⁶, and the fourth value represents the minimum required WCS⁷. Suppose the underlying physical infrastructure has a hierarchical tree topology, consisting of one core switch as root, two aggregation switches, four ToR switches, four host servers per ToR switch, and two VM slots per host server, where a “VM slot” is defined as a standard physical resource unit (e.g., CPU, Memory) provisioned to a VM. Therefore, the VMs are to be allocated against $4 \times 4 \times 2 = 32$ possible slots.

Current solutions, such as [3, 4, 10, 16], overwhelmingly attempt to maximize resource utilization by finding a “best fitted” allocation of tenant application workloads that minimizes some weighted global cost function associated with data center resource usage, and more specifically, bandwidth usage. In theory, such approaches seem effective in maximizing global resource utilization. However, because global cost functions are not capable of representing fine-grained *contention points* of independent tenant application workloads (e.g., CI or DI), the “globally optimal” network allocations produced by current solutions are *unlikely to be optimal for each tenant application comprising it*. This situation is illustrated in Figure 3.1. The left part (3.1a) depicts a best-fitted allocation representative of current solutions, which minimizes network-wide bandwidth usage while satisfying application requirements R1, R2, and R3. However, note that CI applications R1 and R3 are in contention with themselves for CPU and memory on hosts H2, H5, and H6, and DI applications R2 and R4 do not achieve optimal network/storage sharing benefit via host colocation: R2 misses an opportunity for colocation under rack T1 and R4 misses a colocation opportunity under rack T3. Figure 3.1b, in contrast, depicts an *application-aware* best-fitted allocation, which not only minimizes network-wide bandwidth usage, but also *provides optimal conditions for each tenant application*. Although CI applications R1 and R3 compete with each other on H12, they do not compete with themselves on any host, and DI applications R2 and R4 achieve maximum intra-host colocation.

The example in Figure 3.1 also shows that application-aware allocation does not always degrade each global utility metric. Here, the application-aware allocation

⁶Inter-VM bandwidth is reserved according to the “hose” model, as done in [30].

⁷Prior work in datacenter network orchestration [10, 11] asserts that a WCS of 0.5 should be the bare minimum acceptable to the network operator.

method achieves less link congestion (i.e., smaller max BW) by using more physical servers to host the VMs, a tradeoff that typically results in more power usage.

The insight that different types of tenant workloads have different host assignment preferences motivates the need for a metric to capture such preferences. To this end we propose a new Resource Contention Index (RCI) to evaluate the placement of individual tenant applications (Def. 6).

Def. 6. (**RCI** for placement of workload w)

$$RCI(w) = \begin{cases} 0 & \text{workload requires 1 VM} \\ Host_Affinity(w) & \text{workload is CI} \\ 1 - Host_Affinity(w) & \text{workload is DI} \end{cases}$$

where:

$$Host_Affinity(w) = 1 - \frac{(\# \text{ of hosts hosting VMs of } w) - 1}{(\text{total VMs allocated for } w) - 1},$$

and $Host_Affinity()$ is defined only when the total number of VMs allocated for w is greater than 1.

The host affinity for some tenant application represents the degree by which its VMs are intra-host colocated. A value of ‘0’ indicates that no application VMs share the same host, and thus minimizes intra-application computational contention for a CI workload, whereas a value of ‘1’ indicates that all application VMs share the same host, hence maximizing intra-application data throughput for a DI workload. RCI captures this fundamental difference in application preference between CI and DI workloads in a single application-specific metric. Furthermore, we can take the average (mean) RCI across all allocated workloads to provide a measure of global application efficiency. Note that achieving an RCI of zero for all workloads is not realistic because tenants may have competing objectives or constraints. Survivability for instance, is directly at odds with RCI for DI workloads, as a host affinity of ‘1’

implies $WCS = 0$. But having zero survivability is not acceptable for most tenants. As another example, consider tenant budget constraints for a CI workload. It may be cost prohibitive for a tenant to place each workload VM on a separate physical host or CPU core. In such cases, a tenant may choose to settle a higher intra-VM contention ratio (i.e., $RCI > 0$) in the interest of lower SLA costs.

A useful adaptation of RCI, termed *computational resource contention* (CRC), is presented in Def. 7.

Def. 7. (Computational Resource Contention (**CRC**))

$$CRC = 1 - \frac{(\# \text{ of hosts hosting CI VMs}) - 1}{(\text{total CI VMs allocated}) - 1}, \quad (3.1)$$

and *CRC* is defined only when the total CI VMs allocated is greater than 1.

This metric, by treating all CI applications as one group, represents the degree of contention for computational (CPU/Memory) resources throughout the network. We argue that CRC should be added to the collection of global cost functions used for workload placement, as this metric is particularly relevant as an indicator of throughput bottlenecking for CI applications, similar in nature and importance to the commonly used link congestion metrics of mean and maximum bandwidth usage for DI applications. For the example depicted in Fig. 3.1, the application-aware allocation on the right achieves a much lower CRC than its counterpart.

3.3 LAWs for Application Efficiency

To overcome the shortcomings of current, non-application-aware workload placement approaches, we propose a new logical application workload (LAW) abstraction, which serves as the fundamental unit of allocation for an application workload, as opposed to individual VMs, in order to speed up the allocation and *explicitly preserve* the relative positioning of intra-application VMs with respect to the best allocation the operator would desire to obtain for the application on a simulated empty physical infrastructure.

3.3.1 Natures of LAW

Essentially, a LAW represents the best possible allocation for an application workload from the perspective of the operator with respect to a given physical infrastructure. It is *specific* to the allocation method used by the operator to meet the application’s resource and performance requirements, and as such it unambiguously captures the design intent of the operator.

Formally, we define a LAW as follows.

Def. 8. (Logical Application Workload (LAW)) *Given an requirement specification R for workload w , a data center infrastructure P to host VMs of w , and a VM allocation method M used by the operator, a LAW $L(w, P, M, R)$ is a hypothetical allocation for w by applying M on an empty P subject to requirements R .*

When the physical infrastructure has a tree topology, a LAW can be simply modeled as the subtree that contains the hypothetically allocated VMs, with additional annotations of required resources. For the application requirement tuple considered in this paper, i.e., the four-tuple of (appType, numVMs, BW, WCS), each switch of the subtree should be annotated with two parameters: r_slots for the total number of VMs supported underneath, and r_bw for the total amount of bandwidth reservation required on its upstream link. The two parameters are driven by the number of VMs and number of computational resource units (i.e., slots) required by each VM, and the BW requirements, while the relative positions of the VMs should minimize RCI while meeting the WCS bound.

When the physical resource capacity is much larger than what the workload requires, many subtrees can accommodate the LAW, i.e., support the relative positioning of the VMs. In such a case, we model the LAW using the leftmost subtree. Similarly, before the physically infrastructure is heavily utilized, allocation of a LAW should be straightforward, involving a small number of checks of feasibility against the r_slots and r_bw parameters. In other words, heuristics that allocate a workload on the basis of its LAW should run faster than their per-VM counterparts in most scenarios and therefore should be *more suitable as an online solution*.

Intuitively, compared to a finer-grain per-VM allocation, a LAW based allocation method may trade off some network-wide performance such as power efficiency and bandwidth utilization in order to maximize the quality of service of individual applications. However, the former has its own problem as it allocates VMs sequentially and as such VMs allocated earlier in the sequence may prevent the overall allocation from maximizing global utility. An investigation of this interesting tradeoff will be presented in Section 3.4. In the rest of this section, we focus on how to leverage the concept of LAW to create an online VM allocation solution that can respond to dynamic workload input.

3.3.2 Constructing Efficient LAWs

While the existing application agnostic solutions [3, 4, 10, 10, 11, 16, 28] can be directly used to create LAWs, we propose to extend these solutions to consider application characteristics (e.g., CI vs. DI) by applying Algorithm 3.1 post-allocation in order to obtain LAWs with the *smallest RCI possible while meeting other application requirements*. This is accomplished by both a) spreading the CI VMs of each rack over the maximum number of available hosts, and b) concentrating the DI VMs of each rack onto the smallest possible number of physical hosts. Specifically, as presented in Algorithm 3.2, the procedure CONSTRUCT-LAW(P_0, N) first runs an existing per VM allocation scheme on an empty physical topology (simulated), and then calls Algorithm 3.1 on the same input, which, depending on the application type, rearranges some of the VMs to minimize RCI while satisfying other requirements. The selected allocation is then converted into a LAW by (i) removing all elements of the physical infrastructure tree not used by the application, and (ii) tallying up the number of descendant VMs (r_slots) and the reserved upstream bandwidth (r_bw) parameters for each of the remaining nodes.

It is important to note that a data center controller can pre-construct LAWs for expected workloads⁸ against available infrastructures and store the results (LAW subtrees) in a hash table. This way, the online allocation algorithm (presented next) will need minimum time to obtain the LAW for a new workload request.

⁸For the scope of this paper, store all combinations of elements of each of the discrete ranges of numVMs, intra-VM BW, and WCS for both CI and DI app types.

Algorithm 3.1 “App-Aware Allocation Adjustment”

```
1: procedure APP-AWARE-ADJUST( $w, R, P$ )
2:                                     ▷ input  $w$ : workload id
3:                                     ▷ input  $R$ : requirement spec. for  $w$ 
4:                                     ▷ input  $P$ : physical topology; allocated
5:   if  $R.type = CI$  then                                     ▷ Minimize host affinity
6:     for each ToR  $t \in P$  do
7:       Spread VMs of  $w$  across hosts in  $t$ 
8:       while meeting BW and WCS requirements
9:     end for
10:  else if  $R.type = DI$  then                                     ▷ Maximize host affinity
11:    for each ToR  $t \in P$  do
12:      Concentrate VMs of  $w$  onto hosts in  $t$ 
13:      while meeting BW and WCS requirements
14:    end for
15:  end if
16: end procedure
```

Algorithm 3.2 “LAW Construction”

```
1: procedure CONSTRUCT-LAW( $w, R, P_0$ )
2:                                     ▷ input  $w$ : workload id
3:                                     ▷ input  $R$ : requirement spec. for  $w$ 
4:                                     ▷ input  $P_0$ : physical topology; empty
5:   PER-VM-ALLOCATION( $w, R, P_0$ )
6:   APP-AWARE-ADJUST( $w, R, P_0$ )
7: end procedure
```

3.3.3 Allocating LAWs

Allocating a LAW to a physical infrastructure (which may or may not be empty) needs to meet a set of global objectives defined by the network operator. These global objectives have traditionally included minimizing power usage [6] and link congestion [16], but we argue that the CRC objective (Def. 7) defined in Section 3.2 should also be considered.

Conceivably, an algorithm to explore the efficient frontier of LAW allocations with respect to the global objective space could be developed, for instance an algorithm similar to EASO [28] that takes a set of LAWs as input and produces a set of efficient

proposed allocations as output, where each satisfies all LAWs. However, we defer this challenge to future work, and instead focus efforts on the more fundamental, low-level problem of allocating each individual LAW in a manner that is both fast and resource efficient, with the goal of minimizing some global cost function. Specifically, we consider the single-objective cost functions of 1) power usage, 2) link congestion (i.e., mean and max BW usage), and 3) CRC.

In the remainder of this section, we describe a general algorithm for allocating a LAW to the physical infrastructure, and then discuss the use of different heuristics for minimizing the global cost functions described previously. Because we are allocating LAWs, i.e., sets of VMs that are placed with strict relative positioning requirements, we cannot directly apply heuristics for individual VM placement, as done in [36, 37].

Thus, we propose Algorithm 3.3, which takes the precomputed LAW L for an application workload (assumed to satisfy all requirements R of workload w), a physical network state P (i.e., physical infrastructure with current allocation), and a cost-minimizing heuristic h as input, and attempts to find a feasible allocation for the workload. Specially, we consider three heuristics in this paper: (1) “Min Power”: referred to as *best-fit* or *tightest-fit* in the classic bin packing problem literature [36], it seeks to map VMs of L onto hosts with the smallest number of free slots, effectively minimizing the number of active hosts; (2) “Min BW”: similar to the *max-rest* bin packing heuristic [36], it seeks to map VMs of L onto hosts with the largest number of free slots, effectively balancing bandwidth allocation and minimizing BW congestion (i.e., maximum link BW); (3) “Min CRC”: designed to spread CI workloads as evenly among hosts as possible while attempting to place DI workloads within the same subtree as previously allocated CI workloads (to allow maximum CI workload distribution). While the underlying approach of Algorithm 3.3 is generally applicable to tree-based data center topologies including Fat-Tree and VL2, for ease of exposition, we assume that P has a simple tree structure.

At the heart of LAW based allocation is to map each LAW node (ln) to a unique physical node (denoted by $ln.pn$) at the same tree level that has sufficient resources to support the requirements of $ln.r_slots$ and $ln.r_bw$. After considering several different tree search algorithms to explore the feasible $ln \rightarrow ln.pn$ search space, in-

Algorithm 3.3 “Online LAW Allocation”

```
1: procedure ALLOCATE-LAW( $L, P, h$ )
2:                                      $\triangleright$  input  $L$ : LAW,  $P$ : physical topology
3:                                      $\triangleright$  input  $h$ : cost-minimizing heuristic method
4:   for each level  $l$  from “ToR” to “core” do
5:     for each LAW node  $ln \in L$  at level  $l$  do
6:       for each physical ToR  $pn \in P$  at level  $l$  do
7:         if ASSIGN-MIN-COST( $ln, pn$ )  $< \infty$  then
8:           Save best matching of  $ln.children$ ,
              $pn.children$  for  $(ln, pn)$ 
9:         end if
10:      end for
11:      if no saved matchings for  $ln$  then
12:        return false
13:      end if
14:    end for
15:  Assign mapping  $L.root.pn = P.root$   $\triangleright$  core nodes
16:   $ME =$  edge set for saved best matching of
     $L.root.children, P.root.children$ 
17:  while  $ME \neq \emptyset$  do
18:    Extract next edge  $(ln, pn)$  from  $ME$ 
19:    Assign mapping  $ln.pn = pn$ 
20:    Add edge set for saved best matching of
       $ln.children, pn.children$  to  $ME$ 
21:  end while
22:  Allocate VMs according to  $ln \rightarrow ln.pn$  mappings
    and update  $f\_slots$  and  $f\_bw$  of each  $pn$  accordingly.
23:  Call Alg. 3.1 for application-aware adjustment.
24:  return true
25: end procedure
```

cluding depth-first search, breadth-first search, best-first (A*) search, and backtrack-
ing approaches, in the interests of scalability, we ultimately decided to model the
problem of finding the “best” set of feasible $ln \rightarrow ln.pn$ mappings as a minimization
variant of the classical assignment problem, also known as the minimum weighted
bipartite matching problem [38]. To this end, Algorithm 3.3 uses the Hungarian
or Kuhn-Munkres algorithm, originally proposed by Kuhn [39] and later refined by
Munkres [40], as a subroutine in the ASSIGN-MIN-COST sub-procedure. The details
are given below.

Algorithm 3.3 “Online LAW Allocation” (Cont.)

```
19: procedure ASSIGN-MIN-COST( $ln, pn$ )
20:   Initialize a bipartite graph  $G = \langle V_l, V_p, E = \emptyset \rangle$ 
21:   where  $V_l = ln.children$  and  $V_p = pn.children$ 
22:   for each pair of nodes ( $u \in V_l, v \in V_p$ ) do
23:     add new edge  $\langle u, v \rangle$  to  $E$ 
24:     if  $v.f\_slots \geq u.r\_slots \wedge pn.f\_bw \geq u.r\_bw$ 
25:       then
26:          $h\_val = \text{GET-HEURISTIC}(pn, h, L)$ 
27:         set edge weight =  $u.r\_slots \times h\_val$ 
28:       else set edge weight =  $\infty$ 
29:     end if
30:   end for
31:   Add dummy nodes to  $V_l$  and associated  $\infty$  edges
32:   until  $\|V_l\| = \|V_p\|$ 
33:   ME = KUHN-MUNKRES( $G$ )
34:    $\triangleright$  ME stores the minimum weighted edge set
35:   Remove edges with dummy nodes from ME
36:   return sum of edge weights of ME
37: end procedure

37: procedure GET-HEURISTIC( $pn, h, L$ )
38:   if  $h = \text{“Min Power”}$  then return  $pn.f\_slots$ 
39:   end if
40:   if  $h = \text{“Min BW”}$  then return  $pn.r\_slots$ 
41:   end if
42:   if  $h = \text{“Min CRC”}$  then
43:     if  $L.type = \text{“CI”}$  then return  $pn.ci\_slots$ 
44:     else return  $pn.slot\_capacity - pn.ci\_slots$ 
45:   end if
46:   end if
47: end procedure
```

Checking Feasibility

Algorithm 3.3 determines LAW allocation feasibility by first checking at ToR level as follows. For each LAW ToR T_l and each physical ToR T_p , it constructs a bipartite graph consisting of hosts of T_l on one end and hosts of T_p on the other. An edge is added between a pair of nodes on the opposite ends *if and only if* the physical host has sufficient VM slots to support the LAW host. The edge weight assigned is equal to the number of slots required by the LAW host (i.e., the r_slots parameter)

multiplied by a heuristic value associated with the the physical host's current state: The heuristic value equals the number of free VM slots (f_slots) for “Min Power” and the number of reserved slots (r_slots) for “Min BW”. There are two cases for “Min CRC”. If the LAW represents a CI workload, the heuristic value equals the number of reserved CI slots (ci_slots); otherwise, it equals the slot capacity less the number of reserved CI slots. The scaling of r_slots ensures that when a set of LAW hosts can be supported by multiple physical hosts, the LAW host with the largest VM requirement will be matched with the physical host with the smallest heuristic value, and so on.

Because typically T_l has a smaller number of hosts than T_p but the Hungarian algorithm requires a complete bipartite graph of equal size partitions as input, we modify the bipartite graph as done in [41], by 1) adding special edges of infinite weight (representing “infeasibility”) for all pairs of LAW and physical hosts that are not yet connected, and 2) adding dummy nodes to T_l with infeasible (infinite weight) edges from each dummy node to all nodes in T_p . As such, the Hungarian algorithm always returns a complete minimum weight matching, but this matching may include some infinite weight edges, representing “no feasible assignment” [41]. After running the Hungarian algorithm on the modified bipartite graph, we remove the dummy nodes (and associated edges) from the returned matching. If the remaining mapping still contains edge(s) with infinite weight, then it is safe to say that T_p cannot support T_l .

Once the feasibility between each pair of LAW and physical ToR's is determined, the same process repeats for nodes upward the LAW hierarchy (e.g., first the two aggregation switches, and then the core switch for the example scenario in Figure 3.1), until (i) the LAW root (core) is reached, at which point the LAW is determined to be feasible, or (ii) if, for some intermediate node of L , no feasible mappings exist between it and a physical counterpart, then ALLOCATE-LAW returns false and LAW L is determined to be infeasible for P .

ALLOCATE-LAW(L, P, h) returns **true** if and only if the Kuhn-Munkres algorithm returns a finite sum of edge weights. At each level of feasibility checking, a finite edge weight is assigned to a prospective $ln \rightarrow ln.pn$ mapping if and only if the mapping

is feasible, i.e., the physical node $ln.pn$ has sufficient VM slots and available uplink bandwidth to meet the requirements of the LAW node ln . And because the Kuhn-Munkres algorithm returns the set of edges comprising a minimum-weight matching, the sum of the edge weights returned by the Kuhn-Munkres algorithm is finite if and only if there exists a feasible $ln \rightarrow ln.pn$ mapping for each LAW node. Thus, $\text{ALLOCATE-LAW}(L, P, h)$ returns **true** if and only if there exists a feasible mapping for each LAW node.

General LAW Allocation

Once a LAW L is determined to be feasible as described previously (lines 4 to 11), Algorithm 3.3 proceeds to determine the best $ln \rightarrow ln.pn$ mappings using a top-down approach, from the LAW root (core) down to the LAW leaf nodes (hosts), selecting the best LAW-to-physical mappings at each level of the hierarchy based upon the values of the saved matchings that minimize the chosen heuristic (lines 15 to 20). Next, after the ToR mappings have been determined, the LAW VMs are allocated according to their respective $ln \rightarrow ln.pn$ mappings (line 22). Finally, VM placements are adjusted to ideal placement for application objectives by calling Algorithm 3.1.

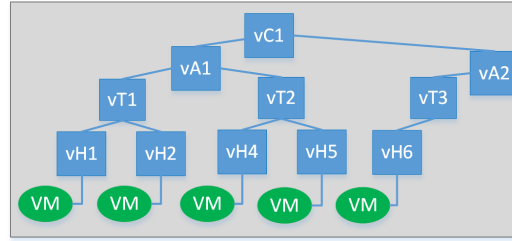
Complexity Analysis

Space Complexity = $O(|L| \cdot C^2)$, where C represents the maximum number of children of any node $pn \in P$. Space complexity is dominated by one of two factors: 1) the number of edges in the largest bipartite graph G , which may contain up to C^2 edges, and 2) the total number of edges maintained in the saved matchings. Because each of the $O(|L|)$ internal LAW nodes may contain up to C matchings of size C , the total number of edges maintained in the saved matchings is $O(|L| \cdot C^2)$. Thus, the resultant space complexity is $O(C^2 + |L| \cdot C^2) = O(|L| \cdot C^2)$.

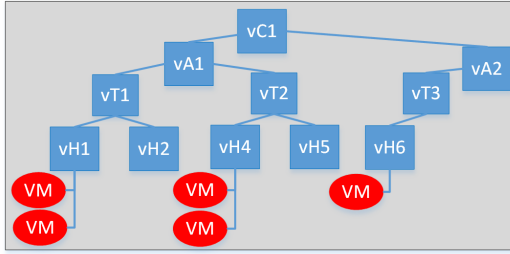
Time Complexity = $O(|L| \cdot C^4)$. Time complexity is dominated by the execution time of the Kuhn-Munkres algorithm, which runs in $O(C^3)$ and executes up to C times for each of the $O(|L|)$ internal LAW nodes, thus yielding a resultant time complexity of $O(|L| \cdot C^4)$.

LAW Allocation Example

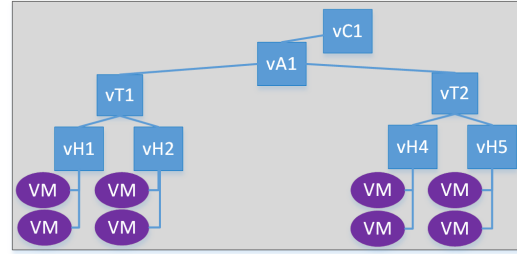
We conclude this section by illustrating the execution of Algorithm 3.3 for each heuristic method to allocate workloads in the example scenario presented in Section 3.2. First, ideal LAWs are constructed for each workload, depicted in Figure 3.2, using Algorithm 3.2. Next, Algorithm 3.3 allocates the workloads in the order they arrive (R1, R2, R3, R4), resulting in the allocations shown in Figure 3.3, depending on the chosen allocation heuristic.



(a) LAW R1, R3



(b) LAW R2



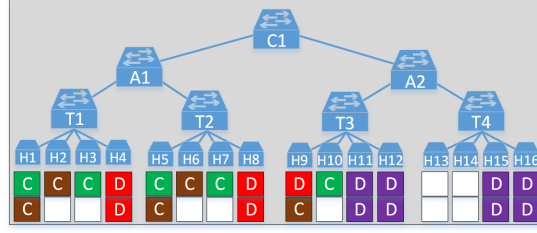
(c) LAW R4

Figure 3.2: LAWs for workloads R1-R4 used in the example scenario of Fig. 3.1. The r_slots and r_bw annotations are omitted for simplicity.

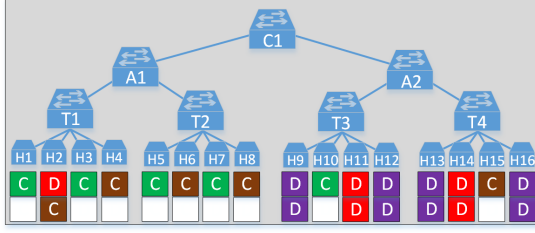
Note that LAW R4 is not feasible using the “Min CRC” heuristic method, since the resultant network state after allocating LAWs R1-R3 does not permit any feasible allocation of LAW R4. Therefore, mapping an entire LAW subtree to the physical infrastructure in a single atomic step appears to incur a tradeoff between performance gain and an increased likelihood of infeasibility.

Observation on Infeasibility

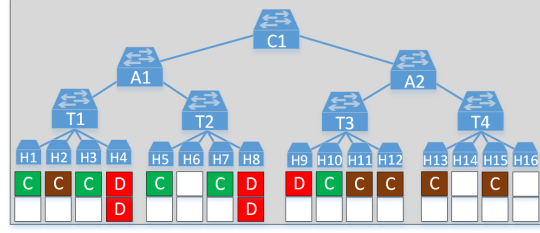
Although not necessarily intuitive, the reason why “Min CRC” encounters infeasibility sooner than the other heuristic methods is relatively straightforward. Because “Min CRC” explicitly seeks the maximum spread of CI workload VMs across ToRs and



(a) “Min Power”



(b) “Min BW”



(c) “Min CRC”

Figure 3.3: LAW allocations using “Min Power,” “Min BW,” and “Min CRC” heuristics for example scenario.

hosts within ToRs, as more CI LAWs are allocated, it becomes increasingly difficult to allocate DI LAWs, which require the use of a large number VM slots on a single host in order to achieve minimum RCI. In the case of the example scenario, LAW R4 could not be feasibly allocated given the allocations of LAWs R1-R3 using “Min CRC” (Figure 3.3c). Thus, in order to allocate workload R4, either the use of a per VM allocation method, such as GASO, or some method for deallocating and reallocating previous LAWs to make room for the next one, e.g., a backtracking approach, is required. In the next section, we will investigate this performance vs. infeasibility tradeoff further using a large-scale data center scenario, and subsequently propose our Statistical LAW solution to this infeasibility challenge in Section 3.5.

3.3.4 Applying LAWs to Different Topologies

In this section we describe how to apply LAWs to different types of physical network topologies like fat-tree [12], VL2 [42], and BCube [43]. Because LAWs are topology-dependent, by definition, it is important to understand how LAW construction and allocation may be performed for different types of physical topologies. Fat-tree and VL2 are both multi-rooted hierarchical trees with redundant links connecting devices between levels of the tree hierarchy. BCube is a non-hierarchical topology that uses a

number of vertically arranged switches in addition to the ToR switches, which provide end-to-end server communication via server-to-switch network links exclusively, i.e. there are no physical connections between switches.

Although the algorithms presented in this chapter assume a single-rooted, simple hierarchical tree topology about which to construct and allocate LAWs, applying the LAW Construction (Alg. 3.2)) and LAW Allocation (Alg. 3.3) algorithms to handle different types of topologies is relatively straightforward. First, Algorithm 3.2 is topology agnostic, and hence applicable “as-is” for constructing LAWs for any arbitrary data center topology, since it takes the physical topology as an input parameter, and subsequently uses an operator-specified per VM allocation method assumed to place the VMs onto the topology in a desirable fashion.

Second, although Algorithm 3.3 is topology dependent, it can be extended to handle multi-rooted, redundant link topologies with the addition of a single “dummy” root node of which each of the core switches comprising the multi-rooted topology are made children of the dummy root using dummy network links of capacity = 0. Redundant links are accounted for by way of setting multiple “uplink BW” variables for devices with multiple available upstream connections. When placing a workload across devices with several available uplinks of sufficient capacity, a method of selecting one must be chosen. Link selection heuristics such as “Min Congestion” or “Min Available BW” may be appropriate. By way of these implementation adjustments, we can soundly perform LAW allocation for multi-rooted and redundant hierarchical network topologies using Algorithm 3.3.

For non-hierarchical data center topologies such as BCube, Algorithm 3.3 can be extended by first using the dummy root strategy described above, and then by making all non-hierarchical component switches children of the root using dummy network links of capacity = 0. However, although this approach remains sound, it may not be very well suited for large non-hierarchical data center networks, as the value of C (maximum number of children of any node), will likely be large due to the high number of dummy root children created when attempting to adapt this approach to handle a non-hierarchical network.

3.4 Evaluation

In this section, we evaluate the performance of proposed LAW based placement heuristics against existing solutions in a simulated (relatively) large-scale data center environment comparable to ones used in related work [4]. The topology consists of 40 aggregation switches, 160 ToR switches (4 x ToRs per aggregate), 2560 hosts (16 x hosts per ToR), and 40960 VM slots (16 x VM slots per host).

3.4.1 Setup

Representation of Existing Solutions

We choose to adapt the GASO allocation algorithm presented in [28] to represent existing solutions. GASO is chosen because it is extensible to an arbitrary number of disparate objectives and customizable by the operator via weightings for each objective, similar to [10, 11]. For our purpose, the global cost function is a weighted sum of power consumption and mean link bandwidth usage. By default, GASO uses a greedy search for solutions that will minimize the global cost, and as such may converge prematurely to local minima [28]. Therefore, we enhance GASO by implementing a genetic algorithm to diversify candidate solutions as done in related work [16, 28]. We set the candidate solution population size to 10, and perform up to 25 evolutions for each workload request; these parameters are shown to be required for the size of our topology [16, 28].

Per VM Application-Aware Solutions

To make our evaluation more complete, we seek to understand how the performance of LAW based heuristics compare to that of solution that is application-aware but places workloads one VM at a time. We call the latter a per VM application-aware solution. We have developed such a solution by further enhancing GASO. Therefore, we evaluate two distinct implementations of GASO: 1) The “default” version, which weighs power and BW usage highest, and 2) an “application-aware” version, which weighs RCI and CRC highest. Both versions of GASO we use the power and BW usage reduction heuristics from [28]. The application-aware version additionally uses an RCI and CRC reduction heuristic that executes Algorithm 3.1 to move (if necessary)

application VMs between hosts in each rack to reduce RCI and CRC. Since the purpose of the default version of GASO is to model non-application-aware solutions, it does not use this application-aware heuristic.

In addition, we refine the LAW based heuristics so that when it is infeasible to allocate a LAW in one atomic step, they resort to per VM allocation for that workload.

Workload Traces

The workloads are randomly generated and their size distributions are comparable to what used in related work [4]. Specifically, four types of workloads are used to model heterogeneous [4] compute-intensive and network-intensive resource requirements:

Type 1: $\langle \text{CI}, \text{numVMs} \times 4 \text{ slots}, 5 \text{ Mbps}, 0.5 \text{ WCS} \rangle$

Type 2: $\langle \text{CI}, \text{numVMs} \times 2 \text{ slots}, 10 \text{ Mbps}, 0.7 \text{ WCS} \rangle$

Type 3: $\langle \text{DI}, \text{numVMs} \times 1 \text{ slot}, 60 \text{ Mbps}, 0.5 \text{ WCS} \rangle$

Type 4: $\langle \text{DI}, \text{numVMs} \times 2 \text{ slots}, 30 \text{ Mbps}, 0.7 \text{ WCS} \rangle$

where numVMs ranges between 40 and 200, and also includes the number of slots per VM to represent heterogeneous workload requirements, as done in [4].

In each run, the workload trace is produced by selecting one of the workload types (1-4) and a value for numVMs (40-200) uniformly at random, and deducting the number of slots required of the workload from the total number of slots available in the physical infrastructure. Each randomly generated workload is added to the trace until the next generated workload exceeds the infrastructure slot capacity. After the workload trace is generated, each heuristic under evaluation attempts to place workloads of the trace in the order that they were generated, i.e., online, to the simulated infrastructure.

Performance Metrics

Our evaluation focuses on the following metrics: (1) power usage, 2) BW usage (mean link BW), 3) link congestion (maximum link BW), 4) mean RCI, 5) CRC, and 6) execution time. We also seek to understand the extent of LAW allocation infeasibility

tradeoff. Specifically, for each run we identify the fraction of VM capacity allocated when the first infeasible LAW occurs.

3.4.2 Results

Here, we present the results of LAW and GASO allocation methods over an average of ten workload traces by depicting their performance in power consumption (Fig. 3.4), BW usage (Fig. 3.5), link congestion (Fig. 3.6), CRC (Fig. 3.7), RCI (Fig. 3.8), and execution time⁹ (Fig. 3.9). Plotted points in each figure represent the average objective metric values of a feasible state for a given fraction of VM capacity allocated (i.e., physical infrastructure utilization) over a series of ten runs using different allocation methods. The vertical lines are color coordinated to match the allocation methods and each represents the average capacity allocated when the first infeasibility occurred for the corresponding allocation method.

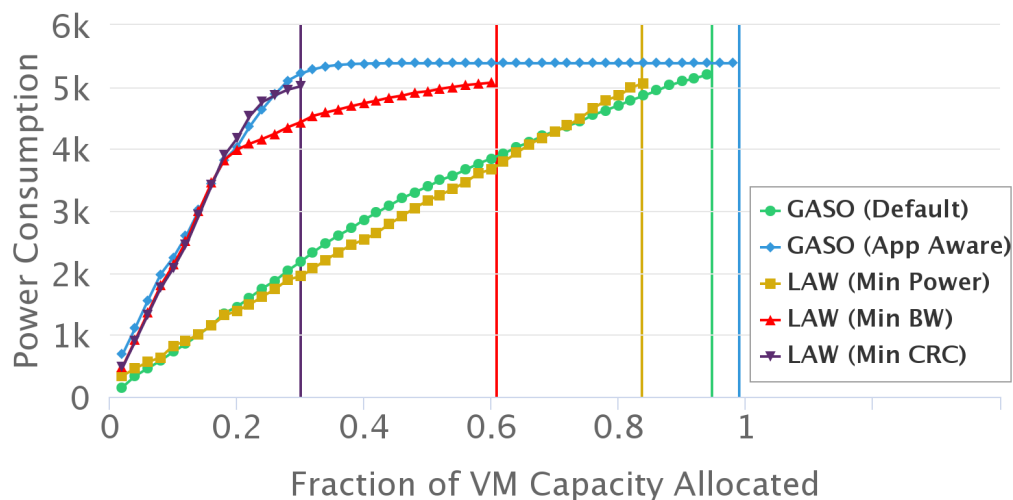


Figure 3.4: Power usage vs. capacity allocated.

Based on these results, we make several observations. First, observe that the “Min Power” LAW allocation heuristic dominates the default GASO per VM allocation method. The “Min Power” LAW allocation heuristic offers similar power conserva-

⁹The simulation consists of approximately 3000 lines of Java code, and was run (serially) using 64-bit JVM. The host PC was running 64-bit Windows on an Intel 2.4 GHz quad-core processor with 24 GB of RAM.

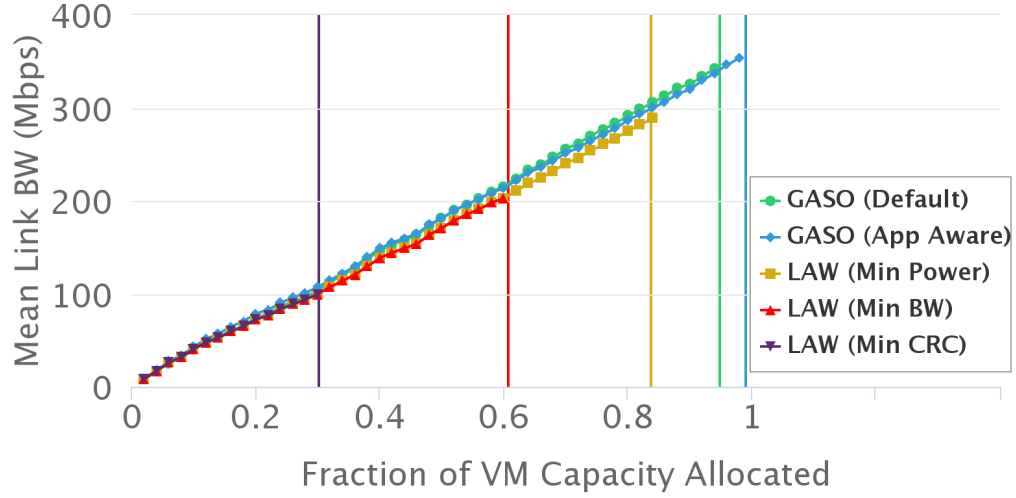


Figure 3.5: Mean BW usage vs. capacity allocated.

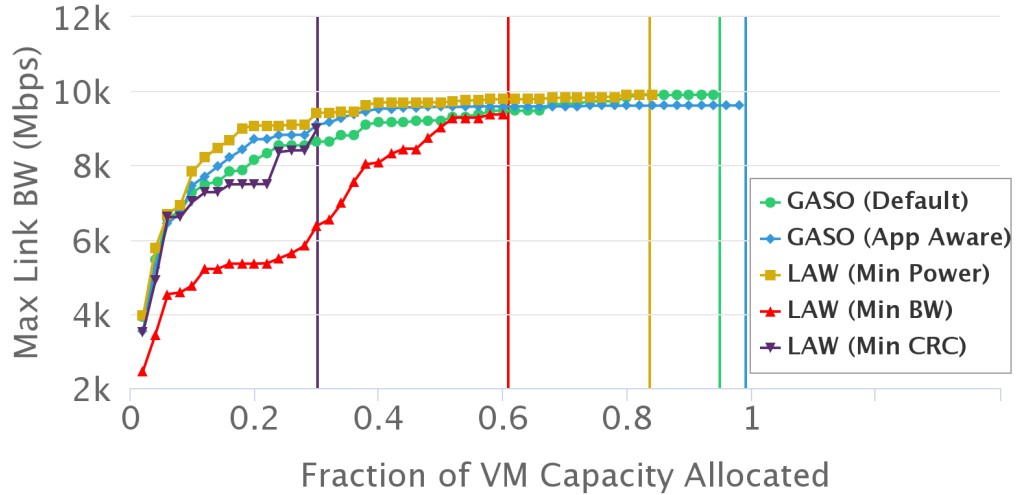


Figure 3.6: Max. link BW usage vs. capacity allocated.

tion, BW usage, and CRC as default GASO, but additionally provides much lower RCI, with an execution time that is an order of magnitude faster. Thus, based on this comparison it is clear that application-aware allocation for this scenario may be strictly superior to non-application-aware allocation, i.e., tenant application objectives may be achieved nearly “for free,” with little to no degradation of operator objectives. Next, observe the straightforward tradeoff between power usage and CRC. Default GASO and “Min Power” maintain relatively high CRC in order to minimize

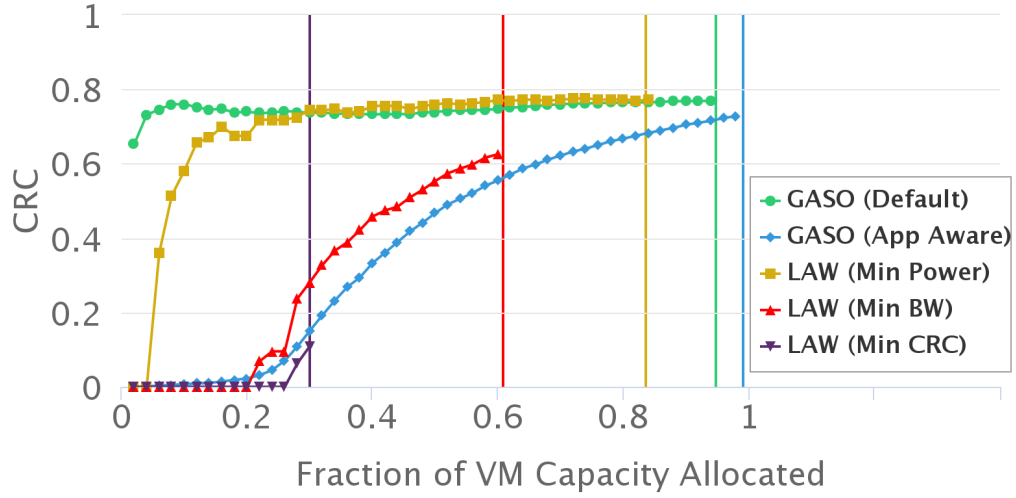


Figure 3.7: CRC vs. capacity allocated.

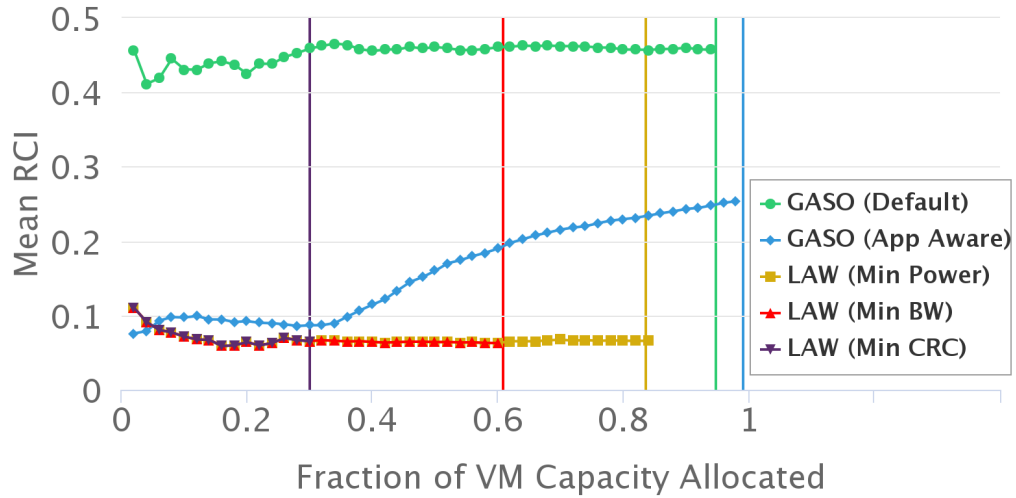


Figure 3.8: Mean RCI vs. capacity allocated.

power usage, whereas App-Aware GASO, “Min BW,” and “Min CRC” tradeoff higher power consumption for reduced CRC. Furthermore, although App-Aware GASO is designed to jointly minimize RCI and CRC, because it is bounded by the parameters and limitations of a genetic algorithm to perform its search (e.g., population size, number of evolutions, etc), it does not achieve ideal Mean RCI as the physical infrastructure becomes utilized when compared to the LAW allocation heuristics, which each explicitly preserve the RCI for each constructed LAW when making allocations.

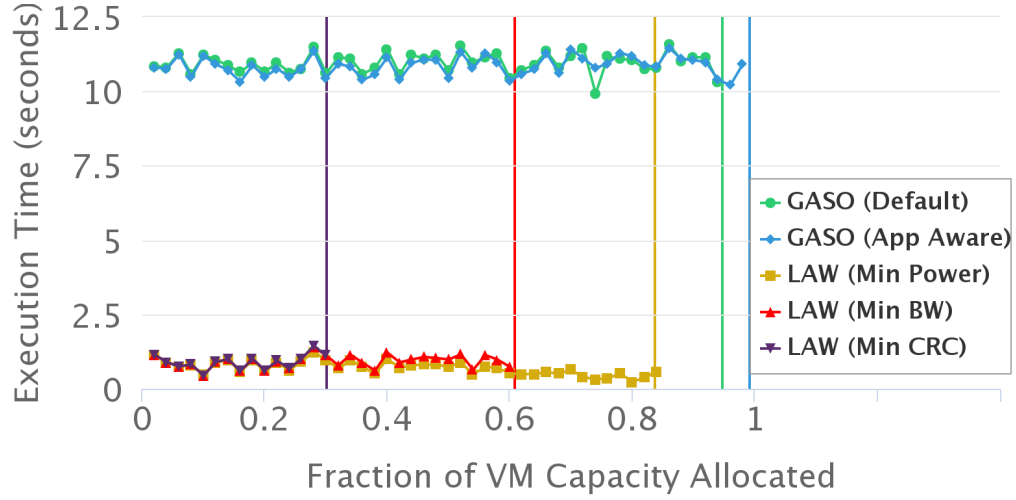


Figure 3.9: Execution time vs. capacity allocated.

This RCI gap between GASO App-Aware and LAW becomes more pronounced as the infrastructure becomes more utilized. At 30% utilization, RCI for GASO App-Aware is only slightly higher than LAW, but at 60% utilization, RCI for GASO App-Aware is more than double that of LAW.

Tradeoffs involving bandwidth are less straightforward. LAW allocation preserves the provisioned BW for each constructed LAW, and both GASO variants include mean link BW as an optimization criterion, so differences in BW usage (Fig. 3.5) are relatively insignificant. The maximum link BW plot (Fig. 3.6) provides an indicator of link congestion. Observe that the “Min BW” allocation heuristic provides significantly lower link congestion than the other allocation methods until the infrastructure becomes about 50% utilized.

Regarding feasibility, observe that each LAW allocation heuristic encounters infeasibility at widely different levels of physical infrastructure utilization. The reason for this is complex, as several factors, including workload size, workload trace composition (CI : DI ratio), physical topology tree structure, physical topology utilization, and heuristic allocation method, determine the likelihood of encountering LAW infeasibility. Moreover, in this scenario there is clear trend indicating that allocation methods that more aggressively attempt to spread CI workloads, i.e., reduce CRC,

are more likely to encounter infeasibility sooner. “Min Power” prefers maximum concentration of all workloads and has the highest feasibility rating (83.6%). “Min BW” prefers maximum distribution of all workloads and has moderate feasibility (60.7%). “Min CRC” explicitly spreads CI workloads and has the least feasibility (30.1%).

Finally, observe that “Min CRC,” while having relatively low feasibility, appears to dominate GASO App-Aware in this scenario, by achieving lower BW usage, link congestion, CRC, and RCI while using generally the same amount of power. Furthermore, regarding algorithm execution time (Fig. 3.9), LAW allocation (hundreds of milliseconds) is an order of magnitude faster than the GASO variants (tens of seconds).

3.4.3 LAW for Architectures of Different Dimensions

In this section, we look at how varying the physical dimensions of hierarchical network architectures affect allocation feasibility of different LAW placement heuristics. Here, physical dimensions are defined in terms the four-tuple (A, T, H, S) , where A represents the number of aggregate switches under each core switch, T represents the number of ToR switches under each aggregate switch, H represents the number of host servers under each ToR switch, and S represents the number of VM slots per host server. Regarding different values for these dimensions, we are particularly concerned with how LAW feasibility is affected as the network architecture is “scaled-out,” or made more horizontal by increasing the number of physical devices (i.e., increasing the values of A , T , and H), as well as “scaled-up,” or made more vertical by increasing the capacity of each physical host server (i.e., increasing the value of S).

In order to provide a fair LAW feasibility comparison across architectures of varying physical proportions, the total VM slot capacity for each architecture is held constant at 10240 slots. Thus, although each architecture evaluated in this section has different values for (A, T, H, S) , they all have the same slot capacity, i.e., for each architecture, $A \cdot T \cdot H \cdot S = 10240$. The workload traces used are the same as before (Section 3.4.1).

Table 3.1 illustrates the feasibility results for each of the “Min Power,” “Min BW,” and “Min CRC” LAW placement heuristics across a range of hierarchical network

architectures with different dimensions and a 10240 VM slot capacity. The range of dimensions in this table represents a spectrum of hierarchical network architectures that are scaled-up and/or scaled-out to varying degrees. The rows closer to the top of the table represent network architectures that are proportionally scaled-up, while those closer to the bottom represent more scaled-out architectures.

<i>Allocation Heuristic / Physical Dimensions</i>	Min Power	Min BW	Min CRC
(5, 4, 16, 32)	0.44	0.18	0.27
(5, 4, 32, 16)	0.58	0.38	0.28
(5, 8, 16, 16)	0.72	0.33	0.33
(10, 4, 16, 16)*	0.70	0.48	0.30
(10, 8, 8, 16)	0.72	0.48	0.38
(10, 8, 16, 8)	0.77	0.50	0.56
(10, 16, 16, 4)	0.81	0.50	0.63
(20, 8, 16, 4)	0.82	0.50	0.66

Table 3.1: LAW feasibility results for “Min Power,” “Min BW,” and “Min CRC” for different 10240 VM slot infrastructures of varying dimensions. The values denote the average physical infrastructure utilization (fraction of VM capacity allocated) when the corresponding allocation heuristic and LAW type first encounter infeasibility. Poor results (less than 50%) are shaded red, moderate (at least 50% but less than 80%) yellow, and good (at least 80%) green. * reference dimensions

From these results, we observe two clear trends, using the (10,4,16,16) dimension architecture as a reference point. First, as an architecture is scaled “out” and “down” from (10,4,16,16) to (20,8,16,4) i.e., approaches dimensions closer to the bottom of Table 3.1, LAW feasibility more than doubles for “Min CRC” (120% increase), and increases significantly for “Min Power” (17% increase), but only increases marginally for “Min BW” (4% increase). Intuitively, these results should be expected: by increasing the number of switches and servers, there is more room to spread CI workloads throughout the infrastructure, resulting in much higher feasibility for “Min CRC,” but less so for the other heuristics, as their allocation feasibilities are not as dependent as “Min CRC” on the ability of the infrastructure to handle distributed CI workloads.

Second, as an architecture is scaled “up” and “in” from (10,4,16,16) to (5,4,16,32), i.e., approaches dimensions closer to the top of Table 3.1, LAW feasibility drastically decreases for all heuristics due to the inability to spread CI workloads throughout the infrastructure. Although allocation feasibilities for “Min Power” and “Min BW” are generally not as infrastructure-sensitive as “Min CRC,” as the network architecture approaches more vertical physical topologies like (5,4,16,32), the resultant inability to spread CI workloads begins to hinder allocation feasibility regardless of heuristic type (less than 50% feasibility across all heuristic types in this example, as illustrated by the first row of Table 3.1).

Therefore, based on these results, we conclude that scaled-out architectures are generally better for achieving higher LAW feasibility versus scaled-up architectures. However, this problem of finding the physical network architecture that yields the highest LAW feasibility for a given workload forecast (e.g., trace) and LAW allocation heuristic seems like a fertile ground for future work.

3.5 Statistical LAWs

As seen from the the execution time plot (Figure 3.9), the GASO per VM workload placement algorithm runs an order of magnitude slower than LAW allocation, and as such, is less suitable for online workload placement tasks, particularly for large data center networks. The GASO execution time is comparable to other workload placement solutions that use genetic algorithms¹⁰, such as [16, 28]. Therefore, although per VM allocation may be used to address the problem of allocating a workload when LAW infeasibility is encountered, we argue that an online solution that gracefully relaxes the ideal application allocation, as represented by the LAW structure, is preferable in the interests of both reduced RCI and reduced execution time. We also considered an alternative LAW backtracking approach to address the issue of LAW infeasibility, but such an approach disrupts current allocations and initial evaluations significantly degraded execution time, so we defer the exploration of such alternative options to future work.

¹⁰Greedy heuristics, like those used in [3, 4] run faster (order of seconds) for placing large workloads, but they are prone to suboptimal convergence to local minima when multiple objectives are considered.

Therefore, we propose a Statistical LAW allocation strategy to explore the LAW feasibility vs. performance tradeoff. A Statistical LAW is natural LAW relaxation which offers a compromise between complete LAW and per VM placement approaches, by representing a percentage of the workload VMs as a “relaxed” LAW, and considering the remainder of the workload VMs as individual units of allocation for some per VM allocation method.

3.5.1 Example

The 70% Statistical LAW model of workload R4 from the Section 3.2 example scenario is illustrated in Figure 3.10, and represents the preservation of at least 70% of the original LAW VMs (6 VMs) while using per VM allocation for the remainder (2 VMs). Figure 3.11 depicts the resultant network state after using the “Min CRC” LAW heuristic to allocate the 70% LAW for R4 (Figure 3.10) to the physical network state depicted in Figure 3.3c.

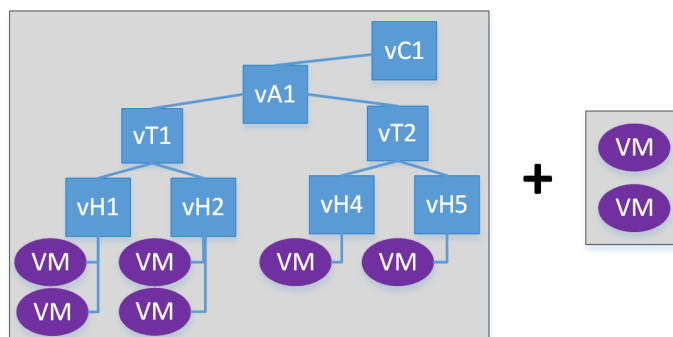


Figure 3.10: Statistical LAW (70%) for workload R4 in example scenario.

By using a statistical measure to relax LAW VM positioning requirements, a compromise between ideal workload allocation and feasibility is achieved, while still maintaining the fast LAW allocation times compared to traditional per VM approaches.

3.5.2 Construction and Allocation

An $x\%$ Statistical LAW for some original LAW L is constructed by removing $(100 - x)\%$ of L 's VMs (floor) from L by removing the VMs sequentially, where each

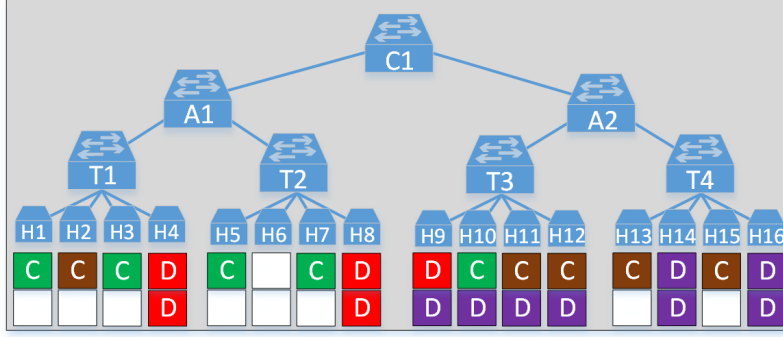


Figure 3.11: “Min CRC” LAW allocation using a 70% Statistical LAW to allocate previously infeasible workload R4 for the example scenario.

VM is removed from the LAW host containing the most VMs (or from a host under the most populated ToR in the case of a tie). The removed VMs are maintained with the LAW, and are allocated in a per VM fashion after the Statistical LAW is allocated. For Statistical LAW, the goal of the per VM allocation heuristic for the “remainder” VMs is to minimize RCI subject to WCS constraints. Hence, for CI workloads, the per VM allocations strive to minimize host affinity by allocating the VMs to hosts with the fewest number of CI slots allocated. For DI workloads, these allocations seek to maximize host affinity by concentrating the VMs as closely as possible to the rest of the LAW VMs without violating the WCS bound.

3.5.3 Statistical LAW Results

Here, we present the results of Statistical LAW allocation using the same workload traces and simulated physical infrastructure as the Section 3.4 evaluation. Statistical LAW allocation methods use a progressive backoff approach. For each workload in the trace, first complete LAW allocation is attempted, then progressively 90%, 70%, 50%, and finally 30% Statistical LAW allocations are attempted if the previous Statistical LAW allocation attempt failed to feasibly allocate the workload. For Statistical LAW, the workload is considered infeasible only if all allocation attempts (complete, 90%, 70%, 50%, 30%) fail. Of course, an even lower Statistical LAW, such as 10% LAW may be attempted if 30% LAW allocation fails, but at that point we observe the resultant LAW structure would be too degraded to provide much benefit.

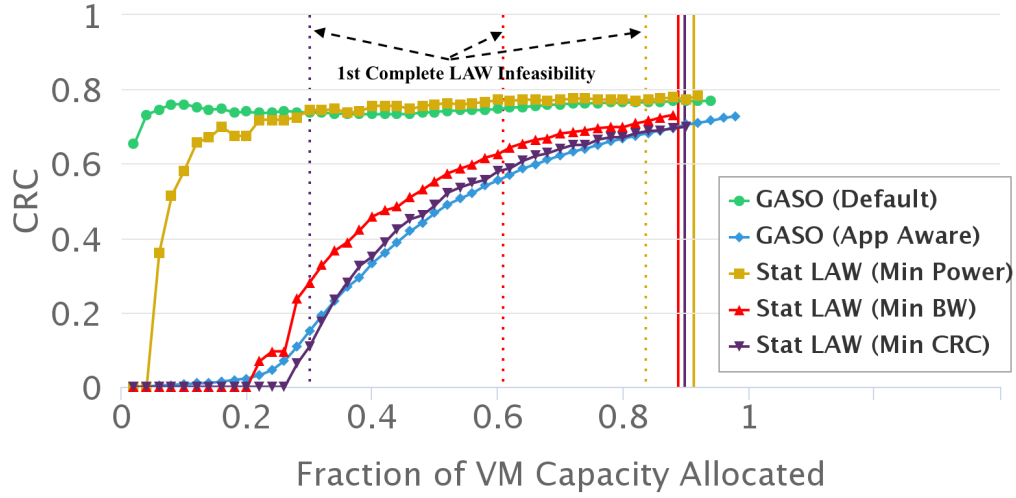


Figure 3.12: Effect of Statistical LAW on CRC.

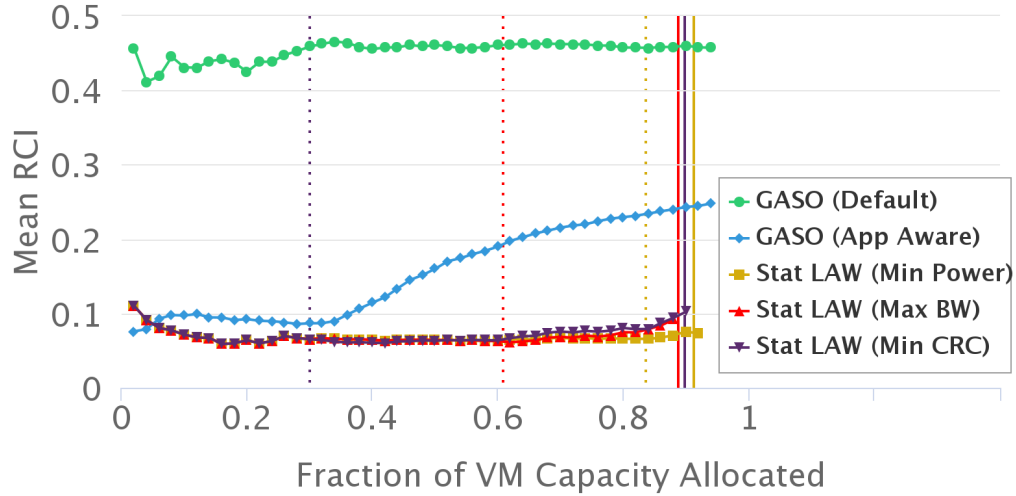


Figure 3.13: Effect of Statistical LAW on Mean RCI.

The Statistical LAW results as shown in Table 3.2 clearly demonstrate the advantage of using a Statistical LAW approach vs. per VM allocation when complete LAW allocation is not feasible. Notably, in this scenario, the Statistical LAW allocation performs similarly to complete LAW allocation, with both RCI and execution time increased by only a small constant factor as workloads become overwhelmingly infeasible for Statistical LAW, which begins to occur around 85% infrastructure utilization, as seen in Figures 3.13 and 3.14, respectively. In other words, for each resource usage

<i>Allocation Heuristic / LAW Type</i>	Min Power	Min BW	Min CRC
Complete LAW	0.84	0.61	0.30
90% Stat. LAW	0.84	0.63	0.35
70% Stat. LAW	0.86	0.67	0.62
50% Stat. LAW	0.87	0.80	0.86
30% Stat. LAW	0.91	0.89	0.90

Table 3.2: Statistical LAW feasibility results for “Min Power,” “Min BW,” and “Min CRC” for the large-scale evaluation. The values denote the average physical infrastructure utilization (fraction of VM capacity allocated) when the corresponding allocation heuristic and LAW type first encounter infeasibility.

metric, there is very little performance degradation for using Statistical LAW compared to complete LAW allocation, as can be seen by observation of plotted points for Statistical LAW allocation methods beyond the amount of physical infrastructure utilization at which complete LAW allocation fails (values in the first row of Table 3.2, denoted by dotted vertical lines in each figure). For instance, in Figure 3.12, although the “Min CRC” LAW placement heuristic begins to incur slightly higher CRC than App-Aware GASO at approximately 40% infrastructure utilization, remains very similar (within a few percentage points) throughout the entire trace.

3.6 LAWs for Workload Prioritization

As demonstrated in the previous section, using Statistical LAW allocation may greatly improve LAW allocation feasibility versus complete LAW allocation, while still producing resource and application efficient allocations with fast workload placement times. However, because Statistical LAW may result in increased application RCI for statistically allocated workloads, the operator may prefer to take a proactive approach in applying Statistical LAW allocations to low priority workloads in order to increase the LAW feasibility of high priority workloads to ensure minimum RCI for them. In this section, we propose the Early Statistical LAW allocation strategy, a proactive approach that trades off suboptimal placement of low priority workloads to

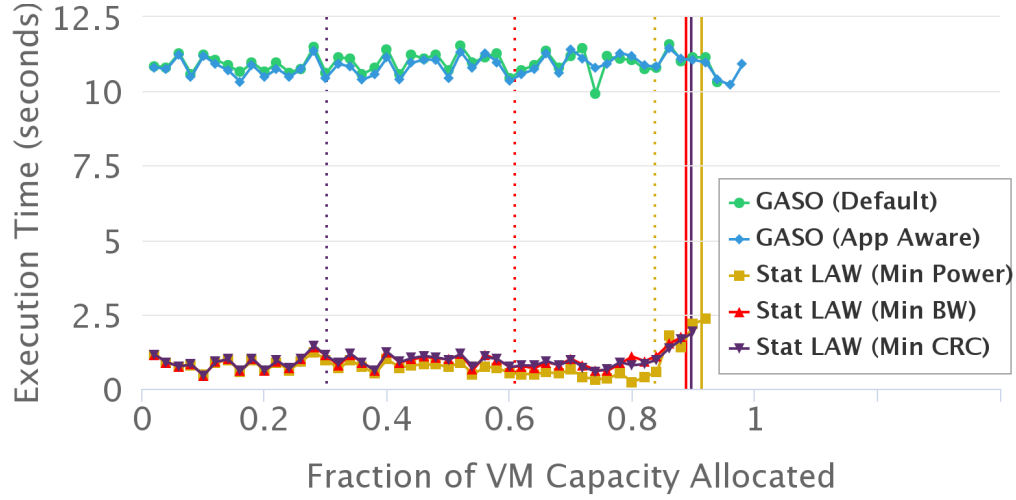


Figure 3.14: Effect of Statistical LAW on execution time.

increase LAW feasibility for high priority workloads.

3.6.1 Early Statistical LAW Allocation

In this work, we explore a simple binary priority scheme and leave more elaborate prioritization to future work. We seek to increase the feasibility of high priority workloads by allocating low priority workloads using Early Statistical LAW, which differs from default Statistical LAW allocation only in the per VM allocation heuristic used to allocate the non-LAW “remainder” VMs. While per VM allocations using default Statistical LAW strive to minimize RCI, Early Statistical LAW uses these per VM allocations as a “compromise” to increase the feasibility for future high priority LAWs.

We achieve these compromise allocations by placing the remainder CI VMs in the infrastructure subtree with the fewest available slots using the classical “tightest fit” bin packing heuristic, with a preference for placement on hosts with more DI VMs. Intuitively, using such a heuristic should provide more “contiguous” VM slot space in other infrastructure subtrees, thus providing a higher likelihood of LAW feasibility for future high priority workloads. Because this additional contiguous slot space comes at the cost of bandwidth efficiency, we do not compromise the placement of DI VMs, but

instead allocate them in the same fashion as default Statistical LAW: with maximum intra-application host affinity while satisfying WCS constraints.

3.6.2 Early Statistical LAW Results

We evaluate Early Statistical LAW using the same scenario as before except this time we randomly assign each workload a binary priority value (“high” or “low”). We compare the performance of Early Statistical LAW to default Statistical LAW, specifically with regard to the feasibility of high priority workloads. Each low priority workload is allocated as a 50% Statistical LAW, using the compromise placement heuristic described in the preceding paragraph, while high priority workloads are allocated identically to Statistical LAW, using the same progressive backoff schedule.

<i>Allocation Heuristic / High Priority LAW Type</i>	Min Power	Min BW	Min CRC
Complete LAW	0.95 (14.5%)	0.63 (5.0%)	0.43 (43.3%)
90% Stat. LAW	0.96 (14.3%)	0.67 (8.1%)	0.51 (45.7%)
70% Stat. LAW	0.97 (12.8%)	0.85 (28.8%)	0.80 (29.0%)
50% Stat. LAW	0.98 (12.6%)	0.89 (11.3%)	0.97 (12.8%)
30% Stat. LAW	0.99 (8.8%)	0.96 (7.9%)	0.99 (11.2%)

Table 3.3: Early Statistical LAW feasibility results for high priority workloads in the large-scale evaluation. The raw values denote the average physical infrastructure utilization when the corresponding allocation heuristic and LAW type first encounter high priority workload infeasibility. The values in parentheses denote the percentage of feasibility increase over default Statistical LAW allocation.

Table 3.3 illustrates the merits of using Early Statistical LAW to improve feasibility for high priority workloads. By proactively allocating low priority workloads using Early Statistical LAW, high priority feasibility is increased for each LAW type versus default Statistical LAW allocation. Consider “Min CRC,” the LAW allocation heuristic that is generally the least feasible. By using the Early Statistical LAW allocation strategy, complete LAW feasibility is increased by 43.3% for high priority workloads. However, while Early Statistical LAW allocation leads to better placements for high priority

workloads (i.e., closer to desired, more likely to meet SLA), this increased feasibility comes at the cost of increased RCI, depicted in Figure 3.15.

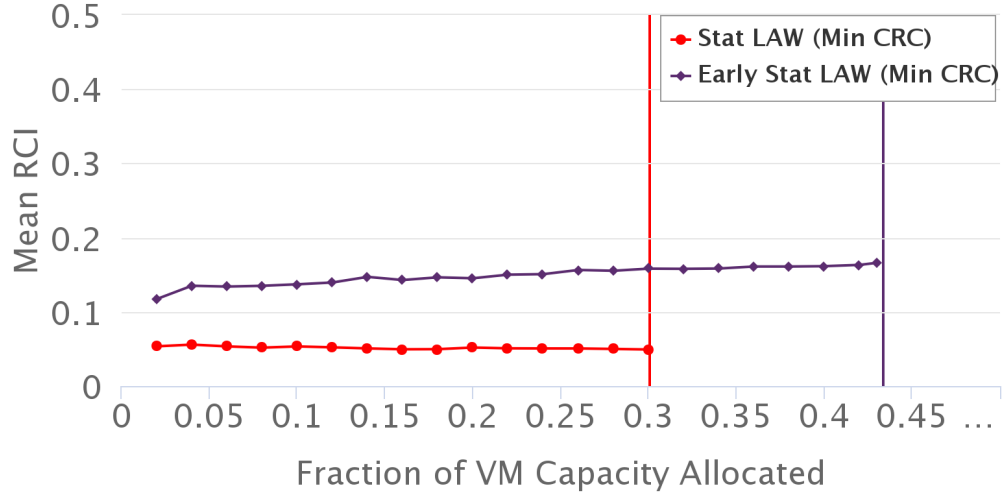


Figure 3.15: RCI vs. feasibility tradeoff introduced by Early Statistical LAW (using Min CRC heuristic).

Regarding this feasibility vs. RCI tradeoff, it is clear that the increase in RCI with Early Statistical LAW is a result of suboptimal allocations for low priority workloads, since complete LAW allocations preserve RCI and high priority LAW feasibility is increased using this strategy. This feasibility vs. RCI tradeoff should be considered by the operator when making a determination to use Early Statistical LAW. For example, suppose an operator decides to use the “Min CRC” LAW placement heuristic based on forecasting evidence that more CI workloads than DI workloads are likely to arrive. Since LAW placement using “Min CRC” has been shown to have relatively low LAW feasibility, he/she may want to invoke Early Statistical LAW allocation for low priority workloads, especially if there are high priority tasks or tenants active, and accept higher RCI for low priority workloads as a tradeoff. On the other hand, if there are relatively few high priority tasks or tenants active, or if the operator is unsure of the types of workloads likely to arrive, then he/she may decide to use the “Min Power” LAW placement heuristic with no Early Statistical LAW allocations, since “Min Power” has been shown to have relatively high complete LAW feasibility

as is, and the relatively benign workload forecast may allow electrical cost savings while still providing good RCI for active tasks and tenants.

3.7 Related Work

Prior work on application-aware workload placement mostly focuses on maximizing the performance of certain applications (e.g., data intensive [20], high-performance computing applications [22], etc.) or optimizing specific aspects of application performance (e.g., network throughput [23,26], fairness [24]). Other related solutions such as CloudMirror [3] and Ostro [4] provide bandwidth guarantees and ensure high application availability. To the best of our knowledge, this work is the first to propose precomputing desired workload placements for individual applications (i.e., LAWs) and subsequently using them to speed up and prioritize workload placement while meeting per application performance requirements.

3.8 Conclusion

We have demonstrated that an application-aware approach, by optimizing the relative positioning of VMs of individual workloads, can meet both per application requirements and cumulative resource usage goals. Furthermore, we have proposed a new abstraction (i.e., LAW) to enable online workload placement that is one order of magnitude faster than existing solutions, and scalable to large data center scenarios.

In the big picture, we view this work as a first step towards understanding the tradeoff between maximizing application performance and optimizing network-wide resource usage. We believe there is a wide design space for new formulations and heuristics to meet specific combinations of application-level requirements and operational goals.

CHAPTER 4:

Conclusion

Here, we summarize the merits of applying each of the design principle individually, i.e., the distinct benefits of applying either the EASO or LAW contribution independently, and then propose a new design strategy for combining these design principles into a synthesized approach that realizes the benefits of each, which we term “EASO-LAW Synthesis.” Finally, the dissertation concludes with a discussion of open issues and future work.

4.1 Summary of Contributions

The first principle, comprehensive tradeoff exploration, provides the operator with a wide range of meritorious placement options for a given workload, but does so at the cost of execution time, and without much consideration for application-specific resource contentions. The second principle, LAW, provides rapid workload allocation times, even for large data center networks, lacks the tradeoff exploration of the former, thus requiring the operator to specify desirable LAW characteristics (i.e., weighted objectives) a priori.

Comprehensive Tradeoff Exploration. We propose a novel, pure MOP formulation for data center orchestration, and demonstrate how our new EASO algorithm can enumerate a wide range of, and potentially better, solutions than current orchestrators for relatively large data center networks by exploring tradeoffs based on Pareto-dominance, rather than attempting to reducing the multi-objective tradeoff space to a single-objective optimization problem as done by existing NCF resource allocation solutions. Also, unlike existing NCF resource allocation solutions, EASO provides a *set* of proposed allocations, each with individual merits, rather than a single “best” allocation (which may not necessarily best achieve operator objectives if NCF weightings are set improperly or if the orchestration algorithm converges to a local minima), as demonstrated in Chapter 2.

LAW-based Resource Allocation. By explicitly modeling application preferences and developing a novel LAW abstraction that preserves them, we have demonstrated how to achieve application-aware allocation for data center workloads at time scales that are an order of magnitude faster than other multi-objective workload placement approaches, while still achieving high quality resource allocations in terms of NCF and operator objectives. This is in contrast to current per VM workload placement approaches that use a finer granularity of allocation for application components (e.g., VMs). By precomputing larger atomic units of allocation (e.g., LAWs) that represent entire workloads, we have demonstrated the feasibility of scalably allocating such LAWs in a manner that further reduces contention among applications while yielding sub-second workload placement times for placing large data center workloads.

4.2 EASO-LAW Synthesis: Better Than the Sum of Its Parts

This strategy involves first performing comprehensive NCF tradeoff exploration using EASO to precompute a wide range of LAWs desirable to the network operator, and then, as new tenant workload requirements arrive, operator preferences and network conditions are queried to select the ideal LAW and corresponding heuristic for workload allocation. By implementing such a design strategy, which should be relatively straightforward given the technical contributions of Chapters 2 and 3, and the content of this section, an orchestration solution that both performs comprehensive tradeoff exploration *and* achieves scalable, resource efficient, and application-aware placement of large-scale data center workloads, can be realized.

The key insight in building an effective data center orchestration solution using comprehensive tradeoff exploration (as implemented by EASO) and LAW lies in understanding how to synthesize these two conceptually distinct principles into one, such that the resultant solution enjoys both the tradeoff exploration of EASO, *and* the rapid and application-aware workload allocation of LAW. To deepen the understanding of how this may be achieved, consider Figure 4.1, which depicts the system flow of (a) standalone EASO, (b) standalone LAW, and (c) EASO-LAW Synthesis.

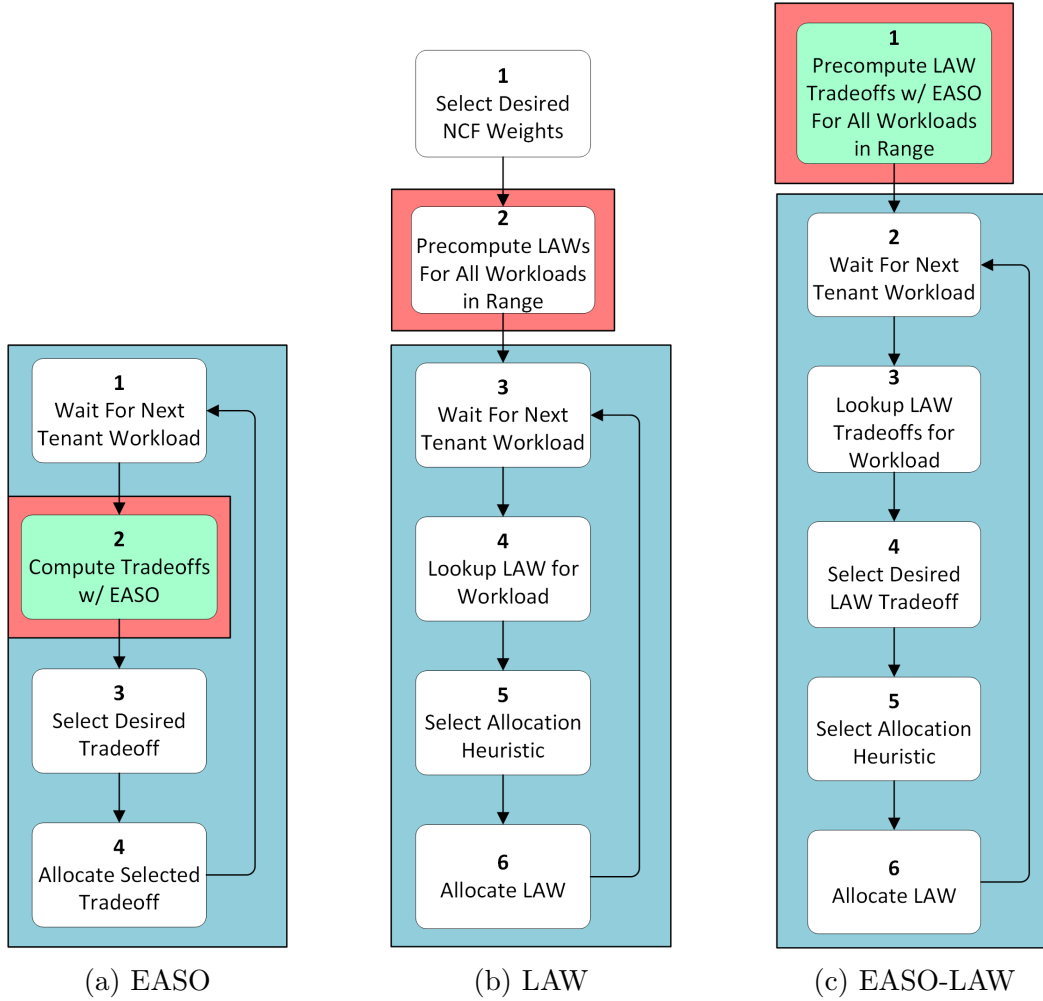


Figure 4.1: System flow diagrams for (a) EASO, (b) LAW, and (c) EASO-LAW Synthesis. Steps highlighted in red are computationally intensive (order of tens of seconds or more), those contained in blue are part of the typical process flow, and those in green provide tradeoff exploration. EASO-LAW Synthesis (c) provides tradeoff exploration while avoiding time intensive computation in the typical process flow.

Each of the above diagrams represents the execution process for its corresponding system. Colors are used to highlight key steps in the execution process. Steps contained in the contiguous blue rectangle are considered part of the typical process flow for the system and will be executed very frequently. These steps describe the system execution after initialization, when it is ready and waiting to process new tenant workload requirements. Steps highlighted require time intensive computation, typically on the order of tens of seconds or more. These steps are associated with

the computation of desirable resource efficient allocations or sets of allocations in the case of tradeoff enumeration. Steps highlighted in green represent the comprehensive exploration of NCF tradeoffs, a desirable feature of an orchestration program, and our first fundamental design principle.

By using these colors to highlight the corresponding aspects of each of the systems depicted in Figure 4.1, the differences between the three designs, and the complementary nature of EASO and LAW are made clear. EASO provides comprehensive tradeoff exploration, thus enabling a variety of proposed allocations to be instantiated on demand according to changing network conditions and operator objectives (e.g., NCF weightings), but it performs this relatively expensive computation within its typical process flow each time a new workload arrives. This expensive computation may hinder the ability of EASO to rapidly allocate tenant workloads during periods of high tenant demand. In contrast, the typical process flow of LAW contains no computationally expensive steps, as the vast majority of tenant workloads are pre-computed, thus permitting the fast allocation times necessary to keep up with high tenant demand. However, because LAW does not accomplish tradeoff exploration, the operator needs to specify NCF weights a priori so that desirable LAWs may be pre-computed. But these weights may change due to varying network conditions, tenant demand level, and SLA, requiring LAWs to be re-precomputed for the new weightings. Thus, LAW may be unsuitable for use in dynamic environments requiring frequent re-precomputation of LAWs.

EASO-LAW Synthesis provides the benefits of both EASO and LAW, with the drawbacks of neither. With EASO-LAW Synthesis, we use EASO in the pre-computation step (Step 1 of Figure 4.1(c)), to generate a set of desirable tradeoff LAWs for each of the vast majority of commonly encountered workloads. So in contrast to standalone LAW, which only stores a single precomputed LAW for each tenant workload requirement tuple, EASO-LAW Synthesis *stores a set of precomputed trade-off LAWs* for each workload. Hence, EASO-LAW Synthesis shares the comprehensive tradeoff exploration benefit of EASO, allowing the operator to dynamically accommodate changing objectives or network conditions on demand, without the need for LAW re-precomputation. Moreover, EASO-LAW Synthesis also offers the speed ad-

vantage of LAW, as LAW tradeoff precomputation is performed in advance of the typical process flow, which is free of steps involving time intensive computation.

Observation 1: EASO and LAW are complementary, and enable better orchestration when used together. By combining EASO and LAW as described above, we can achieve a synthesized data center orchestration solution that yields a net benefit greater than the sum of each of its parts when considered individually, because each compensates for the drawbacks of the other. LAW construction and allocation algorithms permit sub-second placements of large application workloads in large-scale data center networks, but by themselves, only a single application-aware allocation is considered. Complementary to LAW, EASO enables comprehensive exploration of the multi-NCF tradeoff space, but does not explicitly preserve application preferences, and the quality of the resultant nondominated solution set is limited by the “time budget” of the data center operator (i.e., the amount of time the operator can afford to let EASO run). However, when used together as a single orchestration solution, a set of desirable tradeoffs for common or existing application workloads can be discovered by EASO and stored as precomputed LAWs, thus enabling the fast and application-aware allocation of workloads corresponding to the on-demand tradeoff selection preferences of the data center operator, which are likely to change frequently due to dynamic network conditions and varying objective priorities.

Observation 2: EASO-LAW Synthesis enables efficient scaling existing workloads. A common and effective approach for scaling existing application workloads as end user demand increases is to simply allocate additional instances of the workload in demand, as done in the Network Function Center work [16]. Because EASO-LAW Synthesis archives previously encountered workloads, allocating additional instances of in-demand workloads can be achieved in fractions of a second using LAW allocation, enabling efficient workload scaling, even for relatively large workloads and physical topologies. Furthermore, if network conditions or operator priorities change during the life of a workload, because EASO has precomputed LAWs for a wide-range of desirable tradeoffs, the scaling of the workload (i.e., the placement of additional workload instances can be adjusted seamlessly to accommodate the new preferences and requirements of the operator.

Observation 3: Accurate application forecasting enables new workloads to be placed quickly and efficiently using EASO-LAW Synthesis. Although execution for thorough EASO tradeoff exploration (i.e., “long run”) may be too slow for dynamic large-scale data center environments, if the operator can forecast the types of workloads that are due to arrive in advance, he/she can run EASO prior to their arrival and precompute the efficient LAW tradeoffs in advance, thus allowing for fast allocation times when they arrive. In many ways, the challenges associated with achieving fast and efficient allocations using EASO-LAW Synthesis are similar to those associated with developing an effective cache “hit ratio”. Developing and implementing strategies for minimizing real-time EASO execution cycles seems to be a fertile ground for future work.

4.3 Future Work

In this section, we identify the limitations of our contributions, and then go on to outline the next steps in this research to address these limitations. Next, we identify additional research areas that seem a natural fit for our EASO-LAW Synthesis approach, and conclude the dissertation.

Overcoming limitations of EASO. EASO has two limitations. First, for a given tenant application workload requirement, it requires the operator to select a tradeoff from the nondominated set of candidate proposals. Due to the potentially vast multi-NCF tradeoff space (e.g., 843 different nondominated candidate solutions for the large-scale workload presented in in Section 2.5.3), requiring a human to examine each prospective tradeoff individually, without some degree of automated assistance, may require more time and resources than EASO uses to generate the tradeoffs in the first place. Hence, for tradeoff exploration to be more useful in practice, decision support systems and machine learning approaches could be used to reduce the number of tradeoffs available to the operator by automatically discarding those that are least likely to achieve operator goals while meeting tenant SLA, given current or forecasted network conditions.

Second, EASO tradeoff exploration may be time consuming. Depending on the workload size, number of applications, size of physical topology, and the quality and di-

versity of candidate proposals desired by the operator, EASO tradeoff exploration can take from tens of seconds to hundreds of minutes, or longer. However, an implementation of our proposal for EASO-LAW Synthesis should effectively overcome this limitation by precomputing a wide range of LAWs for various tenant workload specifications with respect to the multi-NCF tradeoff space.

Generalizing LAW Construction. In this work, LAWs are constructed to represent a single application (or application-tier). However, complex applications comprised of heterogeneous “tiers,” or sub-applications, such as the 200 VM, 5-tier tenant workload described in Section 2.5.3, are commonly found in practice [3, 4, 25]. Therefore, evaluating LAW construction (precomputation) and allocation for such complex multi-tier workloads in a real-world data center environment is a critical next step towards realizing the deployment of a LAW-based orchestration system. Also, a more granular resource contention classifier is desirable for multi-tier applications, since each individual application tier may have different resource contention characteristics, and the interactions between application tiers may be complex. For example, a recent study [13] demonstrated that resource contention for a large-scale Hadoop TeraSort workload varied based on the number of clusters (i.e., groups of VMs) used to execute the task. When two 4-VM clusters were used, the workload was bottlenecked by storage and network (DI) contention. However, when four 4-VM clusters were used to process the the same task, it was CPU contention that limited throughput.

Considering these challenges, we believe our proposed approach for EASO-LAW Synthesis is well positioned to handle the placement of complex multi-tier applications. We demonstrated in Section 2.5.3 the capability of EASO to generate a diverse set of candidate proposals for allocating complex multi-tier applications, such that each proposal satisfies tenant SLA requirements and is nondominated with respect to the multi-NCF tradeoff space in terms of operator objectives. By retaining the intra-application tier labels for each VM in an EASO proposal, a LAW may be constructed consisting of VMs from heterogenous application tiers, as depicted in Figure 4.2. And because LAW allocation preserves relative positioning of intra-application VMs within the physical infrastructure, Algorithm 3.3 can be used for placing such heterogeneous multi-tier LAWs without modification to achieve application-efficient workload place-

ments that meet tenant-operator SLAs. Statistical LAW allocation, however, becomes more challenging, since per VM allocation of certain application tiers may result in breach of SLA, but not for others. Thus, a promising area for future work for allocating heterogeneous multi-tier LAWs involves understanding and representing the criticality of each intra-application tier, and customizing the Statistical LAW process for VM removal and allocation to meet the requirements of each tier with respect to SLA.

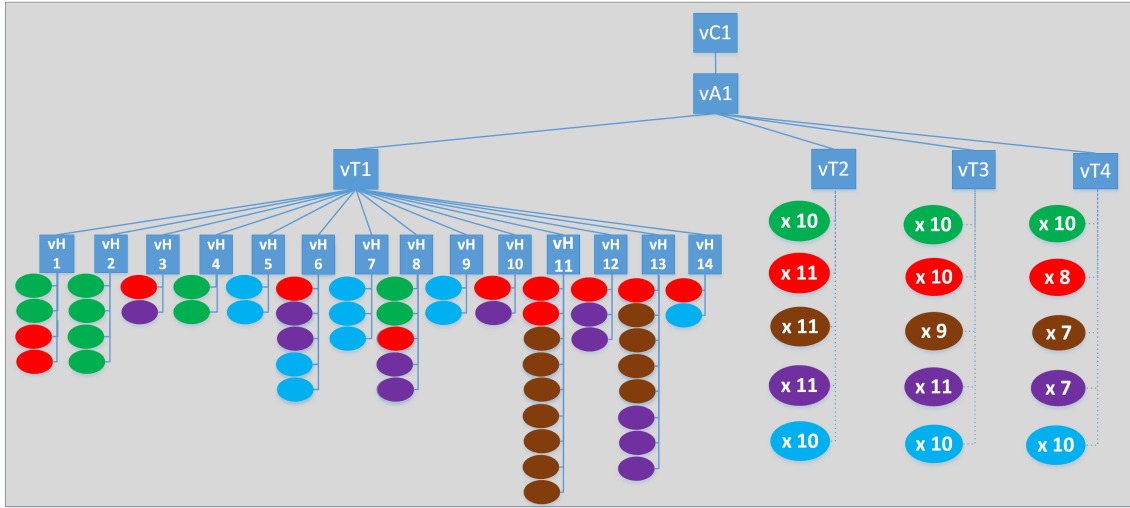


Figure 4.2: Heterogeneous multi-tier LAW constructed from a proposal generated by EASO for allocating the 200 VM, 5-tier (40 VMs per tier) tenant workload described in Section 2.5.3. VMs of tier T1 are represented by green ovals, tier T2 red ovals, tier T3 brown ovals, tier T4 purple ovals, and tier T5 blue ovals. For brevity, logical host allocations for each tier are only shown for those under the first logical ToR (vT1). For the other logical ToRs (vT2-vT4), the number of VMs of each respective tier allocated to logical hosts underneath them is represented by the numbers in the corresponding ovals.

Infeasibility is another limitation of LAW-based orchestration approaches. Clearly, if it is infeasible to allocate a LAW for some tenant workload, it becomes more challenging to guarantee that the tenant-operator SLA is satisfied by some workload allocation. And although our initial Statistical LAW evaluations presented in Section 3.5.3 demonstrate marginal impact on intra-application contention (e.g., RCI), determining whether or not the sub-optimality of some Statistical LAW placement constitutes a breach of SLA, is an open issue.

Of course, one way of combating the potential shortfalls of Statistical LAW allocation is to simply increase complete LAW feasibility. Thus, exploring alternative ways to increase complete LAW feasibility is yet another fertile area for future work. For instance, an offline approach to rearrange previously allocated LAWs in order to maximize the chance of feasibly allocating the next workload seems promising. Even better would be a method of precomputing such rearrangements to best accommodate a range of LAWs of different types and sizes.

Real world data center deployment. In this dissertation, our EASO and LAW workload allocation approaches are evaluated using simulated data center topologies and randomly generated workload traces. Although the range of parameters we use to define the topologies and generate the workload traces are comparable to existing work, our algorithms need to be implemented and evaluated on real data center networks and compared side by side against other solutions, such as Corybantic [10], Athens [11], Ostro [4], or CloudMirror [3], for allocating real tenant workloads with real SLAs, in order to validate our approaches in practice.

Emerging opportunities. In addition to the above, we see opportunities to apply our work to two other emerging network research areas: 1) automated NCF scaling, and 2) network service function chain (SFC) provisioning of virtual network functions (VNFs) for general network topologies.

1. Automatically adjusting the number of application components of workloads (i.e., auto-scaling) based on forecast user demand seems to be another promising area for applying our approach. Auto-scaling essentially adds another dimension to the tradeoff space by considering the number of workload VMs as a variable, rather than a hard constraint. Tradeoffs involving more or less VMs are interesting from an SLA perspective as tenants ideally would like to achieve maximum performance with the fewest number of VMs, whereas cloud providers may want to find opportunities to sell more VMs to tenants with the promise of increased capability.
2. SFCs are similar to application workloads in that they can be represented logically in terms of required resources, but SFCs are more complex as the VNFs that comprise them (i.e., application VMs that perform specific network functions, such

as stateful firewall, traffic load balancing, or web caching) not only require strict component positioning requirements like LAWs, but also require the SFC to send data through the VNFs in a specific order to preserve network policy. This is equivalent to establishing a partial or total ordering of individual VMs within a LAW, and then evaluating resource usage and application contentions subject to the data flow defined by it.

It is an exciting time to study the orchestration of data center networks and the diverse sets of applications hosted within them. These networks are becoming ever more important to the mission sets of governments and corporations, as well as the daily routines of individuals. The size and diversity of cloud services and the data centers that provide them seem poised to grow at unprecedented rates in the years to come. Future data center orchestration approaches will need to continue to scale to even larger networks and application workloads, while remaining flexible and extensible enough to handle a wide range of emerging operator objectives and network environments.

References

- [1] P. M. Mell and T. Grance, “Sp 800-145. the nist definition of cloud computing,” Gaithersburg, MD, United States, Tech. Rep., 2011.
- [2] Cisco Systems, “Cisco global cloud index: Forecast and methodology 2015-2020.” [Online]. Available: <http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf>
- [3] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma, “Application-driven bandwidth guarantees in datacenters,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 467–478, Aug. 2014. [Online]. Available: <http://doi.acm.org/10.1145/2740070.2626326>
- [4] G. Jung, M. A. Hiltunen, K. R. Joshi, R. K. Panta, and R. D. Schlichting, “Ostro: Scalable placement optimization of complex application topologies in large-scale data centers.” in *ICDCS*. IEEE, 2015, pp. 143–152. [Online]. Available: <http://dblp.uni-trier.de/db/conf/icdcs/icdcs2015.html#JungHJPS15>
- [5] P. Bodik, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, “Surviving failures in bandwidth-constrained datacenters,” in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM ’12. New York, NY, USA: ACM, 2012, pp. 431–442. [Online]. Available: <http://doi.acm.org/10.1145/2342356.2342439>
- [6] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, “Elastictree: Saving energy in data center networks,” in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 17–17. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1855711.1855728>
- [7] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, “Less is more: Trading a little bandwidth for ultra-low latency in the data center,” in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI’12. Berkeley, CA, USA: USENIX Association, 2012, pp. 19–19. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2228298.2228324>

- [8] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, “Fresco: Modular composable security services for software-defined networks,” in *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS’13)*, February 2013.
- [9] P. Sun, R. Mahajan, J. Rexford, L. Yuan, M. Zhang, and A. Arefin, “A network-state management service,” in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM ’14. New York, NY, USA: ACM, 2014, pp. 563–574. [Online]. Available: <http://doi.acm.org/10.1145/2619239.2626298>
- [10] J. C. Mogul, A. AuYoung, S. Banerjee, L. Popa, J. Lee, J. Mudigonda, P. Sharma, and Y. Turner, “Corybantic: Towards the modular composition of sdn control programs,” in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, ser. HotNets-XII. New York, NY, USA: ACM, 2013, pp. 1:1–1:7. [Online]. Available: <http://doi.acm.org/10.1145/2535771.2535795>
- [11] A. AuYoung, Y. Ma, S. Banerjee, J. Lee, P. Sharma, Y. Turner, C. Liang, and J. C. Mogul, “Democratic resolution of resource conflicts between sdn control programs,” in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT ’14. New York, NY, USA: ACM, 2014, pp. 391–402. [Online]. Available: <http://doi.acm.org/10.1145/2674005.2674992>
- [12] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM ’08. New York, NY, USA: ACM, 2008, pp. 63–74. [Online]. Available: <http://doi.acm.org/10.1145/1402958.1402967>
- [13] A. Gulati, N. Kodirov, and G. Kulkarni, “Quantifying the noisy neighbor problem in openstack,” in *2016 OpenStack Summit*, Austin, TX, Apr. 2016. [Online]. Available: <https://www.openstack.org/assets/presentation-media/ZeroStack-Austin-Presentation.pdf>
- [14] S. Srikanthan, S. Dwarkadas, and K. Shen, “Data sharing or resource contention: Toward performance transparency on multicore systems,” in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. Santa Clara, CA: USENIX Association, Jul. 2015, pp. 529–540. [Online]. Available: <https://www.usenix.org/conference/atc15/technical-session/presentation/srikanthan>
- [15] D. M. Volpano, X. Sun, and G. G. Xie, “Towards systematic detection and resolution of network control conflicts,” in *Proceedings of the Third Workshop*

- on *Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 67–72. [Online]. Available: <http://doi.acm.org/10.1145/2620728.2620745>
- [16] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo, “Towards making network function virtualization a cloud computing service,” in *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, 2015, pp. 89–97. [Online]. Available: <http://dx.doi.org/10.1109/INM.2015.7140280>
 - [17] M. Ehrgott and M. M. Wiecek, *Multiple Criteria Decision Analysis: State of the Art Surveys*. New York, NY: Springer New York, 2005, ch. Multiobjective Programming, pp. 667–708. [Online]. Available: http://dx.doi.org/10.1007/0-387-23081-5_17
 - [18] E. Zitzler, “Evolutionary algorithms for multiobjective optimization: Methods and applications,” 1999.
 - [19] V. Pareto, *Cours d'Economie Politique*. Genève: Droz, 1896.
 - [20] J. T. Piao and J. Yan, “A network-aware virtual machine placement and migration approach in cloud computing,” in *Proceedings of the 2010 Ninth International Conference on Grid and Cloud Computing*, ser. GCC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 87–92. [Online]. Available: <http://dx.doi.org/10.1109/GCC.2010.29>
 - [21] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, “Application-aware virtual machine migration in data centers.” in *INFOCOM*. IEEE, 2011, pp. 66–70. [Online]. Available: <http://dblp.uni-trier.de/db/conf/infocom/infocom2011.html#ShrivastavaZLJLB11>
 - [22] A. Gupta, L. V. Kale, D. Milojicic, P. Faraboschi, and S. M. Balle, “Hpc-aware vm placement in infrastructure clouds,” in *Proceedings of the 2013 IEEE International Conference on Cloud Engineering*, ser. IC2E '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 11–20. [Online]. Available: <http://dx.doi.org/10.1109/IC2E.2013.38>
 - [23] X. Meng, V. Pappas, and L. Zhang, “Improving the scalability of data center networks with traffic-aware virtual machine placement,” in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.
 - [24] J. Li, D. Li, Y. Ye, and X. Lu, “Efficient multi-tenant virtual machine allocation in cloud data centers,” *Tsinghua Science and Technology*, vol. 20, no. 1, pp. 81–89, Feb 2015.

- [25] X. Li and C. Qian, "Traffic and failure aware vm placement for multi-tenant cloud computing," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, June 2015, pp. 41–50.
- [26] D. S. Dias and L. H. M. Costa, "Online traffic-aware virtual machine placement in data center networks," in *2012 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2012, pp. 1–8.
- [27] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Online allocation of virtual machines in a distributed cloud," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 10–18.
- [28] A. Bairley and G. G. Xie, "Orchestrating network control functions via comprehensive trade-off exploration," in *IEEE NFV-SDN '16*. Palo Alto, CA: IEEE, Nov 2016.
- [29] A. Bairley and G. G. Xie, "An application aware approach to scalable online placement of data center workloads," submitted for publication.
- [30] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 242–253. [Online]. Available: <http://doi.acm.org/10.1145/2018436.2018465>
- [31] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," Tech. Rep., 2001.
- [32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.
- [33] J. Fernández and B. Tóth, "Obtaining an outer approximation of the efficient set of nonlinear biobjective problems," *Journal of Global Optimization*, vol. 38, no. 2, pp. 315–331, 2007. [Online]. Available: <http://dx.doi.org/10.1007/s10898-006-9132-y>
- [34] "Practical guide to cloud service agreements, version 2.0," Cloud Standards Customer Council, Tech. Rep., Apr. 2015.
- [35] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM TRANSACTIONS ON NETWORKING*, vol. 1, 1993.
- [36] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Transactions on Computers*, vol. 99, no. PrePrints, p. 1, 2013.

- [37] C. C. Lin, P. Liu, and J. J. Wu, “Energy-efficient virtual machine provision algorithms for cloud systems,” in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, Dec 2011, pp. 81–88.
- [38] R. E. Burkard and E. Çela, “Linear assignment problems and extensions,” Boston, MA, pp. 75–149, 1999. [Online]. Available: http://dx.doi.org/10.1007/978-1-4757-3023-4_2
- [39] H. W. Kuhn and B. Yaw, “The hungarian method for the assignment problem,” *Naval Res. Logist. Quart.*, pp. 83–97, 1955.
- [40] J. Munkres, “Algorithms for the assignment and transportation problems,” *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, March 1957.
- [41] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982.
- [42] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, “Vl2: A scalable and flexible data center network,” in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM ’09. New York, NY, USA: ACM, 2009, pp. 51–62. [Online]. Available: <http://doi.acm.org/10.1145/1592568.1592576>
- [43] 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,” in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM ’09. New York, NY, USA: ACM, 2009, pp. 63–74. [Online]. Available: <http://doi.acm.org/10.1145/1592568.1592577>

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California