Towards optimal orchestration of network control functions: an evolutionary approach

Bairley, Alan

http://hdl.handle.net/10945/48963

Downloaded from NPS Archive: Calhoun
TOWARDS OPTIMAL ORCHESTRATION OF NETWORK CONTROL FUNCTIONS: AN EVOLUTIONARY APPROACH

by

Alan Bairley
Geoffrey Xie
Matthew Carlyle

9 June 2016

Approved for public release; distribution is unlimited
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 objectives, is a challenging open problem. In this work, we formulate SDN orchestration as a multiobjective optimization problem, and present an evolutionary approach designed to explore the NCF tradeoff space 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 100s of switches, 1,000s of servers, and 10,000s of VM slots.
The report entitled “Towards Optimal Orchestration of Network Control Functions: An Evolutionary Approach” was prepared for the general public and funded by the Department of Defense Information Assurance Scholarship Program (DoD IASP).

Further distribution of all or part of this report is authorized.

This report was prepared by:

__________________________________________
Alan Bairley                             Geoffrey Xie

__________________________________________
Matthew Carlyle

Reviewed by:                           Released by:
__________________________________________
Peter J. Denning, Chairman              Jeffrey D. Paduan
Department of Computer Science           Dean of Research
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 objectives, is a challenging open problem. In this work, we formulate SDN orchestration as a multiobjective optimization problem, and present an evolutionary approach designed to explore the NCF tradeoff space 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 100s of switches, 1,000s of servers, and 10,000s of VM slots.
# Table of Contents

1 Introduction 1

2 Background and Related Work 3
  2.1 Comparison of Related Work 3

3 Design Decisions 9
  3.1 Rethinking Orchestration 9
  3.2 Design Considerations 13
  3.3 Proposed Work vs. State-of-the-Art 13

4 Problem Formulation 17
  4.1 Elements of SDN Orchestration 17
  4.2 Goal 20
  4.3 Multiobjective Optimization 20
  4.4 Specifying fine-grained operator intent 26

5 Evolutionary Approach to SDN Orchestration 31
  5.1 Comparison to general evolutionary algorithm 31
  5.2 Evolutionary Algorithm for SDN Orchestration (EASO) 33
  5.3 Complexity Analysis 35

6 Evaluation Criteria 37
  6.1 Distance 37
  6.2 Coverage 38
  6.3 Approximating $Y_p$ 41
  6.4 Evaluating $X_s$ 45
  6.5 EASO vs. State-Of-The-Art 47
  6.6 Evaluating Scalability of EASO 52
Acknowledgements

The authors would like to thank the DoD Information Assurance Scholarship Program (IASP) and the Office of Naval Research (ONR) for providing general support towards this work.
Automated network management approaches using software-defined networking (SDN) technology are particularly appealing for the management of large-scale enterprise and data center networks (DCNs). The sheer number of devices composing a typical DCN is on the order of thousands [1], making the manual management of such a network tedious and prone to error and inefficiency. SDN offers network administrators the promise of convenient, efficient, and accurate network management by enabling the development and deployment of automated network control functions (NCFs), i.e., SDN controller programs, that automatically perform some set of goal-oriented network configuration actions to achieve predefined high-level policy and performance objectives. Recently, a multitude of NCFs have been developed to achieve various objectives, such as bandwidth and fault tolerance joint optimization [1–3], power conservation [4], QoS control [5], and security services [6].

Orchestrating multiple NCFs in a utility-preserving and conflict-free manner is a challenging open problem. Prior work in SDN orchestration can be categorized as either 1) synchronization approaches, as in [7,8], or 2) resource allocation approaches, as in [9,10]. Synchronization approaches like Statesman [7] view the underlying network as a shared resource contested for by several NCFs, and seek to find a “stable” network state [8] that is free of both conflict and oscillation. These approaches are largely orthogonal to our work, as we are primarily interested in exploring the utility of various feasible configurations within the tradeoff space with respect to competing NCFs.

Hence, our work is motivated by existing resource allocation approaches, namely Corybantic [9] and Athens [10], which attempt to allocate requirements (e.g., VMs, flow rules, network services, etc.) to physical network resources (e.g., hosts, switches, middleboxes) in order to maximize the utility afforded to the network operator. Prior work [2,3,9–11], 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, as in [9–11], or by optimizing one objective function subject to the others cast as constraints as in [2,3].

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 [3,9,10], 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 [12–14], where our goal is to enumerate a diverse set of efficient solutions among competing NCFs, i.e. no solution can be improved in any objective without causing a degradation in at least one other objective.

In this work, our contribution is three-fold. First, we present a new problem formulation for SDN orchestration. Second, we describe a novel evolutionary approach\[^1\] for enumerating a wide range of efficient network states, scalable to topologies of thousands of hosts and hundreds of switches. Third, we present new metrics and use them to evaluate our approach.

\[^1\]Note that evolutionary (genetic) algorithms have been used to perform a variety of specialized network functions, like scaling services to accommodate traffic demands in [11], and SDN multi-path routing in [15], but not for high level network orchestration, as we propose in this work.
CHAPTER 2:
Background and Related Work

The crux of the SDN orchestration problem is achieving an “acceptable” network state. Statesman [7] and Volpano et al. [8] deem an acceptable state as one that is non-conflicting (or stable), whereas Corybantic [9] and Athens [10] define an acceptable state to be one that maximizes some high-level metric, such as cost effectiveness or number of votes. In this study, similar to Corybantic and Athens, we argue that it is not merely freedom of conflict, but rather optimality that is desired for an acceptable network state. However, in contrast to Corybantic and Athens, we argue that 1) a singular high-level metric as used in Corybantic and Athens (e.g. common currency, votes) is unsuitable for addressing complex and disparate operator requirements, 2) a substantial set of noninferior network states may be omitted from consideration if the set of candidate network states is limited exclusively to those generated by specialized NCFs as in Corybantic and Athens, and 3) use of local heuristics, like hill climbing, as used in Corybantic and Athens, will tend to converge towards local optima, but not necessarily global optima.

2.1 Comparison of Related Work

A recently developed state-of-the-art network-state management system called Statesman [7], essentially views the physical infrastructure as a shared resource contested for by multiple NCFs, and aims to provide conflict resolution by way of mutual exclusion. The Statesman system provides a shim layer through which all NCFs must communicate through in order to impose their goal-oriented configuration changes to elements of the underlying network infrastructure. To accomplish this, Statesman provides a global view of the network, represented as a graph of state variables, to each of the NCFs in contention for the network. The NCFs, armed with their observed state of the network, may propose requested changes via Statesman, which essentially provides a device-level synchronization service to ensure that conflicting NCF proposals are either resolved explicitly by priority-based locking when precedence matters, or implicitly by a “last writer wins” policy when precedence doesn’t matter. Additionally, Statesman also provides support for basic invariant checking of the network topology state graph, to ensure that changes proposed by some NCF do not inadvertently render the network inconsistent with respect to some set of predefined safety invariants (e.g. proposing a change that inadvertently disconnects the topology). But while Statesman provides a means of addressing our previously described synchronization problem concerning several NCFs in contention for the el-
lements of the underlying network infrastructure, it does not provide much insight into the addressing the higher-level resource allocation problem. Statesman provides a means to resolve conflict; it does not provide a method of assessing the relative utility of different network states with respect to some notion of optimality.

In Corybantic [9], another recent work pertaining to the SDN orchestration problem, the authors view SDN orchestration as a resource allocation problem, and attempt to reduce a multiobjective optimization problem to a singular one by 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. In contrast to Statesman, Corybantic can be viewed as a higher-level paradigm for addressing the SDN orchestration problem. Where Statesman compares competing NCF proposals for device-level resource conflicts and simultaneously implements non-conflicting proposals (or partial proposals), Corybantic evaluates each NCF proposal in terms of a higher-level metric and implements the best one. Corybantic offers a promising step forward in SDN orchestration by defining a high-level metric about which an optimal NCF proposal may be sought. 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 in practice. For instance, if an operator needs to maximize the fault tolerance of certain critical applications while minimizing core and aggregate bandwidth utilization of others, it would not make sense to represent both the degree of fault tolerance and the amount of bandwidth conserved in terms of the same high-level metric. 2) The range and diversity of candidate network states are limited to proposals made by specialized NCFs. It seems unrealistic to expect such specialized NCFs to offer network state compromise proposals that are mutually beneficial when all NCFs are considered, and hence, it is likely that using such an approach would leave a significant portion of the candidate state space unexplored. 3) Designing NCFs to make small changes to the current network state may aid in exploring more of the state space, but even so, because Corybantic uses a iterative selection process based upon a single greedy criterion (cost effectiveness), it is essentially a hill climbing approach, and is thus likely to converge towards a local optimum when the global objective function is non-convex with respect to incremental changes in the state space.

Another related work, Athens [10] builds on Corybantic by suggesting a family of voting procedures available to each NCF, enabling them to indicate a relative ordering of preferential proposals made by other modules. So while Athens effectively addresses the first of the Corybantic pitfalls described above by substituting a voting paradigm
in place of Corybantic’s universal currency metric, the Athens voting paradigm suffers from yet another difficulty. Namely, the Athens voting paradigm effectively transfers the network state evaluation and selection process to the NCFs themselves, since the best proposal is determined to be the one with the most votes. And although the Athens approach is distinct from Corybantic, as it allows disparate NCFs to weigh in on each other’s proposals, it still falls short in addressing complex and disparate operator requirements, and is subject to pitfalls 2) and 3) of Corybantic as described in the previous paragraph. Consider again the hypothetical scenario described in the previous paragraph concerning the operator that wishes to maximize fault tolerance of certain critical applications while minimizing the core and aggregate bandwidth utilization of others. The network state chosen for implementation is based upon the specialized NCF that wins the election, which may or may not be reflective of the operator’s intent.

Finally, one work by Volpano et al. [8] offers a unique approach in addressing the SDN orchestration problem, 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. These DFTs take elements from the network environment as input, and produce device configuration instructions, i.e. the network state, as output. The primary benefit of representing NCFs as DFTs is that it allows one to decide certain properties regarding the composition of multiple NCFs, such as whether there exists a network state that is mutually beneficial to all concerned NCFs. This “stable region”, as called by the authors, can be viewed as successful convergence to an acceptable network state: one that satisfies the requirements of each NCF under some set of network conditions. Clearly, the operation of a network within the scope of such a stable region among the NCFs that compose it is also free from the dangers of network oscillation. However, despite these benefits, there are three fundamental shortcomings with this approach. Firstly, representing an NCF as a DFT assumes a white box approach, requiring insight into the internal program logic of the NCF itself, which may not be feasible for proprietary NCFs. Secondly, like Statesman, the approach proposed by Volpano et al. does not define a notion of optimality about which distinct network states within the stable region may be evaluated relative to one another. Finally, using DFTs to represent NCFs limits their functionality to that of which can be expressed using regular languages, thus precluding certain types of NCFs from representation, such as those requiring an unbounded amount of counting.

Table 2.1 illustrates key differences between Statesman, Corybantic, Athens, and the DFT approach proposed by Volpano et al. In summary, Statesman, Corybantic, and Athens all present a shim layer representing the physical infrastructure and current
network state to the active NCFs, which in turn propose changes to the network state according to their specialized internal logic in an effort to achieve or optimize some operational objective. The set of changes proposed by an NCF, when applied to the current network state, comprise a new network state, called a “proposal”. Statesman does not use an iterative search process to find an acceptable proposal, but rather directly compares NCF proposals against one another for conflict. If separate NCF proposals are non-conflicting, they may be implemented concurrently, else the priority-based resolution mechanisms described previously are used to choose one for implementation. In contrast, Corybantic and Athens employ an iterative solicitation and selection process to search for the single best proposal among competing NCFs. The approach proposed by Volpano, et al. is unique in that it does not employ a shim layer like the others, but rather models NCF orchestration directly by computing the intersection of each NCF’s representative DFT.

<table>
<thead>
<tr>
<th>Orchestration Scheme</th>
<th>Problem Formulation</th>
<th>NCF Transparency</th>
<th>Operator Intent Specification</th>
<th>Solution Candidate(s)</th>
<th>Search Method</th>
<th>Solution(s) Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corybantic</td>
<td>Single-Objective Resource Allocation (SOP)</td>
<td>Black Box</td>
<td>NCF Granularity Only (Weighting)</td>
<td>NCF Generated</td>
<td>Local Heuristic (Hill Climbing)</td>
<td>Single – Most Cost Effective</td>
</tr>
<tr>
<td>Statesman</td>
<td>Device-Level Synchronization</td>
<td>Black Box</td>
<td>NCF-Granularity Only (Static Priority)</td>
<td>NCF Generated</td>
<td>Deterministic</td>
<td>Single – Non-Conflicting (Priority-Based Resolution)</td>
</tr>
<tr>
<td>DFT</td>
<td>Device-Level Synchronization</td>
<td>White Box</td>
<td>NCF-Granularity Only (Custom DFTs)</td>
<td>DFT Product Construction</td>
<td>Deterministic</td>
<td>Single – Stable (Non-Empty Intersection)</td>
</tr>
<tr>
<td>Athens</td>
<td>Single-Objective Resource Allocation (SOP)</td>
<td>Grey Box</td>
<td>NCF Granularity Only (Weighting)</td>
<td>NCF Generated</td>
<td>Local Heuristic (Hill Climbing)</td>
<td>Single – Most Votes</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of Related Work

Two significant challenges common to prior work are 1) achieving an acceptable network state and 2) defining and implementing operator intent. With respect to the former, Statesman and Volpano et al. attempt to achieve freedom of conflict, whereas Corybantic and Athens strive for optimality, as stated earlier.

In addressing the latter challenge of defining and implementing operator intent, the work proposed by Volpano et al. is unique in that operator intent may be directly incorporated into NCFs by modifying their respective DFT representations such that the parameters of the stable region are acceptable to the operator. However, this requires the operator to have complete knowledge of the NCF control logic, which seems unreasonable in practice. In contrast, Statesman, Corybantic, and Athens only
allow the specification of operator intent at a high level. Statesman allows network operators to assign relative priority values to NCFs, such that the proposal of the higher priority NCF is implemented in case of conflict. Corybantic and Athens incorporate operator intent by permitting the operator to assign weight values to each of the competing NCFs. Similar to static priority assignment, these weight assignments effectively establish a preferential ordering of NCFs by determining how much each NCF’s specific utility function (e.g. bandwidth conservation, power conservation, fault tolerance, etc.) contributes to the global objective function (e.g. cost effectiveness, number of votes) to be optimized.

So although the static priority assignment and weighting methods allow high-level NCF preference information to be supplied by network operators, we argue that such an approach is too course-grained to address complex operator requirements, like maximizing fault tolerance of certain critical applications while minimizing bandwidth used by others. In fact, none of the related work allows specification of operator intent at an application level, but is rather limited to capturing preference information between NCFs only. Considering that an operator may want to specify different NCF preferences for different applications, this limitation of the related work may be quite problematic in practice.
In this chapter, we motivate the design decisions of our approach in contrast to related work. First, we present a scenario motivating a new formulation of the SDN orchestration problem as a multiobjective optimization problem (MOP) comprised of the single-objective optimization problems (SOPs). Next, we state a set of design requirements for the SDN orchestration problem that is only partially fulfilled by related work. Finally, we present a comparison between our proposed work and the current state of the art for SDN orchestration.

3.1 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. 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.

Suppose three independent tenant applications R1, R2, and R3 have requirements <5, 50 Mbps>, <5, 100 Mbps>, and <5, 150 Mbps> respectively. The first value in the tuple represents the number of VMs required, and the second value represents the inter-VM bandwidth (BW) requirement. Suppose the underlying physical infrastructure has a binary 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) [1] over failures of ToR switches and physical servers with a preference for spreading VMs of each application across racks and servers within racks (Figure 3.1a). Specifically, an application’s ToR WCS is defined as the fraction of its VMs that survive a single worst case ToR switch failure. 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, calculated using the hose model, as in [16].
with a preference for consolidating VMs of the some another as possible (e.g. placing on same server or rack (Figure 3.1b). 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 3.1c). 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.

An orchestrator at minimum must solicit and rank candidate proposals from all NCFs. Clearly, the network cannot be in all three depicted states at once. Nor should it oscillate from one to another. If an operator knows a priori how to jointly model the three NCF 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 NCF 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. Conceivably, the operator may 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.

In prior work, Athens [10] and Corybantic [9], and 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 propos-
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. [9,10] then proceed by selecting the NCF proposal that maximizes some global utility function (e.g. votes, cost-effectiveness), by modeling SDN orchestration as a SOP and using a (greedy) hill climbing selection process. Without loss of generality, assume that the power conservation proposal is selected, i.e. assume it is the one that maximizes the global utility function. At this point, NCFs in [9,10] may iteratively suggest counter-proposals to the previously selected network state until an optimum criterion is reached. 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 network state 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 3.1c is selected initially, then it may be the case that while the network state in Figure 3.2a is reachable via a series of incremental counter-proposals, the network state in Figure 3.2b 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 [9,10] in that it is a small-scale change to some previous state 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 non-dominated network state alternatives may be generated and presented to the network operator for consideration. This is better than proposing a single “best” state, since operator requirements are likely to be fluid to accommodate rapidly changing network conditions.

Figure 3.3: Two compromises from successive rounds of recombination and mutation of non-dominated candidates.

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 3.2. In the context of a MOP with respect to the three operator-defined optimization criteria defined previously, these modified proposals dominate the original power conservation and survivability proposals. The mutated power conservation proposal offers better survivability than its predecessor, while maintaining the same power usage and mean link BW. The recombined survivability and BW conservation proposal offers better BW conservation and power usage than the original survivability proposal, while maintaining the same mean application ToR WCS. However, note that these proposals are non-dominated with respect to one another. Although the proposal in Figure 3.2b offers better ToR WCS, it uses more power and BW than the proposal in Figure 3.2a. Since neither proposal dominates the other, both should be maintained as desirable solution candidates. This is especially critical when the operator requirements are complex. The proposal in Figure 3.2b is more appropriate for mission critical applications that require maximum fault tolerance, while the proposal in Figure 3.2a 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, [9, 10] would discard one of these desirable proposals, leaving the operator with only one proposal, without presenting further desirable mutations (as shown in Figure 3.3) to the operator.
3.2 Design Considerations

Upon surveying the SDN orchestration problem space and the current state-of-the-art, we find significant limitations with each existing orchestration scheme that we hope to overcome in our design. In order to capture these limitations and show we intend to advance the state-of-the-art, we define the following set of design considerations:

- **Disparate Objectives**: The orchestration scheme should accommodate an arbitrary number of potentially disparate operator-specified objectives.
- **NCF as Black Box**: It is unreasonable to assume that NCF control logic (e.g. program code, etc) is accessible and interpretable by the orchestrator. However, it seems reasonable for NCFs to indicate levels of preference for different network states.
- **Application-level Specification**: Different (tenant) applications have different requirements. One application may require a certain level of survivability, or for survivability to be maximized, while another may require very low latency or an exceptional amount of bandwidth. Thus, network operators should be able to specify NCF preference with application-level granularity.
- **Offer Diverse Set of Tradeoffs**: The solution should not produce a single solution, but rather a diverse set of tradeoffs. This solution set should be presented to the network operator in a way that clearly identifies the tradeoff costs and benefits associated with each solution.

The fine-grained incorporation of operator intent is a key aspect in developing our novel approach that advances the state-of-the-art. Our proposed approach tackles this issue on two fronts: First, by allowing operators to specify boundary conditions and optimization criteria (Section 4.4) for each application in terms of the available NCFs, the set of network states returned by our orchestration program will be tailored towards their goals. Second, by returning a diverse set of tradeoff solutions, along with metrics pertaining to where each falls in the tradeoff space, operators are afforded a unique opportunity to select the most appropriate solution for their needs.

3.3 Proposed Work vs. State-of-the-Art

Before describing our proposed approach, we place our approach in the context of the SDN orchestration schemes described in Section 2.1. Table 3.1 describes the degree to which these relevant prior works address each corresponding design consideration.

In Table 3.1, Harvey Balls are used to represent the degree to which each SDN
orchestration scheme meets each of our design criteria. A full ball means the scheme fully met that criterion. An empty ball means that the scheme failed to meet that criterion. Other levels imply partial meeting of the criterion with possible caveats.

**Corybantic [Mogul et al. 2013]**

- **Disparate Objectives:** All objectives must be expressible in terms of a singular common currency, thus disparate objectives are not accommodated with this scheme.
- **Black Box:** Corybantic requires no insight into the internal control logic of individual NCFs, nor does it require NCFs to provide feedback regarding a proposed network state (i.e. black box).
- **App-Level Specification:** Although Corybantic allows the specification of certain intra-application requirements, such bandwidth required between the VMs of some tenant application, but it does not provide an interface for granular inter-application dependencies (between tenants), nor does it allow the operator to specify NCF preference information for each application (e.g. different NCF weightings for each application).
- **Offer Diverse Set of Tradeoffs:** Corybantic offers only one network state as a solution: the one that minimizes the common currency used.

**Statesman [Sun et al. 2014]**

- **Disparate Objectives:** Although high-level objectives are not known to the orchestrator, Statesman can effectively avoid conflict from disparate network
control functions.

- **Black Box**: Statesman requires no insight into the internal control logic of individual NCFs, nor does it require NCFs to provide feedback regarding a proposed network state (i.e. black box).

- **App-Level Specification**: The Statesman orchestrator is unaware of application topology, as it only takes inputs from the individual NCFs. Hence, it does not make special provisions for application requirements, nor does it provide an interface for granular specification of application dependencies or NCF preference information among applications.

- **Offer Diverse Set of Tradeoffs**: Statesman offers only one network state as a solution: the product of NCF proposals, by which conflict is resolved by predefined conflict resolution rules.

**DFT [Volpano et al. 2014]**

- **Disparate Objectives**: Although high-level objectives are not known to the orchestrator, the DFT approach can effectively avoid conflict among disparate network control functions.

- **Black Box**: The DFT approach requires complete insight into the internal control logic of individual NCFs (i.e. white box), as each NCF must be expressible as a DFT.

- **App-Level Specification**: The DFT approach is unaware of application topology, as it only takes inputs from the individual NCFs in the form of DFTs. Hence, it does not make special provisions for application requirements, nor does it provide an interface for granular specification of application dependencies or NCF preference information among applications.

- **Offer Diverse Set of Tradeoffs**: The DFT approach does not offer a network state as a solution, but rather indicates whether the product of competing NCF proposals is free of conflict.

**Athens [AuYoung et al. 2014]**

- **Disparate Objectives**: All objectives must be expressible in terms of a voting process where each competing NCF issues a number of votes for the proposals it prefers most. Although this effectively allows the specification of disparate objectives, the voting process itself may not accurately reflect operator intent. For example, a network state may be selected that did not receive any votes from the survivability NCF.

- **Black Box**: Athens requires NCFs to provide preference information regarding a proposed network configurations (i.e. grey box).

- **App-Level Specification**: Although Athens allows the specification of certain
intra-application requirements, such bandwidth required between the VMs of some tenant application, but it does not provide an interface for granular inter-application dependencies (between tenants), nor does it allow the operator to specify NCF preference information for each application (e.g. different NCF weightings for each application).

- **Offer Diverse Set of Tradeoffs:** Athens offers only one network configuration as a solution: the one that receives the most votes.

In the context of these existing and proposed SDN orchestration schemes, we find that our design meets our objectives and represents a unique point in the design space. In the following chapters, we describe an SDN orchestration scheme that will fully achieve each of the above design considerations.
CHAPTER 4: Problem Formulation

In this chapter, we formally define a set of terms to describe SDN orchestration in the context of this study. Next, we specify the SDN orchestration problem as an instance of the multiobjective optimization problem (MOP) using these terms. Finally, we describe the policy and performance graph (PPG), an intuitive abstraction for capturing fine-grained operator requirements, and demonstrate how an operator-specified PPG can be cast into a representative MOP.

4.1 Elements of SDN Orchestration

In order to model the SDN orchestration problem in a manner amenable to formulation as a MOP, we define the following terms:

- Define physical infrastructure as a tree, $T = (H, S, E)$, where each leaf node $h \in H$ represents a physical host (hypervisor) with an associated resource capacity for hosting VMs (e.g. CPU/Memory/Storage), each internal node $s \in S$ represents a physical switch with an associated resource capacity for storing a number of flow rules, and each edge $e \in E$ represents a physical network link with an associated resource capacity of its maximum flow rate. For the purposes of distinguishing between core, aggregation, and top-of-rack (ToR) switches, we use the notation $S_c \subset S$ to represent core switches, $S_a \subset S$ to represent aggregation switches, and $S_t \subset S$ to represent ToR switches.

- Define application requirements or tenant requirements as a PPG (Section 4.4) $G = (M, E)$, where each vertex $m \in M$ represents a tenant application and associated constraints, $c_m$, and each $e \in E$ represents permitted communication between applications subject to communication constraint $c_e$. The PPG uses similar concepts as [2] and [17] to provide a unified abstraction for describing application (or tenant) constraints and optimization criteria. Application constraints $c_m$, may include (but are not limited to) the number of VMs, physical resources (per VM), and level of fault tolerance (among VMs) required by some application $m$, while communication constraints $c_e$ include bandwidth, load balancing factor, and access control rules required between some two applications $u, w \in M$, where $e = (u, w)$.

- Define the resource capacity $r$ as a resource vector of size $|T|$ where each element $r_t$ represents the resource capacity, i.e. total resources available, at element $t$ of
Define the resource constraint $c_T$, as an implicit problem constraint requiring that feasible network states do not require more resources than the resource capacity $r$, i.e. $c_T(x) = TRUE \iff \forall_{t \in T}. r_t - \sum_{m=1}^{M} x_{m,t} \geq 0.$

Define virtual machine (VM) $v \in V(m)$ as the basic unit of allocation comprising an individual $i \in I$. Let $V(m) : m \in M(G)$ represent the set of VMs required by application $m$, and let $V = \{ V(m_1) \cup V(m_2) \cup ... \cup V(m_{|M|}) \}$ represent the set of all VMs.

Define an allocation $i \in I$ as an allocation matrix of size $|M| \times |H|$ where $i_{m,h}$ denotes the number of VMs of application $m \in M(G)$ allocated to host $h \in H(T)$.

Define a network state $x \in X$ as a resource matrix $x = q(i)$ where the mapping function $q$ takes an allocation $i \in I$ as input and produces an $x = |M| \times |T|$ resource matrix as output, representing the resources required by allocation $i$. $x_{m,t}$ denotes a value representing the amount of resources required by application $m$ at element $t$ of the physical infrastructure $T$. If the resources required by some network state $x$ satisfy constraints of both PPG $G$ and $T$, then $x$ is said to be feasible.

Define a configuration as an instantiation of network state $x \in X_f$ across the actual physical infrastructure represented by $T$.

Define the feasible set or feasible region of network states, $X_f \subseteq X$ as the set of network states that satisfy application constraints $c_m$, communication constraints $c_e$, and resource constraint $c_T$, i.e. $x \in X_f \iff (\forall_{m \in M(G)}. c_m(x) = TRUE) \land (\forall_{e \in E(G)}. c_e(x) = TRUE) \land c_T(x) = TRUE$.

Define network control function (NCF) $n \in N$ as an abstraction that takes PPG $G$ and physical infrastructure $T$ as input and produces an allocation $i \in I$ as output. In this study, we assume that each NCF $n$ has a corresponding utility function, $f_n$, where $x = q(i)$ is a network state and $y = f(x)$ represents the utility of $x$. By casting NCFs as single-objective functions, we can represent each NCF as a SOP.

Define a utility vector, $y \in Y$, as an $N$-dimensional vector comprised of the utility function values of each of the $N$ NCFs for some network state $x \in X$, i.e. $y = (f_1(x), f_2(x), ..., f_N(x))$. 

the physical infrastructure $T$. 

- Define the resource constraint $c_T$, as an implicit problem constraint requiring that feasible network states do not require more resources than the resource capacity $r$, i.e. $c_T(x) = TRUE \iff \forall_{t \in T}. r_t - \sum_{m=1}^{M} x_{m,t} \geq 0.$

- Define virtual machine (VM) $v \in V(m)$ as the basic unit of allocation comprising an individual $i \in I$. Let $V(m) : m \in M(G)$ represent the set of VMs required by application $m$, and let $V = \{ V(m_1) \cup V(m_2) \cup ... \cup V(m_{|M|}) \}$ represent the set of all VMs.

- Define an allocation $i \in I$ as an allocation matrix of size $|M| \times |H|$ where $i_{m,h}$ denotes the number of VMs of application $m \in M(G)$ allocated to host $h \in H(T)$.

- Define a network state $x \in X$ as a resource matrix $x = q(i)$ where the mapping function $q$ takes an allocation $i \in I$ as input and produces an $x = |M| \times |T|$ resource matrix as output, representing the resources required by allocation $i$. $x_{m,t}$ denotes a value representing the amount of resources required by application $m$ at element $t$ of the physical infrastructure $T$. If the resources required by some network state $x$ satisfy constraints of both PPG $G$ and $T$, then $x$ is said to be feasible.

- Define a configuration as an instantiation of network state $x \in X_f$ across the actual physical infrastructure represented by $T$.

- Define the feasible set or feasible region of network states, $X_f \subseteq X$ as the set of network states that satisfy application constraints $c_m$, communication constraints $c_e$, and resource constraint $c_T$, i.e. $x \in X_f \iff (\forall_{m \in M(G)}. c_m(x) = TRUE) \land (\forall_{e \in E(G)}. c_e(x) = TRUE) \land c_T(x) = TRUE$.

- Define network control function (NCF) $n \in N$ as an abstraction that takes PPG $G$ and physical infrastructure $T$ as input and produces an allocation $i \in I$ as output. In this study, we assume that each NCF $n$ has a corresponding utility function, $f_n$, where $x = q(i)$ is a network state and $y = f(x)$ represents the utility of $x$. By casting NCFs as single-objective functions, we can represent each NCF as a SOP.

- Define a utility vector, $y \in Y$, as an $N$-dimensional vector comprised of the utility function values of each of the $N$ NCFs for some network state $x \in X$, i.e. $y = (f_1(x), f_2(x), ..., f_N(x))$. 

18
• Define an individual \( i \in I \) as an allocation.

• Define the individual space \( I \) as the set of all allocations.

• Define a decision vector \( x \in X \) as a network state.

• Define the decision space \( X \) as the set of all network states.

• Define a objective vector \( y \in Y \) as a utility vector.

• Define the objective space \( Y \) as the set of all utility vectors.

• Define orchestrator or meta-controller as a high-level control program that communicates with a set of network control functions in order to generate a set of network states within the feasible region, and subsequently implements one of them, i.e. deploys a representative configuration \( C \) of some network state \( x \in X_f \) across the physical infrastructure \( T \).

• Define NCF transparency as the degree of which internal NCF control logic is exposed to the orchestrator. Broadly, the level of NCF transparency can be classified into one of three categories: 1) White box: all internal NCF control logic is made available to the orchestrator, as in [8], for example. 2) Grey box: a limited amount of NCF feedback is made available to the orchestrator, such as the NCF voting mechanism in Athens, or an exposed NCF utility function. 3) Black box: no feedback regarding NCF utility for a given network state is provided to the orchestrator. In this case, any optimization criteria, i.e. the SOPs comprising the MOP, must be specified by the operator.
4.2 Goal

Ultimately, the goal of an SDN orchestration scheme is to select an allocation $i \in I$ that yields a feasible network state $x = q(i) : x \in X_f$, such that the utility $y = f(x)$ best suits the needs of the network operator.

4.3 Multiobjective Optimization

Real-world problems often involve concurrent optimization of several incomparable and potentially competing objectives. Although single-objective optimization is usually well-defined, such is not the case for MOPs. Instead of an absolute optimal solution, there is rather a set of alternative trade-offs, generally known as Pareto-optimal solutions. These solutions are optimal in a broader sense that no other solutions in the state space are superior to them when all objectives are considered. Here, we describe the basic principles and concepts of multiobjective optimization, which correspond to the mathematical formulations most widespread in multiobjective optimization literature, as presented in [18], [19], [20], and [21]. We then formally define the SDN orchestration problem in terms of this framework.

**Def. 1 (Multiobjective Optimization Problem)** A general MOP includes a set of $M$ decision variables, a set of $N$ objective functions, and a set of $L$ boolean constraints. Objective functions and constraints are functions of the decision variables. Maximality is defined in terms of Pareto-optimality (Def. 6), i.e. $y$ is maximal iff it is Pareto-optimal. Note that there may exist several Pareto-optimal solutions for a single MOP instance; in such cases, each Pareto-optimal solution represents a different tradeoff with unique merits.

\[
\begin{align*}
\text{maximize} \quad & y = f(x) = (f_1(x), f_2(x), ..., f_N(x)) \\
\text{subject to} \quad & z = c(x) = (c_1(x) \land c_2(x) \land ... \land c_L(x)) = \text{TRUE} \\
\text{where} \quad & x = (x_1, x_2, ..., x_M) \in X \\
& y = (y_1, y_2, ..., y_N) \in Y
\end{align*}
\]

and $x$ is the decision vector, $y$ is the objective vector, $z$ is the constraint vector, $X$ is denoted as the decision space, and $Y$ is referred to as the objective space.

Without loss of generality, a maximization problem is assumed here. For minimization or mixed minimization and maximization problems, the corresponding definitions are similar to those presented in this section. Since $y$ is a vector, the $N$ objective functions comprising it must be expressed terms of a universal metric, and thus reduce to a
SOP, for the traditional notion of maximality to be defined. However, in contrast to Corybantic and Athens, we argue that the MOP for SDN orchestration should not be reduced to a SOP, as such a formulation limits the diversity of requirements that can be expressed, which ultimately constrains the options available to the network operator. Therefore, in the context of this study, we define maximality in terms of Pareto-optimality (Def. 6). Next, we cast the SDN orchestration problem as an instance of the general MOP (Def. 1).

Def. 2 (SDN Orchestration Problem as MOP) Represent a network state \( x = (x_1, x_2, \ldots, x_M) \) as a decision vector representing the amount of network resources required by each of the \( M \) applications with respect to the physical infrastructure, \( T \), where \( x_m \) denotes the total resources required by \( m \) (resource vector) and \( x_{m,t} \) denotes the resources required by application \( m \) at element \( t \in T \). Represent the objective vector \( y = f(x) = (f_1(x), f_2(x), \ldots, f_N(x)) \) as a vector comprised of the utility functions of each of the \( N \) NCFs, where \( f_n(x) \) represents the utility of \( x \) with respect to NCF \( n \). Represent constraints on application \( m \) and communication path \( e \) as boolean expressions, \( c_m \) and \( c_e \), respectively, and represent the resource constraint as \( c_T \), where:

\[
x \in X_f \iff (\forall m \in M(G).c_m(x) = TRUE) \land (\forall e \in E(G).c_e(x) = TRUE) \land c_T(x) = TRUE.
\]

Now we can state the orchestration problem as the following MOP instance:

\[
\text{maximize } y = f(x) = (f_1(x), f_2(x), \ldots, f_N(x))
\]
\[
\text{subject to } x \in X_f
\]

In the context of our SDN orchestration problem, cast as a MOP, each individual \( i \in I \) represents a possible solution to the problem at hand: an allocation \( i \) such that the objective vector \( y = f(x) : x = q(i) \) is nonnegative and constraint vector \( z = c(x) : x = q(i) \) is satisfied. Note that an individual is not a decision vector but rather encodes it based on an appropriate structure. In the case of our problem, the structure of an individual is simply an allocation of VMs \( v \in V \) to hosts \( h \in H \) and the set of all possible allocations constitutes the individual space \( I \). The quality of an individual with respect to the optimization task is represented by a scalar value, called fitness. Because the quality is related to the objective functions and constraints, an individual must first be decoded before its fitness can be calculated. This situation is illustrated in Figure 4.1. Given an individual \( i \in I \), a mapping function \( q \) encapsulates the decoding algorithm to derive the network state (decision vector) \( x = q(i) \) from \( i \). In the context of our problem, the network state \( x \) is represented as a vector of length
$M$, where $M$ is the number of applications, and each component (resource vector with respect to $T$) represents the physical resources of $T$ required by application $m \in M$.

Consider again the example presented in the previous section. Bandwidth conservation ($f_1$): the inverse of mean link bandwidth, power conservation ($f_2$): the inverse of power used, and ToR WCS ($f_3$), are to be maximized subject to resource constraint $c_T$, application constraints $c_1: numVMs = 5$, $c_2: numVMs = 5$, $c_3: numVMs = 5$, and communication constraints $c_{1,1}: reqBW = 50$, $c_{2,2}: reqBW = 100$, $c_{3,3}: reqBW = 150$. If each of these three optimization functions can be expressed by the network operator in terms of the same universal metric, then we only have to solve a SOP, since the optimal solution is simply the one that maximizes the this value. However, the difficulty in solving MOPs is the common situation when the individual optimization criteria corresponding to distinct of objective functions are sufficiently different, and not amenable to expression by way of a common metric. In this case, if the set of objectives are conflicting and cannot be optimized simultaneously, then a satisfactory trade-off must be found. In our example, bandwidth conservation and fault tolerance are generally competing, and while power conservation generally competes with fault tolerance, it is somewhat orthogonal to bandwidth conservation. Depending on the network operator’s requirements, an intermediate solution (medium bandwidth conservation, medium fault tolerance, medium power conservation), such as one depicted in Figure 3.3(b), might be an appropriate trade-off. Thus, a different notion of optimality is required for MOPs.
**Def. 3 (Feasible Set)** The feasible set $X_f$ is defined as the set of decision vectors $x$ that satisfy the constraints $c(x)$:

$$X_f = \{ x \in X \mid c(x) = TRUE \}$$

The image of $X_f$, i.e., the feasible region in the objective space, is denoted as $Y_f = f(X_f) = \bigcup_{x \in X_f} \{ f(x) \}$.

In single-objective optimization, the feasible set is totally ordered according to the objective function $f$: for two solutions $a, b \in X_f$, either $f(a) \geq f(b)$ or $f(b) \geq f(a)$, where the goal is to find the solution(s) that gives the maximum value of $f$ [22]. However, in the case of multiple objectives, $X_f$ is, in general, not totally ordered, but partially ordered [14]. Consider again the example proposals presented previously. The proposal depicted in Figure 3.2(a) is better than the proposal depicted in Figure 3.1(c), as it provides better tenant ToR WCS while using the same amount of power and mean link bandwidth. Similarly, the proposal depicted in Figure 3.3(b) is preferable to the proposal depicted in Figure 3.2(b), because it requires less power and mean link bandwidth while providing the same tenant ToR WCS. We can express this situation mathematically by extending the relations $=$, $\geq$, and $>$ to objective vectors by analogy to the single-objective case.

**Def. 4** For any two objective vectors $u$ and $v$,

- $u = v \iff \forall i \in \{1, 2, ..., N\} : u_i = v_i$
- $u \geq v \iff \forall i \in \{1, 2, ..., N\} : u_i \geq v_i$
- $u > v \iff u \geq v \land u \neq v$

The relations $\leq$ and $<$ are defined similarly.

Using this notion, the following relationships hold in the context of the proposals represented by the previous example figures: Figure 3.3(b) $>$ Figure 3.2(b), Figure 3.2(b) $>$ Figure 3.1(a), and as a consequence, Figure 3.3(b) $>$ Figure 3.1(a). However, when comparing Figure 3.3(b) and Figure 3.3(a), neither can be said superior, since Figure 3.3(b) $\not\geq$ Figure 3.3(a) and Figure 3.3(a) $\not\geq$ Figure 3.3(b). Although the solution associated with Figure 3.3(a) uses less power and mean link bandwidth, it does not provide as much fault tolerance (ToR WCS) to tenants as the solution represented by 3.3(b). Hence, two decision vectors, $a$ and $b$, can have three possibilities with MOPs regarding the $\geq$ relation (in contrast to two with SOPs): $f(a) \geq f(b)$, $f(b) \geq f(a)$, or $f(a) \not\geq f(b) \land f(b) \not\geq f(a)$. To classify these different situations, the following symbols and terms are used:
Def. 5 (Pareto Dominance) For any two decision vectors $a$ and $b$,

\[
\begin{align*}
    a &> b \quad (a \text{ dominates } b) & \text{iff } f(a) > f(b) \\
    a &\succeq b \quad (a \text{ weakly dominates } b) & \text{iff } f(a) \geq f(b) \\
    a &\sim b \quad (a \text{ is indifferent to } b) & \text{iff } f(a) \not> f(b) \land f(b) \not< f(a)
\end{align*}
\]

The definitions for a minimization problem ($\prec$, $\preceq$, $\sim$) are defined similarly.

Based on the concept of Pareto dominance, the optimality criterion for MOPs can be introduced. If some decision vector, $a$, is not dominated by any other decision vector, then this means that $a$ is optimal in the sense that it cannot be improved in any objective without causing a degradation in at least one other objective. Such solutions are referred to as Pareto-optimal; or noninferior [23].

Def. 6 (Pareto Optimality) A decision vector $x \in X_f$ is nondominated regarding a set $A \subseteq X_f$ iff

\[ \nexists a \in A : a > x \]

If it is clear within the context which set $A$ is meant, then it is simply left out. Moreover, $x$ is said to be Pareto-optimal iff $x$ is nondominated regarding $X_f$.

With respect to the set of proposals represented in Figures 1 - 4, note that the proposals depicted in Figure 3.1(b), Figure 3.2(a), Figure 3.3(a), and Figure 3.3(b) are Pareto-optimal solutions (i.e. network states with corresponding nondominated decision vectors). They are indifferent to each other, and it is this characteristic of indifference that distinguishes MOPs from SOPs: unlike SOPs, in MOPs, there is no single optimal solution, but rather a set of optimal trade-offs. None of these can be identified as better than the others unless preference information is included (e.g. a ranking of the objectives.)

The entirety of all Pareto-optimal solutions is called the Pareto-optimal set; the corresponding objective vectors form the Pareto-optimal front or surface.

Def. 7 (Nondominated Sets and Fronts) Let $A \subseteq X_f$. The function $p(A)$ gives the set of nondominated decision vectors in $A$:

\[ p(A) = \{ a \in A \mid a \text{ is nondominated regarding } A \} \]

The set $p(A)$ is the nondominated set regarding $A$, the corresponding set of objective vectors $f(p(A))$ is the nondominated front regarding $A$. Furthermore, the set $X_p =$
\( p(X_f) \) is called the Pareto-optimal set, and the set \( Y_p = f(X_p) \) is denoted as the Pareto-optimal front.

Although the Corybantic and Athens approaches produce locally nondominated network states with respect to the pool of candidates considered for selection, i.e. \( A \) in Def. 7, the quality of such states are limited. The range and diversity of the candidate network states contained within \( A \) is constrained. Since candidate network states are not proposed by the orchestrator in Corybantic and Athens, certain noninferior compromises may not be explored and hence remain precluded from selection. As result, the network states produced by Corybantic and Athens may be sub-optimal in comparison to an approach that considers a wider range of candidate network states for consideration.

The Pareto-optimal set consists of those solutions that are globally optimal. However as with SOPs, there may also be local optima which constitute a nondominated set within a certain neighborhood. This corresponds to the concepts of global and local Pareto-optimal sets introduced in [24].

**Def. 8** Consider a set of decision vectors \( A \subseteq X_f \).

1. The set \( A \) is denoted as a local Pareto-optimal set iff
   \[
   \forall a \in A : \exists x \in X_f : x \succ a \land \| x - a \| < \epsilon \land \| f(x) - f(a) \| < \delta
   \]
   where \( \| \cdot \| \) is a corresponding distance metric and \( \epsilon > 0, \delta > 0 \).

2. The set \( A \) is called a global Pareto-optimal set iff
   \[
   \forall a \in A : \not\exists x \in X_f : x \succ a
   \]

Essentially, the criteria for local optimality are less stringent than those for global optimality, since local optima need only to be nondominated with respect to some specified range in the decision and objective space, the size of which is denoted by \( \epsilon \) and \( \delta \), respectively. For example, in Corybantic and Athens, a local heuristic known as hill climbing is used to iteratively select NCF proposals that increase value of some universal utility function. However, if the utility function is non-convex with respect to the search space, then the final network state selected, although guaranteed to be locally nondominated, may be globally inferior. NCFs in Corybantic and Athens operate by proposing small, incremental changes to the current network state, and
it may be the case that the value of the universal utility function cannot be further improved by some small change, but rather requires a drastic change to provide an increase in utility. Such a state constitutes a local Pareto-optimal solution, but is in fact inferior with respect to the global state space. With this notion of local vs. global Pareto-optimality in mind, note that a global Pareto-optimal set does not necessarily contain all of the local Pareto-optimal solutions, and that every global Pareto-optimal set is also a local Pareto-optimal set.

4.4 Specifying fine-grained operator intent

In this section we formally define the policy and performance graph (PPG) abstraction used in the scope of this work and subsequently provide an example using this abstraction to describe the tenant requirements described in Section 3.1.

Define a PPG as a directed graph, \( G = (M,E) \), where each node \( m \in M \) represents an application comprised by a set of VMs, similar to the nodes of the tenant application graph (TAG) abstraction presented in [2], and are labeled by the pair, \( (B,O) \), where \( B \) is a set of boundary conditions corresponding to operator-specified constraints and \( O \) is a set of optimization criteria representing the NCF utility functions to be optimized for a particular application. Each edge \( e = (u,v) : u,v \in M \) represents permitted communication from application \( u \) to application \( v \), similar to the edges of the policy graph abstraction (PGA) presented in [17], and is labeled by the pair, \( (AC,NP) \), where \( AC \) is a set of access control conditions, e.g. permitted flows, and \( NP \) is a set of network performance requirements, such as minimum required bandwidth or maximum allowable latency between applications. Self-directed arcs are used to represent permitted communication and network performance requirements among the VMs comprising the same application.

For example, to represent the set of constraints and optimization criteria of the example presented in Section 3.1, let \( AC \) represent permitted flows, \( NP = (reqBW) \), \( B = (numVMs,numSlots) \), and \( O = (torWCS(f_1), bwCons(f_2), powerCons(f_3)) \), where \( reqBW \in [0,MAX\_BW] \) represents required communication bandwidth, \( numVMs \in [1,MAX\_VMS] \) represents the number of VMs required by an application, \( numSlots \in [1,MAX\_SLOTS] \) represents the number of physical server slots required by each VM comprising the application, \( torWCS \) represents the mean application ToR WCS and corresponds to \( f_1 \) of the MOP, \( bwCons \) represents the inverse of mean link bandwidth utilization and corresponds to \( f_2 \) of the MOP, and \( powerCons \) represents the inverse of total power usage and corresponds to \( f_3 \) of the MOP. To illustrate how an operator-specified PPG translates into an operator-specified MOP, consider the PPG depicted in Figure 4.2 representing the example tenant requirements of Section 3.1.
In Figure 4.2, observe that tenant requests, i.e. application requests, R1, R2, and R3 each require five VMs to occupy one physical server slot a piece. It is also required that VMs comprising R1, R2, and R3 are permitted to communicate internally ($AC = \ast$ denotes all flows permitted) and require guaranteed intra-application bandwidth of 50 Mb/s, 100 Mb/s, and 150 Mb/s respectively. Also, see that each of the objective functions, $f_1$, $f_2$ and $f_3$ are to be optimized for each application. Finally, because the PPG allows the specification of optimization criteria and constraints for each application, the corresponding MOP objective functions must be capable of evaluating the utility of a network state at an application level of granularity. To accomplish this, we add an additional argument, $U \subseteq M$ to the specification of MOP objective functions, such that the evaluation of $f(x, U)$ represents the utility of network state $x$ with respect to the set of optimized applications, $U$. In the case of the PPG depicted in Figure 4.2, the set of applications to be optimized for each criteria is $\{R1, R2, R3\}$, or all applications. Hence, this PPG represents the following MOP:

**Def. 9 (MOP represented by Figure 4.2)**

\[
\begin{align*}
\text{maximize} \quad y &= f(x) = (f_1(x, \{R1, R2, R3\}), f_2(x, \{R1, R2, R3\}), f_3(x, \{R1, R2, R3\})) \\
\text{subject to} \quad z &= (\forall_{m \in M}. c_m(x) \land \forall_{e \in E}. c_e(x) \land c_T(x)) = \\
& \quad (numVMs(x, R1) = 5 \land numVMs(x, R2) = 5 \land numVMs(x, R3) = 5 \land \\
& \quad numSlots(x, R1) = 1 \land numSlots(x, R2) = 1 \land numSlots(x, R3) = 1 \land \\
& \quad reqBW(x, (R1, R1)) = 50 \land reqBW(x, (R2, R2)) = 100 \land \\
& \quad reqBW(x, (R3, R3)) = 150 \land c_T(x)) = TRUE \\
\end{align*}
\]

where $f_1$ (torWCS), $f_2$ (bwConsevation), and $f_3$ (powerConsevation) are defined in $O$; $numVMs$ and $numSlots$ are defined in $B$; and $reqBW$ is defined in $NP$.

The strength (and novelty) of the PPG lies in its ability to represent diverse requirements unique to distinct applications or sets of applications. For example, suppose
that application R1 was deemed mission-critical, thus requiring optimal survivability and performance, with a minimum required ToR WCS value of 0.5. To support this new constraint, \( torWCS(f_1) \in [0,1) \) is added to the boundary conditions, yielding \( B = (numVMs, numSlots, torWCS) \). Under these circumstances, a network operator may choose to modify the PPG appropriately, perhaps producing the graph depicted in Figure 4.3.

![Figure 4.3: Example PPG specified to maximize performance and survivability of R1](image)

Note that this graph is identical to the one specified in Figure 4.2, with the exception of the added \( torWCS(f_1) \geq 0.5 \) boundary condition to R1 and application-specific optimization criteria. Namely, the application R1 is exclusively specified for ToR WCS optimization, and exclusively excluded from mean-link bandwidth and power use optimization. Given the scenario of R1 as a mission critical application, this specification of the PPG should make sense intuitively. The network operator desires to maximize the survivability of R1 without consideration for the bandwidth and power used by the VMs comprising it, while also ensuring that in the event of a worst-case ToR switch failure, at least half of R1’s VMs remain operational. Thus, a translation of this PPG produces a similar MOP to Def. 9, with following changes: \( f(x) = (f_1(x, \{R1\}), f_2(x, \{R2, R3\}), f_3(x, \{R2, R3\})) \), and constraint \( f_1(x, \{R1\}) \geq 0.5 \) is added to the set of application constraints for R1 \( (c_1) \).

![Figure 4.4: Example PPG specified for three-tiered web application](image)

Finally, the PPG can also represent complex communication requirements between applications. For instance consider a possible operator-specified PPG to represent the
requirements of three-tiered web application, where R1 represents a web server application tier, R2 represents the application logic tier, and R3 represents the backend database tier, as depicted in Figure 4.4.
Prior MOP work [13,25,26] shows that an evolutionary approach, 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 will be avoided, and 2) a wider range of solution candidates will be considered vs. a greedy approach. In this chapter, we present such an evolutionary algorithm, termed Evolutionary Algorithm for SDN Orchestration (EASO), to solve the MOP problem formulated in the previous chapter.

5.1 Comparison to general evolutionary algorithm

A key difference in our algorithm compared to a general evolutionary algorithm, such as Zitzler’s Strength Pareto Evolutionary Algorithm 2 (SPEA2) [25], is with regard to the nature of recombination and mutation steps, as well as the number of new candidate solutions produced by these steps. In SPEA2, the recombination and mutation steps are destructive, i.e. children produced by recombination replace parents, mutated individuals produced by mutation replace the originals. And although this destructive behavior may not affect the quality of the next generation’s external set (assigned in Step 2 - before destruction occurs), it does adversely affect the diversity of the population, since the parents of recombined children are destroyed and hence cannot be mutated.

In EASO (Algorithm 5.1), the recombination and mutation processes are nondestructive, i.e. the parents of recombined children and the original mutation candidates are preserved. And because these processes are non-destructive, we can perform selection for recombination and mutation deterministically, i.e. equivalent to setting $p_r$ and $p_m$ to 1 in SPEA2. Additionally, this algorithm produces multiple mutations for each mutation candidate, as opposed to just one per individual. The number of mutations performed per candidate is proportional to the number of NCFs, |N|. Specifically, our algorithm aims to produce two mutants per NCF utility function $n$: one that improves the value of $n$, and one that degrades the value of $n$. We hypothesize that such a mutation scheme will ultimately lead to faster convergence towards a more diverse nondominated set. For recombinations, we aim to recombine candidates that are sufficiently different, as opposed to simply using random selection. Recombining sufficiently dissimilar candidate solutions should help explore regions of the state space that may be unreachable via mutation alone. Also, to help simplify the algorithm and further speed up convergence, we modify the selection step by selecting all
and only members of the updated external set into the mating pool.

5.1.1 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 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 3.1, the following NCF-specific mutation heuristics are used within the MUTATE procedure. Here, we use the term \textit{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.
5.2 Evolutionary Algorithm for SDN Orchestration (EASO)

Algorithm 5.1 (Evolutionary Algorithm for SDN Orchestration)

Input: 
\begin{itemize}
    \item $G$ (performance policy graph)
    \item $N$ (set of NCF utility functions)
    \item $T$ (physical infrastructure tree)
    \item $L$ (size of external set)
    \item $K$ (maximum number of generations)
    \item $s$ (mutation size)
\end{itemize}

Output: \begin{itemize}
    \item $A$ (nondominated set)
\end{itemize}

Step 1: **Initialization**: Let $P_k$ and $\overline{P}_k$ denote the population and external set at generation $k$, respectively. Set initial population $P_0 = \emptyset$, $k = 0$. Set initial external set $\overline{P}_0 = \emptyset$.

For each $n \in N$ do
\begin{itemize}
    \item[a)] Solicit the $n^{th}$ NCF for its proposed allocation, $i \in I$.
    \item[b)] Set $P_0 = P_0 + \{i\}$.
    \item[c)] Set $P_0 = P_0 + \text{GENERATE}(L - |N|, G, N, T)$: Repeat $L - |N|$ times:
        Choose $i \in I$ according to an initialization scheme that takes application requirements and network policy as input, set $P_0 = P_0 + \{i\}$. The goal of this function is to produce a diverse set of individuals within the feasible region. A naive scheme could simply place the VMs randomly throughout the infrastructure. Alternatively, the remaining population members could be produced by varying some constraint to help guide initialization towards a diverse set of members.
\end{itemize}

Step 2: **Fitness Assignment**: Set the temporary external set $\overline{P}' = P_k \cup \overline{P}_k$.

Calculate the fitness value $F$ of individuals in $\overline{P}'$ using a fitness assignment scheme based upon Pareto dominance and crowding distance, such as those described in [25, 26], which assign a fitness value equal to the number of candidate solutions that dominate it plus a value proportional to its proximity to other solutions (inverse of crowding distance). Note that a lower fitness value corresponds to a stronger candidate solution.
Step 3: **Update of external set / Termination:**

a) Remove individuals from $P'$ whose corresponding decision vectors are weakly dominated regarding $q(P')$, i.e. while there exists a pair $(i,j) \in P'$ and $q(i) \succeq q(j)$ do $P' = P' - \{j\}$.

b) If $k \geq K$, then return $A = p(q(P'))$

c) If $|P'| > L$, then remove the $|P'| - L$ lowest fitness individuals from $P'$ by using a truncation algorithm such as the one described in [25] or [26].

d) Else if $|P'| < L$, then add the $L - |P'|$ highest fitness individuals to $P'$ from $P_k$.

e) Set $P_{k+1} = P'$

Step 4: **Selection:** Set mating pool $P' = P'$

Step 5: **Recombination:** Set the child pool $P'' = \emptyset$.

Let $x = q(i)$ represent the network state corresponding to individual $i \in P'$, and let objective vector $y = f_1(x), f_2(x), ..., f_N(x)$, where $f_n(x)$ represents the $n^{th}$ NCF utility function.

For each $n \in N$ do

Sort $P'$ in order of $f_n(x)$

For $a$ in 1 to $\lfloor |P'|/2 \rfloor$ do

$b = |P'| + 1 - a$

$j = \text{RECOMBINE}(P'[a], P'[b], G, T)$

Set $P'' = P'' + \{j\}$

Step 6: **Mutation:**

For each $n \in N$ do

For each individual $i \in P'$ do

$j = \text{MUTATE}(i, s, G, n, T, TRUE)$: Mutate $i$ by relocating at most $s$ VMs to improve $f_n(x)$

$l = \text{MUTATE}(i, s, G, n, T, FALSE)$: Mutate $i$ by relocating at most $s$ VMs to degrade $f_n(x)$

Set $P'' = P'' + \{j,l\}$

Step 7: **Loop:** Set $P_{k+1} = P''$, $k = k + 1$, and go to Step 2.
5.3 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 $|T|$). For nontrivial values of $|T|$, totally enumerating $X_p$ can be intractable [3]. In these cases, $X_s$ is rather an inner approximation [13] of $X_p$.

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

Time Complexity: For candidate utility evaluation, each of the $\frac{5NL}{2}$ states are evaluated by $N$ utility functions, and each utility function evaluates at most $|T| \cdot L$ elements of each state, for a resultant complexity of $O(N^2 \cdot L \cdot |T|)$. The loop of Step 1.d runs at most $L$ times, fitness assignment requires at most $O(N \cdot L^2)$ comparisons using the scheme presented in [26], 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 $O(N^2 \cdot L^2 \cdot |T| \cdot K \cdot s)$. 

35
THIS PAGE INTENTIONALLY LEFT BLANK
In this chapter, we describe our criteria for evaluating an SDN orchestration scheme, based upon the insight that the SDN orchestration problem may be cast as an instance of the MOP specified in Chapter 4. Because the SDN orchestration problem is a MOP instance, there is not a single optimal solution, but rather a set of solutions, each representing an optimal outcome for different tradeoff considerations. When all possible tradeoffs are considered, such a set is called the Pareto-optimal set \( X_p \), and its image \( Y_p = f(X_p) \) is known as the Pareto-optimal front. Thus, ideally we seek to compare some set of candidate solutions \( X_s \) produced by an SDN orchestration scheme against \( X_p \). Clearly, if \( f(X_s) = f(X_p) \) then the orchestration scheme produced the entire Pareto-optimal set, and therefore achieved the best results.

However, in the general case, an orchestration scheme will likely produce \( X_s : f(X_s) \neq f(X_p) \), and in such cases it becomes necessary to define evaluation criteria relating \( X_s \) to \( X_p \), in order to provide a basis of evaluation. To provide such criteria, we propose metrics for evaluating the distance and coverage of \( X_s \) with respect to \( X_p \). Such metrics are consistent with Zitler’s criteria [13]. Namely, 1) the distance of the resulting nondominated front \( f(p(X_s)) \) to the Pareto-optimal front \( Y_p \) should be minimized, and 2) the spread (coverage) of \( f(p(X_s)) \) should be maximized, i.e. for each objective a wide range of values should be covered by the set nondominated solutions. Coverage measures how well a solution set illuminates performance tradeoffs among NCFs, whereas distance measures how "close" a solution set is to Pareto-optimality. We also discuss the general intractability of enumerating \( X_p \), and the necessity of generating an outer approximation \( Y_a \) of \( Y_p \) for non-trivial problem inputs (e.g. several NCFs, large-scale infrastructure, complex PPG).

### 6.1 Distance

The distance (Def. 10) of candidate solution objective vectors \( y \in Y_s \), to the nearest point in the reference image \( Y_p \), is representative of its quality. Specifically, the quality of a candidate solution objective vector \( y \) is inversely proportional to its distance to the nearest member of \( Y_p \). A distance of zero indicates that \( \exists y_p \in Y_p : y = y_p \), and hence \( y \) is a member of the reference image, i.e. \( \text{distance}(y, Y_p) = 0 \implies y \in Y_p \); there exists no feasible objective vector that dominates it. Conversely, a large distance indicates that \( y \) is a relatively poor candidate solution, i.e. one or more feasible objective vectors dominate it. Therefore, candidate solutions with lower distances are more desirable.
**Def. 10 (Distance)** The distance, distance\((y, Y_p)\), from some feasible objective vector \(y \in Y_f\) to the nearest point \(w\) in the reference image \(Y_p\):

\[
distance(y, Y_p) = \min_{w \in Y_p} dist(y, w)
\]

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

Within the scope of our example problem, the distance function \(dist(y, w)\) returns the Euclidean distance \((L^2\text{ norm})\) between some two points \(y, w \in Y\). However, depending on the specific problem instance and the goals of the network operator, other distance measurements, such as Manhattan distance \((L^1\text{ norm})\) or maximum distance \((L^\infty\text{ norm})\), may be more appropriate.

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.

### 6.2 Coverage

Although the distance metric presented in previous section provides a representative measure of quality regarding the optimality of the solutions within \(X_s\), it does not reflect the diversity of the solutions, i.e. the area of the tradeoff space covered by \(X_s\). It is important to note that the current state-of-the-art SDN orchestration solutions (Section 2.1) that attempt to optimize the performance of multiple NCFs (e.g. Corybantic, Athens, etc.) are only concerned about minimizing distance, i.e. they seek to enumerate a single “optimal” solution without regard for the (potentially vast) tradeoff space, as illustrated in Figure 6.1. A novel aspect of our approach with respect to the current state-of-the-art is the enumeration of a set of nondominated tradeoffs. Hence, to evaluate the area of the tradeoff space covered by an SDN orchestration solution set \(X_s\), we propose the coverage metric (Def. 11), which represents the fraction of points in the reference image \(Y_p\) that are “covered”, i.e. nearest to objective vectors in \(Y_s = f(p(X_s))\). Hence, solution sets with higher coverage values are more desirable.
**Def. 11 (Coverage)** The coverage, \( \text{coverage}(Y_s, Y_p) \), is the fraction of points in the reference image \( Y_p \) that are nearest to objective vectors \( y \in Y_s \):

\[
\text{coverage}(Y_s, Y_p) = \frac{|\bigcup_{y \in Y_s} \text{nearest}(y)|}{|Y_p|}
\]

where nearest is defined in Def. 12.

**Def. 12 (Nearest)** The nearest point, \( \text{nearest}(y, Y_p) \), is the nearest point \( w \) in the reference image \( Y_p \) to some objective vector \( y \in Y_s \):

\[
\text{nearest}(y, Y_p) = \arg\min_{\forall y_p \in Y_p} \text{dist}(y, w)
\]

If multiple points in \( Y_p \) are equidistant away from \( y \), then one is arbitrarily selected.

Clearly, high coverage values for some set of objective vectors \( Y_s \), e.g. \( \text{coverage}(Y_s, Y_p) = 1 \), imply that the range and distribution of \( Y_s \) is similar to \( Y_p \), as depicted in Figure 6.2(a). However, note that for lower values of coverage, significantly different solution set distributions may produce similar coverage values, as illustrated in Figure 6.3. Therefore, in cases where attaining high coverage values are infeasible (e.g. enumerating a solution set as large as \( Y_p \) is not tractable), it may be helpful to use other metrics, such as range (Def. 13) and largest gap (Def. 14), in addition to coverage, to better elucidate distribution characteristics. In such cases, an ideal coverage area is one such that the points comprising it are distributed with minimal largest gap over the maximum range, i.e. points span the feasible range as uniformly as possible. Figure 6.2(b) depicts an ideal low coverage scenario.
Figure 6.1: Single solution orchestration scheme vs. Pareto-optimal front

(a) High coverage scheme vs. Pareto-optimal front

(b) Low coverage scheme (uniform) vs. Pareto-optimal front

Figure 6.2: High (a) and Low (b) coverage: uniform solution distribution
(a) Low coverage scheme (end-heavy) vs. Pareto-optimal front
(b) Low coverage scheme (mid-heavy) vs. Pareto-optimal front

Figure 6.3: Non-uniform solution distributions for same (low) coverage values

Def. 13 (Range) The range of $Y_s$ in dimension $n$:

$$\text{range}(Y_s, n) = \max_{\forall \vec{y} \in Y_s} y_n - \min_{\forall \vec{y} \in Y_s} y_n$$

Def. 14 (Largest Gap) The largest gap of $Y_s$ between any pair of consecutive points in dimension $n$:

$$\text{largestGap}(Y_s, n) = \max_{\forall \vec{y}, \vec{w} \in Y_s} |y_n - w_n|$$

where $\vec{y}$ and $\vec{w}$ are consecutive points in dimension $n$.

6.3 Approximating $Y_p$

The general distance and coverage metrics described in the previous section may be used to evaluate any MOP instance related to SDN orchestration or otherwise. Although simple and intuitive, these metrics require enumeration or approximation of the Pareto-optimal front $Y_p$ in order to be evaluated. In this section we discuss 1) the general intractability of total enumeration methods, as well as ways to reduce complexity for specific problem instances, and 2) the challenges and pitfalls associated with approximation methods.
6.3.1 Total Enumeration Methods

For trivial problems it may be possible to completely enumerate the set of all feasible network states $X_f$ via brute force in order to obtain $Y_p = f(p(X_f))$. However, note that in the general case, under the assumptions that 1) an application $m \in M(G)$ consists of $n_m = |V(m)|$ indistinguishable VMs, 2) network states are distinguished solely by allocations of VMs to $k = |H(T)|$ hosts, and 3) the resources on each host are unconstrained, i.e. each host can support an unlimited number of VMs; there are at most $\binom{n_1+k-1}{k-1} \times \binom{n_2+k-1}{k-1} \times \ldots \times \binom{n_M+k-1}{k-1}$ distinguishable network states. Hence, total enumeration is intractable in the general case. To see this, observe that the number of distinguishable allocations of $n$ VMs to $k$ hosts is equivalent to the number of ways that $n$ indistinguishable balls can be placed into $k$ bins [27].

Although total enumeration is generally intractable, for specific problem instances there may be ways to reduce the search space such that enumerating the complete set of Pareto-optimal states is tractable. For instance, consider the example problem presented in Section 3.1. In this example scenario, we have $n_1, n_2, n_3 = 5$, and $k = 16$, so by the above analysis, there are at most $\binom{20}{15} \times \binom{20}{15} = 3.727 \times 10^{12}$ distinguishable network states. Of course, because each host is limited to supporting only two VMs, many of these states will be infeasible, hence allowing us to reduce the search space. Furthermore, by examining the utility function of each NCF, we may be able to further reduce the search space if it is clear that all Pareto-optimal states $x_p \in X_p$ must satisfy some constraint, or can be enumerated by some heuristic.

To illustrate this point, consider the specific example problem instance: NCFs $N = \{n_1$ (WCS), $n_2$ (BW), $n_3$ (Power)$\}$, PPG $G$ (Figure 4.2), Phys Inf $T$ (Figure 3.1). Upon examination it becomes clear that 1) any allocation of $V$ VMs to different sets of hosts of the same cardinality under the same ToR yield the same value of $f$, i.e. they are indistinguishable. Next, 2) observe that $f_1$, $f_2$, and $f_3$ may each achieve their maximum value by using only three ToR switches. Furthermore, since $f_2$ and $f_3$ both favor the minimizing the subtree of the physical infrastructure allocated, we can see that the entire Pareto-optimal set consists of states using three or less ToR switches, i.e. any feasible network state using four ToR switches is dominated by some feasible state using three or less ToR switches. Thus, for our example problem, we can reduce the search space of enumerating $X_p$ from $\binom{20}{15} \times \binom{20}{15} \times \binom{20}{15} = 3.727 \times 10^{12}$ to $\binom{7}{2} \times \binom{7}{2} \times \binom{7}{2} = 9261$, by first enumerating the possible allocations of $n_m = 5$ VMs (balls) to $k = 3$ ToR switches (bins) for each application $m$, and subsequently allocating these VMs to hosts under the appropriate ToR switch using a “first-fit” packing heuristic. So in the case of our particular example problem instance, we can tractably enumerate the Pareto-optimal front $Y_p$ (Figure 6.4).
6.3.2 Approximation Methods

From the previous discussion it should be clear that total enumeration of $X_p$ is not tractable in the general case, for if it was then our proposed algorithm (Algorithm 5.1) for solving the SDN orchestration problem would be irrelevant, as we could simply enumerate the set of all Pareto-optimal solutions, $X_p$.

As an example of such intractable complexity, consider the data center infrastructure $T$, and PPG $G$ simulated in Ostro [3]. In Ostro, $T$ consists of a total of $|H| = 2400$ hosts, each capable of hosting up to 16 VMs, and $|S_t| = 150$ ToR switches (16 hosts per rack). $G$ consists of a 5-tiered web application, where each tier may be represented a separate PPG application $m \in M$, consisting of up to 40 VMs (200 VMs total). So even if we did have a scenario similar to our example problem, i.e. $N$ NCFs such that 1) any allocation of $V$ VMs to different sets of hosts of the same cardinality under the same ToR are indistinguishable, and 2) all Pareto-optimal states $x_p \in X_p$ use less than or equal to the number ToR switches required for maximum WCS (40 in case of Ostro), we would still have to enumerate and evaluate up to $\binom{79}{39}^5 = 4.49 \times 10^{113}$ network states, which is clearly intractable.

Thus, in order to evaluate an SDN orchestration scheme $s$ using our criteria in the context of a non-trivial problem, such as the data center scenario considered in the Ostro, it becomes necessary to develop some method specific to a particular problem instance for tractably constructing a representative approximation of $Y_p$. Ideally, a
(a) $f_2$ (BW Cons.) component of $Y_s$ (blue) and $Y_a$ (red). Each is plotted with respect to the discrete values of $f_1$ (WCS).

(b) $f_3$ (Power Cons.) component of $Y_s$ (blue) and $Y_a$ (red). Each is plotted with respect to the discrete values of $f_1$ (WCS).

Figure 6.5: Nondominated Set $Y_s$ (EASO) vs. Outer Approx. of Pareto-optimal front $Y_a$ (OA) for the example scenario presented in Section 3.1, depicted as two separate two-dimensional component plots.

representative approximation $Y_a$ of the Pareto-optimal front $Y_p$ is an outer approximation comprised of points uniformly distributed throughout a similar range. If $Y_a$ is not an outer approximation of $Y_p$, then it is possible for some feasible objective vector $y \in Y_f$ to dominate some point $y_p \in Y_p$, hence invalidating our distance metric when $Y_a$ is substituted for $Y_p$ in Def. 10. And if the range of $Y_a$ is not similar to that of $Y_p$, or if the points throughout that range are distributed dissimilarly, e.g. non-uniformly, then the coverage metric may not reflect accurate coverage of the actual (but intractable) $Y_p$, when $Y_a$ is substituted for $Y_p$ inDefs. 11 and 12.

The remaining challenge lies in tractably constructing a representative outer approximation $Y_a$ of $Y_p$. In the previous section, we established that enumeration of the feasible decision space $X_f$ is generally intractable. In light of this, there appears to be two alternative approaches for tractably constructing $Y_a$. The first involves relaxing certain problem constraints, e.g. application constraints $e_m$, communication constraints $e_c$, or the resource constraint $e_0$, in order to enable the use of some general allocation (bin-packing) heuristic, e.g. first-fit, next-fit, max-rest, etc., at a granularity tractable with complexity of the problem instance (perhaps more coarsely grained, if necessary). The difficulty in using this method deals with proving that $Y_a$ is indeed an outer approximation of $Y_p$; all cases in which the chosen heuristic(s) may yield a sub-optimal value must be compensated for with relaxed constraints in order to ensure that any objective vector produced as by an approximation heuristic cannot be dominated by some feasible objective vector $y \in Y_f$. 

44
Alternatively, $Y_a$ may be constructed using the well known “constraint method” found in MOP literature [12]. Specifically, for the example scenario, we can formulate each ordered pair of NCF utility functions, $(f_1, f_2), (f_1, f_3), (f_2, f_1), (f_2, f_3), (f_3, f_1), (f_3, f_1)$ as a biobjective optimization 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 $C_{f_1} = \{0.05, 0.10, \ldots, 0.60\}$), 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 points comprising $Y_a$ are potentially infeasible, and hence $Y_a$ is an outer approximation. Figure 6.5 depicts such an outer approximation for our example problem. The main challenge in using this method stems from the complexity and opacity of individual NCFs. If NCF utility functions cannot be inferred and/or the complexity associated with enumerating some portion of the discrete range of NCF values is prohibitive, then using this method may be infeasible. Note that tractably constructing a tight outer approximation for nonlinear BOPs is a challenging problem in and of itself [28].

6.4 Evaluating $X_s$

Once the Pareto-optimal front $Y_p$ has been enumerated, or a representative outer approximation $Y_a$ has been constructed for a particular problem instance (NCFs $N$, PPG $G$, Phys Inf $T$), evaluating the set of candidate solutions $X_s$ produced by some SDN orchestration scheme is straightforward: minimal distance (Def. 10) and maximum coverage (Def. 11) are desired. Note that for a given amount of computational effort, there is likely a tradeoff between distance and coverage here, i.e. if high coverage is required, than achieving low distance may be intractable, and vice versa.

<table>
<thead>
<tr>
<th>Solution Set Image</th>
<th>$Y_s$</th>
<th>$Y_p$</th>
<th>$Y_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance ($Y_s$)</td>
<td>(0.0021, 0, 0.030)</td>
<td>(0, 0, 0)</td>
<td>(0.0292, 0, 0.205)</td>
</tr>
<tr>
<td>(Mean, Min, Max)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance ($Y_p$)</td>
<td>(0.0313, 0, 0.205)</td>
<td>(0.0292, 0, 0.205)</td>
<td>(0, 0, 0)</td>
</tr>
<tr>
<td>(Mean, Min, Max)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverage ($Y_s$)</td>
<td>1</td>
<td>1</td>
<td>13/14</td>
</tr>
<tr>
<td>Coverage ($Y_p$)</td>
<td>13/14</td>
<td>13/14</td>
<td>1</td>
</tr>
<tr>
<td>Size</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Range ($f_1, f_2, f_3$)</td>
<td>(0.600, 0.510, 0.205)</td>
<td>(0.600, 0.510, 0.205)</td>
<td>(0.600, 0.510, 0.175)</td>
</tr>
<tr>
<td>Largest Gap ($f_1, f_2, f_3$)</td>
<td>(0.0667, 0.055, 0.175)</td>
<td>(0.0667, 0.055, 0.175)</td>
<td>(0.0667, 0.055, 0.175)</td>
</tr>
</tbody>
</table>

Table 6.1: Evaluation metric values for $Y_s$, $Y_p$, and $Y_a$, for the example problem.
For the example problem instance presented in Section 3.1, we enumerated $Y_p$ (Figure 6.4) and constructed a representative $Y_a$ (Figure 6.5) for separate evaluations against the nondominated front $Y_s = f(p(X_s))$ enumerated by our proposed orchestration solution (EASO: Algorithm 5.1). Figure 6.6 depicts $Y_s$, $Y_p$, and $Y_a$, each plotted in three dimensions. distance, coverage, range, and largest gap metrics for $Y_s$, $Y_p$, and $Y_a$ are presented in Table 6.1.

Note that distance and coverage are dependent upon the reference set, e.g. $Y_p$ or $Y_a$. Thus, in cases when enumerating $Y_p$ is intractable, it is important to realize that poor distance or coverage values may be a result of a loose outer approximation $Y_a$, rather than a poor solution set image $Y_s$. This phenomenon manifests itself somewhat in our example problem: although the EASO solution set image $Y_s$ fully covers $Y_p$, it does not fully cover $Y_a$ i.e. $\text{coverage}(Y_s, Y_p) = 1$ whereas $\text{coverage}(Y_s, Y_a) < 1$. This discrepancy is the consequence of an outer approximation that is tight to $Y_p$ in some places, but loose in others, as depicted in Figure 6.6.

In contrast, range and largest gap are absolute metrics that characterize set distributions. These metrics may be particularly illuminating in scenarios where there are low coverage values and/or large distances between $Y_s$, $Y_p$, and $Y_a$. Consider again the low-coverage examples depicted in Figure 6.3. Although the solution sets depicted in Figure 6.3 (a) and Figure 6.3 (b) may have equivalent coverage and distance...
values, they can be distinguished by their respective range and largest gap values. Additionally, the reference sets themselves may be evaluated using range and largest gap metrics in order to shed light on their distribution characteristics. For example, note in Table 6.1 that $\text{range}(Y_p, f_3) > \text{range}(Y_a, f_3)$. This is an indication that the outer approximation $Y_a$ is not tight in dimension $f_3$ with respect to $Y_p$. Hence, in this scenario it may be possible to improve the approximation algorithm to provide a tighter outer approximation of $Y_p$.

6.5 EASO vs. State-Of-The-Art

In this section we present a comparison between EASO and the current state-of-the-art, namely Corybantic and Athens, in the context of our example problem. Statesman and the DFT approach are not considered here, since we view their respective formulations of the SDN orchestration problem as orthogonal to ours.

6.5.1 Qualitative Comparison

Although Corybantic and Athens adopt a formulation of the SDN orchestration problem similar to our work, their hill-climbing approach used to explore the search space generates only a single preferred candidate solution, where the particular subtree of the state space explored and the resultant candidate solution selected depends upon the respective weightings of each NCF utility function, i.e. common currency cost (Corybantic) or number of votes (Athens). Furthermore, note that the search algorithm used by Corybantic/Athens is equivalent to a greedy version of Algorithm 5.1 that we refer to as GASO (Algorithm 6.1). In GASO, the external set consists only of a single member, $i_k$, the highest fitness individual discovered after $k$ generations, Corybantic’s common currency (or Athens’ voting method) is substituted for the SPEA2 fitness function, the recombination step is omitted, and termination occurs when mutation fails to produce any feasible solution candidates with higher fitness than $i_k$. The mutation heuristics used to improve some $f_n$ in GASO are the identical to those used in EASO. Note that the GASO mutation step to improve some $f_n$ is equivalent to the solicitation of incremental NCF counterproposals in Corybantic/Athens.

Hence, by 1) adding a specification for an $|N|$-dimensional NCF weight vector $\vec{w}$ to our SDN orchestration problem input, i.e. PPG $G$, NCFs $N$, physical infrastructure $T$, and 2) substituting the global objective fitness function $F_{\text{global}}$ (Def. 15) for the SPEA2 fitness function, GASO simulates Corybantic/Athens on EASO inputs. In GASO, $F_{\text{global}}$ is to be maximized: for each generation $k$, the best feasible mutation of $i_k$, i.e. the one that yields the highest value of $F_{\text{global}}$, is selected as $i_{k+1}$ for the next generation.
Def. 15 (Global Objective Fitness Function) The global objective fitness function, $F_{\text{global}}$:

$$F_{\text{global}}(x) = w_1(f_1(x)) + w_2(f_2(x)) + ... + w_N(f_N(x))$$

where $\vec{w}$ is an $|N|$-dimensional NCF weight vector and $w_n$ represents the weighting value (constant) for NCF $n$.

Algorithm 6.1 (Greedy Algorithm for SDN Orchestration (GASO))

Input: $G$ (performance policy graph)  
$N$ (set of NCF utility functions)  
$T$ (physical infrastructure tree)  
$K$ (maximum number of generations)  
$s$ (mutation size)

Output: $A$ (nondominated set)

Step 1: Initialization: Let $P_k$ and $i_k$ denote the population and best individual at generation $k$, respectively. Set initial population $P_0 = \emptyset$, $k = 0$. Set initial best individual $i_0 = \text{EMPTY\_ALLOC}$.

For each $n \in N$ do

a) Solicit the $n^{th}$ NCF for its proposed allocation, $i \in I$.

b) Set $P_0 = P_0 + \{i\}$.

Step 2: Fitness Assignment: Set the temporary external set $P' = P_k \cup i_k$.
Calculate the fitness value $F_{\text{global}}$ of individuals in $P'$.

Step 3: Update of best individual:

$$\text{Set } i_{k+1} = \max_{\forall i \in P'} F_{\text{global}}(q(i))$$

Step 4: Selection / Termination:

If $F_{\text{global}}(q(i_{k+1})) \leq F_{\text{global}}(q(i_k))$, then return $A = \{q(i_k)\}$.
Else if $k \geq K$, then return $A = \{q(i_{k+1})\}$.
Else set mating pool $P' = \{i_{k+1}\}$.
Step 6: **Mutation**: Set the mutant pool $P'' = \emptyset$.

For each $n \in N$ do

For each individual $i \in P'$ do

\[ j = \text{MUTATE}(i, s, G, n, T, \text{TRUE}): \text{Mutate } i \text{ by relocating at most } s \text{ VMs to improve } f_n(x) \]

Set $P'' = P'' + \{j\}$

Step 7: **Loop**: Set $P_{k+1} = P''$, $k = k + 1$, and go to Step 2.

Because the external set $L$ is limited to containing only a single element, the complexity of GASO is significantly reduced in comparison with EASO. Specifically, the maximum population size $S = |P''| = N$, hence the space complexity of GASO is $O(N \cdot |T|)$, compared to $O(N \cdot L \cdot |T|)$ for EASO, where $N$ is the number of NCFs. The loop in GENERATE runs at most $N$ times, fitness assignment and selection requires at most $O(N)$ comparisons, and MUTATE is called $N$ times. Thus, the time complexity of GASO is $O(N \cdot |T| \cdot K \cdot s)$, compared to the $O(N^2 \cdot L^2 \cdot |T| \cdot K \cdot s)$ time complexity of EASO.

However, the reduced complexity of GASO does not come without limitations. Firstly, observe that the candidate solution selected is dependent upon the operator-specified NCF weight vector $\vec{w}$. Such a specification may be unsuitable in real-world scenarios where the operator may wish to dynamically adjust $\vec{w}$ during runtime. For instance, an operator may wish to assign $w_n$ a higher value for lower values of $f_n$, and vice versa, in order help encourage the selection of a balanced (compromise) solution. Secondly, we hypothesize that due to its greedy search heuristic, GASO may prematurely converge (terminate) on some inputs, thus returning a candidate solution that fails to maximize $F_{\text{global}}$.

### 6.5.2 Quantitative Comparison

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 [9,10]. In [9,10], 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 5.1.1) 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 $F_{global}(x) = w_1(f_1(x)) + w_2(f_2(x)) + \ldots + w_N(f_N(x))$ 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 3.1 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 $f_1$, $f_1 \geq 0.00, 0.066, \ldots, 0.594$; 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.

<table>
<thead>
<tr>
<th>Allocation</th>
<th>(WCS, BW, Power)</th>
<th>Allocation</th>
<th>(WCS, BW, Power)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(5,0,0), (0,5,0), (0,5,0), (0,0,0)</td>
<td>(5,0,0), (0,5,0), (0,0,0)</td>
<td>(0.000, 68 Mbps, 23 units)</td>
</tr>
<tr>
<td>2</td>
<td>(4,0,0), (1,5,0), (0,0,0), (0,0,0)</td>
<td>(3,4,0), (2,1,5), (0,0,0), (0,0,0)</td>
<td>(0.067, 72 Mbps, 22 units)</td>
</tr>
<tr>
<td>3</td>
<td>(3,5,0), (2,0,5), (0,0,0), (0,0,0)</td>
<td>(3,4,1), (2,1,4), (0,0,0), (0,0,0)</td>
<td>(0.133, 77 Mbps, 17.5 units)</td>
</tr>
<tr>
<td>4</td>
<td>(2,5,0), (2,0,0), (1,0,5), (0,0,0)</td>
<td>(3,3,1), (2,2,4), (0,0,0), (0,0,0)</td>
<td>(0.200, 84 Mbps, 22 units)</td>
</tr>
<tr>
<td>5</td>
<td>(3,4,0), (2,1,5), (0,0,0), (0,0,0)</td>
<td>(3,3,2), (2,2,3), (0,0,0), (0,0,0)</td>
<td>(0.200, 86 Mbps, 17.5 units)</td>
</tr>
<tr>
<td>6</td>
<td>(2,4,0), (2,1,5), (1,0,0), (0,0,0)</td>
<td>(2,2,3), (2,2,2), (0,0,0), (1,1,0)</td>
<td>(0.267, 93 Mbps, 22 units)</td>
</tr>
<tr>
<td>7</td>
<td>(3,4,1), (2,1,4), (0,0,0), (0,0,0)</td>
<td>(2,2,1), (1,1,2), (1,1,1), (1,1,1)</td>
<td>(0.267, 100 Mbps, 17.5 units)</td>
</tr>
<tr>
<td>8</td>
<td>(2,3,0), (2,2,0), (1,0,5), (0,0,0)</td>
<td>(2,2,1), (1,1,2), (1,1,1), (1,1,1)</td>
<td>(0.333, 102 Mbps, 22 units)</td>
</tr>
<tr>
<td>9</td>
<td>(3,3,1), (2,2,4), (1,1,1), (0,0,0)</td>
<td>(3,3,2), (2,2,3), (0,0,0), (0,0,0)</td>
<td>(0.333, 109 Mbps, 17.5 units)</td>
</tr>
<tr>
<td>10</td>
<td>(3,3,2), (2,2,3), (0,0,0), (0,0,0)</td>
<td>(4,00), (123 Mbps, 17.5 units)</td>
<td>(0.400, 123 Mbps, 17.5 units)</td>
</tr>
<tr>
<td>11</td>
<td>(2,3,1), (2,2,4), (1,0,0), (0,0,0)</td>
<td>(2,2,3), (2,2,2), (0,0,0), (1,1,0)</td>
<td>(0.400, 116 Mbps, 22 units)</td>
</tr>
<tr>
<td>12</td>
<td>(3,5,0), (2,0,5), (0,0,0), (0,0,0)</td>
<td>(4,67, 130 Mbps, 22 units)</td>
<td>(0.400, 164 Mbps, 22 units)</td>
</tr>
<tr>
<td>13</td>
<td>(2,2,3), (2,2,2), (1,1,0), (0,0,0)</td>
<td>(2,2,3), (2,2,2), (1,1,0), (0,0,0)</td>
<td>(0.533, 143 Mbps, 22 units)</td>
</tr>
<tr>
<td>14</td>
<td>(2,2,2), (2,2,2), (1,1,1), (0,0,0)</td>
<td>(2,2,2), (2,2,2), (1,1,1), (0,0,0)</td>
<td>(0.600, 191 Mbps, 25 units)</td>
</tr>
</tbody>
</table>

Table 6.2: $X_s^{EASO}$ 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.

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
solutions\textsuperscript{2} for each of 100 simulation\textsuperscript{3} runs, represented by the $X_s^{EASO}$ solution set (Table 6.2).

For GASO, we performed multiple runs via parametric analysis, across the range of all mutation sizes ($s = 1, 2, \ldots, 15$). 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 6.2). 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 6.2. These suboptimal solutions show that GASO was often stuck in local maxima.

Table 6.3 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 solution set produced by EASO has smaller distance and higher coverage ratio vs. GASO. These results demonstrate that EASO yields a wider range of, and potentially better solutions than the SOP orchestrators in [9,10]. Furthermore, and perhaps the most distinguishing feature of EASO, is how well it enumerates the tradeoff space.

<table>
<thead>
<tr>
<th>Distance (Mean, Min, Max)</th>
<th>$Y_s^{EASO}$</th>
<th>$Y_s^{GASO}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(0, 0, 0)</td>
<td>(0.014, 0, 0.097)</td>
</tr>
<tr>
<td>Coverage</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>Completion Time (seconds)</td>
<td>3.73</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Table 6.3: EASO vs. GASO in distance and coverage of their solution sets w.r.t. $Y_p$, and in avg. execution time.

To illustrate this point, again consider Table 6.2, 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

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

\textsuperscript{3}The 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.
example, run EASO alloc. #11 during peak hours to conserve BW, and EASO alloc. #10 during non-peak hours to save power.

### 6.6 Evaluating Scalability of EASO

To evaluate its scalability, we simulated EASO on a large-scale, multi-tier application data center scenario similar to the one presented in [3], 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 3.1.

| Table 6.4: Evaluation metric values for $Y_s^{EASO}$ using “short-”, “medium-”, and “long-run” parameters, with OA as the reference set. $Y_s^{GASO}$ is included for comparison. |
|---|---|---|---|
| **$Y_s^{EASO}$** | **$Y_s^{GASO}$** |
| | | | |
| $\{L = 25, K = 25, s = 2\}$ | $\{L = 50, K = 50, s = 4\}$ | $\{L = 75, K = 75, s = 6\}$ |
| 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 | 137 | 43 |
| Completion Time (seconds) | 69 | 566 | 5059 | 1184 |

At this scale, enumerating $Y_p$ is intractable [3]. Therefore, we constructed an outer approximation (OA) of $Y_p$ based on the well known “constraint method” found in MOP literature [12]. Specifically, we formulate each ordered pair of NCF utility functions, $(f_1, f_2), (f_1, f_3), (f_2, f_1), (f_2, f_3), (f_3, f_1), (f_3, f_1)$ as a biobjective optimization 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 $C_{f_1} = \{0.005, 0.010, ..., 0.975\}$), 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 [28]. Table 6.4 illustrates the performance of EASO with respect to OA for different sets of input parameters\(^4\). Observe that there is a clear tradeoff between

\(^4\)For 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.
time and optimality.

As the size of input parameters \((L, K, s)\) increase, EASO produces better and more diverse\(^5\) solution sets at the cost of increased completion 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. Figure 6.7 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 \textit{at least as close} to \(Y_p\) as OA, since points in OA are not necessarily feasible.

\(^5\)Because 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 \(L\).
CHAPTER 7: Conclusion

Based upon the results presented in the previous chapter, we have demonstrated that our proposed evolutionary approach can enumerate a wider range of, and potentially better solutions than current orchestrators for relatively large data center networks.

Furthermore, we conclude that the flaws of using a greedy SDN orchestration approach, such as GASO, Corybantic, or Athens, outweigh the reduced complexity when compared to an evolutionary approach, such as EASO. Although EASO is more complex, it consistently yields better solutions, and perhaps more importantly, provides the network operator with a nondominated set of meritorious candidate solutions spread throughout the tradeoff space. In contrast to GASO’s single “best allocation”, EASO generates a set of solutions independent of the weight vector \( \vec{w} \), so even in cases where the operator desires to maximize some global objective function \( F_{\text{global}} \), EASO does not require the operator to specify \( \vec{w} \) a priori. A single run of EASO produces a nondominated set of meritorious candidate solutions, and the operator can retroactively assign \( \vec{w} \) in order to find the nondominated candidate solution that maximizes \( F_{\text{global}} \). We believe that real-world network operators will find this characteristic of EASO desirable, as they may be unsure of how to assign relative weightings to disparate NCFs.

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 [11], 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.
References


Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California