A Formal Model for Multi-objective Optimisation of Network Function Virtualisation Placement

Joseph Billingsley[1(✉)], Ke Li[1(✉)], Wang Miao[1(✉)], Geyong Min[1(✉)], and Nektarios Georgalas[2(✉)]

[1] Department of Computer Science, University of Exeter, Exeter, UK
{jb931,k.li,wang.miao,g.min}@exeter.ac.uk
[2] Research and Innovation, British Telecom, Martlesham, UK
nektarios.georgalas@bt.com

Abstract. Ranging from web caches to firewalls, network functions play a critical role in modern networks. Network function virtualisation (NFV) has gained significant interests from both industry and academia, thus making the study of their placement an active research topic. Due to multiple criteria that must be considered by stake holders, e.g. the minimisation of the end-to-end latency and overall energy consumption, the NFV placement problem is in principle a multi-objective optimisation problem. This paper develops a formal model for the NFV placement problem based on queuing theory. By using the popular NSGA-II as the optimiser, the effectiveness of the proposed model is validated through a series of proof-of-concept experiments. In particular, some genetic operators have been developed to match the characteristics of the problem.

Keywords: Network function virtualisation · Multi-objective optimisation · Telecommunications · Queueing theory

1 Introduction

Virtualisation has transformed data centres in recent years. Despite the installed server base in data centres increasing by an estimated 6 million since 2007, the energy consumption of data centres has remained relatively flat [12]. A key step towards this improved energy consumption was the introduction of elastic scaling through virtualisation. Whilst observations of server workload show the peak workload can exceed the average by factors of 2 to 10 [1], elastic scaling allows the data centre to use only the resources requested at any time.

The next step of this transformation is network function virtualisation (NFV) which targets a key part of data centre infrastructure. Traditional data centres are composed of purpose-built computers called 'middleboxes' that perform a single network function such as deep packet inspection, encryption or analytics.

Supported by EPSRC Industrial CASE and British Telecom under grant 16000177.

K. Deb et al. (Eds.): EMO 2019, LNCS 11411, pp. 529–540, 2019.
https://doi.org/10.1007/978-3-030-12598-1_42

Traditional middleboxes have several drawbacks, the most notable of which is their high specialisation. This leads to high cost and inflexibility in the data centre, making the deployment/redeployment of services challenging and time consuming. Without redeployment of services, any placement will becomes inefficient or unsustainable as demand inevitably changes over time.

NFV is an application of virtualisation technologies to middleboxes. Virtual network functions (VNFs) can be run on off-the-shelf hardware and, as in cloud computing, the resources allocated to the VNFs can easily be scaled in accordance with demand. Furthermore, as no physical components are required, VNFs can easily be moved around in the data centre and virtual network structures can be formed to connect them allowing for services to be optimised over time to meet changing demand. NFV is a powerful technology and has been identified as a key component of 5G [10], the Internet of Things [3] and future data centres [7] all of which have the potential to be multi-billion dollars industries [11,12].

A major challenge of NFV is the optimal placement of VNFs in data centres which meet various criteria from stake holders, such as the demand on services and the energy consumption of placements. Note that it is not uncommon that data centres have tens of thousands of servers, each of which may run many virtual machines (VMs), leading to a problem of a tremendous scale. Dependencies (also know as interactions) among components make it yet more challenging and it is important to find solutions robust enough to handle fluctuations in demand. The need to construct virtual networks reflects the virtual network embedding problem [6] so that the NFV placement problem is NP-hard in principle.

Although many efforts have been devoted to the NFV placement problem, there has been no consensus on the problem formulation, especially the corresponding objective functions. Some researchers have opted to test their solution using actual hardware [13] or to simulate the network/hardware by using a discrete event simulator [9]. Note that both hardware- and simulation-based approaches require time to achieve a stable status. Hence whilst these approaches are useful for validating the effectiveness of some algorithms or heuristics, it is difficult to consider them within the actual placement optimisation process. Another alternative is to use some particular heuristics to evaluate solutions [2]. Although simple heuristics are responsive enough to be used in optimisation, designing appropriate heuristics is no-free-lunch. Furthermore, the final solutions may not be reliable if the heuristic is made with improper assumptions. Besides, it is difficult to compare the effectiveness of two algorithms which separately use different heuristics.

In this work we propose a general purpose model based on queuing theory that balances the speed of a heuristic approach against the accuracy of more expensive simulation or hardware approaches, allowing it to be used in the optimisation process. One of the merits of using queueing theory is its ability to handle complex dependencies among components. Further, it provides additional considerations to the number of network functions needed for a service and the arrival and service rates of VNFs, all of which have been ignored by the existing approaches. To validate the effectiveness of the proposed model, it

is incorporated into a classic evolutionary multi-objective optimisation (EMO) algorithm, i.e. NSGA-II [5], in proof-of-concept experiments.

The remainder of the paper is organised as follows. Section 2 provides a formal definition of the NFV placement problem and develops a formal model to evaluate the end-to-end latency and energy consumption. Section 3 applies the model to a particular network topology and derives genetic operators using information from the model. Section 4 examines the effectiveness of the model through proof-of-concept experiments. Finally, Sect. 5 concludes this paper and outlines some potential future directions.

2 Problem Formulation and Model Building

A service is composed of several network functions that must be visited in a particular order. In traditional data centres, network functions are provided by middleboxes whereas with NFV these are provided by VNFs. In the NFV placement problem, the network can be considered as a graph consisting of servers running VMs connected by an arrangement of switches, as in Fig. 1. We assume that the resources of each server are evenly divided into several slices where each VM is allocated to a slice. A VNF can then be placed on to one of these VMs.

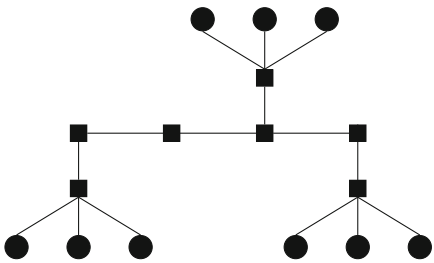


Fig. 1. An example graph with three servers, supporting three VMs (denoted as filled circles) connected by switches (denoted as filled squares).

Each service has a typical arrival rate, which is the amount of traffic it receives over some unit of time on average; while each VNF has a service rate which is the number of packets it can process over the same unit of time when it is placed on a VM slice. If the arrival rate at a switch or VNF is equal to or greater than its service rate, the length of queues at VNFs or switches will tend towards infinity and any dependant services will be inoperable. Similarily if a service requires a particular VNF but no instances of it exist, the service will be inoperable. These characteristics naturally form two constraints on the solution space.

The quality of a particular VNF placement solution can be evaluated by various metrics. In this initial implementation of the model we consider two essential but conflicting objectives, i.e. end-to-end latency and energy consumption. Specifically, the latency for a service is defined as the expected time taken for a

request to visit each VNF in the service whilst the energy consumption can be measured by the number of switches/servers used and the traffic they received. These two objectives should be minimised but they are conflicting with each other. If we consider each objective in isolation, the best solution for latency will use as many servers as possible so as to widely distribute the load, whereas the best solution for energy consumption will lead to the usage of only as many servers as is necessary to obtain a feasible solution.

Before deriving the analytical objective functions, we must consider how packets will be routed through the data centre network since this determines the arrival rate at each switch. To this end, we first need to choose one or more VNFs to forward traffic towards and decide the portion of traffic that each one will receive on average. We also need to determine the path of switches the packets will take to their destination. Generally speaking, the NFV placement problem consists of three inter-connected problems: (1) VNF placement; (2) VNF selection; and (3) packet routing. Depending on the network, the packet routing component may be handled by an existing network protocol and hence not be a part of the optimisation problem. Here we propose a formal model that allows for heuristics or some optimisation techniques to be used to solve for any part of the problem. The model takes two user defined functions, named *selection* and *step*. Specifically, the selection function takes several candidate VNFs as inputs and returns the one or more VNFs that will be selected. The step function takes the current VNF and a target VNF as inputs and returns an object that contains the next possible steps towards the target and the portion of traffic that should be sent down each step.

It is reasonable to assume that requests for a service, which may come from different users or different sources, are independently distributed. Similarly, the time taken to serve a request should not depend on earlier requests. Moreover, the distance between components in a network will likely be very small so that the time spent in flight will be negligible. As a consequence, the end-to-end latency is given by the summation of waiting or processing time at VNFs, servers and switches. Each component in the network contains a packet buffer with a certain capacity, while packet loss occurs when the buffer exceeds this capacity. Bearing these considerations in mind, it is natural to consider using queueing theory as the baseline model. Although finite queues have been well studied in queueing theory, they introduced several additional complexities. If we assume queues have an effectively infinite length, instead of packet loss, the time a packet spends in a queue will increase with the length of the queue. Hence considering infinite queues and optimising for latency should in turn favour solutions that would minimise packet loss. A more thorough analysis of the impact of finite queues and packet loss is planned for future work.

Following the above reasoning, the following assumptions are made with regards to the construction of the network:

1. Every switch and server processes traffic according to a Poisson process with a mean rate of μ_{sw}, while each VNF has a particular service rate with respect to that VNF.

2. For each service there are a set of VMs that contain the first VNF in the service. Each of these VNFs produces traffic according to a Poisson process with a mean rate calculated as the arrival rate of the service divided by the number of VMs in the set.
3. Queues at each network component have an infinite capacity.

Based on these assumptions, we can represent each switch and VNF as an M/M/1 queue. The average time a packet will take to be served is given by [8]:

$$f_w(\mu, \lambda) = \frac{1}{\mu - \lambda} \tag{1}$$

where packets arrive at an average rate λ and are served at an average rate μ.

Subsequently, to determine the latency, we need to first calculate the arrival rate at each component and then calculate the expected latency considering the probability of taking each path. The set of paths can be deduced by using a simple recursive process of selection and step functions as illustrated in Fig. 2. Once the set of paths is determined, the total arrival rate at each node can also be calculated. Once all services have been considered we can calculate the expected latency at each node. This is simply the summation of the waiting time at each node in each path, where the waiting time is given by Eq. (1), then the summation is multiplied by the portion of traffic that is sent down that path. We must also consider that the arrival rate at a node may exceed the service rate and hence make the solution infeasible. In this situation we return the overall constraint violations, allowing a means to evaluate the usefulness of infeasible solutions.

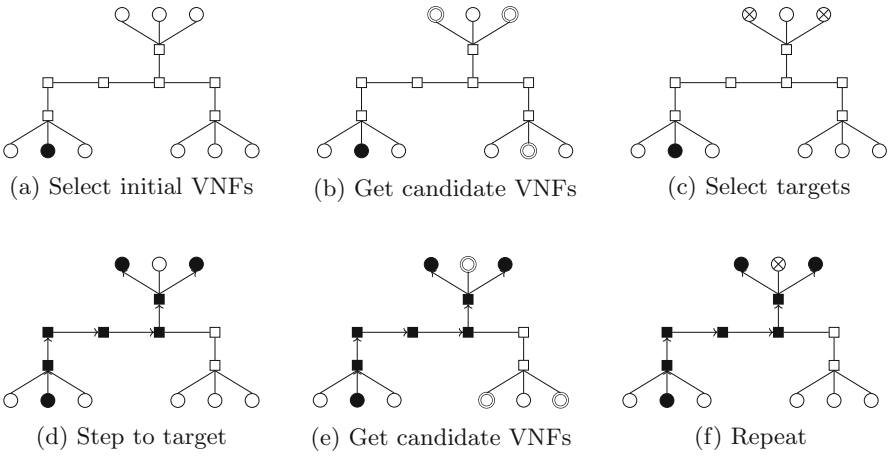


Fig. 2. The paths used by a service can be identified with a simple recursive process.

The energy consumption of the network only depends on the arrival rate at each component. Here we propose a three-level energy model that is suitable for

a range of problems. Each component can either be one of three states, i.e. *off*, *idle* or *active*. A component is off only if it has an average arrival rate of zero. In this case, it is never used. A component is idle when it is not serving a request, otherwise the component is active. In particular, the time a physical switch is active is calculated as the length of its busy period:

$$sw\_busy(i) = \lambda_{sw[i]}/\mu_{sw}; \tag{2}$$

where $\lambda_{sw[i]}$ is the arrival rate at the switch i and μ_{sw} is its service rate. A server is busy if it is serving a request, or if any of the VMs running on the server is serving a request:

$$\begin{aligned} srv\_busy(i) &= 1 - P(server\_idle \cap vms\_idle) \\ &= 1 - (1 - (\lambda_{srv[i]}/\mu_{sw})) \cdot \prod_{j=0}^{k_{vm}} (1 - (\lambda_{vm[i][j]}/\mu_{vm[i][j]})) \end{aligned} \tag{3}$$

where $\lambda_{srv[i]}$ and $\lambda_{vm[i][j]}$ is the arrival rate at the server i and the arrival rate of its jth VM, and μ_{sw} and $\mu_{vm[i][j]}$ are the corresponding service rates.

As services may share components in a data centre, the arrival rate at a component depends on the production rate of each service. Hence we propose to calculate the arrival rate and the expected latency in two steps. Finally the expected energy cost only depends on the arrival rate at each node:

$$\begin{aligned} &\sum_{i=0}^{num\_sw} \begin{cases} sw\_busy(i) \cdot sw_{en\_b} + (1 - sw\_busy) \cdot sw_{en\_i}, & \text{if } \lambda_{sw[i]} > 0 \\ 0, & \text{otherwise} \end{cases} \\ + &\sum_{i=0}^{num\_srv} \begin{cases} srv\_busy(i) \cdot srv_{en\_b} + (1 - srv\_busy) \cdot srv_{en\_i}, & \text{if } \lambda_{srv[i]} > 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

where sw_{en_b}, sw_{en_i} denote energy consumption in the busy and idle states for switches and likewise srv_{en_b}, srv_{en_i} for servers.

The resulting model derives one objective function for latency for each service and another objective function for overall energy consumption. In the next section we will consider a method to find optimal placement solutions to the NFV Placement problem by using this model.

3 Instantiation of NFV Placement Problem

3.1 Fat Tree Networks

Although the model proposed in Sect. 2 is flexible, it is still difficult to define useful step and selection functions for an arbitrary graph. In this paper, for proof-of-concept purposes, we implement these functions for the case of a fat tree network. Fat tree networks are widely used in industry [4] and the underlying principles presented here can be extended to other network structures.

Fat tree networks are described by the number of ports at each switch. We define k as the number of ports for each physical switch and k_{vm} as the number of slices in each VM. In a fat tree topology, there are $(k/2)^2$ core switches. Each core switch connects to one switch in each of k pods. Each pod contains two layers (aggregation and edge) of $k/2$ switches. Each edge switch is connected to each of the $k/2$ aggregation switches of the pod. Each edge switch is also connected to $k/2$ servers. Each server contains a virtual switch connected to k_{vm} VMs. This topology results in $n = (k^3/4) \cdot k_{vm}$ VMs (Fig. 3).

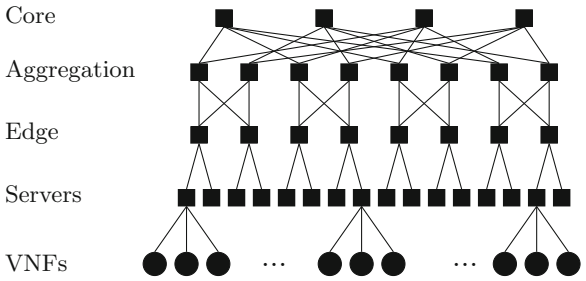


Fig. 3. An example NFV enabled fat tree network with four ports for each hardware switch and space for three VNFs per server.

Based on this definition, a NFV placement solution can be represented as a string of VNFs of length n. As some slices may not be used we also introduce the `None` character to represent an unused slice. To determine the arrival rates we will need to define the selection and routing functions. In this implementation we will use simple heuristics for both. A fat tree network can be efficiently traversed by stepping upwards to the parent node until a common ancestor between the initial and target nodes is reached. Due to the arrangement of aggregate and core switches, there can be several closest common ancestors which lead to equally efficient paths to the target node. In this implementation, the step function is constructed to distribute traffic evenly over each efficient path. As for the selection criteria, we will simply select all candidate VNFs from the closest server.

3.2 Optimisation Algorithm

Having decided on the representation and objective function implementations, we now have enough information to be able to find optimal NFV placement solutions for a fat tree network. Due to its multi-objective nature, we use the most popular EMO algorithm, NSGA-II [5], as the baseline optimiser. However, considering the characteristics of the NFV placement problem and its constraints, some modifications on NSGA-II are developed as follows.

NSGA-II. From the problem formulation introduced in Sect. 2, a solution that violates any constraint will not have meaningful objective values in the NFV placement problem. In this case, we propose to assign all infeasible solutions the lowest possible crowding distance of zero. When comparing two infeasible solutions, the one having the lower constraint violation is preferred.

In principle, when considering the end-to-end latency, each service has its own latency to optimise. In this case, it may end up with a multi-objective optimisation problem with as many objectives as services in the network. However, curse-of-dimensionality is always the Achilles' heel of an optimisation algorithm. To simplify the problem, we propose to combine all services' latencies into a new objective function, i.e. a weighted mean latency over all services as:

$$latency\_agg = \sum_{i=0}^{N_s} \frac{latencies[i]}{N_s} \cdot w_i \tag{5}$$

where N_s is the number of services and w_i gives the relative importance of each service and $\sum_{i=1}^{N_s} w_i = 1$. By setting different w_i, it is easy to prioritise the latency of one service over another. For proof-of-concept purposes, this paper assumes that all services are equally important, i.e. $w_i = 1/N_s, i \in \{1, \cdots, N\}$.

Initialisation. In the original NSGA-II, the initial population is generated by a random sampling over the solution space. Given the existence of constraints, it is highly likely to generate infeasible placement solutions.

To remedy this issue, we need to make sure that at least one instance of each VNF in every service is placed. However, this does not guarantee a feasible solution is made. It is possible that we may need more than one VNF to fulfil a service if the arrival rate exceeds the service rate of a single VNF. In fact, it may be necessary to have multiple instances of earlier VNFs as well, e.g. if the selection function only ever selects one VNF. Whilst these problems mean that we cannot guarantee that all initialised solutions will be feasible, we can determine a lower bound for the required number of instances of a VNF by using the following equation:

$$min\_num(vnf) = \lceil \mu_{vnf} / \lambda_{vnf} \rceil \tag{6}$$

This information can be used to increase the chance of generating feasible solutions. In order to generate a range of solutions, some multiple of the minimum number of each VNF can be placed in the solution. In our proposed initialisation procedure, we opt to randomise the order of these VNFs in the solution but condense the VNFs towards one end of the data centre. This is motivated by the fact that the energy efficiency of a solution is not greatly affected by the placement of VNFs while latency can be significantly harmed if the next VNF in a service is further away. The pseudo code of the initialisation procedure is given in Algorithm 1.

Algorithm 1. Initialisation

$vnfs \leftarrow []$
for all $(i, service)$ in $services$ **do** ▷ Find the minimum number of each VNF
  for all vnf in $service.vnfs$ **do**
    $min\_vnfs \leftarrow ceil(\lambda_{service}/\mu_{vnf})$
    $append(vnfs, [vnf \mid min\_vnfs])$
  end for
end for
$max\_copies \leftarrow floor(n/len(vnfs))$ ▷ Find the maximum copies that could fit
$copies \leftarrow rand(0..max\_copies)$
$vnfs \leftarrow append(vnfs, [vnfs \mid copies])$ ▷ Copy the VNFs
$solution \leftarrow shuffle(vnfs)$ ▷ Shuffle the VNFs
return $append(solution, [None \mid n - len(vnfs)])$ ▷ Pad the output with None

Reproduction Operators. In order to generate new candidate placement solutions, we use crossover and mutation, which are the normal practice in genetic algorithm, to serve as reproduction operators. As for the crossover operation, this paper uses the vanilla uniform crossover without any modification. Due to the consideration of constraint violation, the mutation operator needs some further development to guarantee the feasibility of the mutated solutions. When a mutation occurs, the new value is chosen from the set of VNFs and the empty character, weighted by Eq. (6) – the minimum number of instances of the VNF that is required for a feasible solution.

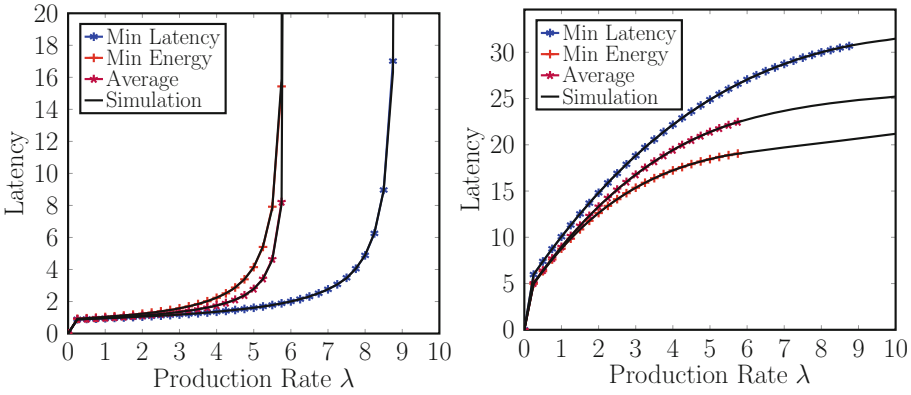
4 Proof-of-Concept Experiments

In this section, we present some proof-of-principle results to demonstrate the effectiveness of the proposed model to serve as objective functions. We also aim to identify some interesting properties of the NFV placement problem that could not have been found with existing models. Due to the page limit, the proof-of-principle experiments here focus on the key results that differentiate the model from previous approaches.

Except where otherwise stated the following parameters are used: $k = 4$, the number of slices on each server $k_{vm} = 3$, 4 services with length 3 are deployed, the service arrival rate for each service is $service_i.arr = 5$, the service rate at each switch $\mu_{srv} = 20$, the service rate at each VNF $\mu_{vnf} = 3$, finally the idle and busy energy consumptions for the servers are $srv_{en\_b} = 2.0$, $sw_{en\_i} = 0.2$, with each switch using $sw_{en\_b} = 1.0$ and $sw_{en\_i} = 0.1$.

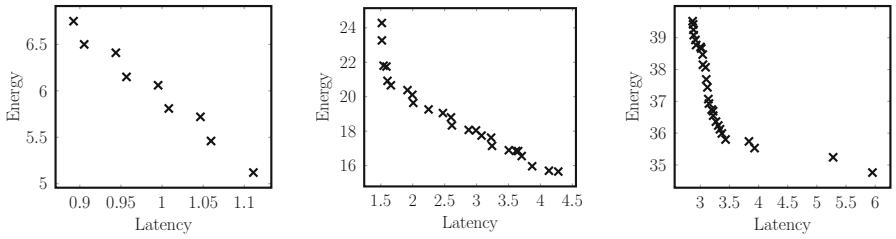
To ensure that the model is accurate we compared the results from the model against those from a discrete event simulation. First the NSGA-II algorithm with a population size of 100 is run for 500 generations to produce a set of non-dominated solutions. Then three representative solutions were selected: one from each objective, and a third solution close to the mean. The objectives calculated by the model where compared against those from a discrete event simulator for the same solutions over a range of arrival rates and are plotted in Figs. 4a and b.

From these two subfigures, we can see that the model and simulation agree to a high level of accuracy up to the point that the queues become saturated. At this point, where the arrival rate exceeds the service rate, the waiting time at a queue will approach infinity and the assumptions the model is constructed with are no longer valid.



(a) Arrival rate vs Latency (b) Arrival rate vs Energy

Fig. 4. Model *versus* simulation results for different arrival rate λ settings.



(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 9$

Fig. 5. Non-dominated solutions from 30 runs of 500 generations for different production rate λ settings.

To gain some insight into the properties of the problem under different parameters, the non-dominated solutions from 30 runs of NSGA-II were gathered and plotted in Figs. 5a to c for different production rates. As expected, we can easily see a trade-off between latency and energy. Less obvious however is the relationship between the two objectives. In Fig. 5a the relationship appears to be linear. However as the production rate increases, improvements in latency require

increasingly more resources. This may be due to the choice of *selection* algorithm which only uses VNFs from the nearest server. If a whole server is required for each VNF to provide enough resources for a feasible solution, groups of VNFs must be deployed to distribute the load. It is unclear at this stage whether this relationship remains under permutations of other parameters but further research could lead to more informed heuristics.

Finally for the sake of future comparison, the hypervolume for different parameter settings was calculated over the course of 30 runs. The nadir point for each setting of α was estimated using the worst objective values for the non-dominated points considering all runs and is used as the reference point. All relevant data including reference points is presented in Table 1 and Fig. 6.

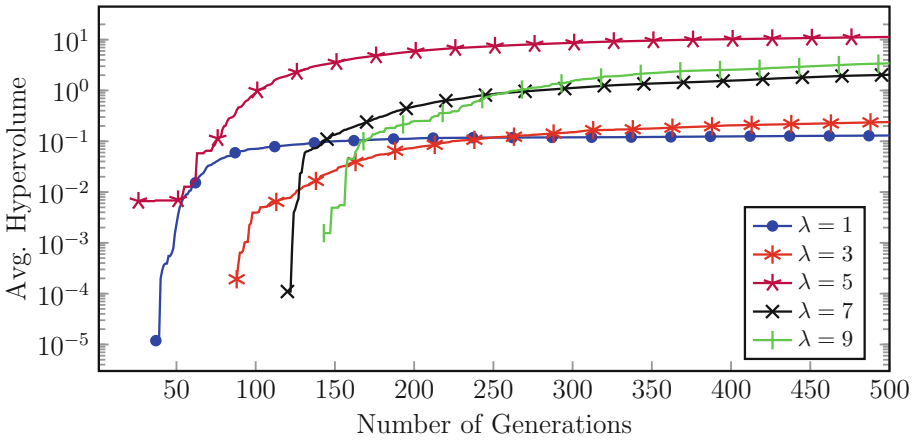


Fig. 6. Average hypervolume for different production rate λ settings.

Table 1. Mean and standard deviation of the hypervolume for different production rates λ and different generations

λ	Ref. point (latency, energy)	Generation				
		100	200	300	400	500
1	(1.111, 6.750)	0.072 ± 0.048	0.114 ± 0.048	0.119 ± 0.048	0.126 ± 0.049	0.129 ± 0.047
3	(1.451, 15.770)	0.004 ± 0.013	0.074 ± 0.085	0.151 ± 0.141	0.206 ± 0.168	0.239 ± 0.164
5	(4.279, 24.280)	0.892 ± 1.436	5.908 ± 2.713	8.613 ± 2.534	10.232 ± 2.705	11.292 ± 2.688
7	(3.421, 32.110)	0 ± 0	0.481 ± 0.797	1.115 ± 1.096	1.560 ± 1.211	2.043 ± 1.286
9	(5.954, 39.520)	0 ± 0	0.250 ± 0.756	1.522 ± 1.858	2.563 ± 2.627	3.415 ± 3.193

5 Conclusion

This paper developed a formal model for the multi-objective telecommunications problem of NFV placement. Whilst this model has been studied previously the proposed model is more thorough than existing approaches and is able to consider important aspects such as the production and arrival rates that existing models cannot. Additionally through the development of new genetic operators we propose an initial algorithm that demonstrates the utility of the model.

There are many practical extensions to the model that could be considered for future work, including the derivation of other objectives using queuing theory. Objectives such as packet loss and service bandwidth have been well studied in queueing theory [8]. Additionally variants of the problem could be considered such as the situation where each service is constrained in each objective e.g. the service is only feasible below some maximum latency. The large scope of the problem and its important real world applications ensure this will be a rich field to explore.

References

1. Armbrust, M., et al.: Above the clouds: a berkeley view of cloud computing (2009)
2. Bari, M.F., Chowdhury, S.R., Ahmed, R., Boutaba, R.: On orchestrating virtual network functions. In: 11th International Conference on Network and Service Management, CNSM 2015, pp. 50–56 (2015)
3. Bizanis, N., Kuipers, F.A.: SDN and virtualization solutions for the internet of things: a survey. IEEE Access **4**, 5591–5606 (2016)
4. Cisco: Cisco global cloud index: forecast and methodology, 2016–2021 (2018). Accessed 03 Oct 2018
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
6. Fischer, A., Botero, J.F., Beck, M.T., de Meer, H., Hesselbach, X.: Virtual network embedding: a survey. IEEE Commun. Surv. Tutor. **15**(4), 1888–1906 (2013)
7. Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: challenges and opportunities for innovations. IEEE Commun. Mag. **53**(2), 90–97 (2015)
8. Kleinrock, L.: Queueing Systems: Theory, vol. 1. Wiley, Hoboken (1975)
9. Mijumbi, R., Serrat, J., Gorricho, J., Bouten, N., Turck, F.D., Davy, S.: Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In: Proceedings of the 1st IEEE Conference on Network Softwarization, NetSoft 2015, London, United Kingdom, 13–17 April 2015, pp. 1–9 (2015)
10. Pei, X., et al.: Network functions virtualisation - white paper on NFV priorities for 5G (2017)
11. Reichert, C.: 5G industry to be worth $1.2 trillion by 2026: ericsson. ZDNet, February 2017. Accessed 03 Oct 2018
12. Shehabi, A., et al.: United States data center energy usage report (2016)
13. Xu, J., Fortes, J.A.B.: A multi-objective approach to virtual machine management in datacenters. In: 8th International Conference on Autonomic Computing, ICAC 2011, pp. 225–234 (2011)