Received: 20 December 2018 | Revised: 6 September 2019 | Accepted: 13 September 2019

DOI: 10.1002/dac.4222

RESEARCH ARTICLE

WILEY

Cost minimization in placing service chains for virtualized network functions

Chien Ting Wang[1] | **Ying-Dar Lin**[2] | **Chih-Chiang Wang**[3] | **Yuan-Cheng Lai**[4]

[1]Graduate Degree Program of Network and Information Systems, National Chiao Tung University and Academia Sinica, Hsinchu, Taiwan

[2]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

[3]Department of Computer Science & Information Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan

[4]Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan

Correspondence
Chien Ting Wang, Graduate Degree Program of Network and Information Systems, National Chiao Tung University and Academia Sinica, Hsinchu, Taiwan.
Email: ctwang@cs.nctu.edu.tw

Funding information
National Chiao Tung University

Summary

Network function virtualization (NFV) places network functions onto the virtual machines (VMs) of physical machines (PMs) located in data centers. In practice, a data flow may pass through multiple network functions, which collectively form a service chain across multiple VMs residing on the same or different PMs. Given a set of service chains, network operators have two options for placing them: (a) minimizing the number of VMs and PMs so as to reduce the server rental cost or (b) placing VMs running network functions belonging to the same service chain on the same or nearby PMs so as to reduce the network delay. In determining the optimal service chain placement, operators face the problem of minimizing the server cost while still satisfying the end-to-end delay constraint. The present study proposes an optimization model to solve this problem using a nonlinear programming (NLP) approach. The proposed model is used to explore various operational problems in the service chain placement field. The results suggest that the optimal cost ratio for PMs with high, hybrid, and low capacity, respectively, is equal to 4:2:1. Meanwhile, the maximum operating utilization rate should be limited to 55% in order to minimize the rental cost. Regarding quality of service (QoS) relaxation, the server cost reduces by 20%, 30%, and 32% as the end-to-end delay constraint is relaxed from 40 to 60, 80, and 100 ms, respectively. For the server location, the cost decreases by 25% when the high-capacity PMs are decentralized rather than centralized. Finally, the cost reduces by 40% as the repetition rate in the service chain increases from 0 to 2. A heuristic algorithm, designated as common sub chain placement first (CPF), is proposed to solve the service chain placement problem for large-scale problems (eg, 256 PMs). It is shown that the proposed algorithm reduces the solution time by up to 86% compared with the NLP optimization model, with an accuracy reduction of just 8%.

KEYWORDS

network function placement, network function virtualization, nonlinear programming, service chains

1 | INTRODUCTION

Network functions play a critical role in any network and are used to provide a wide range of services, including network security and router processing. However, network functions have a high cost and must therefore be carefully deployed. The network function cost comprises two components, namely, the management cost and the operational cost. Network functions are typically provided by customized equipment, and hence, a large human management cost is incurred in setting up the related equipment one by one. In addition, the customized equipment used to provide network functions usually has the form of specialized hardware, such as routers or firewalls, which can process large quantities of packets, but are expensive. In addition, such hardware tends to have poor flexibility. For example, when a network function is operated under low traffic load conditions, or during the off-peak period, all the related equipment must still be kept running despite the low occupancy rate, and hence, a large proportion of the server rental cost is effectively wasted.

Network function virtualization (NFV) mitigates these problems by using software to implement the required network functions rather than traditional specialized hardware. By doing so, both the management cost and the operational cost can be significantly reduced since the network functions can be centralized and placed on the same X86 computer. Furthermore, by using NFV, it is possible to adjust the quantity of virtual machines (VMs) in operation dynamically in accordance with changes in the network load so as to reduce the server rental cost while still maintaining the required quality of service (QoS).

With the technical assistance of NFV, it is easy for a service provider to place the required services onto VMs. These services may be provided by either a single network or by multiple network functions which collectively form a service chain. For example, in a network security service chain, the packets from the customers are first subject to an initial check at the firewall, and any suspicious packets are then sent to an intrusion detection system (IDS) for further checking.

However, to gain the full benefits of NFV, the service chain placement must be carefully optimized such that the rental cost is reduced and the required end-to-end delay constraint is still satisfied. The rental cost scales proportionally with the number of VMs operated within physical machines (PMs). Hence, by placing repeated services on the same VM so as to decrease the total number of VMs, the rental cost can be substantially reduced. However, the end-to-end delay comprises two components, namely, the network delay and the server delay. Thus, while increasing the number of network functions implemented on the same PM reduces the network delay, the processing capacity is reduced, and hence, the server delay may increase. By contrast, when the services in a service chain are placed on physically remote PMs, the network delay increases. A challenge therefore exists in properly controlling the placement of the network functions so as to achieve an acceptable tradeoff between the rental cost and the network performance.

As shown in Table 1, the literature contains many studies on NFV. For example, the authors in Savi et al,[1] Zhang et al,[2] Zeng et al,[3] and Bouet et al[4] discussed the service chain placement problem, while those in Moens and Turck,[5] Addis et al,[6] and Luizelli et al[7] examined the VM placement problem and attempted to minimize the number of PMs or the number of instances. Lin et al[8,9] similarly addressed the VM placement problem. In general, solving the service chain placement problem involves two steps, namely, VNF-CC (ie, the service chain composition problem) and VNF-FGE (ie, the forwarding graph embedding problem). Gupta et al[10] provided feasible solutions for both problems.

Although the studies above provide effective methods for minimizing the service chain placement cost, they neglect the QoS constraints imposed on the service chains. As a result, the service experience of the user may be significantly impaired. Accordingly, the present study proposes an optimization model designed to minimize the rental cost of service chain placement while simultaneously satisfying the specified end-to-end delay constraint. The optimization model is used to explore various related problems, including (a) the optimum arrangement of network functions that minimizes the placement cost under different PM processing abilities; (b) the effects of the end-to-end delay constraint on the optimum arrangement of the network functions; (c) the effect of the PM utilization rate on the cost; and (d) the effect of service repetition on the cost appropriate matches when choosing service chaining.

The main contributions of the present study can be summarized as follows. (a) The service chain placement problem is solved using an optimization model based on a nonlinear programming (NLP) approach and a heuristic algorithm designated as common sub chain placement first (CPF). (b) Five key strategies are identified by the NLP optimization model for reducing the server rental cost, namely, (i) renting low-capacity PMs, (ii) relaxing the QoS constraint, (iii) increasing the service repetition rate, (iv) decentralizing the high-capacity PMs, and (v) controlling the system utilization at 55%. (c) The performance of the CPF algorithm is evaluated and is shown to yield an effective improvement in the solution time compared with that of the NLP optimization model.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the system architecture and proposed optimization model. Section 4 presents and discusses the experimental results. A heuristic

TABLE 1 Reference comparison table

Category	Paper Comparison				
	Paper	Objective	Heuristic	VNF-FGE	VNF-CC
Minimize cost	Our paper	Min cost	Y	Y	Y
	Lin et al[8]	Min cost	N	Y	N
	Lin et al[9]	Min cost	Y	Y	N
	Zhang et al[2]	Min cost	Y	Y	N
	Zeng et al[3]	Min cost	Y	Y	N
	Cohen et al[11]	Min cost	Y	Y	N
	Bouet et al[12]	Min cost	Y	Y	N
	Ghaznavi et al[13]	Min operation cost	Y	Y	N
	Bouet et al[4]	Min cost	Y	Y	N
	Khebbache et al[14]	Min cost	Y	Y	Y
	Soualah et al[15]	Min cost	Y	N	Y
Multiobjective	Jang et al[16]	Min resource usage	Y	Y	N
	Gupta et al[10]	Max remaining network resources	Y	Y	Y
	Riggio et al[17]	Multiobjective (data rate, node, and latency)	Y	Y	N
	Khebbache et al[18]	Multiobjective	Y	Y	N
	Baek et al[19]	Min latency	Y	Y	Y
Minimize #PM/VM	Kuo et al[20]	Min network resource consumption	Y	Y	N
	Moens and Turck[5]	Min PM	N	Y	N
	Addis et al[6]	Min number core	N	Y	N
	Savi et al[1]	Min NFV node	N	Y	N
	Luizelli et al[7]	Min instance	Y	Y	N

Abbreviations: NFV, network function virtualization; PM, physical machine; VM, virtual machine.

algorithm for solving large-scale service chain placement problems is additionally introduced. Finally, Section 5 provides some brief concluding remarks and indicates the intended direction of future research.

2 | RELATED WORK

This section commences by introducing existing proposals for obtaining optimal solutions to the service chain placement problem. Several heuristic techniques for minimizing the rental cost of service chain placement are then briefly described.

2.1 | Optimization solutions

Linear programming algorithms provide an effective means of solving many different types of optimization problems. Problems with low complexity are generally solved using software tools such as CPLEX and GLPK.[21] Previous studies on the service chain placement problem consider five main aspects, namely, (a) minimizing the number of PMs (Moens and Turck[5]); (b) minimizing the network resource usage (Jang et al[16]); (c) adapting the service chain placement to the traffic flow (Gupta et al[10] and Addis et al[6]); (d) maintaining the load balance at the data center (Addis et al[6]); and (e) performing service chain placement subject to physical resource constraints Lin et al.[8]

Broadly speaking, the studies above consider two objectives: (a) reducing the number of PMs, VMs, and used cores, and (2) minimizing the rental cost. Regarding the first objective, the studies in Savi et al,[1] Moens and Turck,[5] Addis et al,[6] and Luizelli et al[7] solve the service chain placement problem subject to three constraints, namely, (a) the server capacity, (b) the link capacity, and (c) the placement of just one service on each VM. The study in Moens and Turck[5] adopts an additional end-to-end delay constraint. As described above, the end-to-end delay comprises two components, namely, the network delay and the server delay. The network delay is equal to the product of the covered path length and a certain constant value, where this value depends on the unit cost. Meanwhile, the server delay is calculated as the product of another constant value and the involved VM quantity (Addis et al[6] and Luizelli et al[7]) or as a combination of the upscaling latency and the context switching latency (Savi et al[1]).

However, the use of constant values to calculate the server delay is unrealistic. For example, under heavy traffic loads, the queuing time increases compared with that under light loads, and hence, the server delay also increases. In other words, the server delay is not in fact constant but varies with the processing load. Accordingly, the present study adopts

M/M/1 queuing theory to more accurately model the server delay variation caused by the quantity of services placed on a PM.

Regarding the second objective (ie, minimizing the rental cost), the rental cost comprises many different components, including most notably the node cost and the link cost. However, the rental cost also includes the account deployment cost (Lin et al[8,9]), the reassignment and migration cost (Ghaznavi et al[13]), and the distance cost of the service chain (Cohen et al[11]). In attempting to minimize the rental cost, most previous studies consider three constraints, namely, (a) the server capacity, (b) the link capacity, and (c) the placement of just one service per VM. However, the studies in Zhang et al[2] and Zeng et al[3] impose an additional multicast path constraint since they mainly discuss a multicast scenario.

In attempting to minimize the rental cost, the present study focuses mainly on the node cost, ie, the server rental cost paid by the consumer to the server provider. Notably, compared with previous studies, in which the rental cost is assumed to have a constant value per PM (Bouras et al[22]), the present study assumes that the rental cost varies depending on the capacity of the PM. Thus, the analysis provides a more realistic assessment of the effect of the PM capacity on the rental cost when evaluating different service chain placement options.

2.2 | Heuristic solutions

When solving problems of high complexity, it is frequently difficult to obtain a solution in good time using exact methods. Accordingly, the use of heuristic algorithms is often preferred. For example, in previous studies aimed at minimizing the rental cost of service chain placement, Ghaznavi et al[13] presented a model to obtain a tradeoff between the bandwidth and the host resource consumption, while Cohen et al[11] adopted a rounding-based heuristic. Bouet et al[12] formulated the problem of minimizing the rental cost in terms of virtual deep packet inspection placement and then solved the problem using a greedy placement strategy. Finally, in Kuo et al[20] obtained the minimal rental cost using a chain deployment algorithm based on the tradeoff between the path length and the VM reuse factor.

3 | SYSTEM ARCHITECTURE AND MODEL

Figure 1 shows the detailed work flow of the service chain placement problem. As shown, the data center comprises a group of PMs, where each PM has many VMs placed on it. One VM can only provide one kind of service. However, services of the same type can be grouped together. When renting VMs to provide service chains, the cost paid by the consumers is referred to as the server rental cost. In practice, this cost depends on many factors, such as the number of servers, the price of each server, the utilization of each server, and so on.

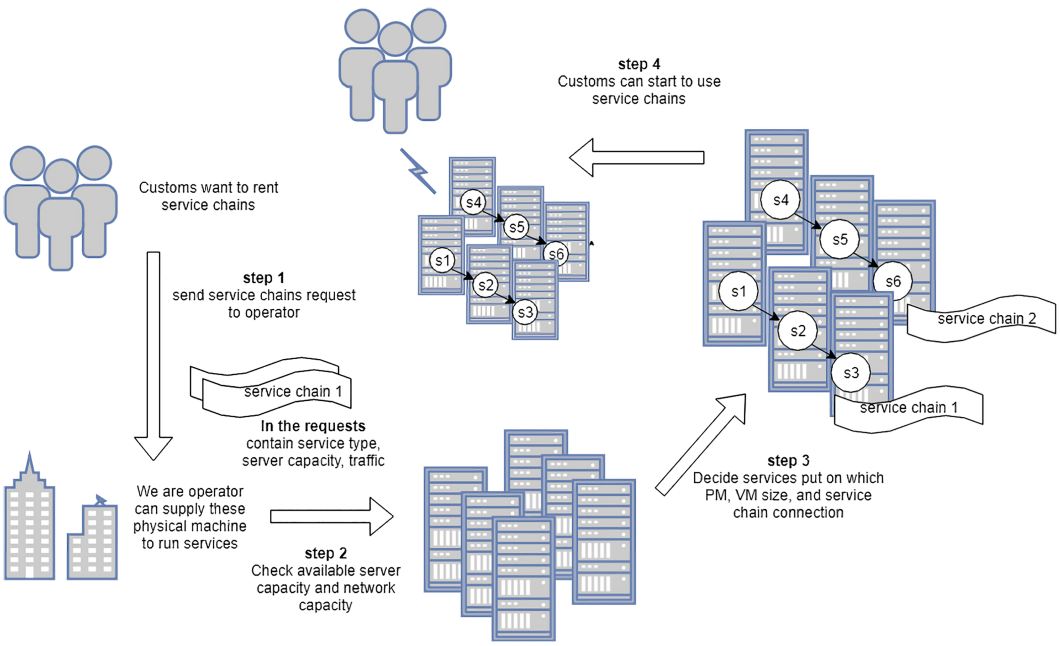


FIGURE 1 Detailed work flow of service chain placement problem

TABLE 2 Key mathematical notations

Symbol	Meaning
M	Number of service chains
N	Number of physical machines
C	Set of service chains
P	Set of physical machines
$d_{m,n}$	Network delay from PM m to PM n
$dis(c)$	Summary network delay of service chain c
p_{min}	Minimum probability that
	end-to-end delay can be bound by t_{max}
t_{max}	Maximum end-to-end delay
μ_p	Processing speed of PM p
λ_c	Traffic of service chain c
$f(c)$	Services in service chain c
$w(p)$	Cost of PM p

Abbreviation: PM, physical machine.

This section presents the NLP model proposed in this study for optimizing the service chain placement problem, ie, to determine the service chain placement which minimizes the server rental cost when placing all of the service chains onto VMs subject to certain constraints end-to-end delay (e.g., 400 ms for VoIP service chain). In formulating the model, the rental cost of each VM is assumed to depend on the processing speed (as specified by the data provided by Amazon and the testing results reported by TPC-C[23]).

In practical applications, the end-to-end delay constraints are designed based on the particular consumer requirement. For example, the end-to-end delay must not exceed 400 ms for VoIP, 150 ms for business usage, and 20 ms for a good QoS Cisco.[24] Table 2 lists the main notations used in the present study. As shown, the set of service chains to be placed is denoted as C, while the number of service chains is denoted as M. Similarly, the set of PMs within the data center is denoted as P, while the number of machines is denoted as N. The network delay from PM m to PM n is denoted as $d_{m,n}$ and is determined by the number of hops between them. Each service chain should lie completely within an end-to-end delay constraint denoted as t_{max}. In particular, each service chain must be completed with a minimum probability that t_{max} can be satisfied equal to p_{min}. Finally, the processing speed of PM p is denoted as p, while the traffic load of service chain c is denoted as λ_c.

As described above, the service chain placement problem is formulated in the present study as a NLP problem, in which the objective is to obtain the minimum server rental cost, ie,

$$\text{Min}\left(\sum_{i\in C}\sum_{j\in f(i)}\sum_{p\in P}F_{i,j,p} * w(p)\right), \tag{1}$$

where $F_{i,j,p}$ is limited to a value of either 0 or 1 (denoting whether service j of service chain i is placed on PM p or not). In addition, $w(p)$ is the rental cost for PM p, and is calculated as

$$w(p) = CT_p * V_p, \tag{2}$$

where V_p is the number of VMs in PM p and CT_p is the VM rental cost. Service j of service chain i can only be placed on one PM, ie,

$$\sum_{i\in C}\sum_{j\in f(i)}\sum_{p\in P}F_{i,j,p} = 1. \tag{3}$$

The end-to-end delay of a service chain, $mc(c)$, comprises two parts, namely, the server delay, $svd(c)$, and the network delay, $dis(c)$, ie,

$$mc(c) = svd(c) + dis(c). \tag{4}$$

The server delay, $svd(c)$, is the total delay time incurred by service chain c at all the PMs used, and is given by

$$svd(c) = \sum_{j\in f(c)}\sum_{p\in P}F_{c,j,p} * delay(p), \tag{5}$$

where $delay(p)$ is the delay time at PM p and is determined by two characteristics of the PM, namely, the processing speed, μ_p, and the traffic load $\lambda_h$derived from service chain h. From M/M/1 queuing theory, the delay time can be computed as (Kumar et al[25])

$$delay(p) = \frac{1}{\mu_p - \sum_{h \in C} \sum_{g \in f(h)} * F_{h,g,p} * \lambda_h}. \tag{6}$$

The network delay of service chain c, $dis(c)$, can be expressed as

$$dis(c) = \sum_{(m,n) \in loc(c)} d_{m,n}, \tag{7}$$

where $loc(c)$ denotes the PM location of service chain c and $d_{m,n}$ is the network delay from PM m to PM n.

The minimum probability p_{min} of the traffic flow satisfying the end-to-end delay constraint t_{max} can be derived from the Chebyshev inequality formula as

$$\frac{(t_{max} - mc(c))^2}{(t_{max} - mc(c))^2 + mc^2} \geq p_{min}. \tag{8}$$

In the present study, the NLP described above is solved using the Basic Open-source Nonlinear Mixed Integer programming (BONMIN) tool (Bonami[26]) based on a branch-and-bound algorithm. For illustration purposes, assume that there exist three PMs (PM1, PM2, and PM3) at the data center, and nine service chains (each containing four services) are to be placed. In other words, a total of 36 services are to be placed onto the three PMs. In solving the placement problem using the NLP model proposed in this study, the first service is simply placed randomly on one of the three PMs in order to estimate the initial lower boundary for the first level of branches. The second service is then placed also randomly on one of the three PMs to estimate the second lower boundary in the second level of branches. If a higher value of the second lower boundary is obtained for one particular branch, calculations cease at this branch; otherwise, calculations continue for the third service in the branch. The procedure continues in this way until all the services have been placed, at which point, the current solution is taken as the optimal solution for the placement problem.

4 | PERFORMANCE EVALUATION

The performance of the proposed service chain placement model was evaluated by means of numerical simulations. The BONMIN solver was implemented using A Mathematical Programming Language (AMPL[27]). In performing the simulations, it was assumed that the PMs were heterogeneous (ie, the PMs at the data center were either all high-capacity or all low-capacity machines, or a mixture of the two) and the network delay over the data link between each pair of devices was constant.

As described in Yoon and Kamal,[28] data centers typically organize PMs into a k-ary fat tree topology to deliver scalable bandwidth at a lower hardware cost and to allow host communication at line speed. The present study considered a fat tree topology with $k = 4$. In other words, as shown in Figure 2, four core switches were connected to four pods, where each pod consisted of a two-layer switch connected to four PMs. The top layer contained two aggregation switches, while the bottom layer contained two edge switches. Furthermore, within every pod, each edge switch was connected to all the aggregation switches and two PMs.

Table 3 summarizes the default settings considered in the experiments. It is noted that the server rental costs of small, medium, and large PMs are taken from the 3-year benchmark research of TPC-C and Amazon (Verbitski et al[29]) and are based on the server processing speed. In addition, the end-to-end delay constraint, t_{max}, and minimum bound probability, p_{min}, are defined in such a way as to prevent the end-to-end delay from being too long. The end-to-end delay constraint indicates the maximum time that the end-to-end delay cannot exceed when completing the entire service chain. Meanwhile, the minimum bound probability indicates the percentage of service chains that can be successfully subject to the end-to-end delay constraint. For example, a probability bound of $p_{min} = 0.9$ indicates that 90% of the data traffic is expected to satisfy the end-to-end delay. For simplicity, the network delay incurred within the same PM is ignored. In other words, the network delay depends only on the number of hops between PMs. The network delay for one hop was set as 1 ms. Thus, for four hops between P1 and P4, the total network delay was calculated as to 4 ms.

The numerical analysis focused on five issues, namely, (a) the server heterogeneity, (b) the operating utilization rate, (c) the effects of QoS relaxation, (d) the server location, and (e) the degree of repetition in the service chains. It is noted

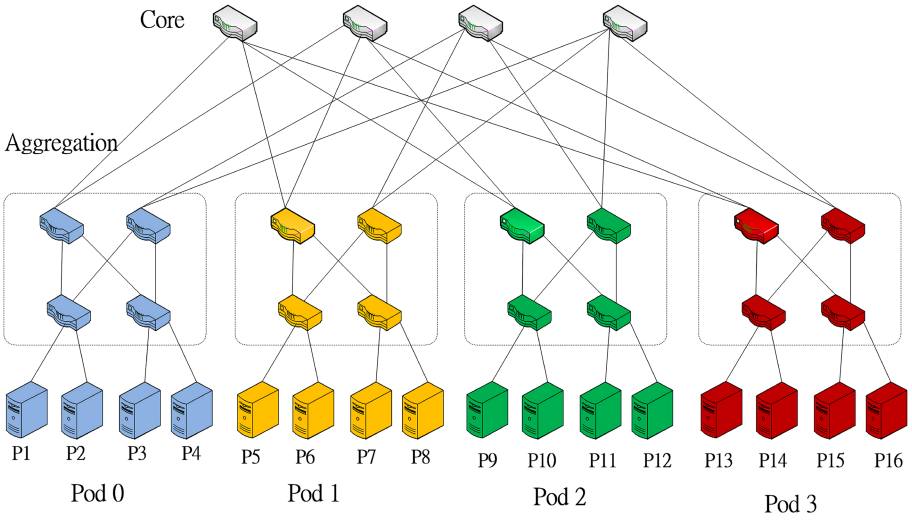


FIGURE 2 4-ary fat tree topology

TABLE 3 Default simulation settings

Parameter	Value
One hop network delay	1 ms
PM number	16
Traffic	100 requests/s
Server rental cost	0.11 USD/hour (small) 0.44 USD/hour (medium) 0.88 USD/hour (large)
Server processing speed	1807 requests/s (small) 4597 requests/s (medium) 5971 requests/s (large)
Average service chain length	4
The number of service chains	10
Max end-to-end delay (t_{max})	20 ms
Bound probability (p_{min})	90%

Abbreviation: PM, physical machine.

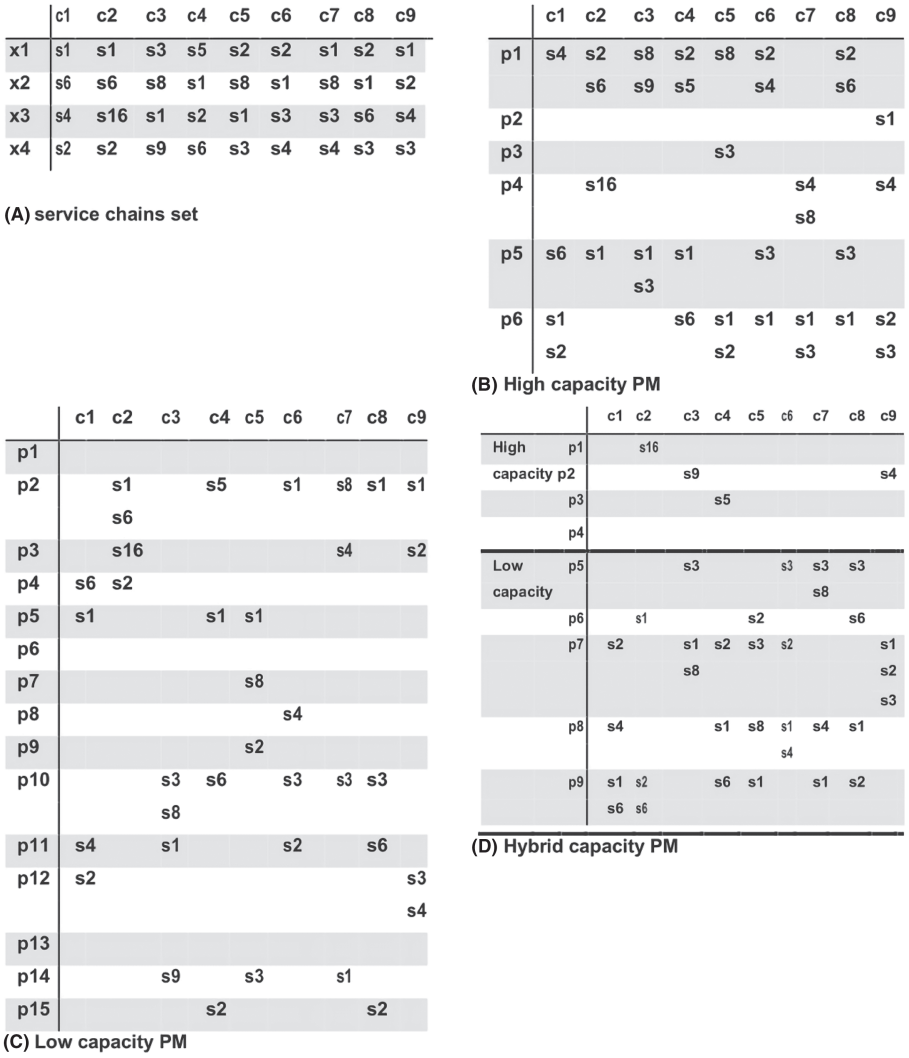
that these issues have not been previously discussed in the literature, and hence, a direct benchmarking of the present results is not possible.

The system performance was evaluated using four metrics: (a) the server rental cost, ie, the total rental cost of all the VMs operated in the PMs; (b) the server delay, ie, the average delay time of a service chain incurred by all its computations; (c) the network delay, ie, the average delay time of a service chain incurred by all its communications; and (d) the end-to-end delay, ie, the sum of the server delay and network delay for the same service chain.

4.1 | Example run

Figure 3A shows the nine service chains considered in the example run of the proposed model. (Note that p_{min} is set as 90%.) The simulation run was designed to explore the effect of the server heterogeneity on the server rental cost. Three scenarios were considered, namely, (a) high-capacity PMs—six high-capacity PMs; (b) hybrid capacity PMs—four high-capacity PMs and five low-capacity PMs; and (c) low-capacity PMs—15 low-capacity PMs. In accordance with Table 3, the price ratio of the low-capacity PMs to the high-capacity PMs was set as 1:4, while the processing speed ratio was set as 2:5.

For the case of high-capacity PMs (see Figure 3B), placing the same type of services on the same VM does not result in a high server delay. The VM sharing ratio (ie, the ratio of saved VMs to the maximum number of VMs) can therefore be increased to a maximum in order to reduce the server rental cost. Thus, as shown in Figure 4, only 18 VMs are required for the 36 services; resulting in a VM sharing ratio of $18/36 = 50\%$. However, as the VM sharing ratio increases, the services within the same service chain may not be located close to one another, and hence, the network delay increases. For the low-capacity scenario (see Figure 3C), placing too many services on a single PM increases the server delay dramatically. The VM sharing ratio is thus equal to just 25% (see Figure 4). However, the network delay must be reduced in order to



(A) service chains set

	c1	c2	c3	c4	c5	c6	c7	c8	c9
x1	s1	s1	s3	s5	s2	s2	s1	s2	s1
x2	s6	s6	s8	s1	s8	s1	s8	s1	s2
x3	s4	s16	s1	s2	s1	s3	s3	s6	s4
x4	s2	s2	s9	s6	s3	s4	s4	s3	s3

(B) High capacity PM

	c1	c2	c3	c4	c5	c6	c7	c8	c9
p1	s4	s2	s8	s2	s8	s2		s2	
		s6	s9	s5		s4		s6	
p2									s1
p3					s3				
p4		s16					s4		s4
							s8		
p5	s6	s1	s1	s1		s3		s3	
			s3						
p6	s1			s6	s1	s1	s1	s1	s2
	s2				s2		s3		s3

(C) Low capacity PM

	c1	c2	c3	c4	c5	c6	c7	c8	c9
p1									
p2		s1		s5		s1	s8	s1	s1
		s6							
p3		s16					s4		s2
p4	s6	s2							
p5	s1			s1	s1				
p6									
p7					s8				
p8						s4			
p9					s2				
p10			s3	s6		s3	s3	s3	
			s8						
p11	s4		s1			s2		s6	
p12	s2								s3
									s4
p13									
p14			s9		s3		s1		
p15				s2				s2	

(D) Hybrid capacity PM

		c1	c2	c3	c4	c5	c6	c7	c8	c9
High capacity	p1		s16							
	p2			s9						s4
	p3				s5					
	p4									
Low capacity	p5			s3			s3	s3	s3	
								s8		
	p6		s1			s2			s6	
	p7	s2		s1	s2	s3	s2			s1
				s8						s2
										s3
	p8	s4			s1	s8	s1	s4	s1	
							s4			
	p9	s1	s2		s6	s1		s1	s2	
		s6	s6							

FIGURE 3 Example run

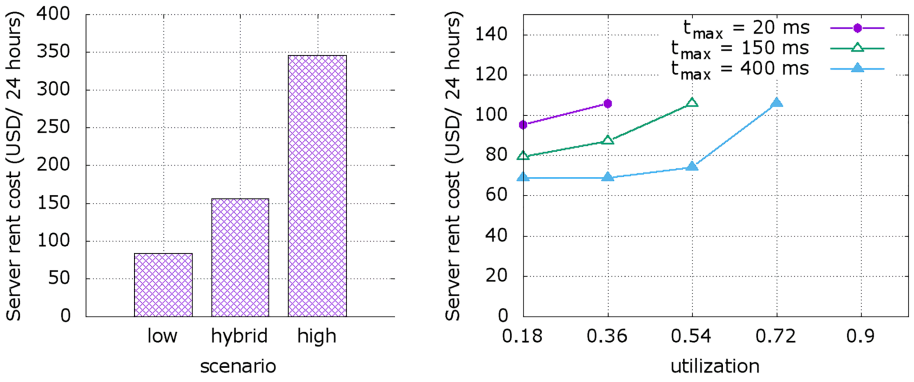
NAME	COST	NETWORK DELAY	SERVER DELAY	VM SHARING RATIO	VM NUMBER	PM NUMBER
Low capacity	2.97	14 ms	3 ms	25%	27	12
High capacity	7.92	17 ms	1 ms	50%	18	6
Hybrid capacity	3.41	12 ms	3 ms	47%	19	8

FIGURE 4 Intermediate results for server heterogeneity

satisfy the end-to-end delay constraint. Consequently, the services within each service chain must be placed close to one another. For the hybrid capacity scenario (shown in Figure 3D), the services are placed preferentially on the low-capacity PMs in order to minimize the rental cost. In particular, only 1/10 of the services are placed on high-capacity PMs.

4.2 | Server heterogeneity: The rental cost ratio between high-capacity PMs, hybrid PMs, and low-capacity PMs should be 4:2:1

Experiments were performed to examine the effect of the server heterogeneity on the service chain placement decision. Three scenarios were considered: (a) high-capacity PMs: using only high-capacity PMs for service chain placement; (b) hybrid PMs: using a mixture of high-capacity PMs and low-capacity PMs for service chain placement; and (c) low-capacity PMs: using only low-capacity PMs for service chain placement. The experiments were performed using the same setup as that described for the example run above. The results of a statistical analysis revealed that 32 VMs were necessary in all three scenarios. In general, the server rental cost of the high-capacity PM set is much higher than that of the low-capacity PM set since the cost ratio between the two types of PM is 8:1, whereas the capacity ratio is only 3.5:1. Thus, the results



(A) Server Heterogeneity

(B) System utilization

FIGURE 5 Effects of server heterogeneity and system utilization

presented in Figure 5A suggest that when placing the same service chains, the server rental cost ratio of the PM sets with high, hybrid, and low capacity, respectively, is equal to 4:2:1.

4.3 | Operating utilization rate: The operating utilization rate should be limited to 55% in order to minimize the rental cost

Further experiments were performed to examine the relationship between the server rental cost and the operating utilization rate under three different end-to-end delay constraints, ie, $t_{max} = 20$, 150, and 400 ms, respectively. The utilization rate depends on the traffic of the service chains. As shown in Figure 5B, the server rental cost increased as the utilization rate increased from 0.18 to 0.72 under all three end-to-end delay constraints. Moreover, for a constant utilization rate, the server rental cost increased with a stricter end-to-end delay.

For a system utilization rate greater than 0.54, the end-to-end delay constraints of $t_{max} = 20$ ms and $t_{max} = 150$ ms could not be achieved, which implies that insufficient resources were available for service chain placement. In general, the results presented in Figure 5B are reasonable since it is intuitively more difficult to satisfy a stricter end-to-end delay constraint, and hence, more PMs are required to handle the traffic. As a result, the server rental cost increases. Furthermore, under a stricter end-to-end delay constraint, a greater utilization increase results in a more rapid increase in the server delay. Overall, the results presented in Figure 5B show that the rental cost increases dramatically as the utilization rate increases beyond 55%. Thus, it is suggested that operators limit the maximum utilization rate to less than 55% in order to minimize the rental cost.

4.4 | QoS relaxation: The rental cost drops by 20%, 30%, and 32% as the delay constraint is relaxed from 40 to 60, 80, and 100 ms, respectively

Experiments were performed to examine the relationship between the server rental cost and t_{max} so as to determine the optimum service chain placement under different probability boundaries, p_{min}. In performing the simulations, t_{max} was varied from 20 to 100 ms, while the probability boundary was varied from 0.3 to 0.9 in uniform intervals of 0.2.

As shown in Figure 6, the server rental cost decreased under a looser end-to-end delay constraint for all values of the probability boundary due to a corresponding increase in the VM sharing ratio. A high VM sharing ratio indicates that the same type of service belonging to different service chains can be placed on the same PM. However, as shown in Figure 6, the server rental cost remained constant as the end-to-end delay constraint was increased beyond 100 ms since by this point, all of the VMs that can be shared have been put together. An inspection of Figure 6 shows that for a probability boundary of 0.9, the rental cost decreases by 20%, 30%, and 32% as the end-to-end delay constraint is relaxed from 40 to 60, 80, and 100 ms, respectively. Notably, the change in the rental cost as t_{max} is relaxed from 20 to 40 ms is very small since both constraints are already very tight, and hence, very little latitude exists for further reductions in the rental cost.

4.5 | Server location: The rental cost decreases by 25% when the high-capacity VMs are decentralized rather than centralized

Further simulations were performed to examine the effect of the server location on the rental cost. The simulations used a fat tree topology to connect the PMs and considered three different physical arrangements of the servers, as shown in Figure 7A, in which L denotes the high-capacity PMs (large PM) and S denotes the low-capacity PMs (small PM).

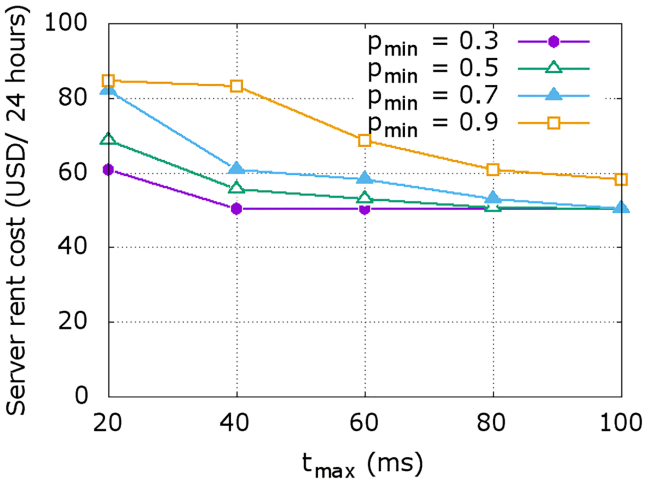
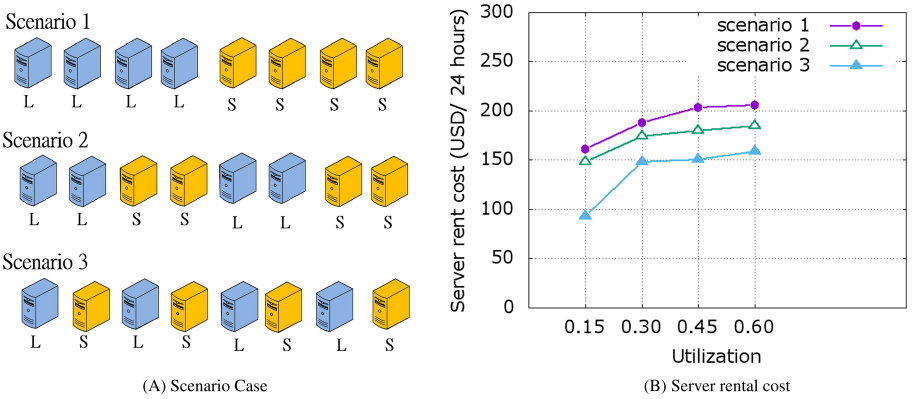


FIGURE 6 The effects of delay constraint t_{max}



(A) Scenario Case

(B) Server rental cost

FIGURE 7 Effects of server location of physical machines (PMs)

The results presented in Figure 7 show that for utilization rates in the range of 0.15 to 0.60, the server arrangement in scenario 3 (ie, decentralized high-capacity PMs) results in the lowest server rental cost since fewer services are placed on the high-capacity PMs. In particular, the statistical results show that 59% of the services are placed on high-capacity PMs in scenario 1, whereas 54% of the services are placed on high-capacity PMs in scenario 2, and just 44% are placed on high-capacity PMs in scenario 3. Overall, the results show that the server rental cost reduces by around 25% when the high-capacity PMs are moved from a centralized arrangement (scenario 1) to a decentralized arrangement (scenario 3).

4.6 | Service repetition in service chains: The rental cost increases by 40% as the service repetition rate is increased from 0 to 2

A final series of experiments was performed to examine the effect of the service repetition rate on the server rental cost under different end-to-end delay constraints. The service repetition rate describes the similarity between different service chains, with a high repetition rate indicating that many of the sub chains are the same. For example, service chains {S1-S2-S3-S4, S2-S3-S4-S5, S1-S2-S4-S5} have a high service repetition rate. In performing the simulations, the repetition of the various services was modeled using the Zipf distribution function (Adamic and Huberman[30]). In particular, a service repetition equal to 0 indicates that all of the services have the same emergence frequency, while a higher value of the service repetition indicates that services with a higher emergence frequency are centralized in fewer services. For example, for the service chains (S1-S2-S3), (S1-S2-S4), and (S1-S2-S5), the services of the three chains are centralized in S1 and S2.

The results presented in Figure 8A,B show that as the service repetition rate increases from 0 to 2, the corresponding increase in the VM sharing ratio causes the server rental cost to decrease by 10% and 40% under the strictest and loosest end-to-end delay constraint conditions, respectively. Furthermore, as t_{max} increases from 20 to 400 ms, the server rental cost decreases by 12% and 27% under service repetition rates of 0.5 and 1.5, respectively. Notably, the VM sharing ratio is close to 0 under service repetition rates of 0.5, 1, and 1.5 when $t_{max} = 20$ ms. This finding implies that only limited VM sharing can be achieved under strict end-to-end delays. However, under more relaxed end-to-end delays, a high service repetition rate results in a lower server rental cost.

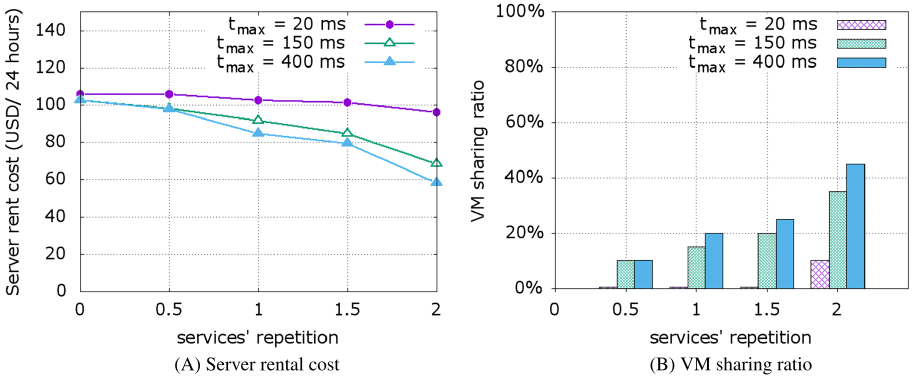


FIGURE 8 Effects of heterogeneity on service repetition rate

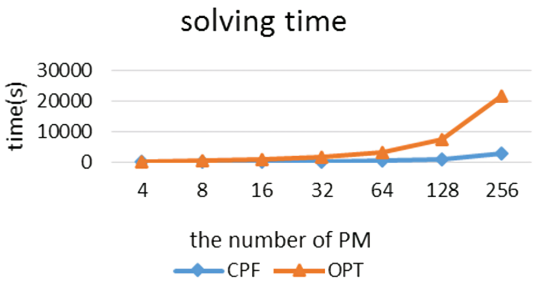


FIGURE 9 Solution time comparison of chain placement first (CPF) and nonlinear programming (NPL) optimization algorithms

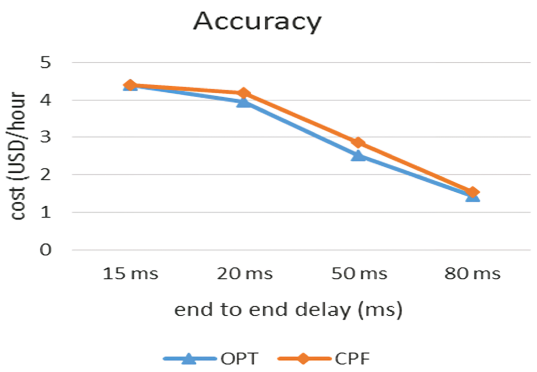


FIGURE 10 Accuracy comparison of chain placement first (CPF) and optimization algorithms

4.7 | Heuristic method

For small-scale problems such as those considered above, the service chain placement problem can be solved relatively easily using the NLP BONMIN solver. However, for typical large-scale problems involving up to 256 PMs, for example, the solution time may be as long as 6 hours. This is clearly impractical for real-world service chain placement scenarios. Consequently, this section of the paper proposes a heuristic algorithm designated as common sub-CPF to reduce the solution time.

The pseudocode of the proposed algorithm is presented in Algorithm 1. As shown, the algorithm takes the service chains and PM capacities as inputs and provides the PMs deployed to each service chain as the output. The main aim of the algorithm is to place common sub chains on the high-capacity PMs such that they can be shared by many service chains. Furthermore, high-capacity VMs are created on high-capacity PMs such that they too can be reused in order to save the server rental cost.

The CPF algorithm comprises five steps. The first step calculates the weights of the individual sub chains. Note that all possible sub chains are considered. For example, service chain S1-S2-S3-S4 comprises six possible sub chains, ie, S1S2, S2S3, S3S4, S1S2S3, S2S3S4, and S1S2S3S4. (Note that sub chain S1S2 is regarded as being the same as sub chain S2S1.) Having identified all possible sub chains, the weights of the sub chains are calculated based on their quantities. For example, the weight of S2S3 is calculated as 2 since it exists in two service chains.

Algorithm 1 Common sub chain placement first (CPF)

Require: $S_{i,j}$: the service chains; P_k: the PM capacity; $|S|$: the number of service chains;
Ensure: $SP_{i,j,k}$: the service chain's placement;

```
# compute sub chains' weight
for i = 1 to |S| do
    for j = 1 to 3 do
        if subchain(S_{i,j}, S_{i,j+1}) or subchain(S_{i,j+1}, S_{i,j}) exist in subchain-list then
            subchain-list[S_{i,j}, S_{i,j+1}]++;
        else
            #subchain-list add subchain
            subchain-list[S_{i,j}, S_{i,j+1}]=0;
        end if
    end for
    for j = 1 to 2 do
        if subchain(S_{i,j}, S_{i,j+1}, S_{i,j+2}) or subchain(S_{i,j+2}, S_{i,j+1}, S_{i,j}) exist in subchain-list then
            subchain-list[S_{i,j}, S_{i,j+1}, S_{i,j+2}]++;
        else
            #subchain-list add subchain
            subchain-list[S_{i,j}, S_{i,j+1}, S_{i,j+2}]=0;
        end if
    end for
    if subchain(S_{i,1}, S_{i,2}, S_{i,3}, S_{i,4}) or subchain(S_{i,4}, S_{i,3}, S_{i,2}, S_{i,1}) exist in subchain-list then
        subchain-list[S_{i,1}, S_{i,2}, S_{i,3}, S_{i,4}]++;
    else
        #subchain-list add subchain
        subchain-list[S_{i,1}, S_{i,2}, S_{i,3}, S_{i,4}]=0;
    end if
end for
# compute subchain's score
for i = 1 to |S| do
    Score_i=subchain-list[S_{i,1}, S_{i,2}, S_{i,3}, S_{i,4}]
    +subchain-list[S_{i,1}, S_{i,2}, S_{i,3}]+subchain-list[S_{i,2}, S_{i,3}, S_{i,4}]
    +subchain-list[S_{i,1}, S_{i,2}]+subchain-list[S_{i,2}, S_{i,3}]+subchain-list[S_{i,3}, S_{i,4}];
end for
# sort score in descending order
MapS = sort(Score_i)
# sort PM by capacity in descending order
MapP = sort(P_k)
#service chain placement
for i in MapS do
    for j = 1 to 4 do
        for k in MapP do
            if S_{i,j} can place on P_k then
                SP_{i,j,k}=k;
            else
                break;
            end if
        end for
    end for
end for
Return SP_{i,j,k}
```

The second step computes the service chains' scores based on their sub chain weights. The third and fourth steps sort the service scores and PM capacities, respectively, in descending order. Finally, the fifth step selects the service chain with the highest order and places it on the PM with the highest capacity. The next service chain is then selected and added to the same PM. If the service chain cannot be allocated to the PM since it lacks sufficient capacity, it is added instead to the PM with the next highest capacity. The allocation process continues in this way until all of the service chains have been allocated to a PM.

As shown in Figure 9, when solving a large-scale problem (eg, 256 PMs), the NLP optimization algorithm proposed in Section 3 requires approximately 6 hours to complete. By contrast, the CPF algorithm requires just 48 minutes. In other words, the solution time is reduced by around 86%. Notably, the results presented in Figure 10 show that the performance improvement is obtained at the expense of an accuracy reduction of no more than 8%.

5 | CONCLUSION

In assigning service chains to VMs, the server rental cost may increase dramatically if the service chains are not properly placed. Accordingly, this paper has proposed a NLP model to solve the service chain placement problem in such a way as to minimize the total server rental cost while simultaneously satisfying the end-to-end delay constraint. Traditionally, the server delay is assigned a constant value when solving the service chain placement problem. However, in practical environments, the delay is not constant but varies depending on the number of VMs used and the traffic load. Consequently, in the present study, the server delay has been modeled using M/M/1 queuing theory. The placement problem has then been solved using a branch-and-bound optimization algorithm.

The simulation results support five main observations. First, the cost ratio of PMs with high, hybrid, and low capacity should be set as 4:2:1. In other words, in solving the placement problem, low-capacity PMs are preferred subject to the proviso that the end-to-end delay constraint is still satisfied. Second, the rental cost increases dramatically as the utilization rate increases beyond 55%. Third, the rental cost reduces by 20%, 30%, and 32% as the end-to-end delay constraint is relaxed from 40 to 60, 80, and 100 ms, respectively. Fourth, the rental cost decreases by 25% when the high-capacity PMs are decentralized. Finally, the rental cost reduces by 40% as the service repetition rate increases from 0 to 2.

Future studies will develop further heuristic algorithms for the service chain placement problem based on the analysis results obtained in the present study. In addition, a more detailed model will be developed to take account of the effect of the traffic load on the network delay. In general, using the management plane alone is insufficient to react to rapid traffic changes. The present study has shown that the proposed common sub-CPF algorithm provides the ability to reduce the solution time by up to 86% compared with the NPL optimization method with an accuracy reduction of no more than 8%. Consequently, future studies will aim to combine the routing algorithm used in the system and the CPF algorithm so as to gain the advantages of both the management plane and the control plane, respectively, in solving real-world service chain placement problems.

ORCID


Chien Ting Wang https://orcid.org/0000-0002-3372-2152

REFERENCES

1. Savi M, Tornatore M, Verticale G. Impact of processing costs on service chain placement in network functions virtualization. In: Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on; 2015:191-197.
2. Zhang SQ, Tizghadam A, Park B, Bannazadeh H, Leon-Garcia A. Joint NFV placement and routing for multicast service on SDN. In: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium; 2016:333-341.
3. Zeng M, Fang W, Zhu Z. Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks. *J Lightwave Tech*. 2016;34(14):3330-3341.
4. Bouet M, Leguay J, Conan V. Cost-based placement of virtualized deep packet inspection functions in SDN. In: MILCOM 2013-2013 IEEE Military Communications Conference; 2013:992-997.
5. Moens H, Turck FD. VNF-P: a model for efficient placement of virtualized network functions; 2014:418-423.
6. Addis B, Belabed D, Bouet M, Secci S. Virtual network functions placement and routing optimization. In: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft); 2015:171-177.
7. Luizelli MC, Bays LR, Buriol LS, Barcellos MP, Gaspary LP. Piecing together the NFV provisioning puzzle: efficient placement and chaining of virtual network functions. In: Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on; 2015:98-106.

8. Lin T, Zhou Z, Tornatore M, Mukherjee B. Optimal network function virtualization realizing end-to-end requests. In: Global Communications Conference (GLOBECOM), 2015 IEEE; 2015:1-6.
9. Lin T, Zhou Z, Tornatore M, Mukherjee B. Demand-aware network function placement. *Journal of Lightwave Technology*. 2016;34(11):2590-2600.
10. Gupta A, Habib MF, Chowdhury P, Tornatore M, Mukherjee B. On service chaining using virtual network functions in network-enabled cloud systems. In: Advanced Networks and Telecommuncations Systems (ANTS), 2015 IEEE International Conference on; 2015:1-3.
11. Cohen R, Lewin-Eytan L, Naor JS, Raz D. Near optimal placement of virtual network functions. In: Computer Communications (INFOCOM), 2015 IEEE Conference on; 2015:1346-1354.
12. Bouet M, Leguay J, Combe T, Conan V. Cost-based placement of vDPI functions in NFV infrastructures. *Int J Netw Manag*. 2015;25(6):490-506.
13. Ghaznavi M, Khan A, Shahriar N, Alsubhi K, Ahmed R, Boutaba R. Elastic virtual network function placement. In: Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on; 2015:255-260.
14. Khebbache S, Hadji M, Zeghlache D. Scalable and cost-efficient algorithms for VNF chaining and placement problem. In: Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on; 2017:92-99.
15. Soualah O, Mechtri M, Ghribi C, Zeghlache D. An efficient algorithm for virtual network function placement and chaining. In: Consumer Communications & Networking Conference (CCNC), 2017 14th IEEE Annual; 2017:647-652.
16. Jang I, Choo S, Kim M, Pack S, Shin M-K. Optimal network resource utilization in service function chaining. In: NetSoft Conference and Workshops (NetSoft), 2016 IEEE; 2016:11-14.
17. Riggio R, Rasheed T, Narayanan R. Virtual network functions orchestration in enterprise WLANs. In: Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on; 2015:1220-1225.
18. Khebbache S, Hadji M, Zeghlache D. A multi-objective non-dominated sorting genetic algorithm for VNF chains placement. In: Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual; 2018:1-4.
19. Baek H, Jang I, Ko H, Pack S. Order dependency-aware service function placement in service function chaining. In: Information and Communication Technology Convergence (ICTC), 2017 International Conference on; 2017:193-195.
20. Kuo T-W, Liou B-H, Lin KC-J, Tsai M-J. Deploying chains of virtual network functions: on the relation between link and server usage. *IEEE/ACM T NETWORK*. 2018;26(4):1562-1576.
21. GLPK. GLPK. https://www.gnu.org/software/glpk/, https://www.gnu.org/software/glpk/
22. Bouras C, Ntarzanos P, Papazois A. Cost modeling for SDN/NFV based mobile 5G networks. In: Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2016 8th International Congress on; 2016:56-61.
23. TPC. TPCC. http://www.tpc.org/tpcc/, http://www.tpc.org/tpcc/
24. Cisco. Quality of service for voice over IP. https://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.pdf, https://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.pdf
25. Kumar BK, Arivudainambi D. Transient solution of an M/M/1 queue with catastrophes. *Comput Math appl*. 2000;40(10-11):1233-1240.
26. Bonami P. Bonmin. https://projects.coin-or.org/Bomin, https://projects.coin-or.org/Bomin
27. optimization INC, AMPL. AMPL. http://ampl.com/products/solvers/all-solvers-for-ampl/, http://ampl.com/products/solvers/all-solvers-for-ampl/
28. Yoon MS, Kamal AE. Power minimization in fat-tree SDN datacenter operation. In: Global Communications Conference (GLOBECOM), 2015 IEEE; 2015:1-7.
29. Verbitski A, Gupta A, Saha D, et al. Amazon Aurora: design considerations for high throughput cloud-native relational databases. In: Proceedings of the 2017 ACM International Conference on Management of Data; 2017:1041-1052.
30. Adamic LA, Huberman BA. Zipf's law and the Internet. *Glottometrics*. 2002;3(1):143-150.

How to cite this article: Wang C T, Lin Y-D, Wang C-C, Lai Y-C. Cost Minimization in Placing Service Chains for Virtualized Network Functions. *Int J Commun Syst*. 2020;33:e4222. https://doi.org/10.1002/dac.4222