Orchestrating Network Control Functions via Comprehensive Trade-off Exploration

Alan Bairley
Department of Computer Science
Naval Postgraduate School
ambairle@nps.edu

Geoffrey G. Xie
Department of Computer Science
Naval Postgraduate School
xie@nps.edu

***Abstract*—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.**

I. Introduction

Software-defined networking (SDN) technology provides an open platform for writing network control functions (NCFs), each designed with a specific goal, such as bandwidth and fault tolerance joint optimization [1], [2], [3], power conservation [4], quality-of-service (QoS) control [5], and security enforcement [6]. Hence, SDN offers the promise of convenient and efficient network management via NCF deployment, but without some arbitration mechanism, the direct deployment of multiple NCFs may cause conflicting and unpredictable network configurations to be instantiated, potentially causing traffic loss and/or network instability [7]. For example, a traffic engineering NCF may want to load balance traffic over multiple switches to preserve QoS, while a power saving NCF may want to power down one of these switches to conserve power. Clearly, shutting down some switch forwarding an active flow may result in traffic loss. Furthermore, if both NCFs are determined to control the same resource, then network instability (oscillation) may result [8], e.g. the NCFs alternate by powering the resource on/off.

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 [7], [8], or 2) resource allocation approaches [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 [11], [3], [2], [9], [10], 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 [11], [9], [10], or by optimizing one objective function subject to the others cast as constraints [3], [2].

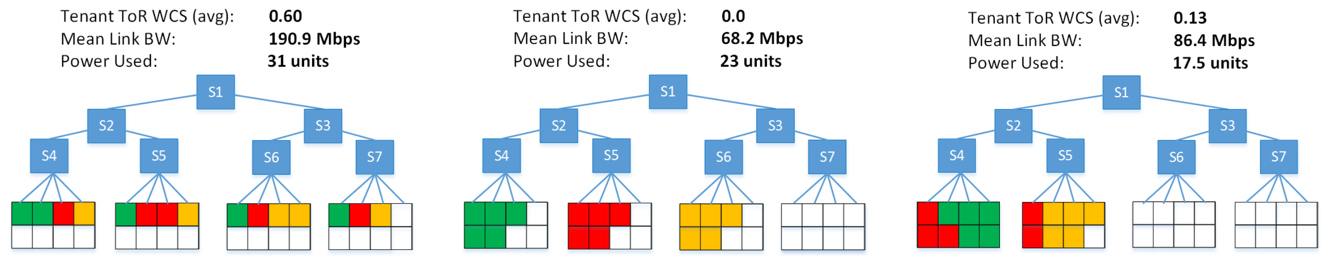
Although SOP formulations of the orchestration problem permit faster solutions, solving a SOP yields only a *single* solution within a potentially vast tradeoff space. Furthermore, many current approaches use search algorithms based on greedy heuristics [3], [9], [10], which may prematurely converge to suboptimal local maxima when applied to non-convex optimization problems. Thus, we believe it prudent to explore an alternative, more basic formulation based on the classical multi-objective optimization problem (MOP) literature [12], [13], [14], where our goal is to enumerate a diverse set of *efficient* solutions among competing NCFs, i.e. each solution in the set cannot be improved in any objective without causing degradation in at least one other objective.

In the following sections, our contribution is three-fold. First, we present a novel MOP problem formulation for SDN orchestration. Second, we describe an evolutionary approach for enumerating a wide range of efficient 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 vs. current solutions in the context of the VM allocation problem.[1].

II. Rethinking Orchestration

Orchestrating multiple NCFs to achieve and maintain stable and desirable network operating conditions face

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



(a) Survivability (b) Bandwidth Conservation (c) Power Conservation

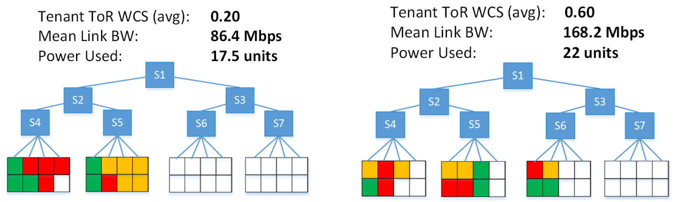
Fig. 1: Example proposals from NCF-S, NCF-B, and NCF-P. Green slots are for VMs of R1, red R2, and gold R3. We assume that each server, ToR switch, aggregation switch, or core switch uses 1, 2, 2.5, and 3 units of power, respectively.

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 simple tree topology, consisting of one core switch as root, two aggregation switches and four top of rack (ToR) switches, four host servers per ToR switch, and two VM slots per host server, where a “VM slot” is defined as a standard physical resource unit (e.g. CPU, Memory) provisioned to a VM. Therefore, the VMs are to be allocated against $4 \times 4 \times 2 = 32$ possible slots.

Suppose the data center utilizes three different NCFs simultaneously: NCF-S, NCF-B and NCF-P. NCF-S is designed to maximize the applications' worst case survivability (WCS) [1] over failures of ToR switches and physical servers with a preference for spreading VMs of each application across separate racks and servers (Figure 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 done in [16], with a preference for consolidating VMs of the same application as close to one another as possible, e.g. placing on same server or rack (Figure 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 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.



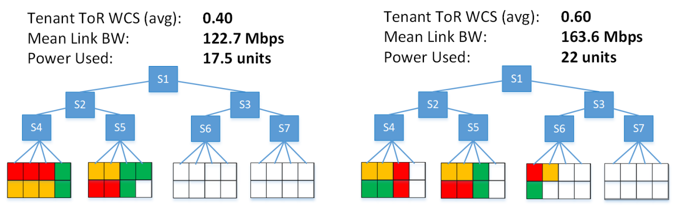
(a) Mutated Power Cons. (b) Recombined BW/WCS

Fig. 2: Mutated power conservation proposal and recombined BW conservation and survivability proposal.

In prior work, e.g. Athens [10] and Corybantic [9], all candidate solutions are exclusively generated by the NCFs. Assuming that each NCF generates one proposal, the initial population in the context of this example is limited to the three proposals depicted in Figure 1. It may be possible for each NCF to generate multiple proposals, but even so, it seems unrealistic to expect a specialized NCF to generate mutually beneficial compromises without knowledge or understanding of the performance criteria of the others. [9], [10] then proceed by selecting the NCF proposal that maximizes some global utility function (e.g. votes, cost-effectiveness), and may iteratively solicit the NCFs for incremental counter-proposals to the previously selected network state until the greedy criterion cannot further be improved by any NCF proposal.

The problem here is that even if a specialized NCF is able to make effective counter-proposals, which may itself be challenging, the region of the 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 1c is selected initially, then it may be the case that while the network state in Figure 2a is reachable via a series of incremental counter-proposals, the network state in Figure 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 nondominated 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.



(a) Mutated from Fig. 2a (b) Mutated from Fig. 2b

Fig. 3: Two compromises from successive rounds of recombination and mutation of nondominated 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 2. Note that while both the mutated power conservation proposal (Figure 2a) and the recombined survivability and BW conservation proposal (Figure 2b) each dominate a predecessor state (Figures 1c and 1b, respectively), these proposals are nondominated with respect to one another. Although the proposal in Figure 2b offers better ToR WCS, it uses more power and BW than the proposal in Figure 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 2b is more appropriate for mission critical applications that require maximum fault tolerance, while the proposal in Figure 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 evolutions (as shown in Figure 3) to the operator.

III. Multi-Objective Optimization Formulation

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

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

Instead of computing a weighted sum or using other techniques to reduce the objective vector to a single scalar metric and then finding one "best" allocation with respect to that metric, as done in prior work, we leverage the classical MOP literature [12] to look for a set of solutions that illuminates the entire trade-off space of the diverse objectives. First, we formally define when two allocations can be ordered in the N-dimension objective vector space, i.e. when one dominates the other, and when they cannot because they may be preferred for different objectives.

Def. 1: (**Pareto Dominance** [14]) For any two allocations $x_1, x_2 \in X_f$,

(i) $x_1 \succ x_2$ (x_1 dominates x_2) **iff** $\exists i, f_i(x_1) > f_i(x_2)$ **and** $\forall j \neq i, f_j(x_1) \geq f_j(x_2)$.

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

This concept of Pareto dominance allows us to define the optimality criterion for the MOP formulation. If some allocation, x, is not dominated by any other allocation, then this means that x is optimal in the sense that it cannot be improved in any objective without causing degradation in at least one other objective. Such solutions are referred to as *Pareto-optimal* [14].

Def. 2: (**Pareto Optimality** [14]) An allocation x is Pareto-optimal regarding the feasible allocation set X_f **iff** $x \in X_f$ **and** $\nexists x' \in X_f : x' \succ x$.

The entirety of all Pareto-optimal solutions is called the *Pareto-optimal set*, denoted by X_p; the corresponding objective vectors form the *Pareto-optimal front* or *surface*, denoted by Y_p. Now a MOP formulation of the VM allocation problem is simply as follows.

Def. 3: (**MOP for VM Allocation**)
maximize $y = (f_1(x), f_2(x), ..., f_N(x))$, i.e., enumerate all Pareto-optimal solutions,
subject to $x \in X_f$ **and** additional criteria such as lower bounds for $f_i(x)$'s.

IV. Evolutionary Approach

Prior MOP work [13], [17], [18] 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 section, we present such an evolutionary algorithm, termed Evolutionary Algorithm for SDN Orchestration (EASO), to solve the MOP problem formulated in the previous section.

A. 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 II, the following NCF-specific mutation heuristics are used within the **MUTATE** procedure. Here, we use the term *affinity* to refer to the number of VMs of a particular application residing in the same subtree.

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

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

B. ***EASO*** Specification

A specification of the EASO algorithm is provided below. In addition to Pareto-dominance, fitness assignment in Step 2 is also based on crowding distance, which measures the uniqueness of a candidate solution with respect to other members in the set, as done in [17], [18].

```
Algorithm 1: EASO
 Input:    $K$: number of generations; $L$: external set size;
           $T$: physical topology tree; $s$: mutation size
 Output:   solution set $X_s$
Step 1: Initialization:
 a) Set initial population $P_0 = \emptyset$, $k = 0$
 b) Set initial external set $ES_0 = \emptyset$
 c) Solicit each of the $N$ NCFs for its proposed allocation
      $x \in X_f$ and add $x$ to $P_0$
 d) Add $(L - N)$ randomly generated allocations to $P_0$
Step 2: Fitness Assignment / Termination:
 a) Calculate fitness $F$ of allocations in $P_k \cup ES_k$
 b) Derive nondominated feasible set $X_s$ from $P_k \cup ES_k$
 c) If $k \geq K$ then return $X_s$
Step 3: Update of external set:
 a) If $|X_s| > L$ then
      Remove $(|X_s| - L)$ worst fitness allocations from $X_s$
 b) Else if $|X_s| < L$ then
      Add $(L - |X_s|)$ best fitness allocations of $P_k$ to $X_s$
 c) Set $ES_{k+1} = X_s$
Step 4: Recombination:
 a) Set mating pool $MP = X_s$, child pool $CP = \emptyset$
 b) For each NCF $i$ do
  1) Sort $MP$ in ascending order of $f_i$
  2) For $a$ in 1 to $\lfloor |MP|/2 \rfloor$ do
       $b = (|MP| + 1) - a$
       $x$ = RECOMBINE($MP[a], MP[b]$)
       Add $x$ to $CP$
```

```
Step 5: Mutation:
 a) For each NCF $i$ do
  1) For each allocation $x \in MP$ do
       $u = \textbf{MUTATE}(x, s, i, goal = \text{"}Improve\text{"})$
       $w = \textbf{MUTATE}(x, s, i, goal = \text{"}Degrade\text{"})$
       Add $u, w$ to $CP$
Step 6: Update Population:
 a) Set $P_{k+1} = CP$, $k = k + 1$; Goto Step 2
```

C. 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 $|T|$, e.g. the large topology used in Section V-C, totally enumerating X_p may require prohibitively large EASO input parameters. In these cases, X_s is rather an inner approximation [13] of X_p.

Space Complexity: The maximum population size contains $|CP| = N * (L/2 + 2L) = \frac{5NL}{2}$ states, and each state contains $|T|$ 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 evaulates at most $|T|$ elements of each state, for a resultant complexity of $O(N^2 \cdot L \cdot |T|)$. Fitness assignment requires at most $O(N \cdot L^2)$ comparisons using the scheme presented in [18], **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 $O(N^2 \cdot L^2 \cdot |T| \cdot K \cdot s)$.

V. Evaluation

In this section, we evaluate EASO using two topologies. First, the simplistic scenario described in Section II is used to illustrate that EASO can produce more diverse, and potentially better solutions than current orchestators. Then, a relatively large topology [3] is used to illustrate that EASO scales to large data centers.

A. Performance Metrics

As discussed in Section IV-C, for nontrivial topologies, EASO may only (inner) approximate the Pareto-optimal set X_p. To evaluate the accuracy of this inner approximation (denoted by X_s), we propose two metrics: *distance* and *coverage* to compare X_s against X_p using their corresponding sets of objective vectors, i.e. images in the objective vector space. Specifically, let Y_s and Y_p denote the image sets of X_s and X_p, respectively. (When it is infeasible to obtain X_p and Y_p due to nontrivial values of $|T|$, we use the "constraint method" well known in MOP literature [12] to first construct an outer approximation of Y_p, and then use it in place of Y_p in equations (1) and (2) to compute distance and coverage. See Section V-C for more details.) The distance of an objective vector $y \in Y_s$ is defined as follows.

Def. 4: (**Distance**)

$$distance(y, Y_p) = \min_{w \in Y_p} dist(y, w), \tag{1}$$

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

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

The current orchestrators, such as [9], [10] strive only to find a single solution that minimizes distance, without regard for the potentially vast tradeoff space. In contrast, a novel aspect of our approach is the enumeration of a set of nondominated tradeoffs. Hence, to evaluate the area of the tradeoff space covered by a solution set produced by EASO or similar algorithms, we propose the *coverage* metric (Def. 5), representing the fraction of points in the reference image set Y_p that are "covered", i.e. nearest to objective vectors in Y_s. Hence, solution sets with higher coverage values are more desirable.

Def. 5: (**Coverage**)

$$coverage(Y_s, Y_p) = \frac{|\bigcup_{y \in Y_s} nearest(y, Y_p)|}{|Y_p|}, \tag{2}$$

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

B. EASO vs. Current Orchestrators

In order to illustrate the unique merits of EASO vs. the current orchestrators, we developed GASO, a greedy version of EASO, to emulate methods proposed in [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 IV-A) 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 $F_{global}(x) = w_1(f_1(x)) + w_2(f_2(x)) + ... + w_N(f_N(x))$ 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 II 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, $f_1 \geq 0.00, 0.066, ..., 0.594$; we used two different weightings: $\vec{w} = (1, 4, 2)$, which clearly favors f_2 over f_3, and $\vec{w} = (1, 2, 4)$, which conversely favors f_3 over f_2. f_1 maintains a minimum weighting here, as the aspiration levels force an enumeration over the its range.

For EASO, we set the size of the external set $L = 25$, the number of generations $K = 25$, and the mutation size $s = 5$. EASO consistently enumerated all 14 Pareto-optimal solutions[2] for *each* of 100 simulation[3] runs, represented by the X_s^{EASO} solution set (Table I).

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

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

TABLE I: 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.

Table II 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,

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

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

and perhaps the most distinguishing feature of EASO, is how well it enumerates the tradeoff space.

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

C. Is EASO Scalable?

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

At this scale, totally 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 $C_{f_1} = \{0.005, 0.010, \ldots, 0.975\}$), 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 [19].

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

[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.

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

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

Table III illustrates the performance of EASO with respect to OA for different sets of input parameters[4]. Observe that there is a clear tradeoff between time and optimality. As the size of input parameters (L, K, s) increase, EASO produces better and more diverse[5] solution sets at the cost of increased execution time. The "short-run" parameter set $(25, 25, 2)$, completes in just over a minute, hence most appropriate for network operators with rapidly changing tenant application requirements. In contrast, the "long-run" parameter set $(75, 75, 6)$ takes over an hour to complete, and thus may be warranted for steady state data center operations where network configurations are unlikely to change frequently. Finally, the "medium-run" parameter set $(50, 50, 4)$ finishes in under ten minutes, and represents a reasonable compromise between agility and quality. The increase of EASO execution time (last row of Table III) corroborates the time complexity analysis in Section IV-C. Figure 4 depicts the EASO "long-run" solution set vs. OA for the large-scale data center scenario. From this figure, we can see that the EASO solution set is well spread and relatively close regarding OA. Also realize that the EASO solutions are *at least as close* to Y_p as OA, since points in OA are not necessarily feasible.

VI. Conclusion

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

For future work, we find several areas intriguing. The mutation and recombination evolutionary primitives may be further refined and adapted for other orchestration tasks, such as traffic engineering, risk management, or cybersecurity. For example, in [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 promsising 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.

[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.

[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.

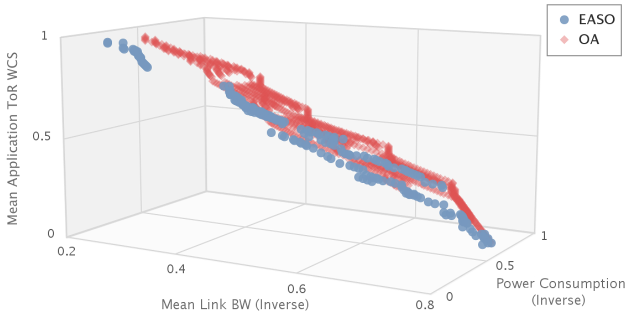


Fig. 4: EASO Nondominated Front vs. Outer Approximation (OA) of Y_p, for a large-scale data center scenario.

Acknowledgment

We would like to thank Matthew Carlyle for his insights on MOP, especially the use of an outer approximation as an evaluation method. We would also like to thank Robert Beverly, Justin Rohrer, Franck Le, Xin Sun, and the anonymous reviewers for their helpful comments. Finally, we thank the NPS Foundation for funding, and the DoD Information Assurance Scholarship Program for providing general support to Alan Bairley.

References

[1] P. Bodik et al., "Surviving failures in bandwidth-constrained datacenters," in SIGCOMM '12. ACM, 2012.
[2] J. Lee et al., "Application-driven bandwidth guarantees in datacenters," SIGCOMM CCR, vol. 44, no. 4, Aug. 2014.
[3] G. Jung et al., "Ostro: Scalable placement optimization of complex application topologies in large-scale data centers." in ICDCS. IEEE, 2015.
[4] B. Heller et al., "Elastictree: Saving energy in data center networks," in NSDI'10. USENIX Association, 2010.
[5] M. Alizadeh et al., "Less is more: Trading a little bandwidth for ultra-low latency in the data center," NSDI'12. USENIX, 2012.
[6] S. Shin et al., "Fresco: Modular composable security services for software-defined networks," in NDSS'13, February 2013.
[7] P. Sun et al., "A network-state management service," in SIGCOMM '14. ACM, 2014.
[8] D. Volpano et al., "Towards systematic detection and resolution of network control conflicts," in HotSDN '14. ACM, 2014.
[9] J. C. Mogul et al., "Corybantic: Towards the modular composition of sdn control programs," in HotNets-XII. ACM, 2013.
[10] A. AuYoung et al., "Democratic resolution of resource conflicts between sdn control programs," in CoNEXT '14. ACM, 2014.
[11] W. Rankothge et al., "Towards making network function virtualization a cloud computing service," in IFIP/IEEE IM, 2015.
[12] M. Ehrgott and M. M. Wiecek, Multiple Criteria Decision Analysis: State of the Art Surveys. Springer, 2005.
[13] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," 1999.
[14] V. Pareto, Cours d'Economie Politique. Droz, 1896.
[15] Y. Liu et al., "The multi-path routing problem in the software defined network," in ICNC 2015, 2015.
[16] H. Ballani et al., "Towards predictable datacenter networks," in SIGCOMM '11. ACM, 2011.
[17] E. Zitzler et al., "Spea2: Improving the strength pareto evolutionary algorithm," Tech. Rep., 2001.
[18] K. Deb et al., "A fast elitist multi-objective genetic algorithm: Nsga-ii," IEEE TEVC, vol. 6, 2000.
[19] J. Fernández et al., "Obtaining an outer approximation of the efficient set of nonlinear biobjective problems," JGO, 2007.