Journal of Ambient Intelligence and Humanized Computing (2020) 11:3963–3974
https://doi.org/10.1007/s12652-019-01630-6

ORIGINAL RESEARCH

Efficient algorithms to minimize the end-to-end latency of edge network function virtualization

Karanbir Singh Ghai[1] · **Salimur Choudhury**[1] · **Abdulsalam Yassine**[2]

Received: 24 July 2019 / Accepted: 30 October 2019 / Published online: 22 January 2020

Abstract

In future wireless networks, network function virtualization will lay the foundation for establishing a new dynamic resource management framework to efficiently utilize network resources. The main problem discussed in this paper is the minimization of total latency for an edge network and how to solve it efficiently. A model of users, virtual network functions and hosting devices has been taken, and is used to find the minimum latency using integer linear programming. The problem is NP-hard and takes exponential time to return the optimal solution. We apply the stable matching based algorithm to solve the problem in polynomial time and then utilize local search to improve its efficiency further. From extensive performance evaluation, it is found that our proposed algorithm is very close to the optimal scheme in terms of latency and better in terms of time complexity.

Keywords Network function virtualization · Hosting device · Latency · Stable matching · Local search

1 Introduction

In today's time, we require an efficient and advanced network model that can support the growing load of the users (Hu et al. 2011). Different models used for network computing are centralized network computing and distributed network computing. In the initial phases of networking the centralized network model was used as there were not many devices that could support the whole networks but trend changed and we now use more of distributed networking model. The main reason behind this is, in centralized networks the complete load of the network system falls on one central machine which increases the risk of network failure but in distributed networks, the network relies on various nodes or network devices which makes it more efficient and thus more reliable (Baran 1964). Edge Networks as the name suggests is a distributed computing paradigm in which computation is wholly or mostly performed on distributed device nodes known as smart devices or edge devices as opposed to primarily taking place in a centralized cloud environment. Here "edge" is defined as any computing and network resources along the path between data sources and cloud data centers (Shi et al. 2016). For example, a smart phone is the edge between smart body sensors and a cloud, a gateway in a smart home is the edge between smart home things and a cloud. Edge computing is related to the concepts of wireless sensor networks, intelligent and context-aware networks and smart objects in the context of human-computer interaction (Satyanarayanan 2017). Edge computing is more concerned with computation performed at the edge of networks and systems whereas the Internet of Things label implies a stronger focus on data collection and communication over networks. Figure 1 illustrates an Edge network in which the core supplies the data to the various edges, which further connects to the users.

In this paper, our primary focus will be on the distributed edge networks. One of the significant factors which lead to increase in the load on the network, is the exponential increase in the number of mobile users, the machine to machine (M2M) communication methods and the Internet-of-things (IoT) as they increase data overhead thus

✉ Karanbir Singh Ghai
kghai1@lakeheadu.ca

Salimur Choudhury
salimur.choudhury@lakeheadu.ca

Abdulsalam Yassine
ayassine@lakeheadu.ca

[1] Department of Computer Science, Lakehead University, Thunder Bay, ON, Canada

[2] Department of Software Engineering, Lakehead University, Thunder Bay, ON, Canada

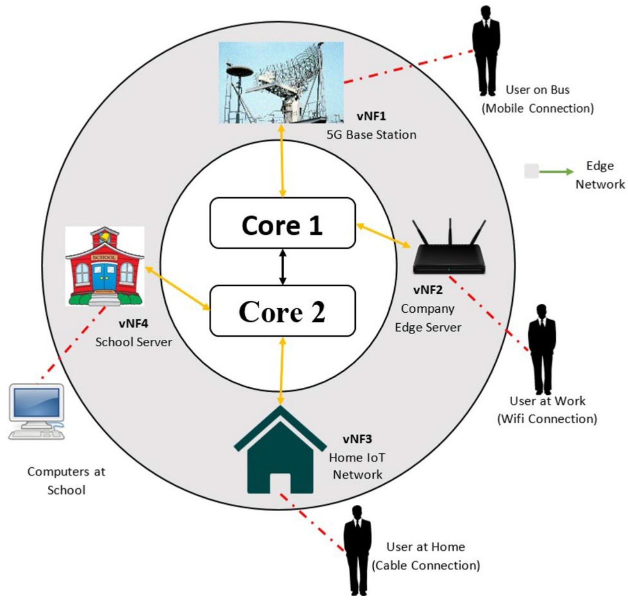


Fig. 1 Example of edge network

increasing the data rate, capacity demands an increase in the need for coverage. So, due to the growth mentioned above, the large volume of raw data is continuously generated by devices, consequently making cloud computing inadequate to efficiently and securely handle the data (Shi and Dustdar 2016). Thus the current trend is shifting from centralized computing to distributed computing as the load in distributed computing is distributed and doesn't fall on the shoulders of one central device. According to the report of Cisco (Cisco 2017), mobile data traffic will grow at a compound annual growth rate (CAGR) of 47exabytes per month by 2021. Meanwhile, M2M connections are calculated to grow from 780 million in 2016 to 3.3 billion by 2021. The modern networks require more hardware base to work efficiently, but this makes the network more complicated and costly.

To overcome these challenges a newly designed technique network functions virtualization (NFV), is being used in which network functions of traditional networks have been converted into software appliances called virtual network functions (vNFs) (Chiosi et al. 2013). This technique was first introduced by a group of researchers from various communication companies in 2012. The objective for introduction of this technique was to counter multiple factors that come into play to launch a new network service mainly including increasing costs of energy, capital investments, the rarity of skills necessary to design, integrate and operate increasingly complex hardware-based appliances. This concept uses the technology of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication or network services. One of the essential and principal uses of this technology is that network functions don't need any sophisticated or high-end hardware; instead, it can be run on general- purpose hardware that is available easily.

It is an emerging network architecture to increase flexibility and agility within the operator's networks by placing virtualized services on demand in the data center. Figure 1 also demonstrates the edges of the network where vNFs can be placed to make it more efficient and reliable. One of the main challenges for the NFV environment is how to efficiently allocate vNFs to virtual machines (VMs) (Luizelli et al. 2015) and get the best out of the whole network with the minimum workload on the network. NFV techniques highly complement the software defined networking (SDN) technology (Chiosi et al. 2013), but these both are not dependent on each other. NFV technique can be implemented without an SDN, but if both methods are used together, more efficient results are obtained (Chiosi et al. 2013). Various security infrastructures that have been developed and matured in cloud computing space are being adopted in NFV technology, few examples of these are in identity services, role-based access control (RBAC) (Chu 2018).

Next-generation networks are expected to support low-latency, context-aware and user-specific services in a highly flexible and efficient manner (Cziva and Pezaros 2017b). Proposed applications include high-definition, low-latency video streaming, remote surgery, as well as requests for the tactile Internet, virtual reality that demand's network-side data processing (such as image recognition, transformation). Mobile networks are the latest and most used type of networks nowadays. The latest in this domain is the 5G network which is on the verge of being deployed for mobile devices. With the arrival of 5G, the mobile networks have increased the demand of the novel, more evolved and scalable network technologies (Bouras et al. 2017) to support this network. 5G will succeed 4G (LTE) which is currently in use, and it will target high data rate, reduced latency, energy saving, cost reduction, higher system capacity, and massive device connectivity (Andrews et al. 2014). It is said to be capable of supporting 20 Gbit/s data rate, 1 ms of latency and mainly it can support up to 10^6 devices per km^2. Using both SDN and NFV techniques, the 5G network can be made more efficient and easier to manage (Chu 2018).

In today's time most of the devices used are smart device, they can be bulbs, appliances or even medical equipment (Park and Yen 2018). As all of these type of devices are connected to the internet. They also contribute towards the usage of NFV technology. Hence, they are an integral part of this research. These devices are called IoT devices. IoT, defined as the Internet of things (IoT) is an emerging technology which was first proposed to study RFID by Ashton, Professor of the MIT Auto-ID Center in 1999 (Wang et al. 2016). The IoT can be used in various ways, and the data transmitted during network communication can take many forms, ranging from personal data to sensing information gathered from the environment (Kim and Lee 2018; Munir et al. 2019; Campioni et al. 2019). IoT is defined as the network of devices such as vehicles, smart devices, and home appliances that contain electronics, software and connectivity, which allows these things to connect, interact and exchange data. The definition of the IoT has evolved due to the convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems (Satyanarayanan 2017). A massive increase in the number of devices in IoT is being predicted (expected to reach 50 billion by 2020). Figure 2 shows us an example of a simple IoT network that can be found in an average household.

Latency is defined as the delay or the interruption in a connection; it can depend on various factors distance, weather, the material used and hardware configurations of hosting devices and users (Pandi et al. 2018). If the latency goes beyond a certain threshold, then the whole network could fail.

In this paper we deal with the vNF placement problem. Cziva and Pezaros (2017b) proposed the vNF placement problem, they propose a mathematical (integer linear programming) model to solve the problem. The mathematical model mentioned is NP-hard in nature which means that it will take exponential time to solve the problem in worst case scenario and on analyzing it is found that the mathematical model is having drawbacks and will lead to failures if the problem persists. No heuristic has been proposed by Cziva and Pezaros (2017b) to solve the problem in polynomial time. This paper modifies the mathematical model to remove the anomaly and to make it more efficient. The modified model also takes exponential time to solve the problem in the worst case scenario. Then we propose an heuristic based on the stable matching (SM) algorithm to solve the modified problem in a polynomial time. The solutions given by the model are then compared to the solutions given by the proposed heuristic (stable matching algorithm) for the allocation of vNFs to hosting devices.We also find that there is a scope of improving the final solution. We design a local search technique to improve the solution.

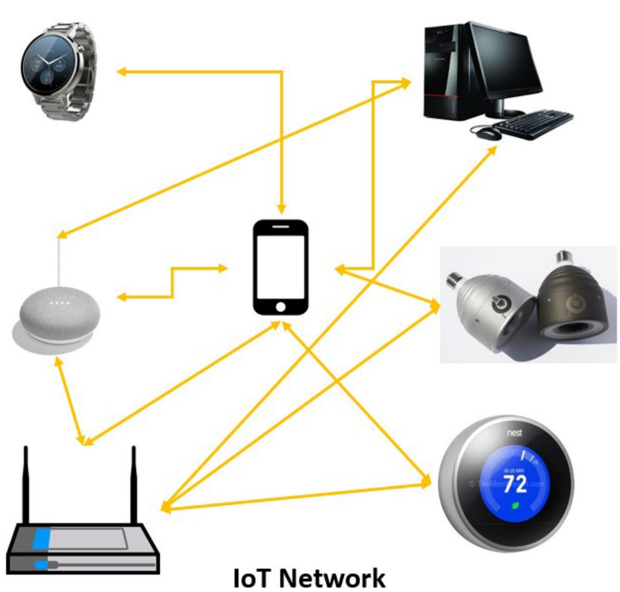


Fig. 2 Example of IoT network

The rest of the paper is designed in the following manner, Sect. 2 canvasses prior works related to our topic of interest. Section 3 contains the problem definition and the system model. Section 4 discusses the modification of the mathematical model, its requirement and implementation. Section 5 presents a detailed discussion of our proposed algorithm and its extension with working procedures. The performance evaluations and simulations are shown in Sect. 6. Finally, Sect. 7 concludes the paper with some future research directions.

2 Related work

Network function virtualization (NFV) is an emerging network architecture and is an efficient technology in the networking area. Current research works are going-on to design or implement new techniques to make this emerging technology more efficient. During the literature review, we can find several studies trying the different scope of vNF technology including scaling, allocation, task scheduling, placement, edge-based models, cloud-based models, and latency optimization. Moving intelligence from traditional servers at the center of the network to the network edge is gaining significant attention from both the research and the industrial communities, as discussed in citep26, (Cziva and Pezaros 2017a). Orchestrating and managing vNFs in different NFV infrastructures has been a popular research topic, and it is often related to traditional Virtual Machine (VM) placement problem, as mentioned in (Moens and Turck 2014). In this research paper, authors have presented vNF-P, a generic model for efficient placement of virtualized network functions. Simultaneous placement of vNFs is used to form a service function chain (SFC), a chain of vNFs, and then uses admission control (AC) to reach the maximum performance state. The main issue of this research is to solve the problem of AC and SFC embedding (Tahmasbi Nejad et al. 2018). They have used relaxation, reformulation, and successive convex approximation methods to solve the problem. In modern data-centers, user network traffic uses a set of vNFs as a service chain to process traffic demands (Tashtarian et al. 2017). Sometimes traffic fluctuations in large-scale data-centers (LDCs) could result in overload and under-load phenomena in service chains. In this research paper, a distributed approach based on alternating direction method multipliers (ADMM) is used to balance the traffic as well as horizontally scale up and down vNFs in LDCs with minimum deployment and forwarding costs. One of the main challenges for the NFV environment is how to efficiently allocate virtual network functions (vNF) to virtual machines (VMs) (Cho et al. 2017). In this research, a more comprehensive model based on real measurements to capture network latency among vNFs with more granularity to optimize placement of vNFs in CDCs.

Stable Match algorithm has been used frequently to solve many problems in various research areas in computer science and other fields of study too. McVitie and Wilson (McVitie and Wilson 1971) pointed out that the algorithm by Gale and Shapley (Gale and Shapley 1962) in which men propose to women, generates a male-optimal solution in which every man gets the best partner he can in any stable matching and every woman gets the worst partner she can in any stable matching. They suggested an equal measure of optimality under which the sum of the ranks of partners for all men and women was to be minimized. An efficient algorithm was provided by Irving et al. (Irving et al. 1987) to find a stable matching satisfying the optimality criterion of McVitie and Wilson.

Stable Matching Algorithm has also been used in scheduling of both computing and storage resources in data centres (Chu et al. 2017). In the research mentioned above paper, authors first define a preference list for each side and stability of their matching, then they propose a useful Stable Matching Based Algorithm (SMB) scheme. This algorithm has given them a stable matching for computing and storage resources as well as applications (virtual machines) for all the performed experimental cases. Authors in (Sugimoto et al. 2009) proposes a fast iteration algorithm for Kansei matching, which is further used as an algorithm to solve the stable matching problem. This is also easy and more transparent than the conventional (extended) Gale–Shapley (GS) algorithm in the sense of programming and debugging. The research shows that the proposed algorithm executes more than six times faster than the Gale–Shapley, while it requires the same memory storage as the GS algorithm. They also present a version of the iteration algorithm that is more efficient and describes the result of comparative experimentation in execution time.

Local search is a technique in which the algorithm tries to find the solution to a problem locally that satisfies the conditions required by the given problem. When the algorithm is done with a state or node, it moves to the next node or state by applying the local changes until it finds an optimal solution. Local search algorithm has been used to design the reliable networks optimally (Dengiz et al. 1997; Islam et al. 2015; Hassan et al. 2017a, b). The research mentioned above paper proposes a genetic algorithm (GA) with specialized encoding, initialization, and local search operators to optimize the design of communication network topologies. The problem taken by the authors is NP-hard and often generates infeasible networks using random initialization and standard genetic operators as it is highly constrained. They found that special purpose GA is more efficient than an enumerative based method on NP-hard problems of realistic size.

3 Problem description and system model

3.1 Problem description

In this paper, we are dealing with a problem in which we need to minimize the latency generated by the newly made connections in a topology. This is done by assigning the vNFs to that hosting devices which gives minimum latency for the topology. This problem can be categorized under the assignment problem in which we need to find that appropriate assignment of all vNFs to hosting devices that minimizes the total expected latencies from all users to its vNFs. The allocation of the vNFs to hosting devices depends on various factors like the requirement of vNFs, the capacity of host devices and mainly on the latency between the hosting device and the vNFs. The allocation is complete when all of the vNFs are allocated, or when the capacity of all the hosting devices gets exhausted.

3.2 System model

In this paper, we are using the same model as used by (Cziva and Pezaros 2017b). Here we consider that vNFs are to be connected to host devices, and further users are connected to vNFs to use the network. The goal of this paper is to allocate vNFs to different hosting devices to minimize the latency caused.

3.2.1 Parameters

We consider a system with vNFs and hosting devices, where $\mathbb{N} = \{n_1, n_2, n_3, \dots, n_i\}$ is the set of all vNFs in the network. For each n_i we can define memory, CPU and IO *requirements* ($\mathbf{R_i}$), as well as *Maxlatency* ($\mathbf{ML_i}$) that denotes the maximum latency which vNF n_i can tolerate. Similarly $\mathbb{H} = \{h_1, h_2, h_3, \dots, h_j\}$ is the set of vNF hosting devices (that represent either a cloud or an edge server). Similar to vNF's requirements, each h_j has capacity ($\mathbf{C_j}$) on its properties, for example; CPU, memory, IO etc. $\mathbf{l_{ij}}$ gives the latency between the user of the n_i vNF in case n_i is located at h_j (Table 1).

$\mathbf{x_{ij}}$ is a binary decision variable that denotes allocation of vNFs to hosts; where

Table 1 Parameters

Variable	Description
$\mathbb{N}$	Set of all vNFs
$\mathbb{H}$	Set of all hosting devices
C_j	Maximum capacity of a hosting device j
R_i	Requirement of vNF i
ML_i	Maximum latency a vNF i can tolerate.
l_{ij}	Latency b/w the user of the n_i vNF in case that vNF is located at h_j

$$x_{ij} = \begin{cases} 1 \text{ if } n_i \text{ is allocated to } h_j \\ 0 \text{ otherwise} \end{cases}$$

3.2.2 Mathematical (ILP) model

The objective of our model is to minimize the Total-Latency value which is given by Eq. (1).

$$\textit{Minimize} \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} x_{ij} l_{ij} \tag{1}$$

Subject To-

$$\sum\nolimits_{n_i \in \mathbb{N}} x_{ij} * R_i \leq C_j, \forall h_j \in \mathbb{H} \tag{2}$$

$$\sum\nolimits_{h_j \in \mathbb{H}} x_{ij} l_{ij} \leq \mathrm{ML}_i, \forall n_i \in \mathbb{N} \tag{3}$$

$$\sum\nolimits_{h_j \in \mathbb{H}} x_{ij} = 1, \forall n_i \in \mathbb{N} \tag{4}$$

- *First constraint* (2) ensures that vNFs are placed to hosting devices with sufficient capacity. This constraint also defines that vNFs can't be allocated to the hosting device if its capacity gets filled.
- *Second constraint* (3), ensures that latency-sensitive vNFs are placed subject to not violating the max latency requirement from their users. The latency of the selected pair should always be less than the Maxlatency for the vNF.
- *Third constraint* (4), constraint ensures that all vNFs are allocated to hosting devices exactly once. A single vNF can't be connected to two hosting devices, but one hosting device can connect to two vNFs.

The above-mentioned ILP problem is a minimizing problem in which our objective is to minimize the total latency obtained by the allocation of the vNFs to the hosting devices. It can be noted that the above ILP is an NP-hard problem

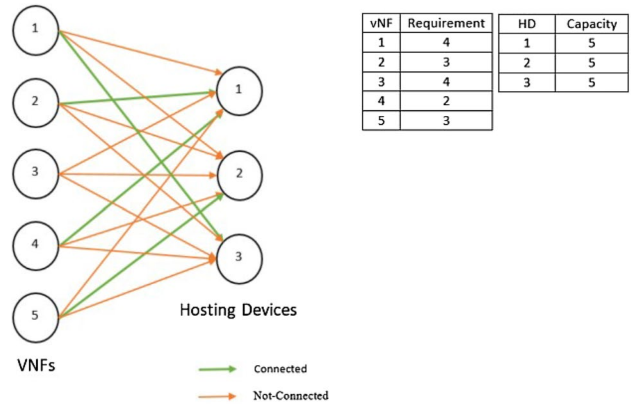


Fig. 3 Fail case scenario for 5 vNFs

(Cziva and Pezaros 2017b) and can be solved by optimally by an ILP solver, for example, IBM CPLEX or Gurobi. For our simulations, we used IBM CPLEX to solve it optimally.

4 Mathematical model modification

In this problem statement, the allocation constraint Eq. 4 states that every vNF should be connected to at least one host device. If this constraint fails in any circumstance, the whole model fails. Some of the scenarios that lead to model failure are:

Let us consider a scenario with five vNFs that want to connect to three different hosting devices as represented in Fig. 3. It can be seen that all the devices want to connect with the hosting devices, but due to the insufficient capacity of the hosting devices, all of the vNFs won't be connected and the connections in green will only be connected. Though it doesn't have much problem but due to the allocation constraint Eq. 4 the model will fail and will give an infeasible solution.

4.1 Proposed modification

Thus, to fix the above problem and make the problem more general, we used another mixed integer linear programming (MILP) problem model to find the maximum number of vNFs that can be connected optimally to the hosting devices. The allocation constraint (4) is thus changed to:

$$\sum\nolimits_{n_i \in \mathbb{N}} \sum\nolimits_{h_b \in \mathbb{H}} x_{ij} = M \tag{5}$$

$$\sum\nolimits_{h_b \in \mathbb{H}} x_{ij} <= 1, \forall n_i \in \mathbb{N} \tag{6}$$

where 'M' is the total number of vNFs that can be connected optimally and another constraint (6) is added, which ensures that one vNF connects to a maximum of one Hosting Device. M is calculated using another ILP formulation which is as follows:

$$Maximize\ M = \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} x_{ij} \tag{7}$$

Subject To−

$$\sum_{i \in B_j} x_{ij} * R_j <= C_j, \forall j \in \mathbb{H} \tag{8}$$

$$\sum_{j \in A_i} x_{ij} <= 1, \forall i \in \mathbb{N} \tag{9}$$

where,

$$x_{ij} = \begin{cases} 1 & \text{if } n_i \text{ is allocated to } h_j \\ 0 & \text{otherwise} \end{cases}$$

For each vNF $i \in \mathbb{N}$, $A_i \subseteq \mathbb{H}$ is a set of hosting devices that can hold vNF i (satisfying constraint 3).

$$A_i = \begin{cases} 1 & \text{if } n_i \text{ can be accommodated by } h_j \\ 0 & \text{otherwise} \end{cases}$$

Similarly $B_j \subseteq \mathbb{N}$ be the set of vNFs that can be assigned to hosting devices j. vNF i is connectable to hosting device j, if it satisfies constraint 3.

$$B_j = \begin{cases} 1 & \text{if } h_j \text{ can accommodate } n_i \\ 0 & \text{otherwise} \end{cases}$$

C_j is capacity of hosting devices and R_i are the requirements for vNFs. Further the constraint 8 is similar to constraint 2. Constraint 9 states that one vNF can't be connected to more than one hosting device.

The problem model used to find the "M" is also an NP-Hard problem and it can be defined as Multiple Knapsack Problem with Assignment Restrictions (MKAR) (Dawande et al. 2000). The model can be solved optimally by an ILP solver, such as IBM CPLEX or by a $\mathbb{A}\frac{1}{2}$ Approximation Algorithm as proposed in (Dawande et al. 2000).

The complete new model with modification becomes:

$$Minimize \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} x_{ij} l_{ij} \tag{10}$$

Subject To-

$$\sum_{n_i \in \mathbb{N}} x_{ij} * R_i \ \leq\ C_j, \forall h_j \in \mathbb{H} \tag{11}$$

$$\sum_{h_j \in \mathbb{H}} x_{ij} l_{ij} \ \leq\ \mathrm{ML}_i, \forall n_i \in \mathbb{N} \tag{12}$$

$$\sum_{n_i \in \mathbb{N}} \sum_{h_b \in \mathbb{H}} x_{ij} = M \tag{13}$$

$$\sum_{h_b \in \mathbb{H}} x_{ij} <= 1, \forall n_i \in \mathbb{N} \tag{14}$$

5 Proposed algorithm

5.1 Stable matching algorithm

Stable matching based solution has been proposed in various domain in the case of assignment problems (Hossen et al. 2018; Ghai et al. 2019). Stable Matching start's by creating two priority matrices for the two groups that we want to match. These matrices are created on the basis of the latencies in which the lesser latencies are given the more priority for both the groups that are vNFs and hosting devices. Then the matching is done according to the priority matrix, where the vNF wants to connect to the hosting device that is first on its priority list. The same case exists for hosting devices as they want to connect to the vNF that is first on their priority list. The algorithm runs for all the vNFs and matches them to hosting devices until a stable matching is achieved.

5.1.1 Algorithm

In the above Algorithm 1, we start with all the vNFs and hosting devices as free, and take Total latency and Count as 0. The algorithm will run until maximum number of devices that can be connected (M) are connected, as shown in line 4, where M is calculated in the modified model using 7. Then a vNF, n proposes to the hosting device h that has the highest priority for vNF if the conditions as specified in line 6 are met then the vNF and hosting device is engaged. The count, capacity, and total latency are then updated. The other aspect is that if the hosting device is connected to another vNF n', as shown in line 11. Then from line 12, if the hosting device prefers the selected vNF n over the currently engaged n', the hosting device will be engaged with n and n' will become free. In this case the count remains same but capacity and total latency are updated. If the hosting device does not prefer the selected vNF, n over the currently engaged n', then the pair remains engaged. The proposed algorithm has a complexity of $\mathcal{O}(n * m)$ in the worst case where n is the number of vNFs, and m is the number of host devices (while $n >> m$). So, generalizing we can say that the complexity of the algorithm is $\mathcal{O}(n^2)$.

Algorithm 1 Stable Matching

```
procedure MATCHING vNFs TO HOSTING DEVICES UNTIL A STABILITY IS ACHIEVED
    All vNFs and Hosting devices are free.
    Initialize both totalLatency and count as 0.
    while (There Exist a Free vNF (n) who has not proposed to hosting device (h) and count is less than M) do
        h→ is the first preferred Hosting Device
        if (h is free and Constraints 11 and 12 are satisfied) then
            (n, h) become engaged
            Update count = count + 1
            Update capacity = capacity of h - requirement of n
            Update totalLatency = totalLatency + latency between the n and h
        else(Some pair (n′, h) already exists)
            if (h prefers n to n′) then
                (n, h) become engaged
                n′ becomes free
                Count remains same
                Update capacity and totalLatency
            else
                (n′, h) remain engaged
            end if
        end if
    end while
    Return totalLatency and matched pairs
end procedure
```

5.2 Local search

Though the stable match based algorithm discussed in the previous section works efficiently, it can be improved using a local search algorithm. In this procedure, we start with an initial feasible solution that is provided by the Stable Match algorithm and then tries to improve the solution iteratively. The local search begins by picking two connected pair of hosting devices and vNFs and checks whether the latency can be improved by changing the connections locally. The algorithm stops when there is no further improvement is possible.

5.2.1 Algorithm

In this local search Algorithm (2) we start with a feasible solution provided by the Stable Match algorithm. All the connected pairs (i, j) are checked from the provided solution by comparing them (7) with all the other connected pairs (i', j'). We even compare the selected pair with all the unpaired vNFs (13) and hosting devices (19). IF the comparison leads to improvement (reduction) in the total latency and they satisfy the constraints 11 and 12 , then the connection is either swapped or moved. The whole procedure is performed while there is still a chance of improvement. At the end the solution is provided using the updated allocations. The improvement is calculated as follows:

- For Case I (Swapping), we calculate and compare the sum of the latencies for the connected pairs and for the swapped connections. If the sum of latency for the swapped pair is lesser, it can be said that there is improvement in solution. This way we don't have to calculate the whole total latency each time.
- For Case II (Moving for free vNF), we just check that if the latency of the new connection is lesser than the selected connection then there is an improvement in solution.
- For Case II (Moving for free hosting device), we just check that if the latency of the new connection is lesser than the selected connection then there is an improvement in solution.

Complexity of above algorithm is $\mathcal{O}(n * m * W)$ in the worst case where n is the number of vNFs, m is the number of hosting devices (where n >> m) and "W" is the latency given by the stable matching solution (taken as initial feasible solution). So, generalizing we can say that the complexity of the algorithm is $\mathcal{O}(n^2 * W)$.

Algorithm 2 Local Search (Swapping\Moving)

procedure SWAP OR MOVE THE CURRENT MATCHED PAIRS TO FIND MORE EFFICIENT SOLUTION.
  Using initial feasible solution from Algorithm 1.
  Initialize improvement = true.
  while (improvement) **do**
    improvement = false.
    Checking for all connected pairs (i, j) and (i′,j′), where “i” vNF is connected to “j” hosting device and similarly “i'” is connected to “j'”. ▷ Case I
    if ($l_{ij} + l_{i'j'} > l_{ij'} + l_{i'j}$ and Constraints 11 and 12 are satisfied) **then** ▷ Swapping
      Assign vNF i to hosting device j' and vNF i' to hosting device j.
      Update Capacity for hosting devices.
      improvement = true.
    end if
    Check for other unconnected vNFs (i''). ▷ Case II
    if ($l_{ij} > l_{i''j}$ and Constraints 11 and 12 are satisfied) **then** ▷ Moving
      Assign vNF i'' to hosting device j and vNF i will get free.
      Update Capacity for hosting devices.
      improvement = true.
    end if
    Check for other unconnected hosting devices (j''). ▷ Case III
    if ($l_{ij} > l_{ij''}$ and Constraints 11 and 12 are satisfied) **then** ▷ Moving
      Assign vNF i to hosting device j''.
      Update Capacity for hosting devices.
      improvement = true.
    end if
  end while
  Print New minimum totalLatency using current allocation.
end procedure

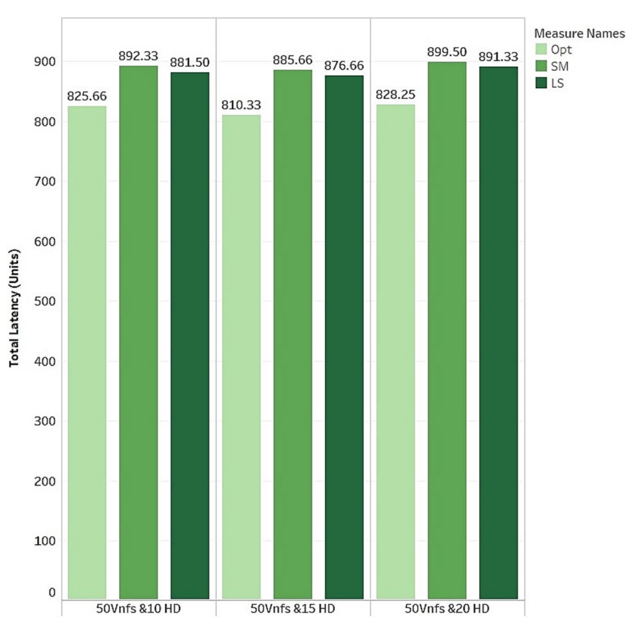


Fig. 4 Latency result comparisons between optimal (ILP), stable matching and stable match with local search for 50 vNFs

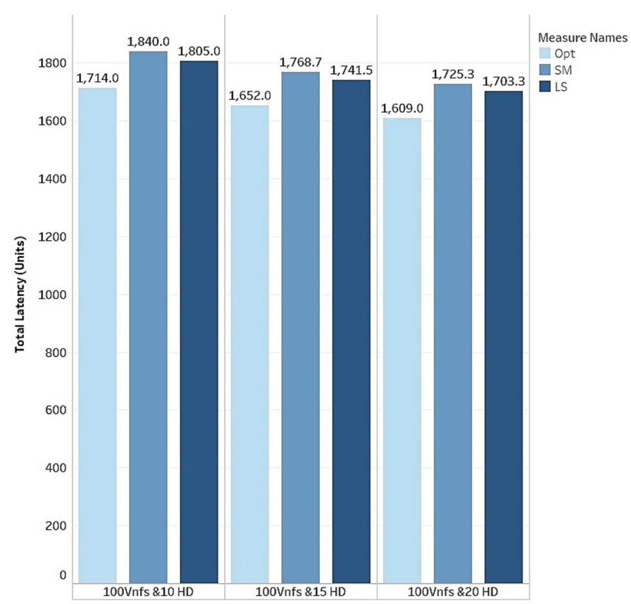


Fig. 5 Latency result comparisons between optimal (ILP), stable matching and stable match with local search for 100 vNFs

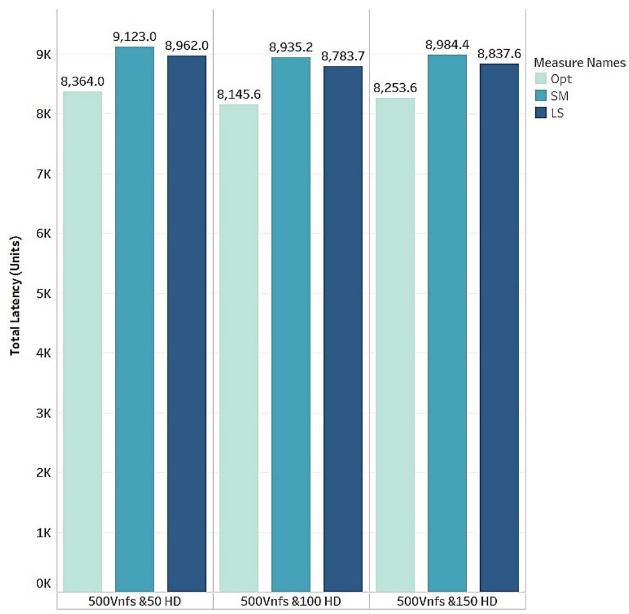


Fig. 6 Latency result comparisons between optimal (ILP), stable matching and stable match with local search for 500 vNFs

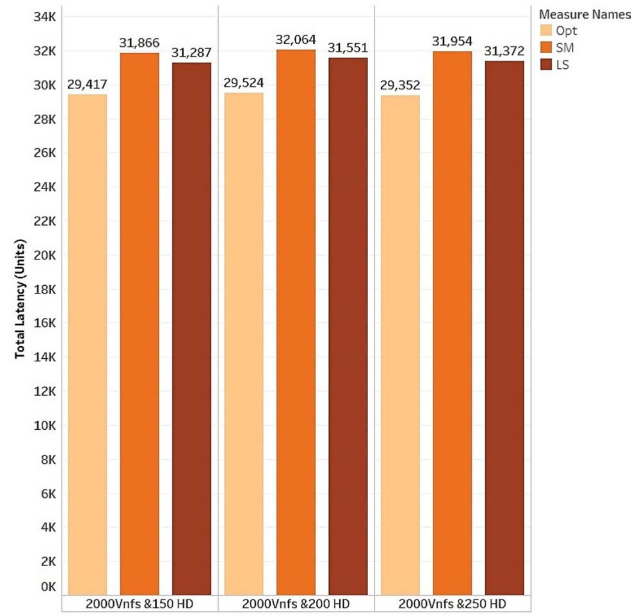


Fig. 8 Latency result comparisons between optimal (ILP), stable matching and stable match with local search for 2000 vNFs

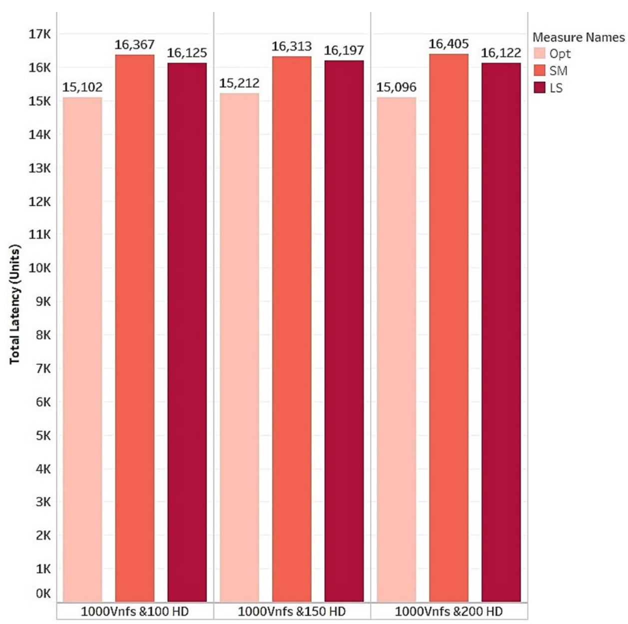


Fig. 7 Latency result comparisons between optimal (ILP), stable matching and stable match with local search for 1000 vNFs

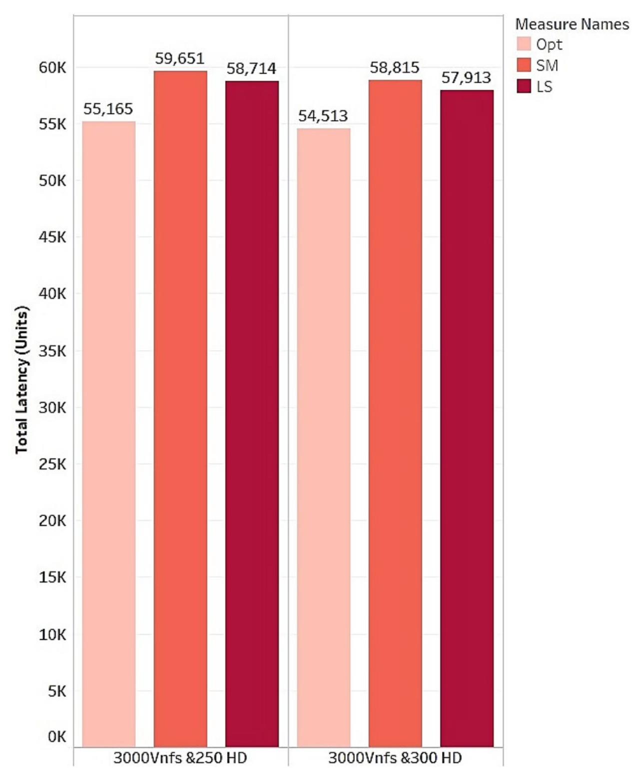
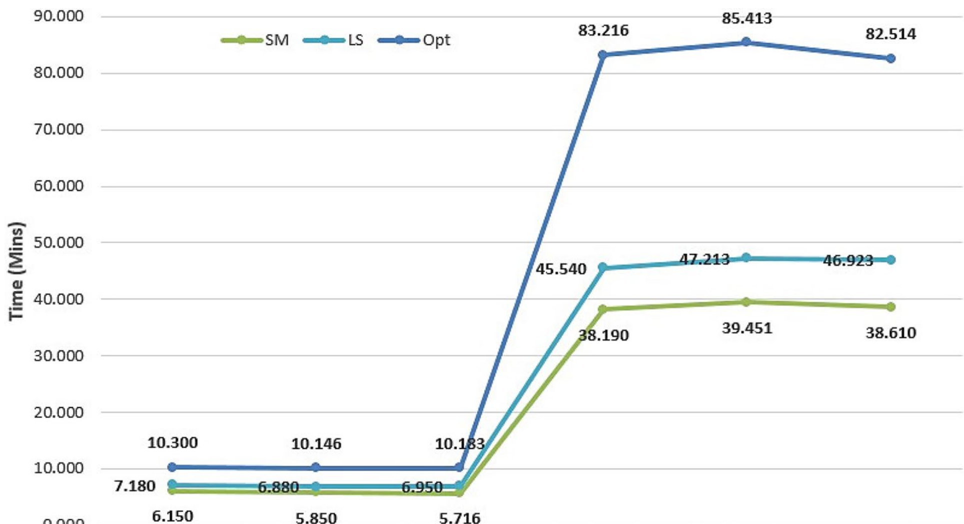


Fig. 9 Latency result comparisons between optimal (ILP), stable matching and stable match with local search for 3000 vNFs

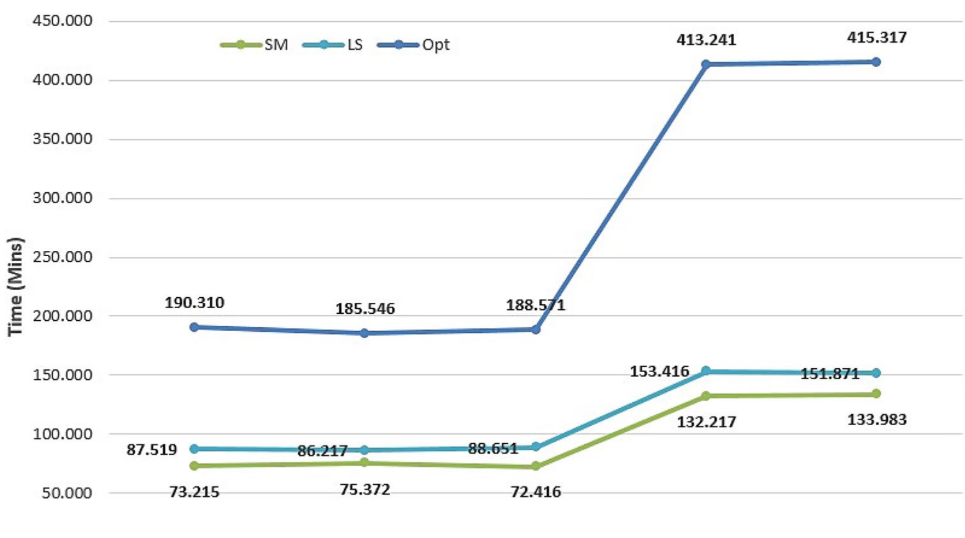
6 Results

The ILP model used is implemented in IBM CPLEX, and our proposed algorithm has been implemented in C++. In this process, we don't use a network simulator as we are not solving any network layer research problems. For input, the data taken includes the number of vNFs, hosting devices, users. The other values taken as input are capacity of hosting devices, requirements and a maximum latency of vNFs and latency between the vNF and hosting device as these



	500Vnfs &50 HD	500Vnfs &100 HD	500Vnfs &150 HD	1000Vnfs &100 HD	1000Vnfs &150 HD	1000Vnfs &200 HD
SM	6.150	5.850	5.716	38.190	39.451	38.610
LS	7.180	6.880	6.950	45.540	47.213	46.923
Opt	10.300	10.146	10.183	83.216	85.413	82.514

Fig. 10 Time (min) comparison between optimal (ILP), stable matching and stable match with local search for 500 and 1000 vNFs



	2000Vnfs &150 HD	2000Vnfs &200 HD	2000Vnfs &250 HD	3000Vnfs &250 HD	3000Vnfs &300 HD
SM	73.215	75.372	72.416	132.217	133.983
LS	87.519	86.217	88.651	153.416	151.871
Opt	190.310	185.546	188.571	413.241	415.317

Fig. 11 Time (min) comparison between optimal (ILP), stable matching and stable match with local search for 2000 and 3000 vNFs

all are the properties of hosting devices and vNFs which are used in the simulations. For latency between the vNFs and the hosting devices, we take random values between 15–40 as it depends on various factors such as distance, the material used, and the performance of hosting devices and vNFs. Similarly, the random values of the capacity of the hosting devices are taken between 10–75. Requirements and a maximum latency of vNFs have also been taken randomly between 1–15 and 20–50 respectively. For all the simulation results presented in this section, we start with 20 vNFs and 5 hosting devices. The different instances that are used in this scenario are 20, 30, 50 and 100 for vNFs. 5, 10, 15, 20 are a different number of host devices which are then used to form different cases and use them to compare results for

Table 2 Working time comparison (s) and latency comparisons

vNFs	Hosting devices	Opt	LS	% Decrease (time)	% Increase (latency)
50 vNFs	10 HD	0.090	0.076	20.00	6.76
	15 HD	0.117	0.108	11.97	8.19
	20 HD	0.137	0.129	10.95	7.62
100 vNFs	10 HD	0.131	0.125	8.40	5.31
	15 HD	0.120	0.110	13.33	5.42
	20 HD	0.124	0.116	12.10	5.86
500 vNFs	50 HD	10.300	7.180	30.29	7.15
	100 HD	10.146	6.880	32.19	7.83
	150 HD	10.183	6.950	31.75	7.08
1000 vNFs	100 HD	83.216	45.540	45.27	6.77
	150 HD	85.413	47.213	44.72	6.48
	200 HD	82.514	46.923	43.13	6.79
2000 vNFs	150 HD	190.310	87.519	54.01	6.36
	200 HD	185.546	86.217	53.53	6.87
	250 Hd	188.571	88.651	52.99	6.88
3000 vNFs	250 HD	413.241	153.416	62.87	6.43
	300 HD	415.317	151.871	63.43	6.24

Opt (optimal result from ILP), stable match (SM) and local search (LS) algorithms. All of the simulation results presented in this section are an average of 10 different runs for a particular scenario. The figures ahead (Figs. 4, 5, 6, 7, 8, 9, 10 and 11) illustrate us the result comparison between the ILP result given by Opt (mathematical model), SM (stable match) and LS (SM with local search) on basis of TL (total latency) for different cases. Table 2 shows the comparisons between the time taken by both optimal and stable match with the local search for different number of vNFs and varied number of host devices. It shows that the local search takes 20–30% less time compared to the optimal. The table also represents the comparisons in terms of latency. It is found that local search solution costs around 7–8% more latency compared to the optimal solutions. In summary, considering all experimental results, it is clear that the stable match algorithm performs very close to the optimal (costs 9–10% more than the optimal latency). However, when the local search is added, an even better result is achieved (7–8% more than the optimal latency).

7 Conclusion and future work

In this paper, we modify an existing latency minimization problem (to make it more general) for edge NFV. Since the problem is NP-hard, we introduce two heuristics (one is based one stable matching and another one is based on local search) to solve the problem efficiently. Our results suggest that our local search provides results quite close to the optimal in a very reasonable time.

In future we plan to consider a problem which will deal in fair allocation of the vNFs to the hosting devices and minimizing the total latency simultaneously. Another future aspect can be to design an effcient algorithm to do the assignment of vNFs to hosting devices dynamically. This algorithm will automatically start re-assigning the vNFs when there is a change in scenario and change in latency (goes beyond a specified limit). A similar type of problem has been defined in (Cziva et al. 2018); in this problem, the authors give an ILP model first to allocate vNFs to a distributed edge infrastructure, minimizing end-to-end latency. Then they dynamically re-schedule the optimal placement of vNFs based on temporal network-wide latency fluctuations using optimal stopping theory. Since, the problem can take exponential time in the worst case, designing an efficient heuristic to solve this problem is an interesting research topic.

References

Andrews JG, Buzzi S, Choi W, Hanly SV, Lozano A, Soong ACK, Zhang JC (2014) What will 5g be? IEEE J Select Areas Commun 32(6):1065–1082. https://doi.org/10.1109/JSAC.2014.2328098

Baran P (1964) On distributed communications networks. IEEE Trans Commun Syst 12(1):1–9. https://doi.org/10.1109/TCOM.1964.1088883

Bouras C, Kollia A, Papazois A (2017) Sdn nfv in 5g: Advancements and challenges. In: 2017 20th Conference on innovations in clouds, internet and networks (ICIN), pp 107–111. https://doi.org/10.1109/ICIN.2017.7899398

Campioni F, Choudhury S, Al-Turjman F (2019) Scheduling rfid networks in the iot and smart health era. J Ambient Intell Hum Comput:1–15

Chiosi M, Clarke D, Willis Cablelabs P, Donley C, Johnson Centurylink L, Bugenhagen M, Feger J, Khan W, China C, Cui H, Chen China Deng C, T, Baohua L, Zhenqiang S, Wright S (2013) Network functions virtualisation (nfv) network operator perspectives on industry progress https://doi.org/10.13140/RG.2.1.4110.2883

Cho D, Taheri J, Zomaya AY, Wang L (2017) Virtual network function placement: Towards minimizing network latency and lead time. In: 2017 IEEE International conference on cloud computing technology and science (CloudCom), pp 90–97. https://doi.org/10.1109/CloudCom.2017.12

Chu W (2018) NFV and NFV-based security services, vol 15. Wiley, Oxford, pp 347–372. https://doi.org/10.1002/9781119293071.ch15

Chu Q, Cui L, Zhang Y (2017) Joint computing and storage resource allocation based on stable matching in data centers. In: 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids), pp 207–212. https://doi.org/10.1109/BigDataSecurity.2017.36

Cisco (2017) Cisco visual networking index: global mobile data traffic forecast update, 2017–2022. Cisco White Paper

Cziva R, Pezaros DP (2017a) Container network functions: bringing nfv to the network edge. IEEE Commun Mag 55(6):24–31. https://doi.org/10.1109/MCOM.2017.1601039

Cziva R, Pezaros DP (2017b) On the latency benefits of edge nfv. In: 2017 ACM/IEEE symposium on architectures for networking and communications systems (ANCS), pp 105–106. https://doi.org/10.1109/ANCS.2017.23

Cziva R, Anagnostopoulos C, Pezaros D (2018) Dynamic, latency-optimal vnf placement at the network edge, pp 693–701. https://doi.org/10.1109/INFOCOM.2018.8486021

Dawande M, Kalagnanam J, Keskinocak P, Salman F, Ravi R (2000) Approximation algorithms for the multiple knapsack problem with assignment restrictions. J Comb Optim 4(2):171–186. https://doi.org/10.1023/A:1009894503716

Dengiz B, Altiparmak F, Smith AE (1997) Local search genetic algorithm for optimal design of reliable networks. IEEE Trans Evol Comput 1(3):179–188. https://doi.org/10.1109/4235.661548

Gale D, Shapley LS (1962) College admissions and the stability of marriage. Am Math Mon 69(1):9–15. http://www.jstor.org/stable/2312726

Ghai KS, Choudhury S, Yassine A (2019) A stable matching based algorithm to minimize the end-to-end latency of edge nfv. Proced Comput Sci 151:377–384

Hassan MY, Hussain F, Hossen MS, Choudhury S, Alam MM (2017a) A near optimal interference minimization resource allocation algorithm for d2d communication. In: 2017 IEEE international conference on communications (ICC), IEEE, pp 1–6

Hassan Y, Hussain F, Hossen S, Choudhury S, Alam MM (2017b) Interference minimization in d2d communication underlaying cellular networks. IEEE Access 5:22471–22484

Hossen MS, Hassan MY, Hussain F, Choudhury S, Alam MM (2018) Relax online resource allocation algorithms for d2d communication. Int J Commun Syst 31(10):e3555

Hu F, Qiu M, Li J, Grant T, Tylor D, McCaleb S, Butler L, Hamner R (2011) A review on cloud computing: design challenges in architecture and security. CIT 19:25–55

Irving RW, Leather P, Gusfield D (1987) An efficient algorithm for the“optimal” stable marriage. J ACM 34(3):532–543. https://doi.org/10.1145/28869.28871

Islam MT, Taha AEM, Akl S, Choudhury S (2015) A local search algorithm for resource allocation for underlaying device-to-device communications. In: 2015 IEEE global communications conference (GLOBECOM), IEEE, pp 1–6

Kim S, Lee I (2018) Iot device security based on proxy re-encryption. J Ambient Intell Hum Comput 9(4):1267–1273. https://doi.org/10.1007/s12652-017-0602-5

Luizelli MC, Bays LR, Buriol LS, Barcellos MP, Gaspary LP (2015) Piecing together the nfv provisioning puzzle: efficient placement and chaining of virtual network functions. In: 2015 IFIP/IEEE international symposium on integrated network management (IM), pp 98–106. https://doi.org/10.1109/INM.2015.7140281

McVitie DG, Wilson LB (1971) Three procedures for the stable marriage problem. Commun ACM 14(7):491–492. https://doi.org/10.1145/362619.362632

Moens H, Turck FD (2014) Vnf-p: A model for efficient placement of virtualized network functions. In: 10th International conference on network and service management (CNSM) and workshop, pp 418–423. https://doi.org/10.1109/CNSM.2014.7014205

Munir A, Laskar MTR, Hossen MS, Choudhury S (2019) A localized fault tolerant load balancing algorithm for rfid systems. J Ambient Intell Hum Comput 10(11):4305–4317

Pandi S, Wunderlich S, Fitzek FHP (2018) Reliable low latency wireless mesh networks —from myth to reality. In: 2018 15th IEEE annual consumer communications networking conference (CCNC), pp 1–2. https://doi.org/10.1109/CCNC.2018.8319326

Park JH, Yen NY (2018) Advanced algorithms and applications based on iot for the smart devices. J Ambient Intell Hum Comput 9(4):1085–1087. https://doi.org/10.1007/s12652-018-0715-5

Satyanarayanan M (2017) The emergence of edge computing. Computer 50(1):30–39. https://doi.org/10.1109/MC.2017.9

Shi W, Dustdar S (2016) The promise of edge computing. Computer 49(5):78–81. https://doi.org/10.1109/MC.2016.145

Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: vision and challenges. IEEE IoT J 3(5):637–646. https://doi.org/10.1109/JIOT.2016.2579198

Sugimoto S, Hattori T, Izumi T, Kawano H (2009) Fast kansei matching method as an algorithm for the solution of extended stable marriage problem. In: 2009 International conference on biometrics and kansei engineering, pp 209–214. https://doi.org/10.1109/ICBAKE.2009.55

Tahmasbi Nejad MA, Parsaeefard S, Maddah-Ali MA, Mahmoodi T, Khalaj BH (2018) vspace: Vnf simultaneous placement, admission control and embedding. IEEE J Select Areas Commun 36(3):542–557. https://doi.org/10.1109/JSAC.2018.2815318

Tashtarian F, Varasteh A, Montazerolghaem A, Kellerer W (2017) Distributed vnf scaling in large-scale datacenters: an admm-based approach. In: 2017 IEEE 17th international conference on communication technology (ICCT), pp 471–480. https://doi.org/10.1109/ICCT.2017.8359682

Wang S, Hou Y, Gao F, Ji X (2016) A novel iot access architecture for vehicle monitoring system. In: 2016 IEEE 3rd world forum on internet of things (WF-IoT), pp 639–642. https://doi.org/10.1109/WF-IoT.2016.7845396