MORSA: A Multi-objective Resource Scheduling Algorithm for NFV Infrastructure

M. Yoshida, W. Shen, T. Kawabata, K. Minato and W. Imajuku
NTT Network Innovation Laboratories
1-1 Hikari-no-oka, Yokosuka, Kanagawa 239-0847 Japan
yoshida.masahiro@lab.ntt.co.jp

***Abstract*— Capital expenditure (CAPEX) and operating expenses (OPEX) have been practical concerns for telecommunication companies, and Network Function Virtualization (NFV) has attracted significant attention in recent years. However, the NFV Infrastructure (NFVI) includes a wide variety of resource types and stakeholders. Hence, there is still a need for the existing resource scheduling approaches, but they are no longer sufficient. In this paper, we propose a Multi-objective Resource Scheduling Algorithm (MORSA) to optimize the NFVI resources. In the development of the MORSA, we designed a plug-in architecture to satisfy various requirements related to NFVI resources and stakeholder policies. The MORSA allows the NFVI Resource Scheduler to optimize simultaneously the combination of possibly conflicting objectives with multifaceted constraints in complex real world situations. Evaluation results show that the MORSA obtains approximate solutions among conflicting objectives in a reasonable computation time.**

Keywords—NFV-MANO; NFV-Orchestrator; OpenStack; Resource Scheduling; Genetic Algorithm

I. INTRODUCTION

In current telecommunication networks, network functions (NFs) are implemented as a combination of vendor specific hardware and software. On the other hand, to improve capital efficiency compared to dedicated hardware and software implementations, ETSI ISG [1] proposed a promising concept called Network Function Virtualization (NFV) [2]. NFV focuses on using commercial off-the-shelf (COTS) hardware, i.e., general-purpose servers, to provide NFs through software virtualization techniques. NFV aims to move critical infrastructure services, e.g., the Evolved Packet Core (EPC), to NFV Infrastructure (NFVI) (described in Section II). The breakthrough that the NFV concept represents may cause a paradigm shift for many stakeholders including network operators, solution vendors, service integrators, providers, and service users. However, several issues continue to hinder the management of NFVI resources.

First, the NFV concept has many types of NFVI resource requirements in order to launch requested network services. The NFVI includes not only physical and virtual resources in a single data center but also network resources between multiple data centers. Therefore, software network appliances would be deployed in the NFVI with various hardware and software requirements, quality of service (QoS) provisioning levels, and/or negotiated service level agreements (SLAs). Accordingly, scheduling, i.e., managing, NFVI resources will be more complex than that currently used in managing cloud environments [3]-[9]. The existing approaches lack the capability (and modularity) to satisfy the complex combinations of various NFVI resource requirements.

Second, there are various stakeholders regarding the NFVI and their operational policies may differ with the others on some essential points. For example, even when deploying one software network appliance at a single data center, service users, cloud, i.e., data center, operators and telecommunication network operators exist as stakeholders. A service user may want to select the lowest priced data center for the placement of the software network appliance [3]. On the other hand, the software network appliances may have large quantities of data center resources. Moreover, individual data centers have several operational policies regarding, e.g., minimizing the total energy consumption [4], intra-data center traffic [5], and physical machine (PM) load VI. Therefore, existing data centers may not be able to schedule resources due to the NFVI resource requirements. In addition, to avoid a bottleneck link in telecommunication networks, telecommunication network operators may suggest using alternative data centers regardless of other stakeholder policies [7]. The proposed NFVI Resource Scheduler must consider trade-offs among the operational policies from each stakeholder.

In this paper, we propose the Multi-objective Resource Scheduling Algorithm (MORSA) for network services on the NFVI. In developing the MORSA, we designed a plug-in architecture to satisfy various NFVI resource requirements and stakeholder policies. The MORSA allows the NFVI Resource Scheduler to optimize simultaneously the combination of possibly conflicting objectives with multifaceted constraints in complex real world situations. Because the formulations of the NFVI resource scheduling are classified as NP-hard, MORSA uses a modified Multi-objective Genetic Algorithm (MOGA) to obtain approximate solutions in a reasonable computation time.

The main contributions are described below.

- We designed two types of resource scheduling mechanisms called the NFVI Resource Filter and the NFVI Resource Scheduler to implement various NFVI resource requirements and stakeholder policies as plug-ins. These mechanisms are intended to simplify greatly the addition of support for new requirements resulting

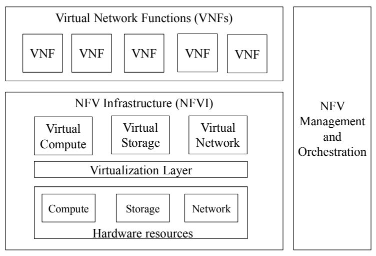

Fig. 1. The high-level NFV framework [1].

in much less initial and ongoing effort than would be required to add a new monolithic plug-in.

- We conducted a gap analysis of the NFVI resource scheduling in current standards. OpenStack [10] is quickly becoming the de-facto standard for the open source cloud-computing platform. We provide gap analysis for OpenStack to be a standard Virtualized Infrastructure Manager (VIM) to fulfill NFVI resource requirements and stakeholder policies [11].
- As a usage case of the MORSA, we optimize virtual machine (VM) placement with a combination of conflicting objectives. Although the objective functions are classified as NP-hard, simulation results show that the MORSA effectively obtains Pareto optimal solutions (described later) within seconds.

To the best of our knowledge, this is the first challenge for NFVI resource scheduling that considers both the NFVI resource requirement and stakeholder policies. The main contribution of this paper is that the MORSA not only solves the joint optimization problem of NFVI resources under multiple stakeholder policies, but also suggests that the framework of the NFVI resource scheduling can improve the capability of NFV management and orchestration. This paper is rather describing a framework than an algorithm.

II. The NFV Framework

The NFV concept enables implementation of NFs as software appliances that run over NFVI on COTS hardware [1]. Fig. 1 shows a high-level NFV framework. The NFV framework comprises three components: Virtual Network Functions (VNFs), the NFVI, and NFV Management and Orchestration (NFV-MANO).

The VNFs are software implementations of NFs that can be executed over the NFVI. For example, Lagopus [12] and Open vSwitch [13] are typical VNFs used as software L2 switches.

The NFVI includes all hardware resources, e.g., physical machines, storage, and network; virtualized resources, e.g., virtual machines, storage, and network; and virtualization layers, e.g., hypervisors. The NFVI includes both intra-data center resources and inter-data center resources, e.g., inter-provider links. The NFVI supports the execution of the VNFs.

NFV-MANO covers the orchestration and lifecycle management of network services including VNFs. The NFV Orchestrator (NFV-Orch.) and the VIM are responsible for

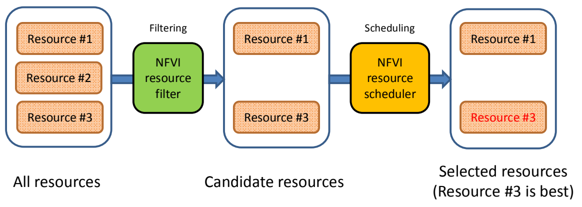

Fig. 2. Outline of NFVI resource scheduling for MORSA.

controlling and managing the NFVI resources. The MORSA is implemented as part of NFV-MANO.

III. The MORSA Framework

The MORSA is implemented as part of vConductor [14], which is regarded as an NFV management solution. Fig. 2 shows an outline of the NFVI resource scheduling for the MORSA. The NFVI Resource Filter sifts out the NFVI resources that are sufficient to meet the NFVI resource requirements. Then, the NFVI Resource Scheduler finds the best combination of resources and stakeholder policies. Fig. 3 shows the MORSA plug-in framework. The MORSA consists of the NFVI Resource Filter, NFVI Resource Scheduler, Requirement Plug-ins, and Scheduling Policy Plug-ins.

A. MORSA Output

The MORSA maps the requested network service to the available NFVI resources. Table I gives the MORSA output. The NFVI Resource Filter and NFVI Resource Scheduler generate several outputs for VNF deployment. Possible output types include VM resources, virtual link (VL) resources, VM placement, and virtual network embedding (VNE) [15]. These output types are a specific set of NFVI resources that fulfill the required topology and QoS, which are capable of accommodating the requested VNFs, according to their characteristics and type of resources.

B. Requirement Plug-ins

Requirement Plug-ins allow the MORSA to utilize simultaneously the variety of requirements for VNF deployment. Table II gives a list of NFVI Requirement Plug-ins in the MORSA. We implemented four types of plug-ins. Note that both the NFVI Resource Filter and NFVI Resource Scheduler use Requirement Plug-ins.

First, Operational Requirement Plug-ins define the requirements for network service operations on the NFVI. In particular, our implementation offers requirements for a high-availability system design for telecommunication companies. For instance, Machida *et al.* [8] proposed host packing, e.g., deploying two related VMs on the same host machine, and host slicing, e.g., deploying two related VMs on different host machines, to achieve fault tolerance in host servers. The MORSA extends this to racks and availability zones, e.g., data centers. Moreover, the MORSA takes into account the data center network architecture to generate a redundant configuration of access switches, aggregation switches, and core switches.

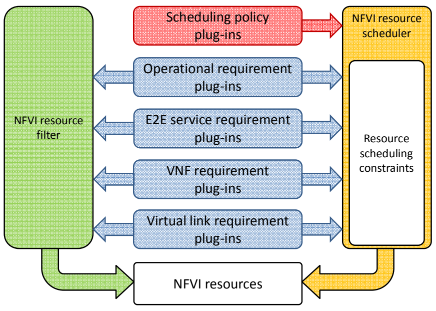
Scheduling policy plug-ins
NFVI resource scheduler
Operational requirement plug-ins
NFVI resource filter
E2E service requirement plug-ins
Resource scheduling constraints
VNF requirement plug-ins
Virtual link requirement plug-ins
NFVI resources

Fig. 3. MORSA plug-in framework.

Second, to ensure the performance of end-to-end (E2E) network services, Network Service Requirement Plug-ins define the QoS requirements such as those for network latency, bandwidth, and packet loss between two endpoints.

Third, VNF Requirement Plug-ins and Virtual Link Requirement Plug-ins supply definitions for requirements such as physical/virtual infrastructure resources and network protocols.

C. *Scheduling Policy Plug-ins*

Scheduling Policy Plug-ins allow the MORSA to generate outputs with a combination of various objectives for each stakeholder. In other words, the Scheduling Policy Plug-ins enhances the modularity of the NFVI resource scheduling algorithms. Numerous resource scheduling policies were proposed in [3]-[9] and the MORSA provides Scheduling Policy Plug-ins to satisfy these policies. For the sake of brevity, we show two different types of Scheduling Policy Plug-ins, one to minimize the PM load and the other to minimize the intra-data center traffic. In this paper, we limit the scheduling policies to show how MORSA optimizes the VM placement in a single data center with a combination of conflicting objectives.

1) *Minimizing the PM load*

The MORSA with the minimizing the PM load policy places a new VM on the least loaded host machines. Although there are a wide variety of resource definitions on both the VM and PM, we assume CPU, memory, and storage resources for this objective function.

$$l_x = w_{cpu}\frac{\sum_{i=1}^{|VM_x|} u_i^{cpu}}{c_x^{cpu}} + w_{memory}\frac{\sum_{i=1}^{|VM_x|} u_i^{memory}}{c_x^{memory}} + w_{storage}\frac{\sum_{i=1}^{|VM_x|} u_i^{storage}}{c_x^{storage}} \tag{1}$$

where w is the weight associated with each resource and u is the utilization of each resource by VM i. We define c as the total capacity of PM x in terms of each resource. Term VM_x is the set of VMs hosted on PM x and PM is the set of all PMs. Term $|VM_x|$ is the total number of VMs hosted on PM x. Thus, we can use l to denote the total load of PM x in (1). The mean of l values of the PMs in set L is denoted by μ_l and the standard deviation of l values on the same load set is denoted by σ_l. Finally, we define the traditional resource optimization objective in a single data center: to minimize the PM load.

$$\min_{\forall l \in L}\left(\frac{\sigma_l}{\mu_l}\right) \tag{2}$$

TABLE I. OUTPUTS OF MORSA

Output Type	Output Examples
Virtual Machine Resources	# CPU/vCPU core, Memory size, Storage size, vNIC bandwidth
Virtual Link Resources	Latency, Bandwidth, and Packet loss between VMs
Virtual Machine Placement (Single Datacenter)	An optimized placement of VMs on host machines in a single DC
Virtual Machine Placement (Multiple Datacenters)	An optimized placement of VMs on host machines in multiple DCs
Virtual Network Embedding [15]	An optimized allocation of virtual resources both in nodes and links

$$s.t. \sum_{i=1}^{|VM_x|} u_i^{cpu} \le c_x^{cpu}, \sum_{i=1}^{|VM_x|} u_i^{memory} \le c_x^{memory}, \sum_{i=1}^{|VM_x|} u_i^{disk} \le c_x^{disk} \tag{3}$$

The VM placement problem with objective function (2) can be classified as the multiple-knapsack problem (MKP), which is known to be NP-hard, i.e., $O(2^n)$ [16]. Equation (3) represents the PM capacity constraints.

2) *Minimizing the intra-datacenter traffic*

The new VMs with a large mutual bandwidth usage are assigned to PMs in close proximity to the minimizing-the-intra-data-center-traffic plug-in. We assume static and single-path routing in the data center network. Dynamic and multi-path routing is outside of the scope of this paper. The intra-data center traffic cost (product of bandwidth and distance) is calculated according to the following equation.

$$\min_{\substack{\forall i,j \in VM,\ \forall k \in GW: \\ \pi_i, \pi_j \in \Pi}} \left(\sum_{i=1}^{|VM|} \left(\sum_{j=1}^{|VM|} I_{ij} C_{\pi_i \pi_j} + \sum_{k=1}^{|GW|} E_{ik} C_{\pi_i k} \right) \right) \tag{4}$$

$$\pi : [1, \dots, |VM|] \rightarrow [1, \dots, |VM|] \tag{5}$$

where I_{ij} denotes the traffic rate from VM i to j and E_{ik} denotes the external traffic rate from VM i to gateway (GW) k. Term C_{ij} denotes the communication cost from VM i to j. Although C_{ij} can be defined in many ways, we define C_{ij} as the number of switches on the routing path from VM i to j. Term $|VM|$ is the total number of VMs and $|GW|$ is the total number of GWs in a data center. We consider a scenario where there are $|VM|$ VMs and $|VM|$ slots on a one-to-one basis, and there is a corresponding permutation function, (5). Due to the complexity of the recursive permutation function, the VM placement problem with objective function (4) can be classified as a quadratic assignment problem (QAP) that is known to be NP-hard, i.e., $O(n!)$ [17].

D. *NFVI Resource Filter*

The NFVI Resource Filter supports filtering to make informed decisions on where the requested network service should be created. When the NFVI Resource Filter receives the Requirement Plug-ins in Table II, it first creates filtering rules. Then, the NFVI Resource Filter applies the filtering rules to determine which resources are eligible for consideration when dispatching a resource. The NFVI Resource Filter is binary, i.e., either a resource is accepted by the filter or it is rejected. NFVI resources that are accepted by the NFVI Resource Filter are then processed by the NFVI

TABLE II. A LIST OF REQUIREMENT PLUGINS IN MORSA

MORSA				Gap analysis
Requirement Plugins	**Type**	**NFVI Resource Filter**	**Constraints for the NFVI Resource Scheduler**	**OpenStack (nova-scheduler)**
Operational Requirement Plugins	IT Resource redundancy	N/A	Host packing/slicing	Yes (Different host filter, Same host filter)
		N/A	Rack packing/slicing	Yes (Availability zone filter)
		N/A	Availability zone packing/slicing	Yes (Availability zone filter)
	Network resource redundancy	N/A	Access switch packing/slicing	No
		N/A	Aggregation switch packing/slicing	No
		N/A	Core switch packing /slicing	No
Network Service Requirement Plugins	QoS	N/A	Network latency/bandwidth/packet loss between two endpoints	No
VNF Requirement Plugins	IT resource requirements	Physical (Host) machine resources (e.g. Physical CPU/memory/storage capacity, Vendor specification and etc.)	N/A	Yes (Compute capabilities filter)
		Virtual (Guest) machine resources (e.g. Virtual CPU/memory/storage capacity, Hypervisor type and etc.)	N/A	Yes (Compute capabilities filter)
Virtual Link Requirement Plugins	QoS	N/A	Network latency/bandwidth/packet loss between two VNFs	No
	IP Address requirements	Host or subnet range IP address	N/A	Yes (Simple CIDR affinity filter)
	Protocol requirements	Intra-Datacenter network protocol (e.g. VLAN, VxLAN, OpenFlow, Flat and etc.)	N/A	No
		Gateway tunnel protocol (e.g. IPSec, GRE, OpenVPN and etc.)	N/A	No
		Inter-Datacenter routing protocol (e.g. BGP, OSPF, MPLS and etc.)	N/A	No

Resource Scheduler to decide which resources to use for the requested network services.

In a vConductor [14] implementation, we use OpenStack with the latest Havana version as our VIM [10]. Based on the OpenStack nova-scheduler service [11] we implement the NFVI Resource Filter. Since OpenStack is already widely deployed, we carefully analyze the capabilities of resource filters already implemented on OpenStack and we provide gap analysis as shown in Table II. Currently, OpenStack has a function to consider IT resource redundancy, e.g., different host filters, the same host filter, and an availability zone filter; IT resource requirements, e.g., computing capability filter; and IP address requirements, e.g., a simple CIDR affinity filter. However, OpenStack does not provide a function to take into account network topology redundancy, resources, and QoS in a data center. Certainly, the network resources are very important to NFV. Therefore, we need to modify the OpenStack nova-scheduler to achieve the NFVI Resource Filter. Note that binary filtering cannot deal with QoS requirements and fault-tolerant requirements. For this reason, we implement them as constraints for the NFVI Resource Scheduler.

E. NFVI Resource Scheduler

For each network service request, the NFVI Resource Scheduler analyzes them and schedules the NFVI resources, i.e., we apply the MORSA in the online scenario. When supplying resources to network services, the NFVI Resource Scheduler schedules each resource in the NFVI Resource Filter list of accepted resources. We recall that there are various stakeholders in NFV and they have different resource scheduling policies, i.e., there are possibly conflicting objectives. Deployment of the network services with conflicting objectives is a challenging task for the MORSA

For a NFVI resource scheduling problem, there is no single solution that simultaneously optimizes objectives. In this case, the MORSA has conflicting objective functions for each stakeholder and we must obtain Pareto optimal solutions [18]. Fig. 4 shows the concept of Pareto optimal solutions. A solution set on the Ideal Pareto Optimal Front is called Pareto efficient, i.e., none of $f_1(x)$ can be improved in value without degrading the objective values of $f_2(x)$ - and vice versa. All Pareto optimal solutions are considered equally good; hence, the MORSA aims to find a representative set of Pareto optimal solutions in a reasonable computation time. Note that automatically suggesting a single solution that satisfies the subjective preference of each stakeholder is outside the scope of this paper (it is still an open problem). The MORSA effectively suggests options for the NFVI resource scheduling to stakeholders and each stakeholder can choose from the options.

To obtain a better Pareto optimal solution with a reasonable computation time, we propose a modified MOGA for the NFVI Resource Scheduler. A genetic algorithm is beneficial in obtaining a Pareto optimal solution with the combination of conflicting objectives in a short time. The genetic algorithm concept is described in [19]. When the NFVI Resource Scheduler receives Scheduling Policy Plug-ins, it creates objective functions. Table III shows a high-level description of MOGA and Table IV gives the MOGA parameters in the MORSA. First, the MOGA randomly generates an initial population of S individuals. Then it evaluates population P

TABLE III. THE HIGH-LEVEL DESCRIPTION OF MOGA

```
Algorithm: Multi-Objective Genetic Algorithm in MORSA
1  P ←generate an initial population of S individuals
2  evaluate each individual in population P
3  B ←select the best individual in P with the elite selection
4  test convergence of B (if true, go to step 12)
5  for (i = 1; i < N; i++)
6      evaluate each individual in population P
7      produce offspring by the uniform crossover with $R_C$
8      probabilistically apply the mutation in each individual with $R_M$
9      B ←select the best individual in P with the elite selection
10     test convergence of B (if true, go to step 12)
11 end
12 decode the best individual and output it
```

with objective functions created by the NFVI Resource Scheduler. In fact, there are many multi-objective evaluation techniques [20]. In our preliminary experiments, we found that the weighting and global criterion method [21] is suitable for NFVI resource scheduling. Furthermore, to obtain a weak Pareto optimal solution [18], we respectively calculate optimal solutions using each objective function and add them to the Pareto optimal solution. We select the best individual among *P* with the elite selection [22]. The uniform crossover operator produces a set of offspring for each generation. The crossover and mutation rate are constant values (R_C is set to 0.5 and R_M is set to 0.05 in our evaluation). To handle constraints effectively for the MOGA, we use the penalty function method [23]. When the NFVI Resource Scheduler receives the Requirement Plug-ins in Table II, it analyzes constraints and then creates penalty functions. With the evolution of *S* individuals, the average fitness of population *P* and the fitness of the best individual *B* increase gradually.

IV. EVALUATION

For the sake brevity, we report on only a small fraction of our evaluations. In this paper, we limit ourselves to our scheduling policies to show how the MORSA optimizes the VM placement in a single data center with a combination of conflicting objectives. For the evaluations, we use two Scheduling Policy Plug-ins, minimizing the PM load and minimizing the intra-data center traffic as described in Section III. All weights in (1) are set to 1 in the evaluation experiments. The evaluation experiments are based on simulations and we developed a single data center simulator written in C#. The simulation parameters are given in Table IV and the data center network architecture is Fat-Tree [24]. We conduct the simulation three times with the same settings for each evaluation and estimate the average value and standard deviation.

We evaluate the computation time of the MORSA to optimize the VM placement. The number of VMs in each network service request is 10 and the total number of PMs in the data center is 5 in this evaluation. To compare the MORSA with the naïve method, we also optimize the VM placement with the traditional integer linear programing (ILP) method [25]. Note that MORSA's genetic algorithm returns exactly the same results as naïve ILP in this evaluation. Fig. 5 shows the averag e computation time for each network service request. We observe that the MORSA reduces the average

TABLE IV. MOGA PARAMETERS AND SIMULATION SETUP

Parameter	Description	Value
S	Population Size	100
L	Chromosome Length (i.e. # physical machines in a datacenter)	5 ~ 1,000
R_C	Crossover Rate	0.5
R_M	Mutation Rate	0.05
N	Max Generation	100~10000

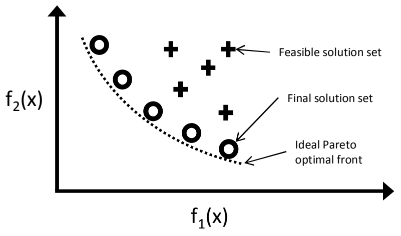

Fig.4. Concept of Pareto optimal solutions.

computation time by approximately 57.51 times compared to naïve ILP. Even for a small number of PMs, e.g., 5 PMs, the computation time of the naïve ILP would yield a considerable overhead for the resource scheduling. In fact, the objective function of minimizing the intra-data center traffic is QAP [17] and it is one of the most difficult NP-hard problems, e.g., QAP with size ≥ 30 has remained unsolved for decades. This result suggests that the MORSA is beneficial in solving NP-hard objective functions for the NFVI resource scheduling.

We evaluate the scalability of the MORSA. We change the total number of PMs in the data center and evaluate the computation time. Unfortunately, we cannot compare the MORSA output and the optimal solution because the naïve ILP does not scale well for this evaluation. (i.e., objective functions are NP-hard). Fig. 6 plots the results for the scalability of the MORSA. Even considering the maximum number of PMs in the evaluation, the MORSA optimizes the VM placement within 40 s. Moreover, parallel distributed genetic algorithms [26] would urge further increases in scale and speed. Accordingly, the MORSA overhead would not be a heavy burden for the NFVI resource scheduling.

Fig. 7 illustrates the Pareto optimal solutions obtained from the MORSA with two conflicting objective functions. The number of VMs is 50 and the total number of PMs is 100 in this evaluation. The MORSA can concurrently obtain both Pareto optimal and weak-Pareto solutions within 5.41 seconds. This means that the MORSA can effectively suggest a wide variety of options for the NFVI resource scheduling to stakeholders. Accordingly, we conclude that the MORSA is a reasonable approach to enhance the NFVI resource scheduling.

V. RELATED WORK

A great amount of work has been devoted to optimizing cloud resources. For example, Verma *et al.* [4] proposed a power and migration aware VM placement algorithm. Similarly, Meng *et al.* [5] provided an algorithm that places VMs to minimize the inter-VM traffic. Mishra *et al.* VI pointed out the drawbacks in existing methods and they proposed a vector arithmetic based method. Mechtri *et al.* [3] proposed the Cloud Broker Framework, which acquires resources from multiple data

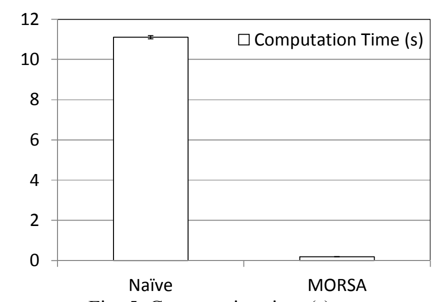

Fig. 5. Computation time (s).

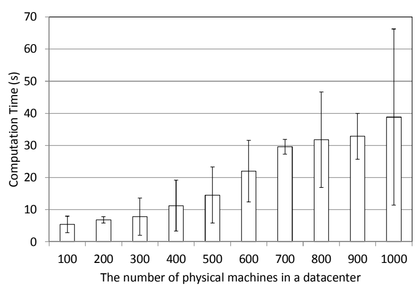

Fig. 6. Scalability of MORSA.

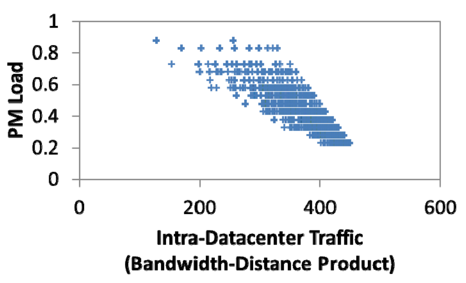

Fig.7. Pareto optimal solutions obtained from MORSA with two conflicting objective functions.

centers. Sch□ller *et al.* [7] proposed the Tenant Infrastructure Management System and they implemented a prototype based on OpenStack. Although the above approaches are effective in scheduling NFVI resources, their studies do not focus on multi-objective scenarios in complex real world situations. There are a few existing approaches for multi-objective scenarios. Xu *et al.* [9] proposed an algorithm with fuzzy multi-objective evaluation for VM placement. Both the MORSA and the algorithm in [9] simultaneously optimize the combination of possibly conflicting objectives; however, the main difference is that the MORSA can treat various requirements of NFVI resources and stakeholder policies with modular plug-ins.

VI. CONCLUSION

We proposed the MORSA for network services on the NFVI. In developing the MORSA, we designed a plug-in architecture to treat various NFVI resource requirements and stakeholder policies. Moreover, the MORSA uses a modified multi-objective genetic algorithm to obtain Pareto optimal solutions with a combination of possibly conflicting objectives. Evaluation results show that the MORSA effectively obtains Pareto optimal solutions in a reasonable computation time. In the future, we plan to scale our evaluation closer to the scale of real data centers rather than that in our evaluation environments. We will also conduct multi-data center scenarios as needed in the NFV context.

REFERENCES

[1] "The European Telecommunications Standards Institute (ETSI)," http://www.etsi.org/ (Ref. May 2014)

[2] M. Chiosi, *et al.*, "Network functions virtualisation - Introductory white paper," http://portal.etsi.org/nfv/nfv_white_paper.pdf (Ref. May 2014).

[3] M. Mechtri, I. Houidi, W. Louati, and D. Zeghlache, "SDN for inter cloud networking," Proc. of SDN4FNS 2013, pp. 25-31, 2013.

[4] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems," *Proc. of ACM/IFIP/USENIX Middleware 2008*, pp. 243—264, 2008.

[5] X. Meng, P. Vasileios, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," *Proc. of IEEE Infocom 2010*, pp. 1154-1162, 2010.

[6] M. Mishra and A. Sahoo, "On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," *Proc. of IEEE CLOUD 2011*, pp. 275-282, 2011.

[7] M. Sch□ller, M. Stiemerling, A. Ripke, and R. Bless, "Resilient deployment of virtual network functions," *Proc. of IEEE RNDM 2013*, 2013.

[8] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters, " *Proc. of IEEE/IFIP NOMS 2010*, pp. 32-39, 2010.

[9] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," *Proc. of IEEE/ACM GreenCom 2010*, pp. 179-188, 2010.

[10] "OpenStack," https://www.openstack.org/ (Ref. May 2014).

[11] "Scheduling - OpenStack Configuration Reference – icehouse," http://docs.openstack.org/trunk/config-reference/content/section_compute-scheduler.html (Ref. May 2014).

[12] Y. Nakajima, T. Hibi, H. Takahashi, H. Masutani, K. Shimano, and M. Fukui, "Scalable, high-performance, elastic software OpenFlow switch in userspace for wide-area network," *Proc. of USENIX ONS2014*, 2014.

[13] "Open vSwitch," http://openvswitch.org/ (Ref. May 2014).

[14] W. Shen, M. Yoshida, T. Kawabata, K. Minato, and W. Imajuku, "vConductor: An NFV management solution for conducting end-to-end virtual network services, " *Submitted to APNOMS 2014.*

[15] A. Fischer, J. F. Botero, M. T. Beck, H. Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888-1906, 2013.

[16] H. Shachnai and T. Tamir, "On two class-constrained versions of the multiple knapsack problem, " *Springer Journal of Algorithmica*, vol. 29, no. 3, pp. 442-467, 2001.

[17] R. E. Burkard, S. E. Karisch, and F. Rendl, "QAPLIB – A quadratic assignment problem library," *Springer, Journal of Global Oprimization*, vol. 10, no. 4, pp. 391-403, 1997.

[18] V. Pareto, "Manual of political economy: a critical and variorum edition," *Oxford University Press*, 2014.

[19] J. H. Holland, "Genetic algorithms," *Elsevier, Artificial Intelligence*, vol. 40, no. 1, pp. 235-282, 1989.

[20] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Springer, Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369-395, 2004.

[21] K. A. Proos, G. P. Steven, O. M. Querin, and Y. M. Xie, "Multicriterion evolutionary structural optimization using the weighting and the global criterion methods," *AIAA Journal*, vol. 39, no. 10, pp. 2006-2012, 2001.

[22] S. Jayaprakasam, S. Rahim, K. A. Sharul, and C. Y. Leow, "A pareto elite selection genetic algorithm for random antenna array beamforming with low sidelobe level," *Proc. of Electromagnetics Reserch B*, vol. 51, pp. 407-425, 2013.

[23] K. Deb, "An efficient constraint handling method for genetic algorithms," *Elsevier, Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311-388, 2000.

[24] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Trans. on Computers*, vol. 100, no. 10, pp. 892-901, 1985.

[25] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *Elsevier, European Juornal of Operational Research*, vol. 176, no. 2, pp. 657-690, 2007.

[26] M. Rodríguez, D. M. Escalante, and A. Peregrín "Efficient distributed genetic algorithm for rule extraction, " *Elsevier, Applied Soft Computing*, vol. 11, no. 1, pp. 733-743, 2011.