Placements of Virtual Network Functions for Effective Network Functions Virtualization

by

Karanbir Singh Ghai
Lakehead University

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTERS

in the Department of Computer Science

Lakehead University

Placements of Virtual Network Functions for Effective Network Functions Virtualization

by

Karanbir Singh Ghai
Lakehead University

Supervisory Committee

Dr. Salimur Choudhury, Supervisor
(Department of Computer Science, Lakehead University, Canada)

Dr. Zubair Fadlullah, Departmental Member
(Department of Computer Science, Lakehead University, Canada)

Dr. Waleed Ejaz, External Member
(Department of Applied Science and Engineering, Thompson Rivers University, Kamloops, BC, Canada)

ABSTRACT

In the future wireless networks, network function virtualization will lay the foundation for establishing a new resource management framework to efficiently utilize network resources. The first part of this thesis deals in the minimization of the total latency for a network and how to solve it efficiently. A model of users, Virtual Network Functions (vNFs) and hosting devices have been considered and was used to find the minimum latency using Integer Linear Programming (ILP). The problem is NP-hard and takes exponential time to solve in the worst case. A Stable Matching based heuristic has been proposed to solve the problem in polynomial time and then the local search is utilized to improve the efficiency of the result.

The second part of this thesis proposes the problem of fair allocation of the vNFs to hosting devices. A mathematical programming based model (ILP) has been designed to solve the problem which takes exponential time to solve in the worst case, due to its NP-hard nature. Thus an heuristic approach has been provided to solve the problem in polynomial time.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

First of all **Dr. Salimur Choudhury**, for providing me with the opportunity to work under his supervision and for supporting me through all the tough times. It was not an easy journey, but I have been very fortunate to have a supervisor who cared this much about my work. His guidance, constructive suggestions, and encouragementare the reason I was able to learn and grow as a researcher. It was an absolute privilege to work with him on this research.

Dr. Zubair Fadlullah, for his patience and support in overcoming numerous obstacles in my thesis writing.

Dr. Waleed Ejaz, for his constructive suggestions that helped me in polishing my thesis.

Faculty of Graduate Studies & Faculty of Science and Environmental studies for their financial support. This research would not have been possible without it.

DEDICATION

"What we find changes who we become."

- Peter Morville

I would like to dedicate this thesis to,

My parents, who gave me the foundation of something they always enjoyed, education. For inspiring me and always believing in my ability to accomplish anything I set my mind to. For making sure I had access to all the resources I need to succeed no matter the circumstances. My sister, she always took care of my chores when I was busy studying, and my cousins, they all made me laugh whenever I would feel low and all the other family members who were there to guide and support me at each step of my life. They are my driving force, which keeps me going.

My friends who became family, when my family was far away from me, and they wholeheartedly supported me in all my decisions. They were significant support to me in all my ups & downs and helped me focus on my goal that I was trying to achieve. I thank each of you for your efforts and support towards me.

I would also like to dedicate this thesis to all the teachers and mentors who inspired, empowered, taught, and shaped me into the person I am today.

Chapter 1

Introduction

1.1 Overview

This thesis revolves around the technologies and techniques used to enhance working of a network. So, we start with the question: What is a network? A network is defined as the connection between two or more devices that can be communication devices, Internet of Things (IoT) devices and other smart devices. The connection can be made over various media, either wired such as cable connections, fibre optics or can be wireless such as Wireless Local Area Network (WLAN commonly refereed to as WiFi), satellite connections, sensors and others. In today's time, we require an efficient and advanced network model that can support the growing load of the users [3]. 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 with central devices supporting the whole network but with change in trends, we are now using more of distributed networking

model. The main reason behind this is, in centralized networks the complete load of the network system falls on a single (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 [51].

After discussing the centralized networking computing and distributed network computing, we will now discuss the different types of distributed network technologies as our research revolves mainly around them. The first one is **Cloud Networks**.

Cloud Networks is a network model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [70]. It relies on sharing of resources to achieve coherence and economies of scale, similar to a public utility. The availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing has led to growth in cloud computing. As mentioned in [3], Cloud computing is a conglomerate of several different computing technologies and concepts like grid computing, virtualization, autonomic computing [63], Service Oriented Architecture (SOA) [64], Peer-to-Peer (P2P) computing [65], and ubiquitous computing [66]. The main characteristics of cloud computing are, first, it is a large-scale environment consisting of many physical hosts and virtual machines (VMs). Second, the configuration of a cloud computing environment is complicated. So in order to manage it, we should consider a large number of diverse, networked physical/virtual machines. Third, it is dynamic as the services in cloud computing can run on-demand [54].

The second distributed network technology that is now emerging and plays an essential role in this research is **Edge Network**.

Edge Network 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 [67]. For example, a smart phone is the edge between body things and cloud, a gateway in a smart home is the edge between home things and 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 [55]. Edge computing is more concerned

with computation performed at the edge of networks and systems whereas the Internet of Things (IoT) implies a stronger focus on data collection and communication over networks. Figure 1.1 illustrates an Edge network in which the core supplies the data to the various edges, which further connects the users.

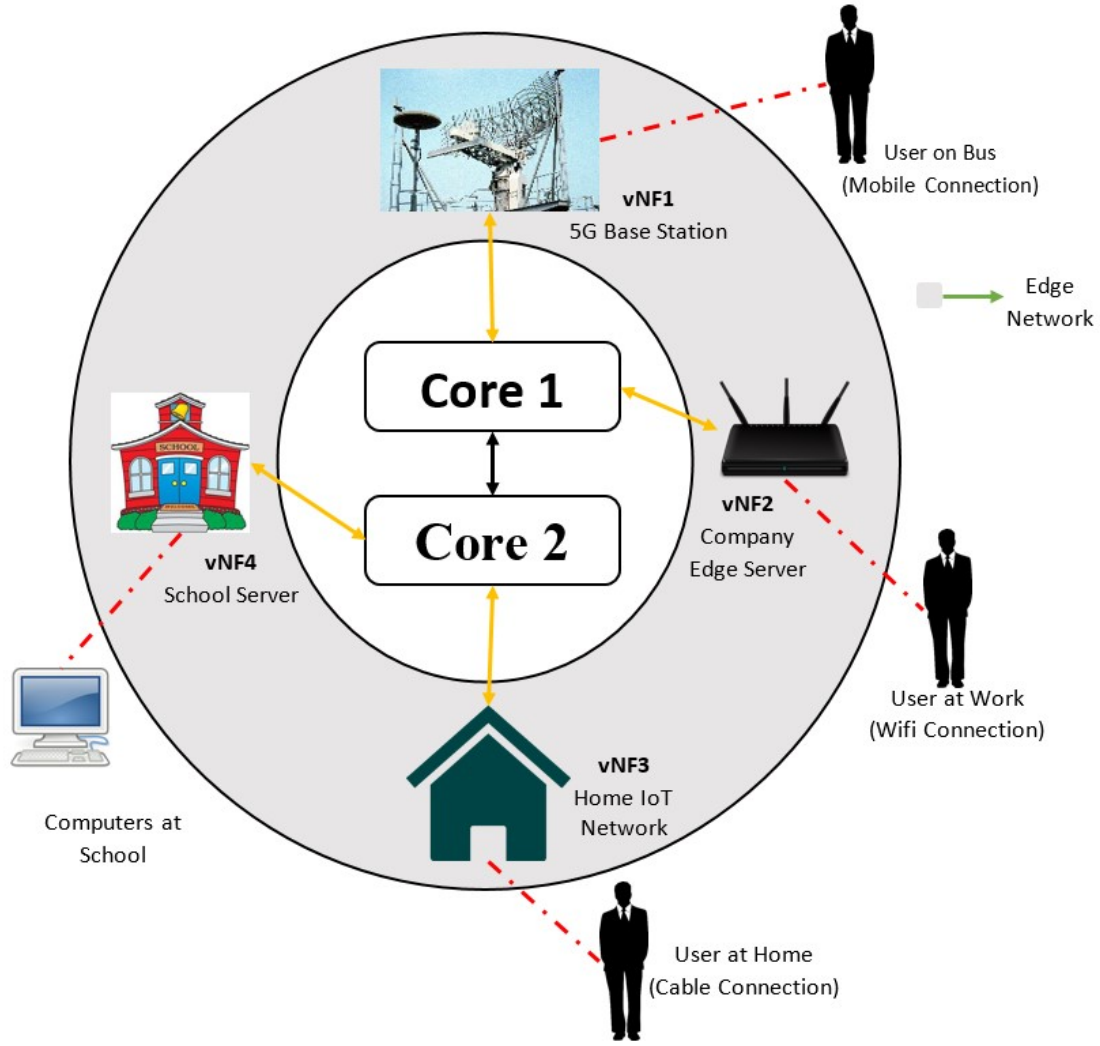


Figure 1.1: **Example of Edge Network**

In this thesis, our primary focus will be on the Distributed Networks. One of the significant factors which leads 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 IoT as they increase data overhead thus increasing the data rate, capacity demands and 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 [68]. Thus, the current trend is shifting from centralized computing to distributed computing as the load in distributed computing is distributed and does not fall on the shoulders of one central device.

According to the report of Cisco [5], mobile data traffic will grow at a compound annual growth rate (CAGR) of 47 percent from 2016 to 2021, reaching 49 exabytes 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 called **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)** [1]. 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 virtualizes 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 do not need any sophisticated or high-end hardware; instead, it can be run on general-purpose hardware that are available easily.

NFV replaces traditional, custom-designed network equipment (black boxes) that continue to dominate the installed base of networks. 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 1.1 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) [52] and get the best out of the whole network with the minimum workload on the network. NFV technique can be implemented without an Software Defined Networking **(SDN)**, but if both methods are used together, more efficient results are obtained [1].

NFV has a dynamic and loosely-coupled nature, which makes it vulnerable to threats; thus, reliability measures should be included in it from the start as a basic need. When a physical network device is introduced into an operational network,

there is established the trust of that device as everything related to it can be trusted like configuring, installing and manufacturing. But for vNFs, the chain of trust relationships needs to be created and maintained in an NFV environment throughout its lifecycle. 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**) [4]. Security, related regulations and even mobile health-care services can be integrated seamlessly by software on top of a shared operator infrastructure based on NFV. The technique mentioned above will be beneficial for the medical institutions like hospital and other care providers who do not wish to spend the time and money to deal with IT systems, communication systems, and security and patient privacy law compliance. Next-generation networks are expected to support low-latency, context-aware and user-specific services in a highly flexible and efficient manner [8]. Proposed applications include high-definition, low-latency video streaming, remote surgery, as well as requests for the tactile Internet, virtual reality that demands 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 [2] 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 [57]. It is said to be capable of supporting 20Gbit/s data rate, 1ms 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 [4].

A distributed and on-demand deployment of network functions, service guaranteed network slicing, flexible orchestration of network functions and optimal workload allocation can be achieved using both the SDN and NFV techniques [6]. In management proposed by L. Ma et al. [6] 5G Service Portal acts as the entrance of network functions to provide different services for users. The Service Management Layer is responsible for orchestrating and configuring of Network Function (NF) modules based on the policy made by the Operation Support System (OSS) and Business Support System (BSS). The 5G Infrastructure Management Layer manages the infrastructure of the 5G core, including flow scheduling, NF deployment, network slicing, etc. Core

SDN Controller and Flow SDN Controller are the two types of controllers that are present in this layer. The first one is in charge of NF management and coordination and the other one in charge of efficient traffic dispatch in the backhaul network. Controlled by SDN controller, the 5G core user plane (UP) and some applications can be deployed on the edge servers as mobile edge core (MEC), while the control plane (CP) is deployed in the cloud data center as mobile cloud core (MCC).

3rd Generation Partnership Project (3GPP) is a standards organization which develops protocols for mobile telephony, and they have seven primary members. An integration of NFV into the **3GPP**- 5G system including distributed NFVs, MEC and NFV integration, and Manamgement and Orchestration (MANO) support in 3GPP management system have been done which helps in virtualizing and managing 3GPP 5G NFVs on top of NFV platform [7]. Here M. Shin et al. [7] show the transformation of networks from 4G to 5G, according to them a lot of new network functions that have been created for a 5G system such as Access and Mobility Management Function (AMF), Session Management Function (SMF), User Plane Function (UPF), Network Repository Function (NRF), Network Exposure Function (NEF), etc. AMF and SMF are well-known Control Plane (CP) functions for mobility and session management of 5G systems. To support low latency services and access to local data networks, UPFs have been created that enable distribution and flexible location.

In today's time most of the devices used are smart device, they can be a light bulb, any appliance or even a medical equipment. 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 thesis. 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[56]. 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 [55]. A massive increase in the number of devices in IoT is being predicted (expected to reach 50 billion by 2020). Figure 1.2 shows us an example of a simple IoT network that can be found in an average household.

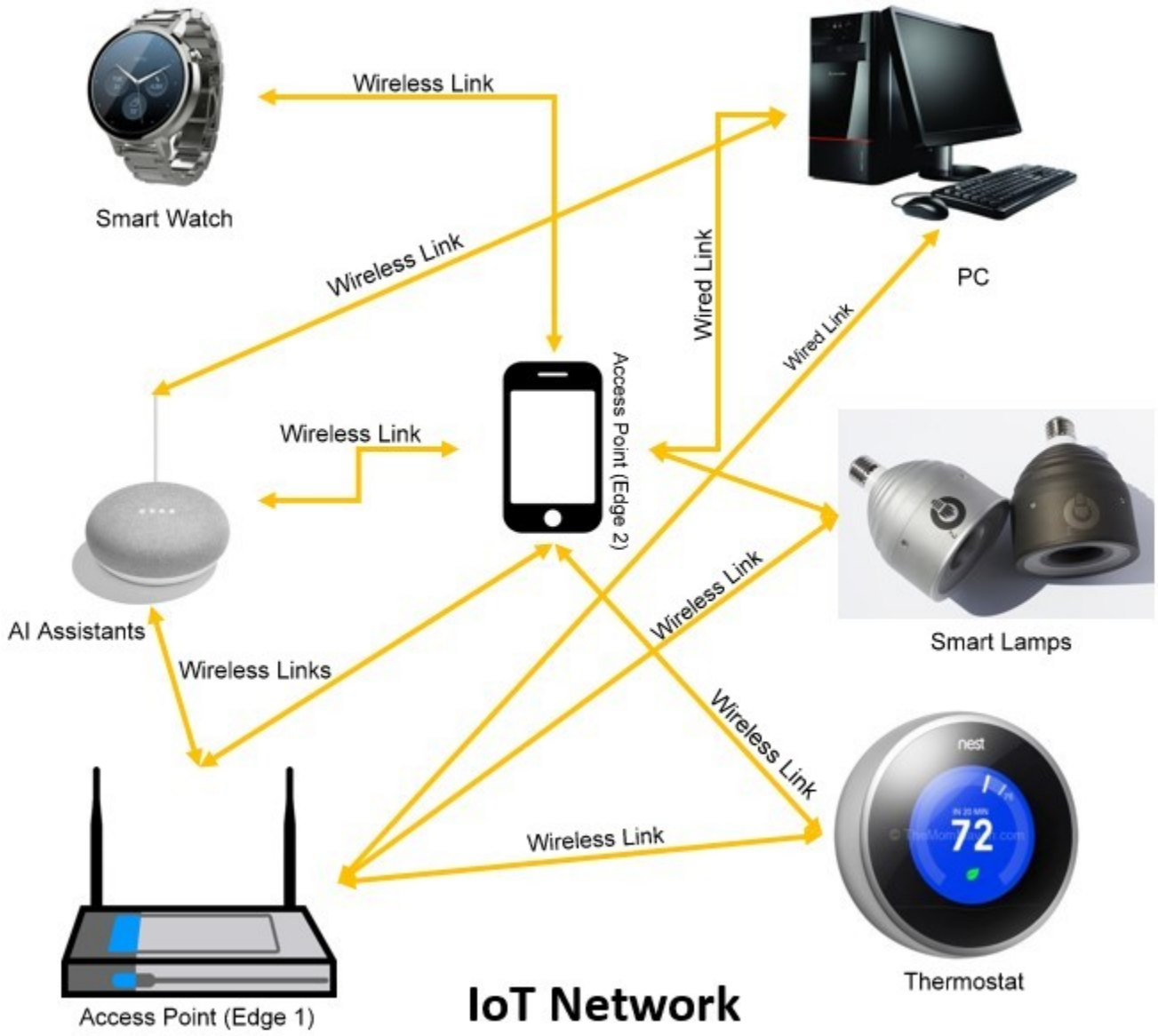


Figure 1.2: **Example of IoT Network**

1.2 Motivation

As discussed earlier the number of IoT devices are said to increase soon which leads to growth in the load on the network. The load can be of different types such as the cost of maintenance, bandwidth requirements and latency which is the most dominant factor in the working of the network efficiently. 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 [69]. If the latency exceeds a certain threshold the whole network could fail.

In this thesis, the research is performed considering a network which includes hosting devices, vNFs and users and anyone of these can be an IoT device. **NFV** technique can convert these devices to a **vNF** and thus these can be used to make the network more efficient. But as the IoT devices are increasing resulting in more choices for creating a NFV network, they also increase the load and latency of the network. So, efficient and advanced techniques are required to maintain the efficiency of the network, which can include modelling the network in such a way that all the users get connected while maintaining the requirements for the network properties. Other types of techniques include the use of the network resources in such a way that we get more efficient solution (better latency) while maintaining the minimal cost and load for the network. This research also handles the management of the resources and optimal modelling of the network to achieve the minimum latency.

Though we can maintain the latency generated by a network with help of modelling a network, sometimes it also leads to another problem, dealing with the unfair distribution or utilization of the resources for some devices. Even if a network is generating less latency does not assure that it will have fair distribution of resources. On deeply analyzing the solutions given by the allocation model, we can observe that in some of the cases, the latency difference in the selected vNFs is vast, which leads to varying difference in the utilization of the resources. It is impartial to the vNF that has to use more of its resources when there is a way in which they both can get connected in an arrangement that both of them employ fewer resources thus no vNF gets partial treatment. Thus, we propose a vNF fair allocation problem to deal with this scenario.

1.3 Problem Description

In this thesis, 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 latency generated by the network. 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.

The problem of allocating the vNFs to hosting devices is to find the minimum latency that is generated by the network has been dealt in the first part of thesis. Now if the allocation of the vNFs is not fair that will lead to another problem as it leads to the inefficient utilization of the resources. Second part of this thesis introduces and gives a solution for a fair vNF allocation problem which is based on the predefined vNF placement problem, which while dealing in the optimal placement of the vNFs to hosting devices will maintain the fairness of the allocation. This problem deals mainly with the fair allocation of the vNFs, maintaining the equal resource allocation for connection. This will lead to the removal of the unfair allocation of vNFs, which leads to inefficient utilization of the resources.

1.4 Contribution

In this thesis we are dealing with two main problems related to the NFV technology, the first one being vNF placement problem and second is fair placement of vNF to hosting device problem.R. Cziva et al. [8] 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 R. Cziva et al. [8] to solve the problem in polynomial time. First part of this thesis 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.

The second part of this thesis deals with the problem for fair allocation of the vNFs to hosting devices. We propose a problem to solve the issue of allocating the vNFs to hosting devices in a fair manner such that no device over-utilizes its resources and

there is an even distribution of latency among the connected pairs. Then we propose a mathematical (ILP) model to solve the above explained problem. The proposed mathematical model is NP-hard in nature and will take exponential time to solve the problem in worst case. So, we provide a heuristic that uses both Local Search and Stable Matching techniques to solve the problem in polynomial time. The solutions given by the ILP model (optimal) have been compared to the solutions given by the heuristic approach (Local Search after SMA) proposed in this research.

The research problem that we have considered in this thesis can be applied in Facility Location research [77].

1.5 Organization of Thesis

This section was all about introduction and rest of the thesis proceeds as follows,
Chapter II gives detailed insight about the background for the various techniques used in this thesis and then discussing the different research works being performed in the field of Function Virtulization and the techniques that are used to perform the researches.
Chapter III, includes the details about the initial considered problem [8] and then further explains its parameters and the constraints that are used in the ILP model given for the problem. Then, the initial modification done to the given [8] model to make it more efficient has been explained. It also explains the proposed algorithms which are based on the **Greedy Approach** and **Stable Matching** techniques to solve the problem defined in the previous chapter in polynomial time.
Chapter IV explains the second proposed algorithm which works using the solutions provided by the Stable Matching Algorithm to make it more efficient. It uses the technique of **Local Search**.
Chapter V explains the newly proposed fair vNF allocation problem which deals in the problem of **fair allocation** and efficient resource utilization of the vNFs. Furthermore, the chapter explains the heuristic given to solve this problem in polynomial time. The algorithm uses the Local search algorithm after the solution provided by Stable Matching Algorithm and minimizes the maximum selected latency and increases the fairness measure.
Chapter VI is the last chapter in this thesis, which concludes the whole work done in this thesis and further explaining the future prospects of the research done.

Chapter 2

Background and Related Work

2.1 Background

2.1.1 Mathematical Modelling

Linear Programming (LP) is the field of mathematical optimization in which the best outcome (such as max profit or low latency) is achieved using the mathematical model whose requirements are represented by linear relationships. It is one of the simplest ways to perform optimization. **Integer Linear Programming** (ILP) is an optimization technique in which the variables are restricted to be integers; these can either be some variables or all of them.

The standard form of LP problem as defined in [59] is:

$$
\begin{aligned}
&\textit{maximize } c\hat{x} \\
&\text{Subject To -} \\
&A\hat{x} \leq b \\
&\hat{x} \geq 0
\end{aligned}
$$

The above LP is a maximizing problem in which $\mathbf{c\hat{x}}$ is objective function with $\mathbf{c}$ and $\mathbf{\hat{x}}$ as vectors and $\mathbf{\hat{x}}$ represents a decision variable. Here $\mathbf{A}$ is a matrix of known coefficients and $\mathbf{b}$ is another vector. Similarly, we can also model a minimizing problem as follows:

$$
\begin{aligned}
&\textit{minimize } c\hat{x} \\
&\text{Subject To -} \\
&A\hat{x} \leq b \\
&\hat{x} \geq 0
\end{aligned}
$$

Simplex Algorithm is the most used algorithm to solve the LP problems [58]; the only drawback of this algorithm is that it takes exponential time to solve the problem in the worst case scenario.

The ILP problems are typically solved using branch-and-bound and cutting plane algorithms [59]. Branch-and-bound is an algorithmic technique to find the optimal solution by keeping the best solution found so far and uses it to prune the search space. It typically enumerates all the possible candidate solutions for a problem implicitly.

Integer Linear Programming is mainly used to obtain a solution for NP-hard problems. These problems are first converted to ILP models, and then the ILP solvers give the solution that satisfies all the required conditions and completes the task required. The solution provided by the ILP solvers is the optimal solution that we can get for that problem. ILP models can be solved using many solvers like Gurobi [71], IBM CPLEX [72], etc. In this research, I have created and solved the ILP model using IBM CPLEX.

2.1.2 Stable Matching

Stable Matching or **Stable Marriage Problem** is a problem in which we have to find a stable matching between two equal sized sets of elements given an ordering of preferences for each item. Matching can be defined as the mapping from the elements of one set to the elements of the other set. Matching can be defined as not stable if:

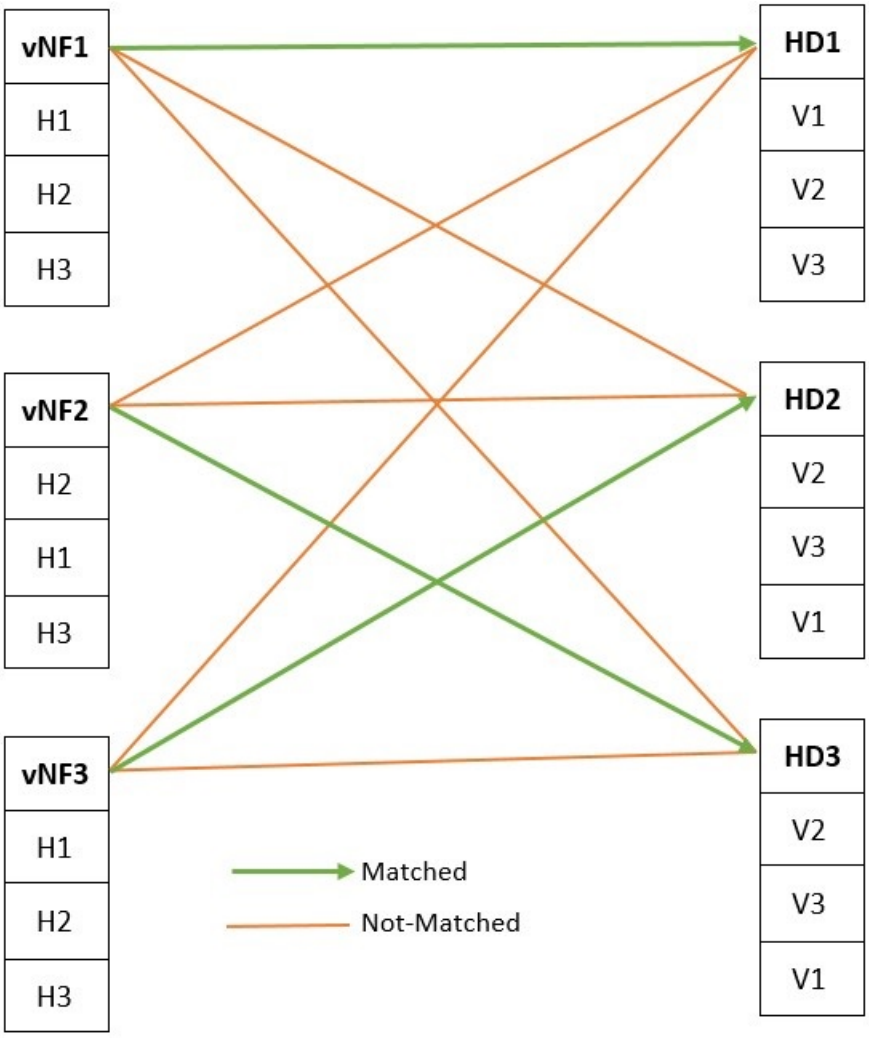
- The element of A of first matched set which prefers some given element B of the second matched set over the element to which A is already matched.
- B also prefers A over the element to which B is already matched.

Algorithm 1 shows the pseudo-code for the Stable matching algorithm.

Algorithm 1 Stable Matching Pseudo-code

```
1: procedure MATCHING MEN TO WOMEN UNTIL A STABILITY IS ACHIEVED
2:     Initialize all men and women as free
3:     while For every man (m) who has women (w) to propose to, where w is first woman on list
       of m whom m has no yet proposed do
4:         if (w is free) then
5:             (m,w) become connected
6:         elseSome pair (m' , w) already exists
7:             if (w prefers m to m') then
8:                 m' becomes free and (m, w) become engaged
9:             else(m', w) remain engaged
10:            end if
11:        end if
12:    end while
13: end procedure
```

David Gale and Lloyd Shapley [34] proved that for any number of men and women it is possible to solve the stable matching problem and make all the marriages stable and they also presented an algorithm for solving the problem. Figure 2.1 shows us a simple example of the stable match problem in which three vNFs want to get engaged with three hosting devices. It shows us all the engagements that can be done (all lines) and the only stable engagement (green lines). Stable Match has lots of application in real-world scenarios, including allocation of resources to users, allocating nodes to a network and medical student allotment to the medical schools.



vNFs	HDs		
	1	2	3
1	5	10	12
2	6	4	9
3	7	8	10

Figure 2.1: **Stable Matching Example**

2.1.3 Local Search

In the field of Computer Science, Local Search is a heuristic method solving computationally hard optimization problems [73]. Local search can be used on issues that can be formulated as finding a solution maximizing (or minimizing) a criterion among several candidate solutions. The algorithms in this category move from solution to solution in the space of solutions by applying local changes, until an optimal solution is found. These algorithms have broad applications in hard computational problems, including problems from computer science (particularly artificial intelligence), mathematics, operations research, engineering, and bioinformatics [73]. The only drawback of local search is that for some cases it gives us the solutions in lesser time than the

other algorithms but the solutions are not efficient.

Algorithm 2 shows the pseudo-code for the local search using random selection.

Algorithm 2 Local Search Pseudo-code

```
1: procedure Local Search Using Random Selection
2:     Find initial solution x
3:     Select a neighbour
4:     while Stop criteria is not met do
5:         Find neighbourhood A_x
6:         Find best solution x_best in A_x
7:         x ← x_best
8:     end while
9: end procedure
```

The local search starts with selecting a local feasible solution and then moving forward trying to find a better solution. The initial solution that we need to start the process can be selected by various techniques namely random selection, random walk, greedy search, hill climbing and genetic algorithm etc.

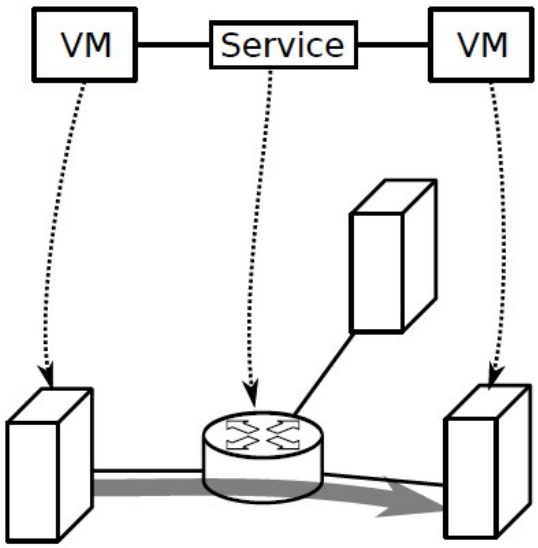
2.2 Related Works

2.2.1 Network Function Virtualization (NFV)

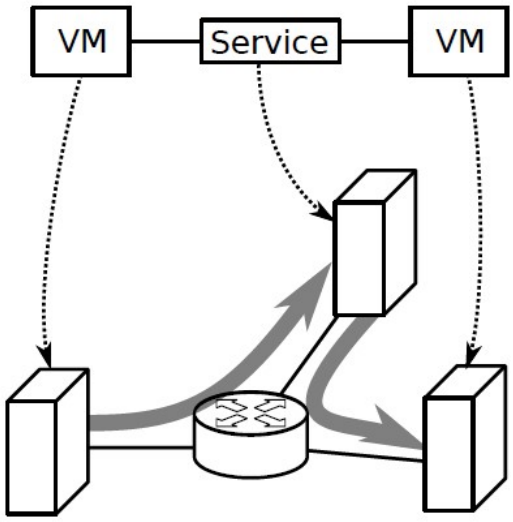
Network Function Virtualization (NFV) is an emerging network architecture and is an efficient technology in the networking area. Current researches are going-on to design or implement new techniques to make this emerging technology more efficient. During the literature review, we can find several studies 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 [10]. A similar case for the trend mentioned above can also be found in [11].

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 [12]. In this research, authors have presented **vNF-P**, a generic model for efficient placement of virtualized network functions.

Figure 2.2 shows us the different approaches that are explained in the research paper [12] to allocate NFV chains.



(a)



(b)

Figure 2.2: **NFV service chain allocation approaches.**
(a)- Services can be allocated on physical devices
(b)- Services offered using virtualized instances.

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 per-

formance state. The main issues of this research are to present a system model that formulates the desired resource allocation problem for different types of SFCs and to tackle the computational complexity of the problem [13]. They have used relaxation, reformulation, and successive convex approximation methods to solve the problems.

In modern data-centers, user network traffic uses a set of vNFs as a service chain to process traffic demands [14]. 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.

The deployment of vNF service chains (**vNF-SCs**) and task scheduling for bulk-data transfers in inter-datacenter (**inter-DC**) elastic optical networks (**EONs**) [15] is the main aim of their study. They propose a DP-based vNF-SC deployment and task scheduling algorithm, which can find the solution with a minimum service completion time (SCT). For multi-branch data-intensive vNF-SC requests, a correlation-aware vNF-SC deployment and task scheduling algorithm (which minimizes the average SCT) is proposed as the approach to solving the above problem.

Virtualized network function (vNF) service chaining in optical data center networks (**DCN**) is a more complex problem than in packet-switched networks, as it introduces additional constraints related to the optical network [18]. The most common example of this is in an optical DCN one needs to make sure that visual network resources are efficiently utilized, which requires multiplexing of several vNF chains to fill the optical pipes. In [18] authors propose a novel and flexible DCN architecture based on optical circuit switching technology and supporting service chaining in the optical domain. The problem has been formulated using ILP and heuristic methods.

One of the main challenges for the NFV environment is how to efficiently allocate Virtual Network Functions (vNF) to Virtual Machines (VMs) [19]. 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. Figure 2.3 shows the proposed arrangement of the vNFs by the above research authors.

vNF placement is a phase to allocate vNFs in a network infrastructure [20] optimally. Figure 2.4 shows the NFV environment in which the transformation is being performed using the consolidation of different vNF types in standard general purpose computers (servers, storage devices, etc.). This may be located in data centers, network nodes and close to end user premises.

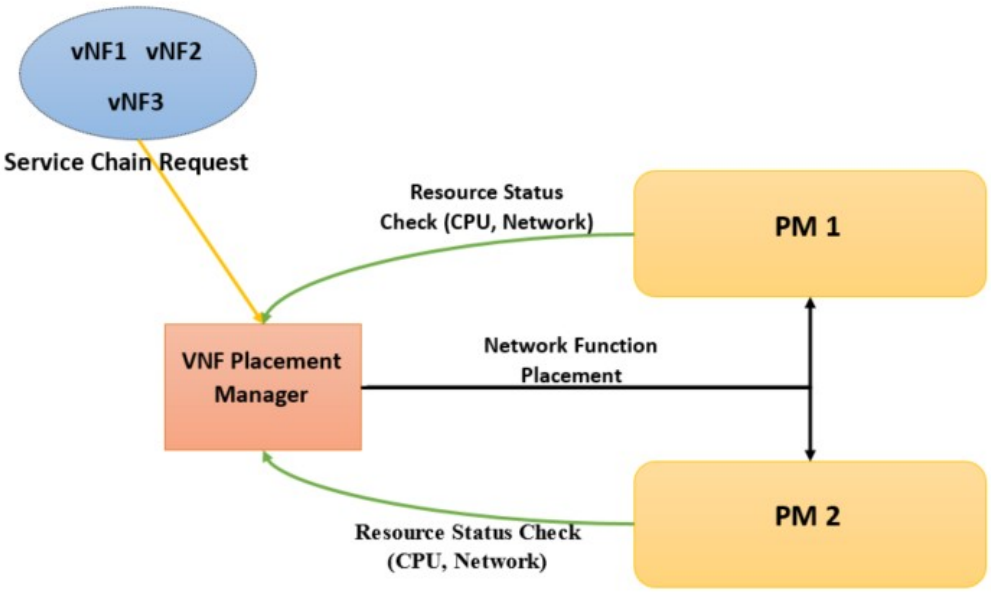


Figure 2.3: **Architecture of the System for the vNF placement**

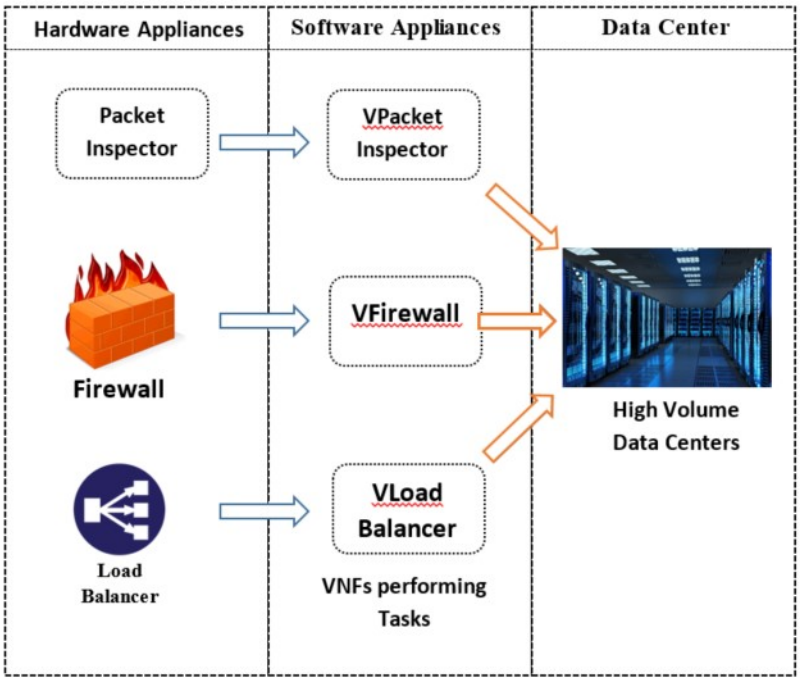


Figure 2.4: **NFV Environment**

Several approaches are already proposed to model the vNF placement for efficient resource allocation. Moen and Turck [21] presented a formal vNF-Placement (vNF-P)

for resource allocation in hybrid network environments in which network functions can be allocated on both physical hardware and virtualized instances. With the rise of 5G networks, ultra-low and predictable end-to-end latency is becoming increasingly important as the critical enabler for many new and visionary applications. Achieving ultra-low latency has been attempted at various points of the networking stack, from OS kernel [22] to 5G millimetre-wave cellular networks [23].

2.2.2 Stable Matching

Stable Match algorithm has been used frequently to solve many problems in various research areas in computer science and other fields of study too. A stable matching-based virtual machine (VM) allocation mechanism for Cloud data centers has been proposed by [24]. In this paper, the authors have used a different approach using Stable Match as in this research both involving party groups matching process have a mutual objective, which is to reduce the total energy consumption of a Cloud Data centre while giving a high Quality of Service. They have compared their result with Local Regression Robust (LRR) algorithm as it performs exceptionally well than other algorithms. The comparison has been made using four different metrics: energy consumption, Service Level Agreement (SLA) violations, migration number, and Energy & SLA violations (ESV).

McVitie and Wilson [35] pointed out that the algorithm by Gale and Shapley [34] 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. [36] to find a stable matching satisfying the optimality criterion of McVitie and Wilson.

In the normal many-to-many matching problem, a person may have preferences defined over subsets of the members of the other set. Two approaches have been defined based on the assumptions placed on the preference function of a person over the members of the other set. The **first approach** comes from **economics area**, in which they assume that each person (or a firm) specifies a strict preference ordering on all possible subsets of the set of acceptable partners (or workers). In this matching approach workers and firms regard each other as substitutes, that is, if a worker is a desirable employee to a firm amongst a subset of workers, then he continues to be so

even amongst a less desirable subset of workers. Many solutions were developed using this assumption for the one-to-one stable matching have their counterparts for one-to-many (Roth and Sotomayor [37]) and many-to-many situations (Roth [38], Sotomayor [39], Martinez et al. [40] and Alkan [41], [42]). The above-defined approach has a computational limitation due to the exponential nature of the preference function, which puts a lower bound on what any algorithm can achieve.

The **second approach** comes from the **computer science domain** which is closely related to the original stable marriage problem of Gale and Shapley. This approach states that each man and woman has an upper limit on the number of partners and specifies a preference ordering on acceptable individuals of the opposite sex (and not on combinations of them). Bansal et al. [33] uses the above approach to the many-to-many stable matching problem and generalize the notion of optimality proposed for one-to-one matching by McVitie and Wilson [35]. They show that the optimality criterion makes sense provided that we include a no-complementarities condition for preferences on combinations of partners.

Another research has been done in vehicular network technology in which Stable Matching has been used to replicate content in the given networks [25]. Vehicular networks are ad-hoc networks composed of mobile vehicular nodes and fixed roadside units. They propose a content replication scheme using Stable Matching, naming it **RSM**. RSM constructs an initial bipartite graph (a graph whose vertices can be divided into two disjoint and independent sets such that all edges connect a vertex of both sets) with the contents as the left vertices and the storage cells of roadside units as the right vertices as the first step. They found that RSM has a small storage consumption and a high access ratio with adequate access latency. Figure 2.5 shows the process in which a roadside unit u_1 obtains a request from a vehicle v_1 at time $\mathbf{t}$, and forwards this request to the Internet. Then the online control centre computes a content replication solution and allocates the data to another roadside unit u_2 accordingly. Thereafter, u_2 responds to v_1 at a later time ($\mathbf{t}+ \Delta\mathbf{t}$).

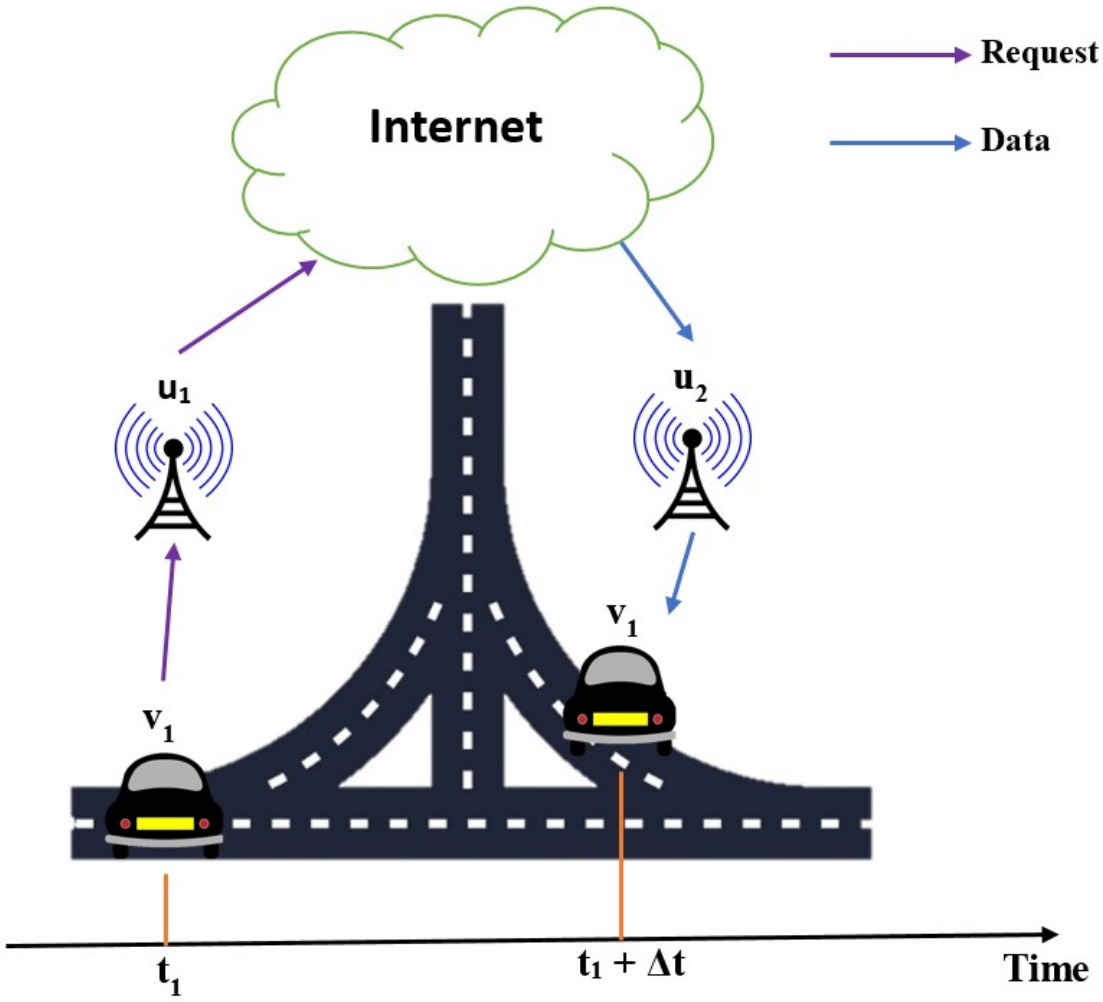


Figure 2.5: **Sketch Map of Data Request and Accusation Process**

Stable Matching Algorithm has also been used in scheduling of both computing and storage resources in data centres [26]. 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.

This research paper [30] 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.

Another research that is recently done using the Stable Matching theory in which the SM has been used to analyze vehicle stowage problem and establish the matching model [32]. The method they proposed was helpful to the logistics parks and websites to improve the vehicle stowage service. As stated by them, this research can be beneficial to real businesses, and it can avoid the supply side into a crisis of survival and prevent market imbalance.

Content Delivery Network [43] is defined as an extensive distributed system consisting of multiple servers deployed in much geographical location that delivers the content to end user with high performance and high availability. Due to the increase in the load of the content delivery server and network degrades the quality of service as 70% of unintentional failures are Single link failure, so it creates potential failure along the delivery chains. To solve the above problem, Gupta et al. [43] targeted the stability of network reliability by providing hop length stability and network connectivity. They use the stable matching approach on the network to find the vital link to be protected so that users can access the server within a small hop count even if the non-essential links fails. They also found that the problem can be solved in polynomial time when the hop count is determined to be one.

2.2.3 Local Search

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 based genetic algorithm has been used to design the reliable networks optimally [27]. 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.

Travelling Salesman Problem (TSP) is one of the significant problems that are NP-complete. TSP requires that starting from place "**A**" the salesman should travel to each city once before returning home. There are many algorithms used to solve this problem, namely k-opt algorithm, Christofides algorithm, pairwise exchange and ant colony optimization. A traditional ant colony algorithm (ACA) for solving TSP easily gets stuck into a locally optimal solution and slow convergence, also the quality of the solution is not ideal [28]. Authors propose a Dynamic Local Search based Ant Colony Algorithm (DLACA) in which each and has the ability of local search and it can use this ability according to the real-time condition which enhances the search quality of algorithm and stabilizes the solution. They use Matlab for solving TSP using ACA and DLACA, and the solutions obtained by them show that the DLACA achieves the known optimal solution within the stipulated time and the stabilization of solution is also better.

Given a graph or hypergraph, the graph or hypergraph Max-k-Cut problem is to partition the vertices into k nonempty sets such that the sum of weights of edges across different sets is maximized. Wenxing Zhu et al. [49] proposed deterministic local search algorithm for the problem, which has a performance ratio 1 - 1/k for Max-k-Cut of the graph, and has a similar result for Max-k-Cut of hypergraph. Safaa Alqallaf et al. [50] new hybrid local search approximation algorithm for solving the capacitated Max-k-cut problem and contrast its performance with two local search approximation algorithms. The first algorithm uses swapping neighbourhood search technique, whereas the second algorithm uses a vertex movement method. They analyze the behaviour of the three algorithms concerning running time complexity, several iterations performed and the total weight sum of the cut edges where algorithms are "Vertex swapping local search algorithm", "Vertex Movement Local Search Algorithm" and "Hybrid Local Search Algorithm" respectively. It is clear from their performed research that the time required to solve each problem is almost the same for the three algorithms for small values of n where n is the number of vertices. As n increases, the running times of the three algorithms differ noticeably.

2.3 Conclusion

We found that there is currently no heuristic can solve the problem proposed by R. Cziva et al. [8] in a polynomial time. So, we are the first ones to propose the heuristic to solve this problem in a polynomial time. Thus, it is not possible to compare our

obtained solutions with any other heuristic but they will only be compared with the optimal solution given by the mathematical model.

Chapter 3

System Model, Modification and Proposed Heuristic

3.1 Introduction

In this chapter we discuss the system model, parameters and the modification proposed in the system model. We also discuss the two approaches that were used to solve the vNF allocation problem. **Greedy Approach** and the **Stable Matching** based approach have been used to match the vNFs and hosting devices. The results obtained have been compared to optimal result obtained by the mathematical model and are then analyzed.

3.2 System Model

3.2.1 Overview

In this research, we are using the same model as used by [8]. 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 research problem is to allocate vNFs to different hosting devices to minimize the latency caused.

3.2.2 Parameters Used

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 between the user of the n_i vNF in case that vNF is located at h_j.

Table 3.1: **Parameters**

We consider a system with vNFs and hosting devices, where $\mathbb{N} = \{n_1, n_2, n_3, ..., n_i\}$ 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 $\mathbb{H} = \{h_1, h_2, h_3, ..., h_j\}$ 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.

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

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

3.2.3 ILP Model

The objective of our model is to minimize the **Total-Latency** value which is given by equation (3.1).

$$Minimize \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} x_{ij} l_{ij} \tag{3.1}$$

Subject To-

$$\sum\nolimits_{n_i \in \mathbb{N}} x_{ij} * R_i \ \leq \ C_j, \forall h_j \in \mathbb{H} \tag{3.2}$$

$$\sum\nolimits_{h_j \in \mathbb{H}} x_{ij} l_{ij} \ \leq \ \mathrm{ML}_i, \forall n_i \in \mathbb{N} \tag{3.3}$$

$$\sum\nolimits_{h_j \in \mathbb{H}} x_{ij} = 1, \forall n_i \in \mathbb{N} \tag{3.4}$$

- **First constraint (3.2)** ensures that vNFs are placed to hosting devices with sufficient capacity. This constraint also defines that vNFs can not be allocated to the hosting device if its capacity gets filled that is the total of the requirements of the vNFs connected to a hosting devices should be less than the capacity of that hosting device.
- **Second Constraint (3.3)**, 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 (3.4)**, constraint ensures that all vNFs are allocated to hosting devices exactly once. A single vNF can not 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 also an NP-hard problem [8] 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.

3.2.4 Simulation Environments

The ILP models used in this thesis are implemented in IBM CPLEX, and our proposed algorithms have been implemented in C++. In this process, we do not use a network simulator as we are not solving any network research problem but computational problem.

For input, the data taken includes the number of vNFs, hosting devices, users. The other values taken as input are capacity of hosting devices, requirement and a maximum latency of vNFs and latency between the vNF and hosting device. For latency between the vNFs and the hosting devices, we take random values between 15 to 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 to 75. Requirements and a maximum latency of vNFs have also been taken randomly between 1 to 15 and 20 to 50 respectively.

Since we are just solving a computational problem considering latency as the input of our problem, unit for latency can be any time unit (for example, milliseconds, microseconds etc.)

Similarly we are considering the system to be reliable as we will not be considering the factors effecting reliability like overloading, physical damages, software anomalies, power failures, device failures and others.

The properties for the system used for performing all the simulations have been illustrated in Table 3.2.

System Properties	Value
Windows Version	**Windows 10 Home Version**
Processor	**Intel i5-5200U**
Random Access Memory (RAM)	**8 GB**
System Type	**64-bit OS, x64-based processor**
GPU	**NVIDIA GeForce 830M**
Graphic Memory	**2 GB**

Table 3.2: **System Properties**

3.3 Initial ILP Model Modification

It can be observed that only one constraint gives in-feasible solutions under many scenarios. The allocation constraint **Eq. 3.4** states that every vNF should be connected to at most one hosting device. If this constraint fails in any circumstance, the whole model fails. Some of the example cases are given next.

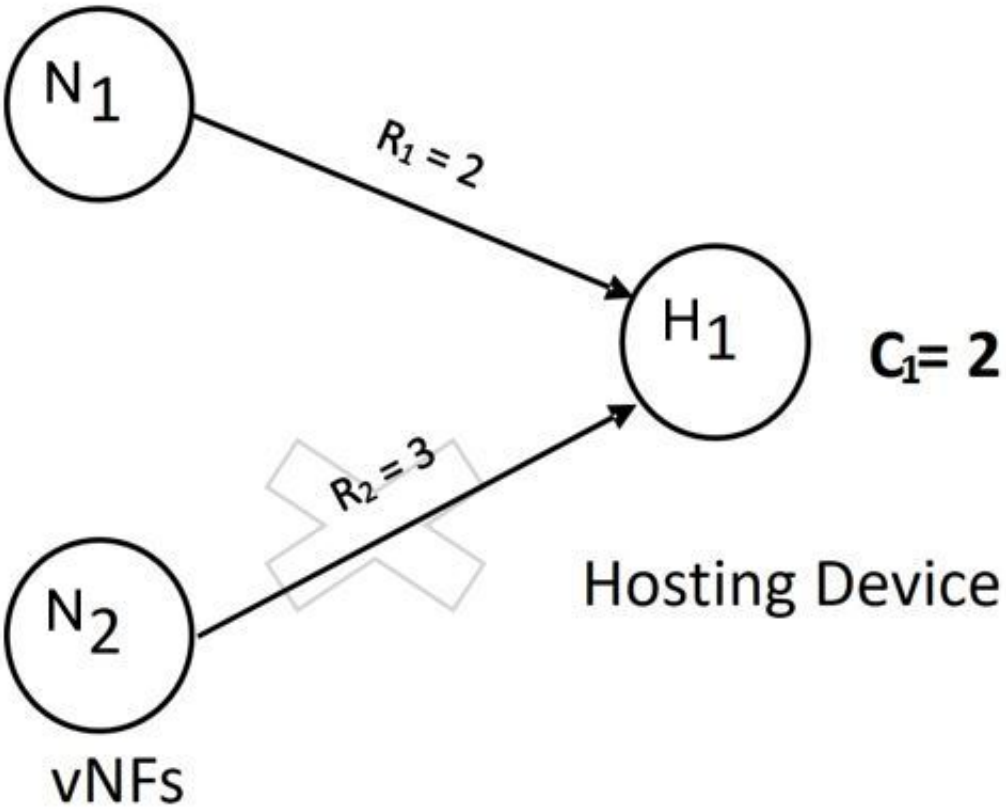
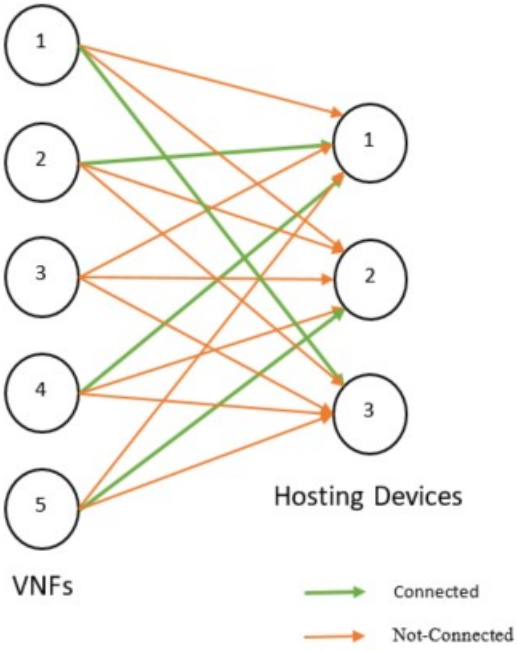


Figure 3.1: **Fail Case Scenario for 2 vNFs**

Let us consider a case in which we have two vNFs that want to connect to a hosting device, as shown in Figure 3.1. The vNFs have 2 and 3 as requirements respectively, and the hosting device had a capacity of 2. In the above case, the hosting device can not connect to the second vNF as it does not have the required capacity to connect with both the vNFs as a result due to constraint Eq. 3.4, the model will fail.



vNF	Requirement
1	4
2	3
3	4
4	2
5	3

HD	Capacity
1	5
2	5
3	5

Figure 3.2: **Fail Case Scenario for 5 vNFs**

Let us consider another scenario with five vNFs that want to connect to three different hosting devices as represented in Figure 3.2. 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 would not be connected and the connections in green will only be connected. Though it does not have much problem but according to the allocation constraint Eq. 3.4 the model will fail and will give an infeasible solution.

3.3.1 Generalization for Failures

Theorem 1: For any network with any number of vNFs and hosting devices the network model will fail when the total capacity of the hosting devices is less than the total requirements of the vNFs.

Proof: From the above two cases, we can say there can be many more cases that can lead to failure of the network model. To generalize the above cases, let us consider a model with a total number of vNFs (G) and the total number of hosting devices (H). R_n is the requirement for the vNF "n" and C_m is the capacity of the hosing device "m". Let us consider the case given in Figure 3.2 here we can see that the total of

requirement for the vNFs is 16 but the total capacity of the hosting devices is 15 so even if the hosting devices exhaust their whole capacity they cannot pair with all the hosting devices. Hence proved that the network model will fail when the total capacity of the hosting devices is less than the total requirements of the vNFs.

Theorem 2: For any network with any number of vNFs and hosting devices the network model will fail when the maximum latency tolerance of any of the vNF is less than the latencies in the l_{mn} matrix.

Proof: To generalize the above cases, let us consider a model with a total number of vNFs (G) and the total number of hosting devices (H). ML_n is maximum latency vNF "n" can tolerate and l_{mn} which is the matrix having the latencies between the user of vNF "n" in case that vNF is located at "m". Now let us consider a case in which the maximum latency tolerance of a vNF is lesser than the latency that it transmits when connecting to various hosting devices. Due to constraint 3.3 it would not be able to connect to any of the hosting devices and the model will fail. Hence proved that the network model will fail when the maximum latency tolerance of any of the vNF is less than the latencies in the l_{mn} matrix.

So, we can generalize that if:

1. The total capacity of the hosting devices is less than the total requirements of the vNFs. This ensures that all the vNFs would not be connected and the model will fail.

2. The maximum latency tolerance of any of the vNF is lesser than the latencies in the l_{mn} matrix. This will also lead to the rejection of that vNF and the model will fail.

3.3.2 Proposed Modification

Thus, to fix the above problem, 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 (3.4)** is replaced by:

$$\sum\nolimits_{n_i \in \mathbb{N}} \sum\nolimits_{h_b \in \mathbb{H}} x_{ij} = M \tag{3.5}$$

$$\sum\nolimits_{h_b \in \mathbb{H}} x_{ij} <= 1, \forall n_i \in \mathbb{N} \tag{3.6}$$

where '**M**' is the total number of vNFs that can be connected optimally and another **constraint (3.6)** 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:

$$Maximize\ M = \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} x_{ij} \tag{3.7}$$

$$Subject\ To-$$

$$\sum_{i \in B_j} x_{ij} * R_j <= C_j, \forall j \in \mathbb{H} \tag{3.8}$$

$$\sum_{j \in A_i} x_{ij} <= 1, \forall i \in \mathbb{N} \tag{3.9}$$

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 3.3).

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

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, it satisfies constraint 3.3.

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

C_j is capacity of hosting devices and R_i are the requirements for vNFs. Further the constraint 3.8 is similar to constraint 3.2 but with different input. Constraint 3.9 states that one vNF can not 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) [9]. 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 [9].

The complete new model with modification becomes:

$$Minimize \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} x_{ij} l_{ij} \tag{3.10}$$

Subject To-

$$\sum\nolimits_{n_i \in \mathbb{N}} x_{ij} * R_i \ \leq \ C_j, \forall h_j \in \mathbb{H} \tag{3.11}$$

$$\sum\nolimits_{h_j \in \mathbb{H}} x_{ij} l_{ij} \ \leq \ \mathrm{ML}_i, \forall n_i \in \mathbb{N} \tag{3.12}$$

$$\sum\nolimits_{n_i \in \mathbb{N}} \sum\nolimits_{h_b \in \mathbb{H}} x_{ij} = M \tag{3.13}$$

$$\sum\nolimits_{h_b \in \mathbb{H}} x_{ij} <= 1, \forall n_i \in \mathbb{N} \tag{3.14}$$

From hereon this model has been taken as the mathematical model to solve the problem proposed by [8] and it is refereed wherever the vNF allocation model has been mentioned.

3.4 Greedy Approach

In this approach we start by selecting the pair that has lowest latency value and then move on to select the next pair with minimum latency. This process is repeated till either all the vNFs are connected or the capacity of the hosting devices gets exhausted. This is the simplest way to find a solution satisfying the requirements of the problem.

3.4.1 Algorithm

Algorithm 3 Greedy Approach

```
1: procedure Matching vNFs to Hosting Devices using a greedy approach
2:     Initialize x_ij as 0 for all vNFs and hosting devices.
3:     Initialize both totalLatency and count as 0
4:     vNFs have requirements and maximum latency and Hosting Devices have capacity
5:     while (Count < Number of vNFs) do
6:         for (All number of vNFs) do
7:             for (All number of Hosting Devices) do
8:                 Selecting the pair with minimum latency from latency matrix and satisfying constraints 3.11 and 3.12.
9:                 Update the x_ij as 1 where i is selected vNF j is selected hosting device
10:                Update the capacity
11:                Remove selected pair for next iteration
12:            end for
13:        end for
14:        Increment Count by 1.
15:    end while
16:    if (All vNFs are connected) then
17:        Print totalLatency
18:    else
19:        Print "Infeasible Model".
20:    end if
21: end procedure
```

In the above algorithm 3 we try to give a greedy approach heuristic for problem proposed by R. Cziva et al. [8] in this approach we find that pair first, which has minimum latency in latency matrix and it satisfies the constraints, capacity (3.11) and latency (3.12), we then update the allocation matrix and capacity of the hosting device. Then we remove that pair and start the process again; it continues until the count is less than the total number of vNFs. Then we calculate the total latency using the allocation matrix (x_{ij}) and latency matrix, then solution is provided.
For the modified model proposed by us (3.10) the If condition at line 16 will change to if "**M**" devices are connected only then it will give a solution, where "M" is calculated by 3.7.

3.4.2 Results

The different cases that were used are 20 and 30 for vNFs. 10, 15, 20 are a different number of host devices which are then used to form different cases and use them to compare solutions for both CPLEX and proposed greedy approach. All of the simulation results illustrated are an average of 10 different runs for a particular scenario.

Table 3.3 shows the solutions obtained by the **Greedy Approach (GrA)** and their comparison with the **Optimal (Opt)** solution given by the ILP model.

vNFs	**Hosting Devices**	**Opt**	**GrA**	**% Diff**
20 vNFs	**10 HD**	389.66	550.33	41.23338
	15 HD	335.67	505.13	50.48411
	20 HD	309.33	513.66	66.05567
30 vNFs	**10 HD**	313.33	497.33	58.72403
	15 HD	526.66	643.5	22.18509
	20 HD	503.67	639.33	26.9343

Table 3.3: **Latency Comparison between Optimal and Greedy Approach**

3.4.3 Result Analysis

This technique was used initially to solve the vNF allocation problem in polynomial time. But it is clear from the results illustrated by the Table 3.3 that the technique is not good to solve this problem. Another approach (Stable Matching) was then tried and on comparison the solutions given by stable matching heuristic were far better than the solutions given by greedy approach. Thus, greedy approach was rejected, and we we went on with the Stable Matching approach which is explained in detail in the coming section.

3.5 Stable Matching Algorithm

Stable Matching is initiated 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.

3.5.1 Algorithm

Algorithm 4 Stable Matching

```
1: procedure Matching vNFs to Hosting Devices until a stability is achieved
2:     All vNFs and Hosting devices are free.
3:     Initialize both totalLatency and count as 0.
4:     while (There Exist a Free vNF (n) who has not proposed to hosting device (h) and count
   is less than M) do
5:         h→ is the first preferred Hosting Device
6:         if (h is free and Constraints 3.11 and 3.12 are satisfied) then
7:             (n, h) become engaged
8:             Update count = count + 1
9:             Update capacity = capacity of h - requirement of n
10:            Update totalLatency = totalLatency + latency between the n and h
11:        else(Some pair (n′, h) already exists)
12:            if (h prefers n to n′) then
13:                (n, h) become engaged
14:                n′ becomes free
15:                Count remains same
16:                Update capacity and totalLatency
17:            else
18:                (n′, h) remain engaged
19:            end if
20:        end if
21:    end while
22:    Print totalLatency
23:    Print matched pairs
24: end procedure
```

In the above algorithm 4, 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 3.7. 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 $\mathcal{O}(n^2)$

3.5.2 Flowchart for Algorithm

The Figure 3.3 shows the flow chart for working of our Stable Matching Algorithm. The algorithm is implemented in **C++** language.

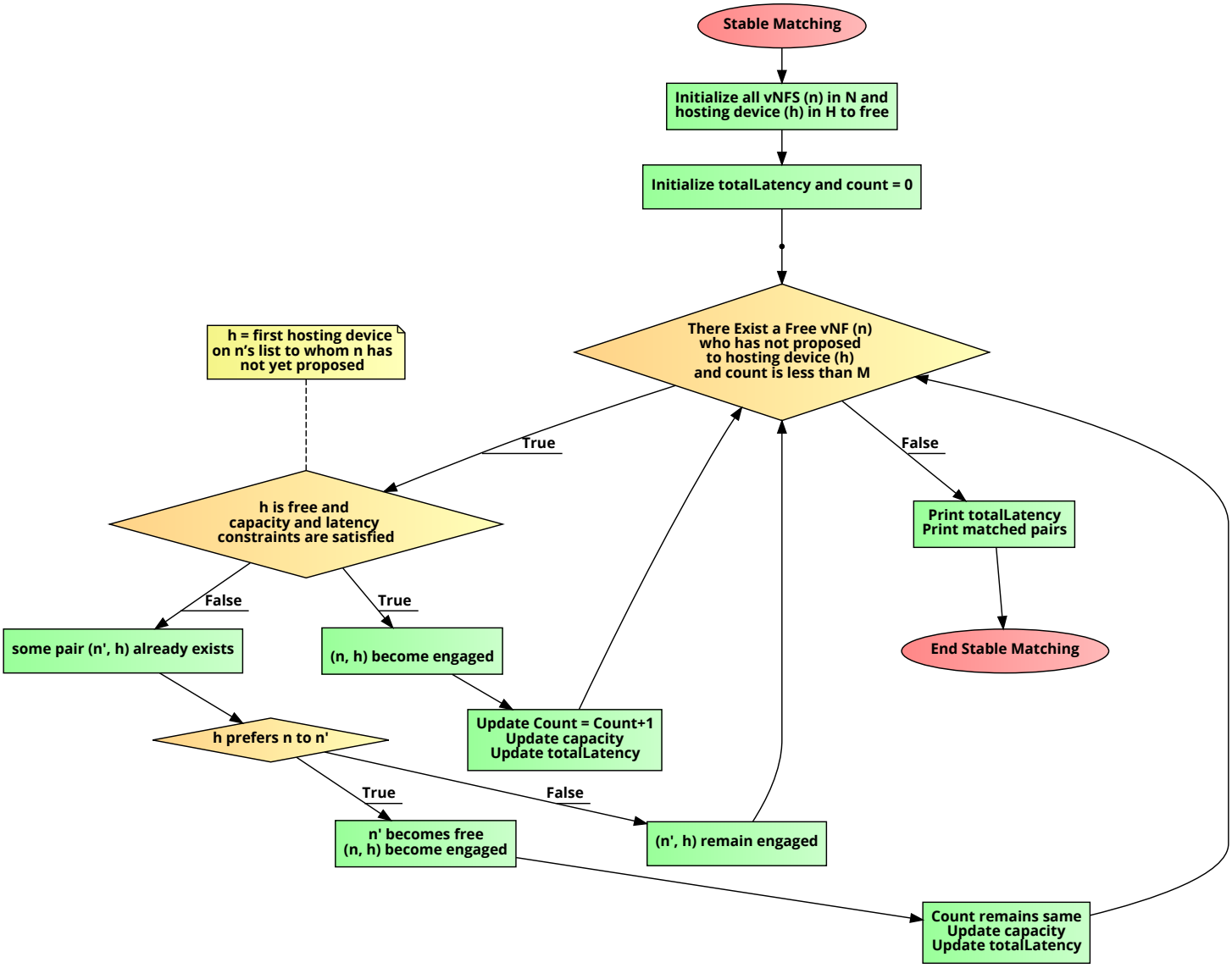


Figure 3.3: **Flowchart for SMA**

3.5.3 Simulation Results

Scenario used for calculating these solutions is similar to that used in Greedy Approach but with more instances. The different instances that are used in this scenario are 50, 100, 500, 1000, 2000 and 3000 for vNFs. 5, 10, 15, 20, 50, 100, 150, 200, 250 and 300 are the different number of host devices which are then used to form different cases and use them to compare solutions for both CPLEX and stable match. In the solutions provided, **Opt** is defined as Optimal (ILP Model Result) and **SMA** is defined as Stable Match Algorithm. All of the simulation solutions presented in this section are an average of 10 different runs for a particular scenario.

The figures ahead illustrate us the solution comparison between the ILP solution given by IBM Cplex (Optimal) and solution given by our proposed heuristic approach(Stable Matching) on basis of **TL** (Total Latency) for different cases.

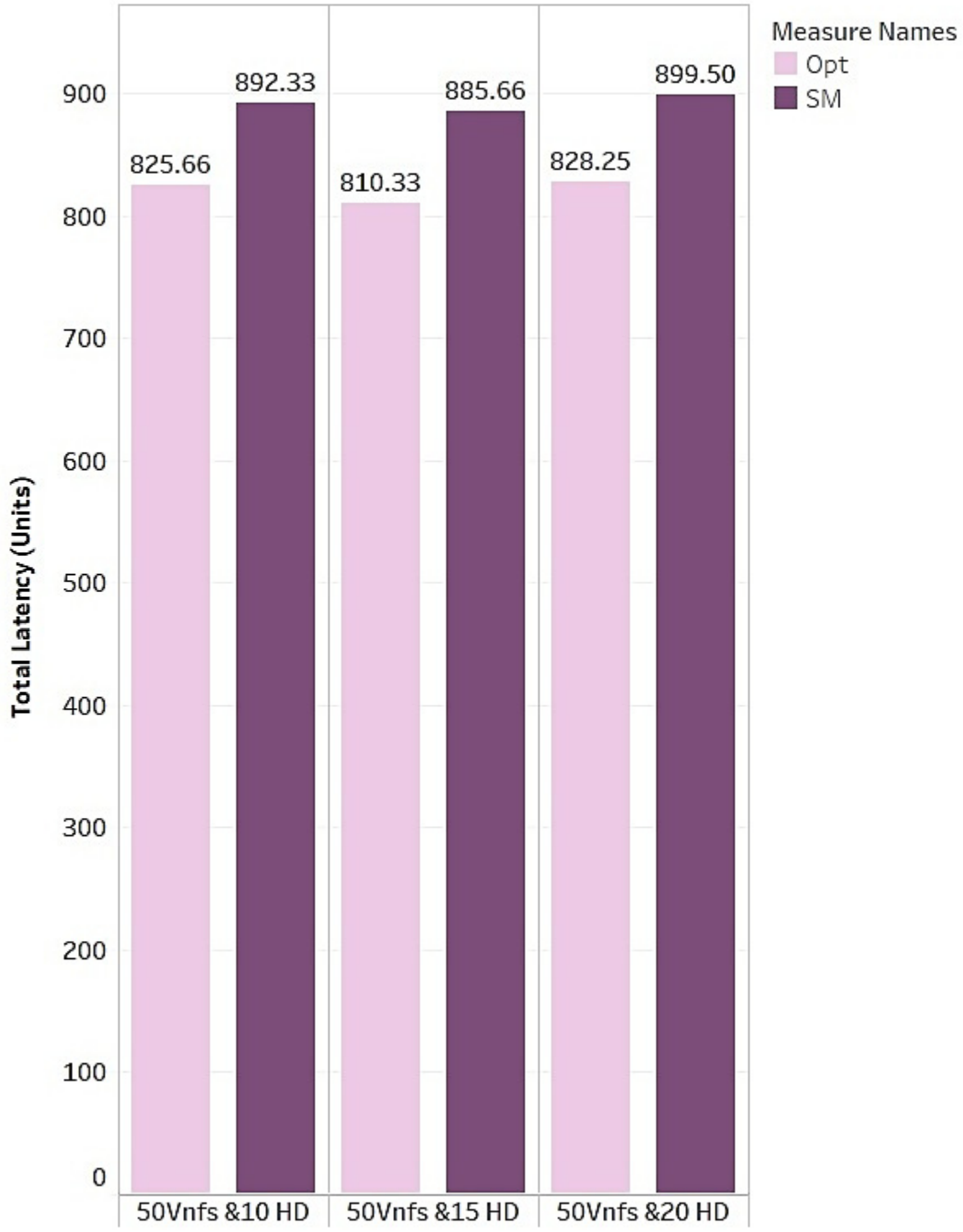


Figure 3.4: **Resulting Latencies for 50 vNFs using ILP and Stable Matching for different number of Hosting Devices**

Figure 3.4 shows the graphical comparison between the ILP and Stable Matching algorithm for 50 vNFs when we have a different number of hosting devices.

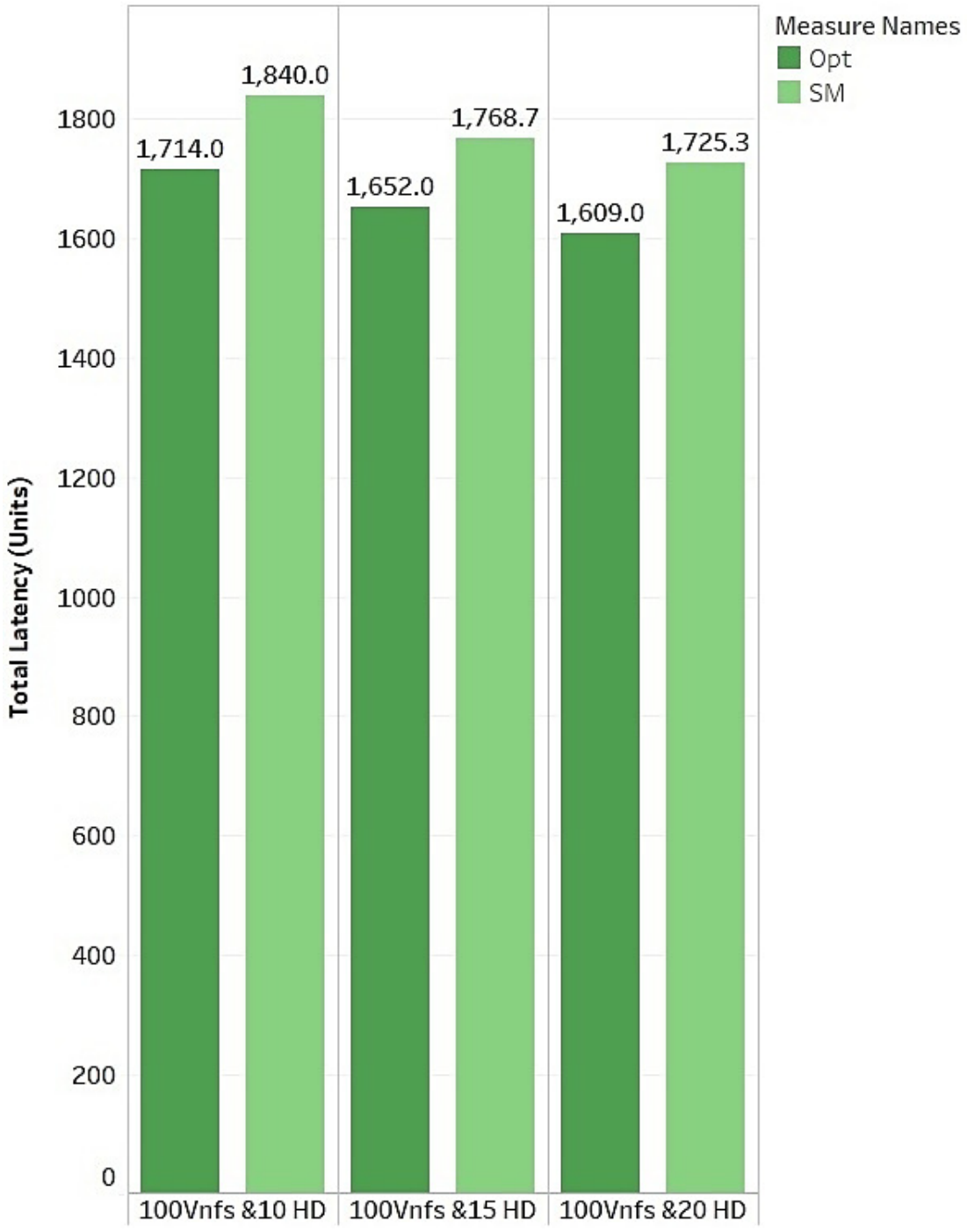


Figure 3.5: **Resulting Latencies for 100 vNFs**

Figure 3.5 shows 100 the comparisons between the ILP and Stable Matching al-

gorithm for 30 vNFs when we have different number of hosting devices.

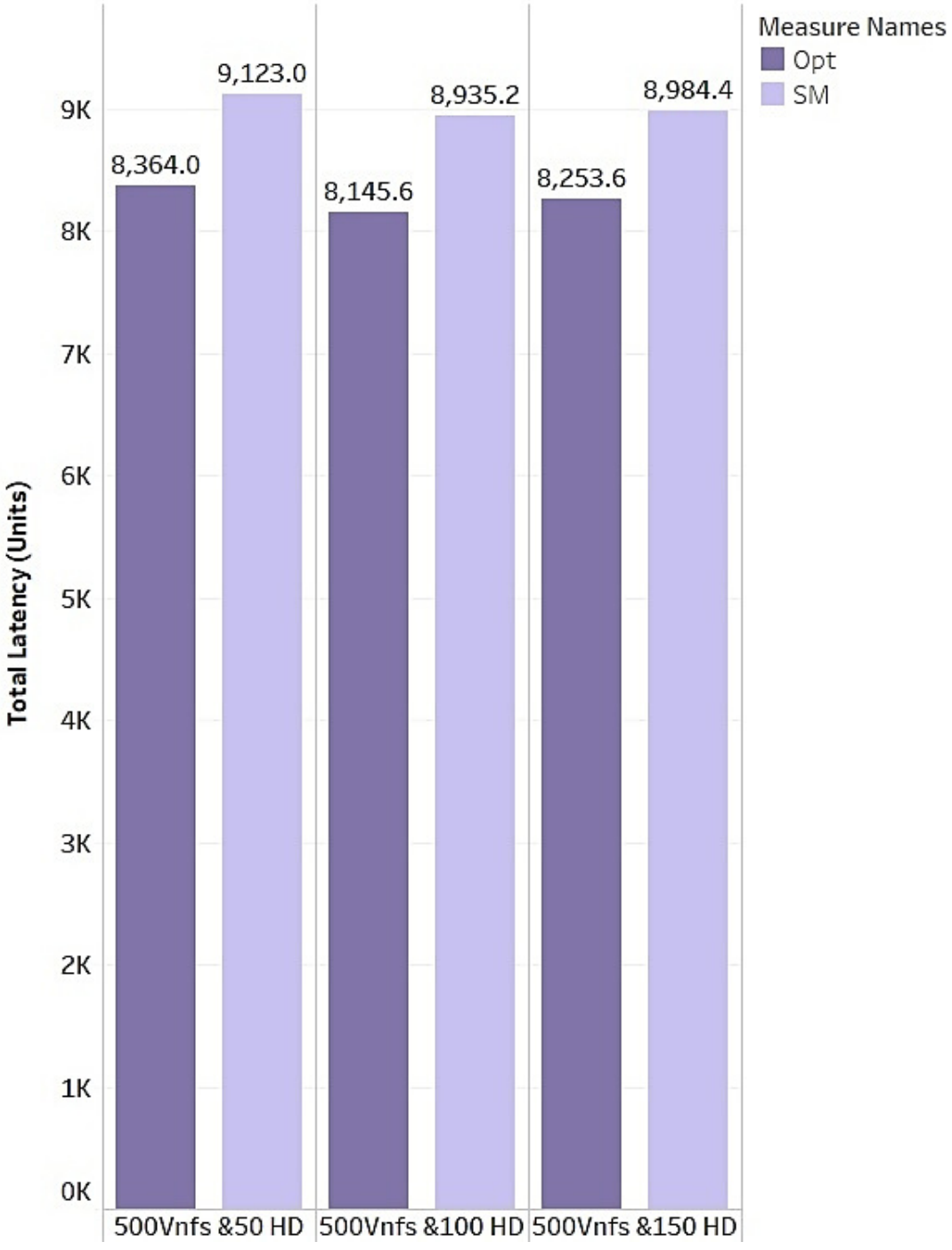


Figure 3.6: **Resulting Latencies for 500 vNFs**

Figure 3.6 shows the comparisons between the ILP and Stable Matching algorithm for 500 vNFs when we have different number of hosting devices.

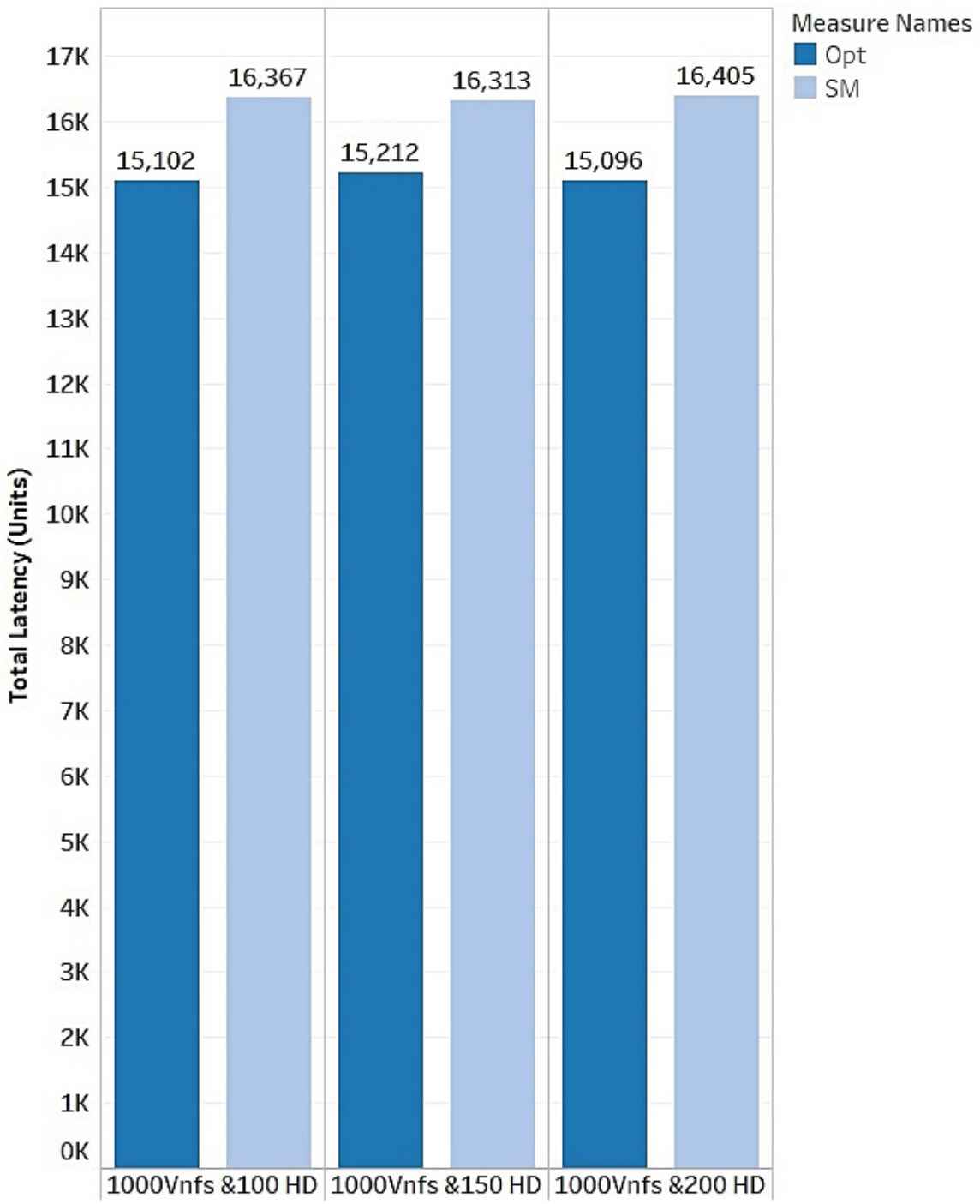


Figure 3.7: **Resulting Latencies for 1000 vNFs**

Figure 3.7 shows the comparisons between the ILP and Stable Matching algorithm for 1000 vNFs when we have different number of hosting devices.

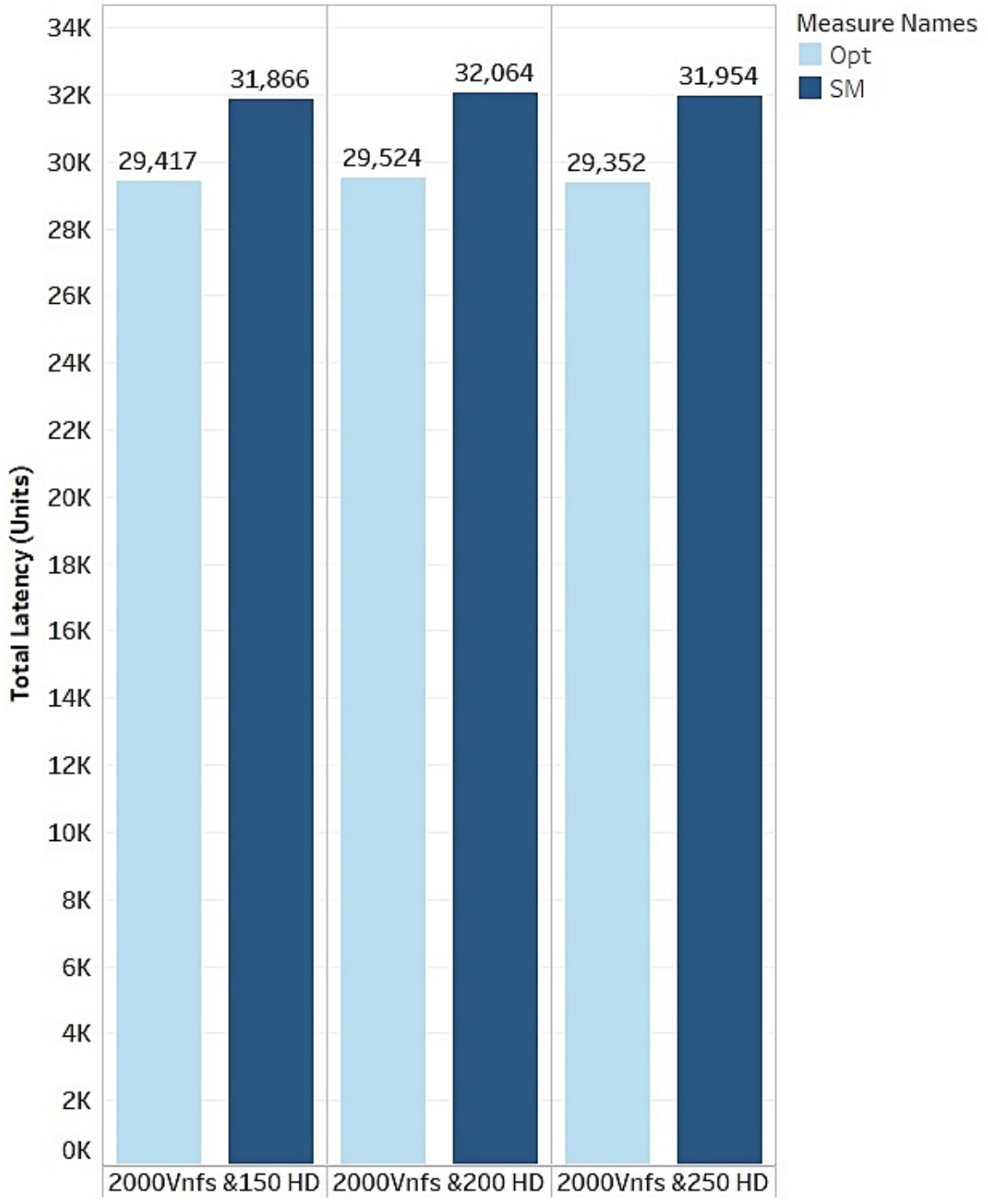


Figure 3.8: **Resulting Latencies for 2000 vNFs using ILP and Stable Matching**

Figure 3.8 depicts the comparison between the ILP model solutions and the solutions obtained by the stable matching algorithm when we have 2000 vNFs and various

number of hosting devices.

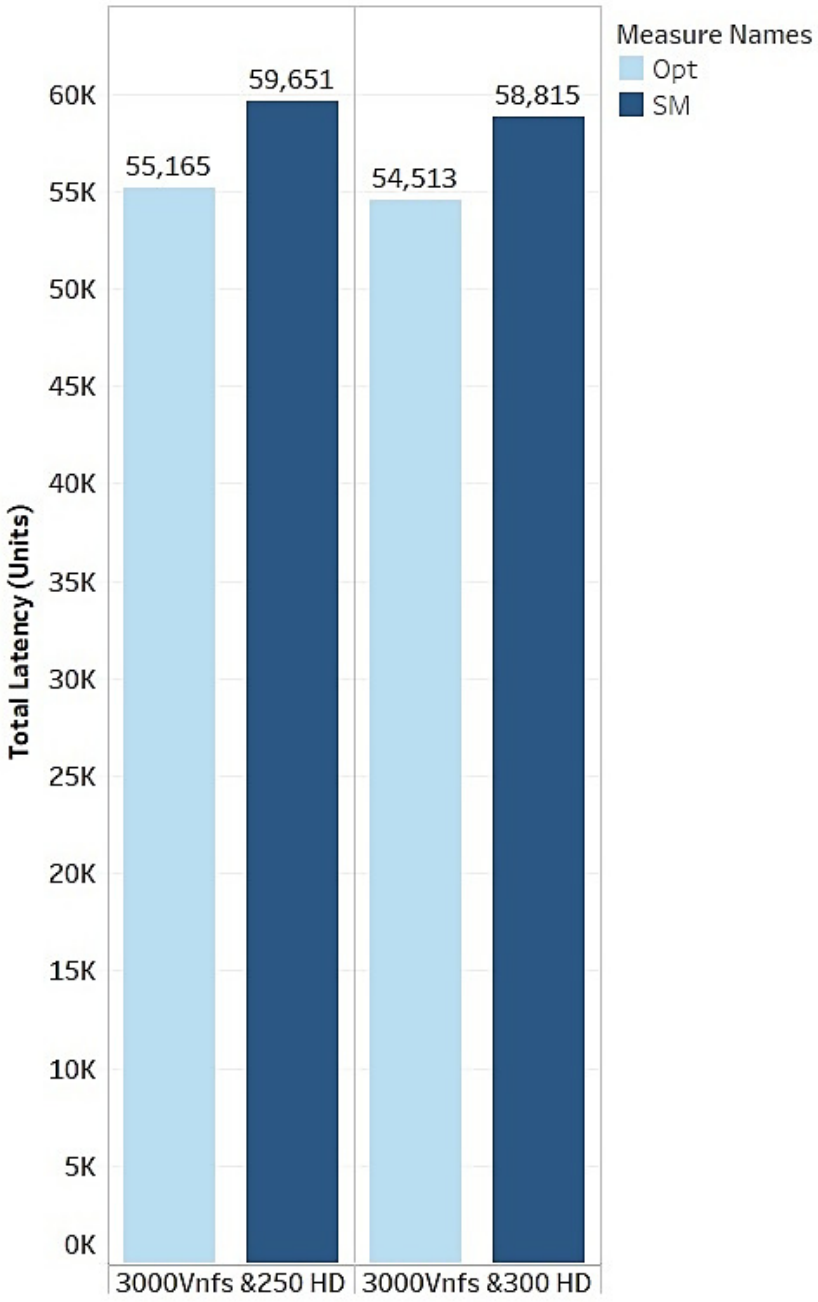


Figure 3.9: **Resulting Latencies for 15 Hosting Devices using ILP and Stable Matching**

Figure 3.9 depicts the comparison between the ILP model solutions and the results obtained by the stable matching algorithm when we have 15 hosting devices and various number of vNFs.

vNFs	Hosting Devices	Opt	SM	% Decrease (Time)
20 vNFs	5 HD	4.47	3.53	21.03
	10 HD	6.17	4.44	28.04
	15 HD	7.63	5.76	24.51
	20 HD	8.51	7.31	14.10
30 vNFs	10 HD	3.14	2.12	32.48
	15 HD	2.93	2.03	30.72
	20 HD	5.97	4.63	22.45
50 vNFs	10 HD	5.45	4.32	20.73
	15 HD	7.03	6.20	11.81
	20 HD	8.23	7.37	10.45
100 vNFs	10 HD	7.89	7.23	8.37
	15 HD	7.21	6.27	13.04
	20 HD	7.45	6.57	11.81

Table 3.4: **Working Time Comparison for Different vNFs (Seconds)**

Table 3.4 depicts the comparison between the time complexity of optimal and the time taken by the stable matching algorithm when we have 20, 30, 50 and 100 vNFs respectively and a various number of host devices.

3.6 Conclusion

The anomaly in the initial ILP was critical as even if there is a small case that one vNF does not get connected can lead to the failure of the whole network. Thus the model has been modified to make it more efficient. The modified problem is still an NP-hard problem which takes exponential time in worst case scenario. No heuristic has been proposed by R.Cziva et al. [8] for solving the vNF allocation problem in polynomial time. In the next chapter we discuss the heuristic approach proposed by us to solve the vNF allocation problem in polynomial time.

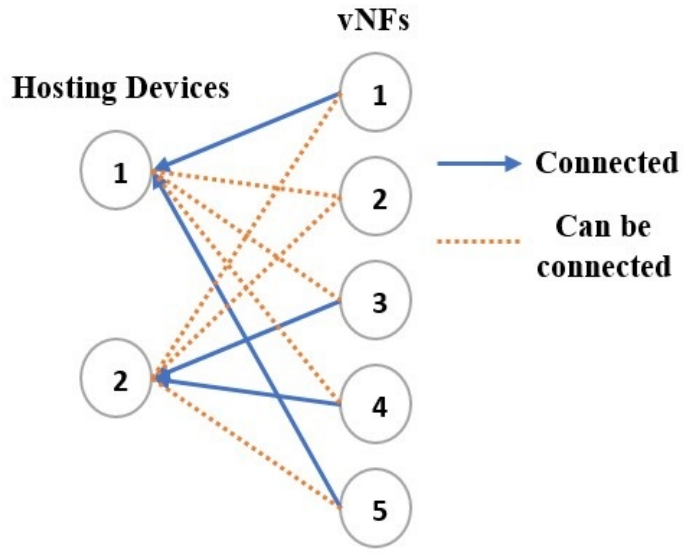
The problem proposed by R. Cziva et al. [8] can be categorized as an assignment problem. In this type of problem we want to find an efficient assignment of the vNF to the hosting devices to reduce the totalLatency generated by the network model. Many heuristic approaches were tried to solve it efficiently in polynomial time namely Hungarian Approach, Greedy approach and Stable matching approach.

This chapter has also provided a structured methodology and systematic evaluation of our proposed heuristics based on Stable Matching Algorithm and Greedy Algorithm. From all the above experimental solutions and working time comparisons it is clear that **Greedy Search** based algorithm was not suitable for this problem and our **Stable Matching** based algorithm is operating efficiently and performs close to the optimal ($8\% - 9\%$ more than the optimal latency). Though all the approaches gave feasible solutions but the best and most efficient (nearest to optimal) solution was provided by the Stable Matching Technique.

This part of the thesis has been selected in **The 10$^{\text{th}}$ International Conference on Ambient Systems, Networks and Technologies** and published in **Procedia Computer Science** journal [76].

3.6.1 Scope of Improvement

Let us take a case in which all the vNFs have been allocated to the hosting devices, but we can find that if the allocated connection can either be swapped or moved to get even better solutions.



vNFs	HD	
	1	2
1	2	1
2	7	9
3	2	3
4	1	5
5	2	8

Latency Matrix

Figure 3.10: **Example to Show that Improvement can be Done**

Figure. 3.10 shows an example in which vNFs 1 & 5 have been allocated to hosting device 1 and vNFs 3 & 4 have been allocated to hosting device 2. The total latency for the above configuration is 12 $(2+2+3+5)$.

Now using local search, first hosting device is picked and we check for connection (1,2) that is if the second vNF is allocated to first hosting device if it satisfies the capacity and latency constraints before allocation. This will be moving the connection, and the total latency will become 17$(7+2+3+5)$. Thus this pair will be discarded, and we move on to the next pair. We would not consider (1,3) & (1,4) as those vNFs are connected to second hosting device and we will check them with (1,1) once they are selected. (1,5) also would not be considered as it is already connected to first hosting device and it would not change total latency. Now we select connection (2,1), that is if the first vNF is allocated to the second hosting device if it satisfies the capacity and latency constraints before allocation. This again will be moving the connection, and the new total latency would be 11$(1+2+3+5)$ thus first vNF will now be an allocation to the second hosting device. This process will go on until all the pairs are checked for either swapping or moving.

Thus **Local Search** technique can be used to enhance the solutions, and the process is thoroughly explained in the next chapter.

Chapter 4

Extending SMA using Local Search

4.1 Overview

Though the algorithm discussed in the previous chapter was working efficiently, but as shown in the end of the previous chapter the solutions can be enhanced by extending Stable Match technique using the Local Search technique to make it more efficient. 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 a random connected pair and then checking it with other pairs and devices for finding even lower total latency if possible and then completes the process for all other pairs. Local search algorithm stops when there is no chance left for further improvement.

4.2 Algorithm

Algorithm 5 Local Search (Swapping\Moving)

1: **procedure** SWAP OR MOVE THE CURRENT MATCHED PAIRS TO FIND MORE EFFICIENT SOLUTION.
2: Using initial feasible solution from Algorithm 4.
3: Initialize improvement = true.
4: **while** (improvement) **do**
5: improvement = false.
6: **Checking for all connected pairs (i, j) and (i′, j′)**, where "i" vNF is connected to "j" hosting device and "i'" vNF is connected to "j'" hosting device. ▷ Case I
7: **if** ($l_{ij} + l_{i'j'} > l_{ij'} + l_{i'j}$ and Constraints 3.11 and 3.12 are satisfied) **then** ▷ Swapping
8: Assign vNF i to hosting device j' and vNF i' to hosting device j.
9: Update Capacity for hosting devices.
10: improvement = true.
11: **end if**
12: Check for other unconnected vNFs (i''). ▷ Case II
13: **if** ($l_{ij} > l_{i''j}$ and Constraints 3.11 and 3.12 are satisfied) **then** ▷ Moving
14: Assign vNF i'' to hosting device j and vNF i will get free.
15: Update Capacity for hosting devices.
16: improvement = true.
17: **end if**
18: Check for other unconnected hosting devices (j''). ▷ Case III
19: **if** ($l_{ij} > l_{ij''}$ and Constraints 3.11 and 3.12 are satisfied) **then** ▷ Moving
20: Assign vNF i to hosting device j''.
21: Update Capacity for hosting devices.
22: improvement = true.
23: **end if**
24: **end while**
25: Print New minimum totalLatency using current allocation.
26: **end procedure**

In this local search algorithm (5) 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 3.11 and 3.12, 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 $\mathcal{O}(n^2 * W)$.

4.3 Simulation Results

Scenario used for calculating these solutions is similar to that used in Greedy Approach but with more instances. The different instances that are used in this scenario are 50, 100, 500, 1000, 2000 and 3000 for vNFs. 5, 10, 15, 20, 50, 100, 150, 200, 250 and 300 are the different number of host devices which are then used to form different cases and use them to compare solutions for Opt (mathematical model), SMA and LS algorithms. In the solutions provided **Opt** is defined as Optimal (ILP Model Result), **SMA** is defined as Stable Match Algorithm and **LS** which is Local Search on top of Stable Match Algorithm. All of the simulation solutions presented in this section are an average of 10 different runs for a particular scenario.

The figures ahead illustrate us the comparison between the optimal solution presented as **Opt** by the mathematical (ILP) model, **SMA** (Stable Match) and **LS** (SMA with Local Search) on basis of **TL** (Total Latency) for different cases.

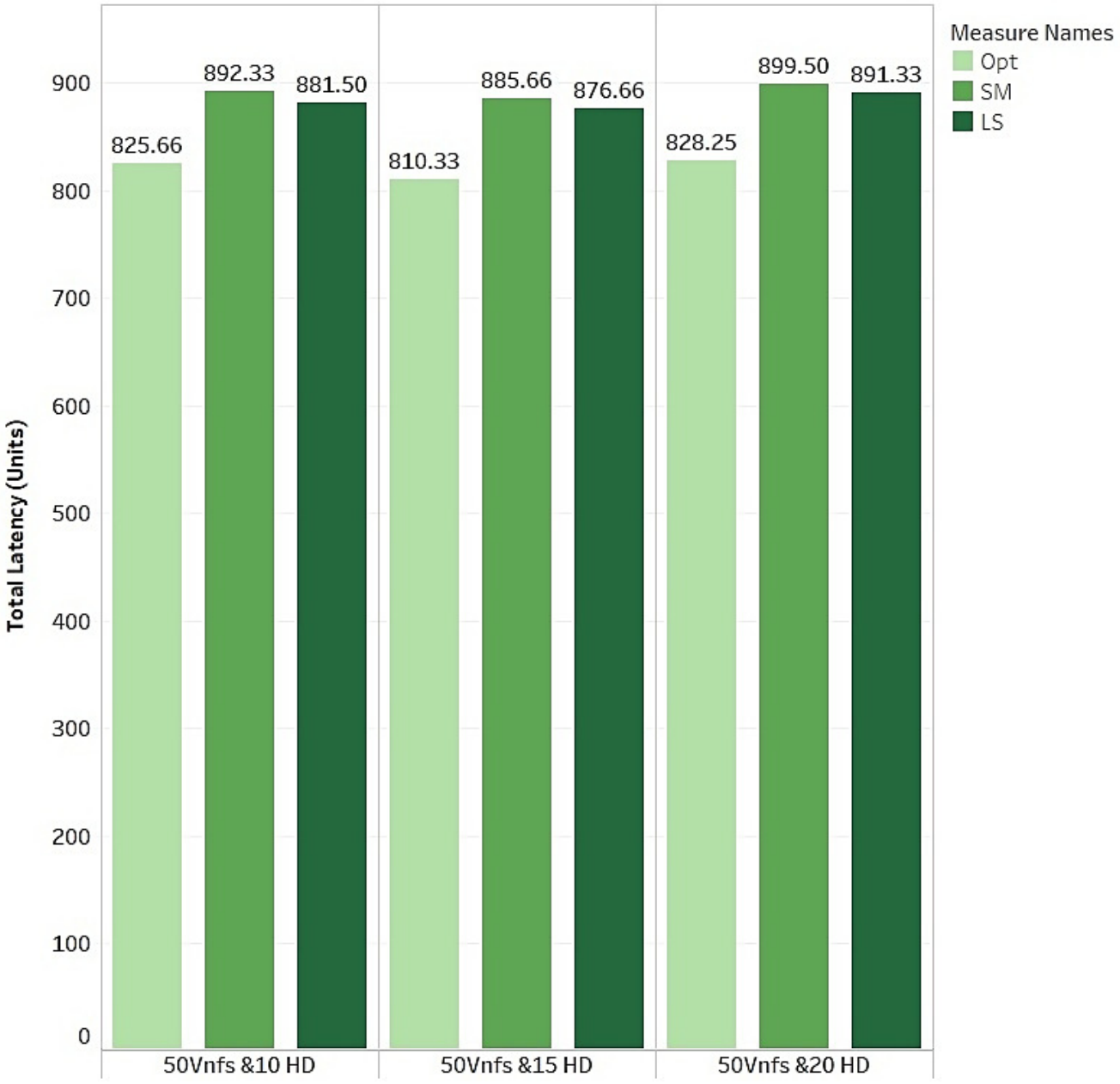


Figure 4.1: **Latency Result Comparisons Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 50 vNFs.**

Figure 4.1 shows the graphical comparison between the Optimal (ILP), Stable Matching algorithm and SMA with Local Search for 50 vNFs when we have a different number of hosting devices.

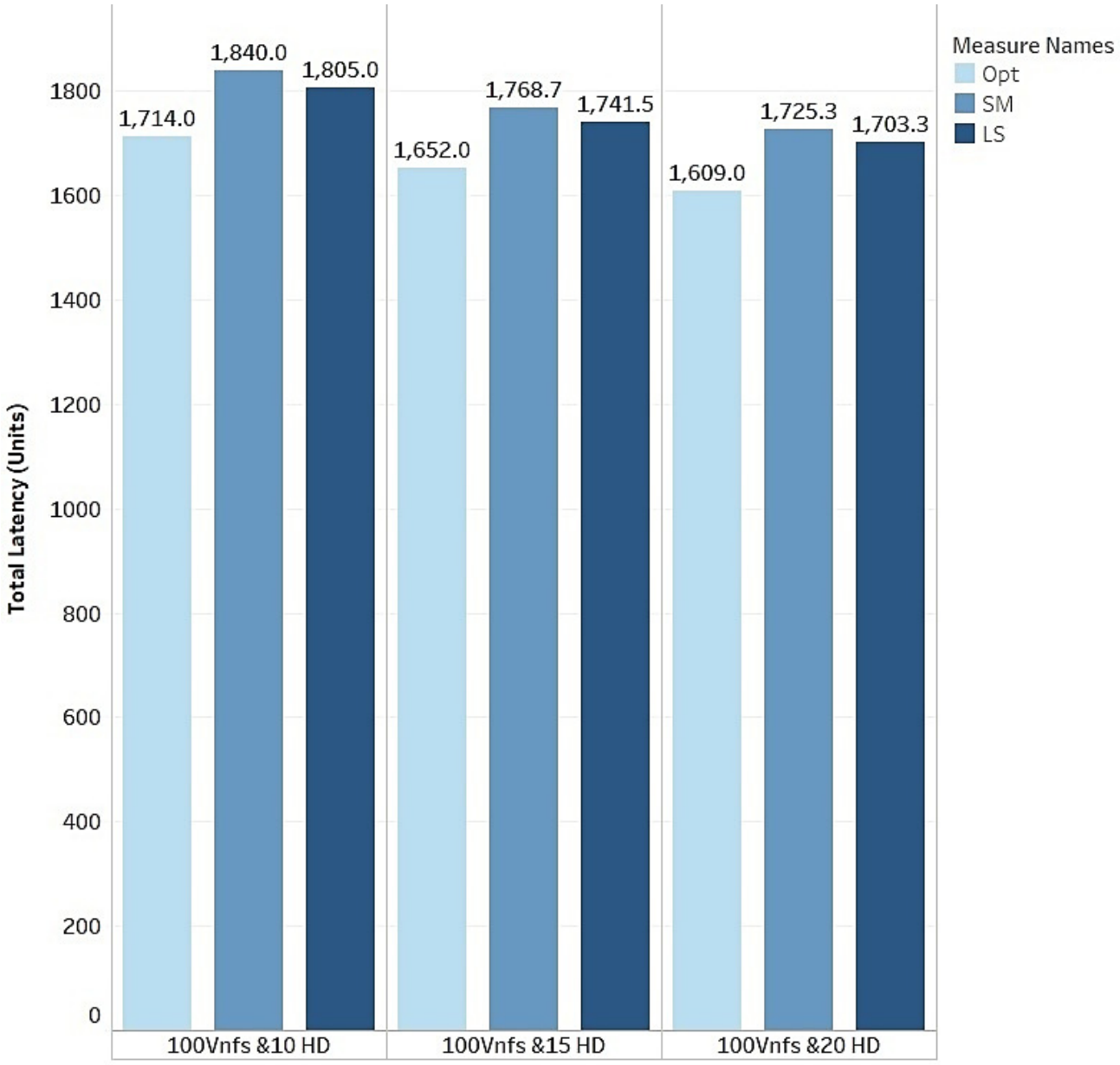


Figure 4.2: **Latency Result Comparisons Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 100 vNFs.**

Figure 4.2 shows the graphical comparison between the Optimal (ILP), Stable Matching algorithm and SMA with Local Search for 100 vNFs when we have a different number of hosting devices.

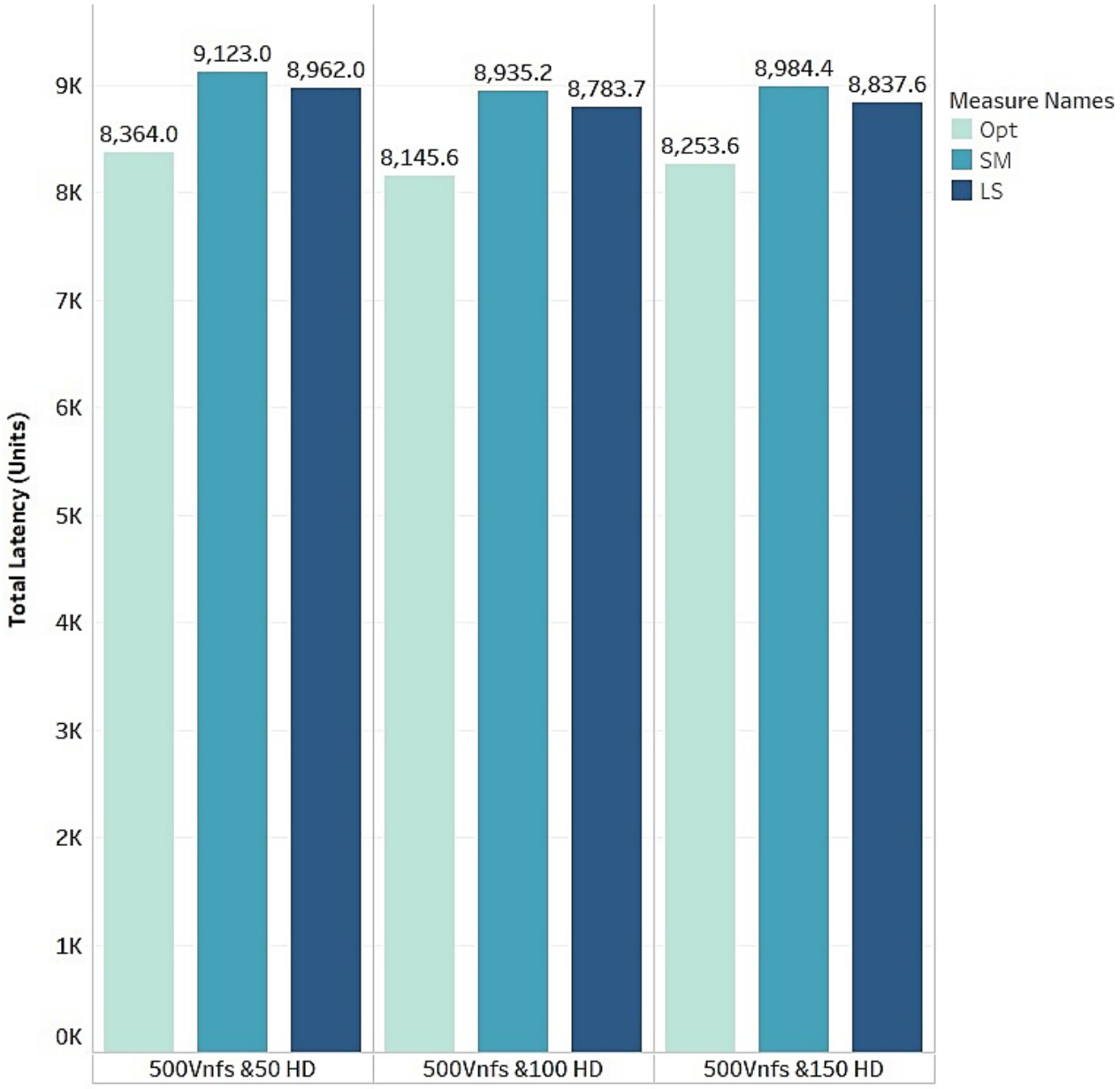


Figure 4.3: **Latency Result Comparisons Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 500 vNFs.**

Figure 4.3 shows the graphical comparison between the Optimal (ILP), Stable Matching algorithm and SMA with Local Search for 500 vNFs when we have a different number of hosting devices.

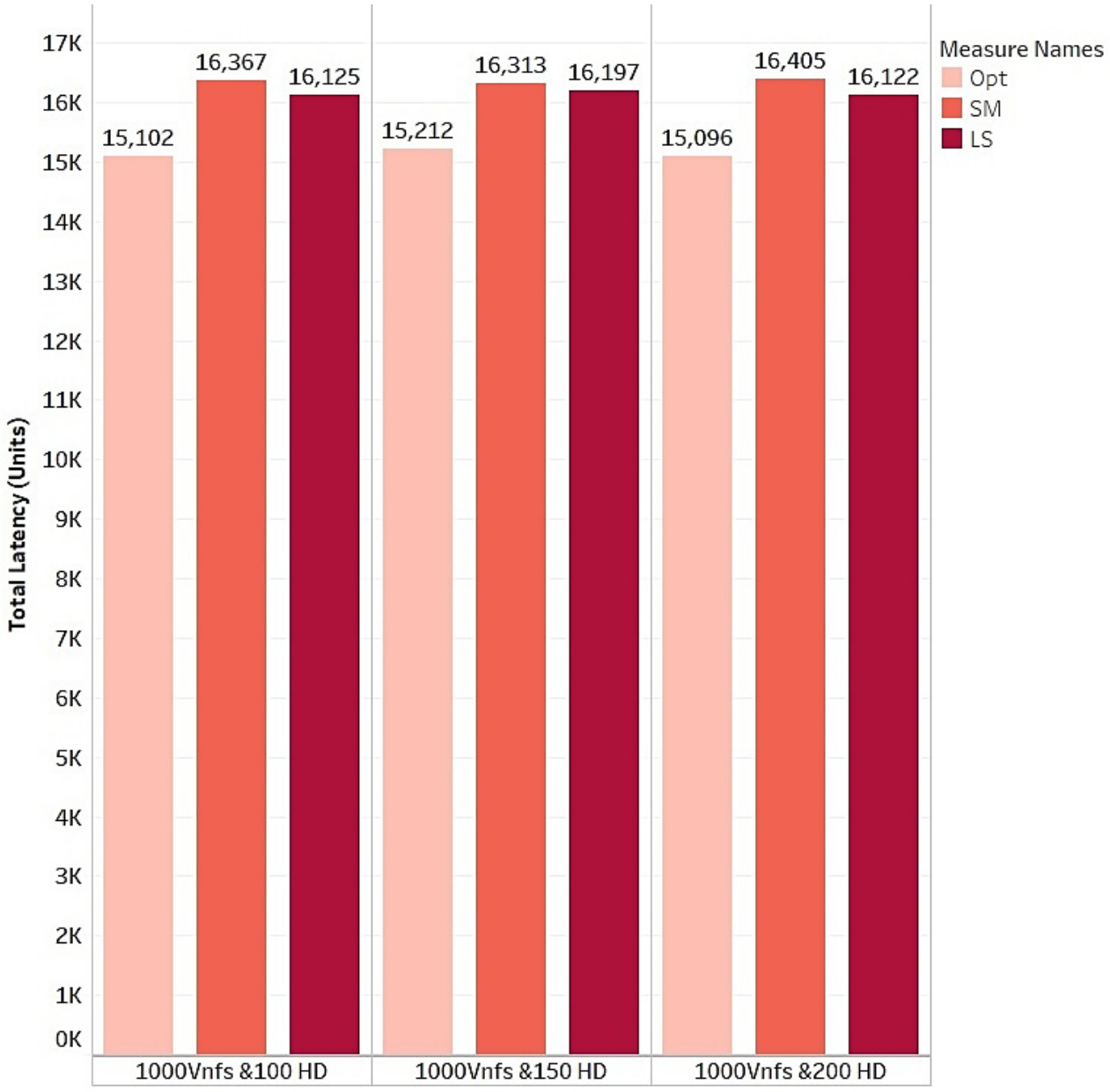


Figure 4.4: **Latency Result Comparisons Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 1000 vNFs.**

Figure 4.4 shows the graphical comparison between the Optimal (ILP), Stable Matching algorithm and SMA with Local Search for 1000 vNFs when we have a different number of hosting devices.

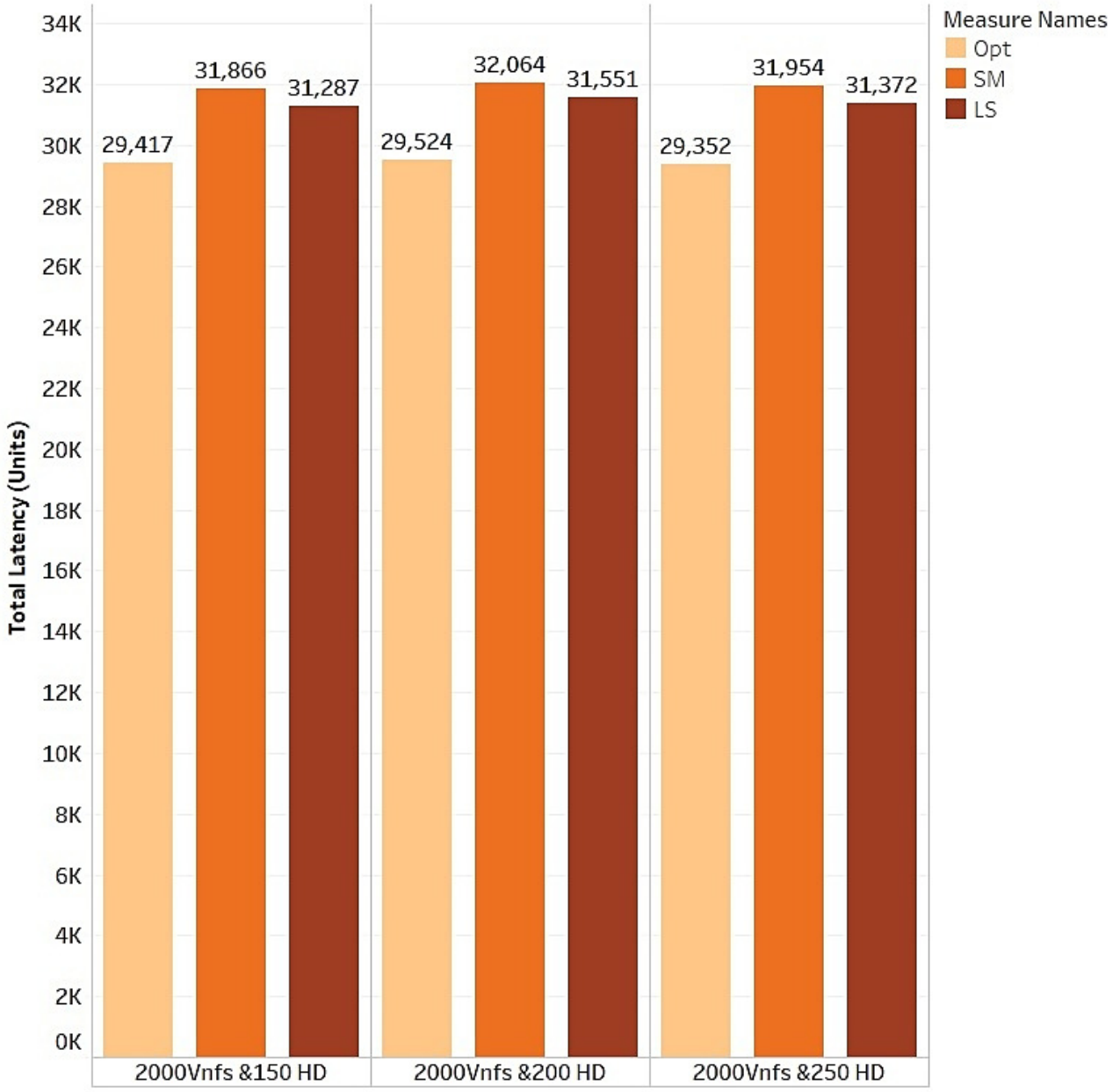


Figure 4.5: **Latency Result Comparisons Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 2000 vNFs.**

Figure 4.5 depicts the comparison between the Optimal (ILP), Stable Matching algorithm and SMA with Local Search when we have 2000 vNFs and various number of hosting devices.

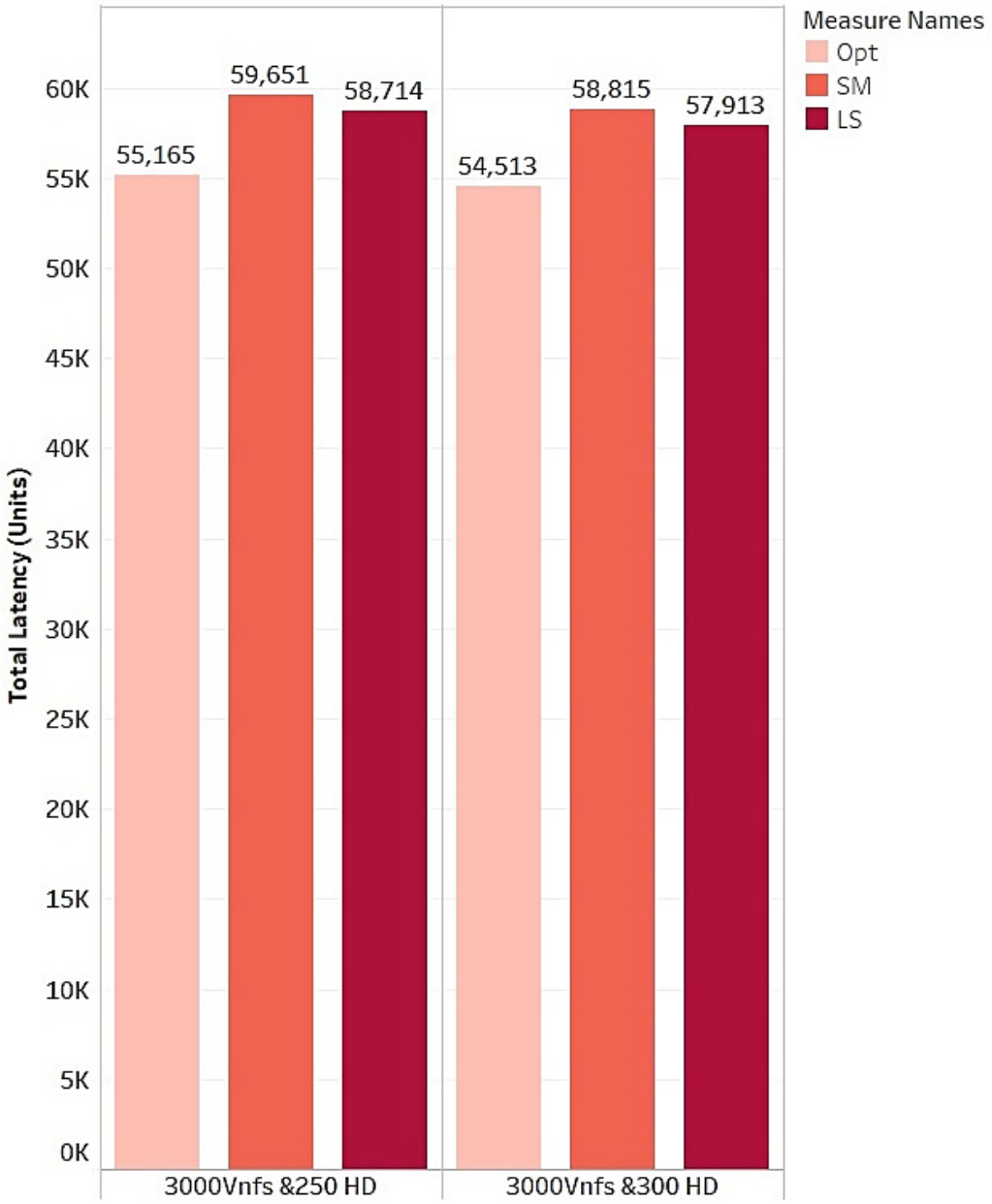


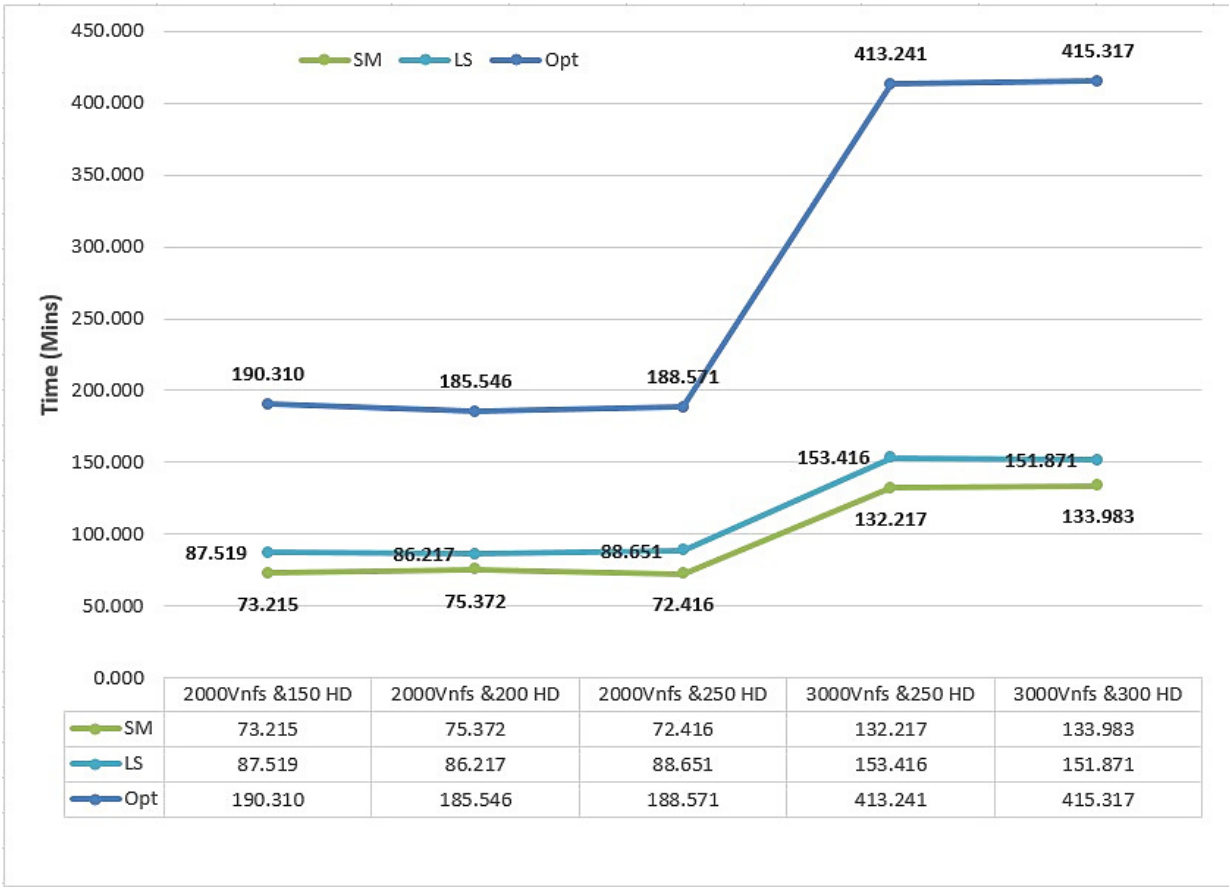
Figure 4.6: **Latency Result Comparisons Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 3000 vNFs.**

Figure 4.6 depicts the comparison between the Optimal (ILP), Stable Matching algorithm and SMA with Local Search when we have 3000 vNFs and various number of hosting devices.

	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

Figure 4.7: **Time (Min) Comparison Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 500 & 1000 vNFs.**

From the above figure it is clear that the time taken by the model (Opt) to solve the problem increases exponentially with the increase in the number of vNFs and hosting devices. But the proposed algorithms do not illustrate the similar kind of behaviour.



	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

Figure 4.8: **Time (Min) Comparison Between Optimal (ILP), Stable Matching and Stable Match with Local Search for 2000 & 3000 vNFs.**

From the above figure it is clear that the time taken by the model (Opt) to solve the problem increases exponentially with the increase in the number of vNFs and hosting devices.

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

Table 4.1: **Working Time Comparison (Seconds) & Latency Comparisons**

Table 4.1 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 to 30 percent 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 to 8 % more latency compared to the optimal solutions.

4.4 Conclusion

Local Search technique works by starting with an initial feasible solution and then tends to improve it with each iteration. The analysis of the solutions provided by the Stable Matching algorithm illustrated a scope of improvement and thus an extension based on local search technique was tried to obtain even more efficient solutions.

This chapter has provided a structured methodology and systematic evaluation of our proposed extension based on Local Search after finding a solution by Stable Matching Algorithm. The solution comparison done is between the minimum latencies and time taken by the optimal solution, stable match, and stable match with the local search for 50, 100, 500, 1000, 2000 and 3000 vNFs (different number of host devices (10, 15, 20, 50, 100, 150, 200, 250, 300)). Considering all experimental solutions, it is clear that the stable match algorithm performs very close to the optimal ($8\% - 9\%$ more than the optimal latency). However, when the local search is added, an even better solution is achieved ($6\% - 7\%$ more than the optimal latency).

It can be mentioned here that the best solution for the problem can only be given by the mathematical (ILP) model and the aim of our research is to go as close as to the optimal solution as possible. Lesser the difference (%) between the heuristic approach solution and the optimal solution, better is the performance or efficiency of the heuristic.

Chapter 5

Fair Allocation Problem for Allocating the vNFs

5.1 Overview

In this chapter, we introduce a new problem in which we try to balance the latencies for each connection in which vNFs connect to the hosting devices. In the previous chapters, we gave the algorithms just to minimize the total latency, but they don't care about the fair allocation of the vNFs or the hosting devices. The main reason being that they only minimize the latency without considering any latency imbalance that can be there. Fairness is one of the main issues in networking domain [60] that needs to be checked before moving forward with any of the networking implementations.

This is the main reason behind this chapter, and in this chapter, we propose an ILP model for the fair allocation of the vNFs to hosting devices. Then we propose an algorithmic solution to increase the fairness for the connections using the Local Search technique on top of the solutions obtained by the Stable Matching algorithm.

5.2 ILP Formulation

5.2.1 Parameters Used

Variable	Description
$\mathbb{N}$	Total number of vNFs
$\mathbb{H}$	Total number of vNF hosting devices
$\mathbb{U}$	Total number of users
C_j	Maximum capacity of a hosting device j.
R_i	Requirement of vNF i.
MaxL_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.

Table 5.1: **Parameters**

We consider a system with vNFs and hosting devices, where $\mathbb{N} = \{n_1, n_2, n_3, ..., n_i\}$ 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* ($\mathbf{MaxL_i}$) that denotes the maximum latency which vNF n_i can tolerate. Similarly $\mathbb{H} = \{h_1, h_2, h_3, ..., h_j\}$ 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 its own capacity ($\mathbf{C_j}$) properties CPU, memory, IO. $\mathbf{l_{ij}}$ gives the latency between the user of the n_i vNF in case the vNF is located at h_j.

5.2.2 ILP Model

We propose an ILP model to solve the above mentioned problem. The objective of the model will be to minimize the maximum selected latency, this will be an min max

model.

$$Minimize\ (Max\ (x_{ij} * l_{ij})) \tag{5.1}$$

Subject To-

$$\sum\nolimits_{n_i \in \mathbb{N}} x_{ij} * R_i \leq C_j, \forall h_j \in \mathbb{H} \tag{5.2}$$

$$\sum\nolimits_{h_j \in \mathbb{H}} x_{ij} l_{ij} \leq \mathrm{MaxL}_i, \forall n_i \in \mathbb{N} \tag{5.3}$$

$$\sum\nolimits_{n_i \in \mathbb{N}} \sum\nolimits_{h_j \in \mathbb{H}} x_{ij} = M \tag{5.4}$$

$$\sum\nolimits_{h_j \in \mathbb{H}} x_{ij} \leq 1, \forall n_i \in \mathbb{N} \tag{5.5}$$

where "M" is the total number of devices which can be connected and it is calculated in similar fashion as done in vNF allocation model using 3.7.

To formulate this problem we use the same constraints (Eq 3.1, Eq 3.2, Eq 3.3, Eq 3.4 & Eq 3.5) as used in the previous problem as we want the same conditions for the allocation of the vNFs to hosting devices but with finding lesser maximum selected latency as objective of the model. Thus only the objective function is changed, which minimize the maximum of the latency. The main work of the objective function is first to maximize the selected latency for the allocated vNFs to hosting devices and then minimizing that latency value. Then we calculate the fairness measure for the outcome generated by the model.

5.3 Fairness Measure

Fairness Measure is the metric which is used in network engineering to check the fairness of the resource sharing of a network model. It is calculated using **Raj Jain's** equation [62] which is as follows:

$$\mathbb{J}(x_1, x_2, ..., x_n) \ = \ \frac{(\sum_{i=1}^{n} x_i)^2}{n * \sum_{i=1}^{n} x_i^2} \tag{5.6}$$

It rates the fairness of a set of values where there are $\mathbf{n}$ users (vNFs), $\mathbf{x_i}$ is the throughput (latency) for the $\mathbf{i^{th}}$ connection. The result ranges from $\frac{\mathbf{1}}{\mathbf{n}}$ (worst case) to $\mathbf{1}$ (best case), and it is maximum when all users receive the same allocation.

We are using fairness measure as only a metric to compare fairness for the mathematical model and the heuristic approach. The model discussed in the previous

section is only minimizing the maximum selected latency not maximizing fairness. If we want to maximize the fairness using fairness measure we need to modify the mathematical model to make a multi-optimization model which will minimize the maximum selected latency and maximize the fairness simultaneously.

5.4 Proposed Heuristic

5.4.1 Local Search Algorithm

Algorithm 6 Using Local Search after SMA to minimize ML and increase FM

procedure SWAPPING OR MOVING MATCHED PAIRS TO MINIMIZE ML AND INCREASE FM
    Using initial solution obtained by SMA.
    Initialize improvement = true.
    while (improvement) **do**
        improvement = false
        Pick connected pair (i, j) with maximum latency, where "i" vNF is connected to "j" hosting device.
        Check for all other connected pairs (i', j'). ▷ Case I
        if ($Max(l_{ij}$ & $l_{i'j'}) > Max(l_{i'j}$ & $l_{ij'})$ and Constraints 3.11 and 3.12 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
        Checking for other unconnected vNFs (i''). ▷ Case II
        if ($l_{ij} > l_{i''j}$ and Constraints 3.11 and 3.12 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
        Checking for other unconnected hosting devices (j''). ▷ Case III
        if ($l_{ij} > l_{ij''}$ and Constraints 3.11 and 3.12 are satisfied) **then** ▷ Moving
            Assign vNF i to hosting device j''.
            Update Capacity for hosting devices and **improvement** = true.
        end if
    end while
    Print New maximum latency.
    Calculate and print**Fairness Measure** for new allocations.
end procedure

In the above algorithm (6), we use the solution given by Stable Matching algorithm similar to algorithm 4 as the initial feasible solution for this algorithm, where **ML** is maximum latency and **FM** is Fairness Measure. The algorithm picks the connected pair *with maximum latency value* and then it is compared (8) with all the other connected pairs. We even compare the selected pair with all the unpaired vNFs (14) and hosting devices (20). IF the comparison leads to improvement (reduction) in the total latency and they satisfy the constraints 3.11 and 3.12, then the connection is either swapped or moved. The improvement is calculated as follows:

- For **Case I** (**Swapping**), we find the largest value of latency between the selected pair and pair to be checked. It is compared to the largest value of latency between the swapped pairs. If the value of latency (maximum latency) for the swapped pairs is lesser, it can be said that there is an improvement in solution.
- 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.

This process is done until there is no scope of the improvement. New maximum latency for the updated allocations is supplied as a solution. Fairness Measure is calculated for the updated allocations and is also supplied as a result.

5.5 Results

The scenario used for calculating these solutions is similar to that used in Greedy Approach but with more instances. The different instances that are used in this scenario are 20, 30, 50, 100 and 200 for vNFs. 5, 10, 15, 20 and 50 are a different number of host devices which are then used to form different cases and use them to compare solutions for VA Model **(vNF Allocation Model)**, FA Model **(Fair Allocation Model)** and SMA & LS Algorithm **(Stable Matching along with Local Search Algorithm)**. All of the simulation solutions presented in this section are an average of 10 different runs for a particular scenario.

vNFs	Hosting Devices	Maximum Latency (ML)		Fairness Measure (FM)	
		VA Model	FM Model	VA Model	FM Model
50 vNFs	**10 HD**	24	22	0.935	0.985
	15 HD	23	23	0.897	0.946
	20 HD	23	21	0.921	0.974
100 vNFs	**10 HD**	25	23	0.981	0.983
	15 HD	22	22	0.987	0.989
	20 HD	22	21	0.965	0.976
200 vNFs	**20 HD**	21	21	0.969	0.976
	50 HD	23	21	0.973	0.978
500 vNFs	**50 HD**	24	22	0.989	0.994
	100 HD	22	21	0.951	0.977
	150 HD	23	22	0.987	0.991

Table 5.2: **Comparison between the Results given by VA Model and FM Model**

Table 5.2 shows the solution comparison between VA Model and SMA & LS Algorithm on basis of **Maximum Latency** and **Fairness Measure**. Where VA is **vNF Allocation Model** and FA is **Fair Allocation Model**.

vNFs	Hosting Devices	Maximum Latency (ML)		Fairness Measure (FM)	
		FM Model	SM & LS	FM Model	SM & LS
50 vNFs	10 HD	22	23	0.985	0.932
	15 HD	23	24	0.946	0.913
	20 HD	21	25	0.974	0.916
100 vNFs	10 HD	23	24	0.983	0.976
	15 HD	22	23	0.989	0.961
	20 HD	21	24	0.976	0.954
200 vNFs	20 HD	21	23	0.976	0.953
	50 HD	21	25	0.978	0.969
500 vNFs	50 HD	22	23	0.994	0.976
	100 HD	21	22	0.977	0.949
	150 HD	22	24	0.991	0.979

Table 5.3: **Comparison between the Results given by FM Model and SM & LS**

Table 5.3 shows the solution comparison between VA Model and SMA & LS Algorithm on basis of **Maximum Latency** and **Fairness Measure**.

5.6 Conclusion

In this chapter, we define a new problem based on vNF allocation problem proposed in [8]. The main reason behind doing this was that, the vNF allocation problem only dealt with minimizing the total latency for a network model without worrying about the fair allocation of the resources. From the solutions, it is clear that the mathematical model created for the newly proposed fair allocation problem is working fine and giving desired solutions.

The newly defined problem is also NP-hard and takes exponential time in worst case scenario. Then a heuristic is provided to solve the problem in polynomial time. As the optimal (ILP Model) solution given by newly defined fair allocation problem

are either similar or better in some cases than the ones given by the predefined vNF allocation problem. We decided to try the stable match approach to solve the problem in polynomial time as done in chapter 3, the solutions were good but not that close to the optimal provided by the FA Model and thus the Local Search was tried as an extension to make the solutions more efficient.

From the table 5.2 of solutions, it is clear that the ILP model proposed for the problem is working well and giving efficient solutions. The model is improving the maximum latency and fairness measure. It is also clear from solutions that as the size of the network increases, both the models proceed towards similar solutions. According to this information, we can state that at some point both the models would give the same solution.

For our proposed algorithm the solutions show that it is working fine and gives better solutions than the VA Model for initial cases and when the size of the network increases the algorithm proceeds towards giving equivalent solutions though they are a bit less than the ILP models.

Chapter 6

Conclusion & Future Work

6.1 Overview

The approaches proposed in this thesis has shown good overall accuracy with room for improvement. The findings of this research can be used in future for solving similar problems in polynomial time.

6.2 Main Contributions

This research addresses the question of minimizing the total latency of a given network or model when the vNFs are assigned to the hosting devices. In this research, we assign vNFs to hosting devices in such a way that we can get minimum network latency. This problem was initially defined by R. Cziva et al. [8]; they proposed an mathematical (ILP) model to solve the problem. This is an assignment problem which can be solved using various approaches.

There are three key contributions of this thesis:

- **First** a problem proposed by [8] is analyzed, as there are some anomalies in the problem formulation. So, it is modified to make it more efficient; this is done using another problem which is categorized as multiple knapsack problem with assignment restriction problem [9]. The mathematical model for this problem is an Integer Linear Programming (ILP) model, and it is implemented in CPLEX.
- **Second** as both the given problem and modified one are **NP-hard** problems and they take exponential time in the worst case. No heuristic has been provided till now to solve them in polynomial time. A heuristic approach based on Stable Match technique has been provided in this thesis to solve the problem polynomially. Further Local Search has been used to enhance the solution given by SMA and make the proposed heuristic more efficient.
- **Third** we define a new problem (Fair Allocation Problem) to deal with the fair allocation of the vNFs in the vNF allocation problem but it is also a **NP-hard** problem. A mathematical model has been proposed and then a heuristic has been provided to solve the newly defined problem in polynomial time.

6.3 Conclusion

In this research, we gave algorithmic approaches to solve an assignment problem to minimize the end-to-end latency of edge NFV. The problem is an NP-hard one. The original problem statement defined in the research paper [8] had some technical drawbacks. Thus it was further enhanced to overcome those difficulties and make it more general. Our proposed algorithm is based on stable matching and then increasing efficiency using a local search technique. The solutions for optimal latency and working times are used for comparison between the techniques proposed.

The IBM CPLEX Solver is used for solving the ILP model. According to the experimental results, it is clear that our proposed heuristic approach is working efficiently as it is giving us solutions which are approximately $8\% - 9\%$ more for only stable matching and $6\% - 7\%$ more when the local search is used on top of the stable match algorithm than the optimal latency. Therefore, we can state that our proposed algorithmic approach can be used to solve the given problem efficiently (close to the optimal) in polynomial time.

6.4 Future Work

There can be many prospects using the research done in this thesis. One of which can be to design an 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). The change can be the result of various scenarios, mainly:

1. If a new vNF is introduced in the topology.

2. If a vNF leaves the topology or gets wrecked.

A similar type of problem has been defined in [61]; 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.

The paper mentioned above though, gives an ILP model to solve the problem. Designing an efficient heuristic is an interesting research topic.

Another future work related to the second part of the thesis can be designing a mathematical multi-optimization model to minimize the maximum selected latency and maximize the fairness simultaneously.

Appendix A

List of Abbreviations

- **vNF** Virtual Network Function
- **M2M** Machine to Machine
- **SOA** Service Oriented Architecture
- **P2P** Peer-to-Peer
- **IT** Information Technology
- **IaaS** Information as a Service
- **PaaS** Platform as a Service
- **SaaS** Software as a Service
- **IoT** Internet-of-Things
- **NFV** Network Functions Virtualization
- **VM** Virtual Machines
- **SDN** Software Defined Networking
- **RBAC** Role-based Access Control
- **ILP** Integer Linear Programming
- **NP-hard** Non-Deterministic Polynomial-Time Hardness
- **SMP** Stable Matching Problem

- **LS** Local Search
- **ML** Maximum Latency
- **TL** Total Latency
- **Opt** Optimal Solution
- **GA** Genetic Algorithm
- **GrA** Greedy Approach
- **VM** Virtual Machine
- **SFC** Service Function Chain
- **AC** Admission Control
- **LDCs** Large-scale Data Centres
- **ADMM** Alternating Direction Method Multipliers
- **vNF-SCs** vNF Service Chains
- **EONs** Elastic Optical Networks
- **DCN** Data Center Networks
- **vNF-P** vNF-Placement
- **LRR** Local Regression Robust
- **RSM** Replication Stable Matching
- **TSP** Traveling Salesman Problem
- **ACA** Ant Colony Algorithm
- **DLACA** Dynamic Local Search based Ant Colony Algorithm
- **MILP** Mixed Integer Linear Programming
- **HD** Hosting Device
- **SM** Stable Match

- **SMA** Stable Matching Algorithm
- **FM** Fairness Measure
- **SOCP** Second-Order Cone Programming
- **OPL** Optimization Programming Language
- **IDE** Integrated Development Environment
- **CSV** Comma Separated Value
- **GS** Gale-Shapley Algorithm
- **AG** Auxilary Graph
- **DP** Dynamic Programming
- **WSN** Wireless Sensor Network

Bibliography

[1] Network Functions Virtualisation - Introductory White Paper, doi:http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[2] C. Bouras, A. Kollia and A. Papazois, "SDN & NFV in 5G: Advancements and challenges," 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, 2017, pp. 107-111. doi: 10.1109/ICIN.2017.7899398

[3] F. Hu, M. Qiu, J. Li, T. Grant, D. Tylor, S. McCaleb, L. Butler, R. Hamner,"A Review on Cloud Computing: Design Challenges in Architecture and Security", Journal of Computing and Information Technology - CIT 19, pp. 25-55, 2011. doi: https://doi.org/10.2498/cit.1001864

[4] W. Chu, "NFV and NFV-based security services," 2018. doi: 10.1002/9781119293071.ch15

[5] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021, Cisco White Paper, 2017.

[6] L. Ma, X. Wen, L. Wang, Z. Lu and R. Knopp, "An SDN/NFV based framework for management and deployment of service based 5G core network," in China Communications, vol. 15, no. 10, pp. 86-98, Oct. 2018. doi: 10.1109/CC.2018.8485472

[7] M. Shin, S. Lee, S. Lee and D. Kim, "A way forward for accommodating NFV in 3GPP 5G systems," 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2017, pp. 114-116. doi: 10.1109/ICTC.2017.8190953

[8] R. Cziva and D. P. Pezaros, "On the Latency Benefits of Edge NFV," 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Beijing, 2017, pp. 105-106. doi: 10.1109/ANCS.2017.23

[9] M. Dawande, J. Kalagnanam, P. Keskinocak, F.S. Salman, R. Ravi, "Approximation Algorithms for the Multiple Knapsack Problem with Assignment Restrictions" in Journal of Combinatorial Optimization, vol. 4, no. 2, pp. 171-186, June 2000. doi: 10.1023/A:1009894503716

[10] A. Manzalini and R. Saracco, "Software Networks at the Edge: A Shift of Paradigm," 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Trento, 2013, pp. 1-6.doi: 10.1109/SDN4FNS.2013.6702555

[11] R. Cziva and D. P. Pezaros, "Container Network Functions: Bringing NFV to the Network Edge," in IEEE Communications Magazine, vol. 55, no. 6, pp. 24-31, 2017.doi: 10.1109/MCOM.2017.1601039

[12] H. Moens and F. D. Turck, "vNF-P: A model for efficient placement of virtualized network functions," 10th International Conference on Network and Service Management (CNSM) and Workshop, Rio de Janeiro, 2014, pp. 418-423.

[13] M. A. Tahmasbi Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi and B. H. Khalaj, "vSPACE: vNF Simultaneous Placement, Admission Control and Embedding," in IEEE Journal on Selected Areas in Communications, vol. 36, no. 3, pp. 542-557, March 2018. doi: 10.1109/JSAC.2018.2815318

[14] F. Tashtarian, A. Varasteh, A. Montazerolghaem and W. Kellerer, "Distributed vNF scaling in large-scale datacenters: An ADMM-based approach," 2017 IEEE 17th International Conference on Communication Technology (ICCT), Chengdu, China, 2017, pp. 471-480. doi: 10.1109/ICCT.2017.8359682

[15] W. Lu, L. Liang and Z. Zhu, "On vNF-SC deployment and task scheduling for bulk-data transfers in inter-DC EONs," 2017 IEEE/CIC International Conference on Communications in China (ICCC), Qingdao, 2017, pp. 1-4. doi: 10.1109/ICCChina.2017.8330366

[16] H. Zhu and C. Iluang, "vNF-B&B: Enabling edge-based NFV with CPE resource sharing," 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, 2017, pp. 1-5. doi: 10.1109/PIMRC.2017.8292421

[17] J. Kong et al., "Guaranteed-Availability Network Function Virtualization with Network Protection and vNF Replication," GLOBECOM 2017 - 2017 IEEE

Global Communications Conference, Singapore, 2017, pp. 1-6. doi: 10.1109/GLOCOM.2017.8254730

[18] V. Nikam, J. Gross and A. Rostami, "vNF service chaining in optical data center networks," 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, 2017, pp. 1-7. doi: 10.1109/NFV-SDN.2017.8169845

[19] D. Cho, J. Taheri, A. Y. Zomaya and L. Wang, "Virtual Network Function Placement: Towards Minimizing Network Latency and Lead Time," 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, 2017, pp. 90-97. doi: 10.1109/CloudCom.2017.12

[20] J. Gil Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," in IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 518-532, Sept. 2016. doi: 10.1109/TNSM.2016.2598420

[21] H. Moens and F. D. Turck, "vNF-P: A model for efficient placement of virtualized network functions," 10th International Conference on Network and Service Management (CNSM) and Workshop, Rio de Janeiro, 2014, pp. 418-423. doi: 10.1109/CNSM.2014.7014205

[22] S. Luz, M. Masoodian, D. McKenzie and W. V. Broeck, "Chronos: A Tool for Interactive Scheduling and Visualisation of Task Hierarchies," 2009 13th International Conference Information Visualisation, Barcelona, 2009, pp. 241-246. doi: 10.1109/IV.2009.88

[23] R. Ford, M. Zhang, M. Mezzavilla, S. Dutta, S. Rangan and M. Zorzi, "Achieving Ultra-Low Latency in 5G Millimeter Wave Cellular Networks," in IEEE Communications Magazine, vol. 55, no. 3, pp. 196-203, March 2017. doi: 10.1109/MCOM.2017.1600407CM

[24] J. V. Wang, K. Fok, C. Cheng and C. K. Tse, "A Stable Matching-Based Virtual Machine Allocation Mechanism for Cloud Data Centers," 2016 IEEE World Congress on Services (SERVICES), San Francisco, CA, 2016, pp. 103-106. doi: 10.1109/SERVICES.2016.21

[25] X. Tang, D. Hong and W. Chen, "Content Replication Scheme Using Stable Matching in Vehicular Networks," 2016 International Conference on Identification,

Information and Knowledge in the Internet of Things (IIKI), Beijing, 2016, pp. 351-356. doi: 10.1109/IIKI.2016.18

[26] Q. Chu, L. Cui and Y. Zhang, "Joint Computing and Storage Resource Allocation Based on Stable Matching in Data Centers," 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), Beijing, 2017, pp. 207-212. doi: 10.1109/BigDataSecurity.2017.36

[27] B. Dengiz, F. Altiparmak and A. E. Smith, "Local search genetic algorithm for optimal design of reliable networks," in IEEE Transactions on Evolutionary Computation, vol. 1, no. 3, pp. 179-188, Sept. 1997. doi: 10.1109/4235.661548

[28] H. Qin, S. Zhou, L. Huo and J. Luo, "A New Ant Colony Algorithm Based on Dynamic Local Search for TSP," 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, 2015, pp. 913-917. doi: 10.1109/CSNT.2015.241

[29] Laborie P, Rogerie J, Shaw P, Vilim P (2018). "IBM ILOG CP optimizer for scheduling". Constraints. 23 (2): 210250. doi:10.1007/s10601-018-9281-x.

[30] S. Sugimoto, T. Hattori, T. Izumi and H. Kawano, "Fast Kansei Matching Method as an Algorithm for the Solution of Extended Stable Marriage Problem," 2009 International Conference on Biometrics and Kansei Engineering, Cieszyn, 2009, pp. 209-214. doi: 10.1109/ICBAKE.2009.55

[31] T. Pino, S. Choudhury and F. Al-Turjman, "Dominating Set Algorithms for Wireless Sensor Networks Survivability," in IEEE Access, vol. 6, pp. 17527-17532, 2018. doi: 10.1109/ACCESS.2018.2819083

[32] Y. Yong and H. Guang, "Research on the Vehicle Stowage Problem Based on Stable Matching Theory," 2013 International Conference on Computer Sciences and Applications, Wuhan, 2013, pp. 442-445. doi: 10.1109/CSA.2013.110

[33] V. Bansal, A. Agrawal, V.S. Malhotra, "Stable Marriages with Multiple Partners: Efficient Search for an Optimal Solution", in Baeten J.C.M., Lenstra J.K., Parrow J., Woeginger G.J. (eds) Automata, Languages and Programming. ICALP 2003. doi: 10.1007/3-540-45061

[34] D. Gale, L. S. Shapley (1962), "College Admissions and the Stability of Marriage," The American Mathematical Monthly, 69:1, 9-15. doi: 10.1080/00029890.1962.11989827

[35] D.G. McVitie, L.B. Wilson, "The Stable Marriage Problem". Commucations of the ACM, Vol 114, (1971) 486492. doi: 10.1145/362619.362631

[36] R.W. Irving, P. Leather, D. Gusfield, "An Efficient Algorithm for the "Optimal Stable Marriage". Journal of the ACM, Vol 34(3), (Jul 1987) 532543. doi: 10.1145/28869.28871

[37] A. Roth and M. Sotomayor, "Two-sided Matching: A Study in Game-Theoretic Modeling and Analysis", Econometrica Society Monographs, Vol. 18, Cambridge University Press, 1990.

[38] A. Roth, "Stability and Polarization of Interests in Job Matching", Econometrica, Vol 52, (1984) 4757

[39] M. Sotomayor, "The Lattice Structure of the Set of Stable Outcomes of the Multiple Partners Assignment Game," International Journal of Game Theory, Vol 28, (1999) 567583.

[40] R. Martinez, J. Masso, A. Neme, J. Oviedo, "An Algorithm to Compute the Set of Many-to-many Stable Matchings," Mathematical Social Sciences, 2001.

[41] A. Alkan, "On Preferences over Subsets and the Lattice Structure of Stable Matchings", Review of Economic Design, Vol 6, (2001) 99111.

[42] A. Alkan, "A class of Multipartner Matching Markets with a Strong Lattice Structure", Economic Theory, Vol 19(4), (2002) 737746.

[43] E. Gupta and N. Nitin, "Stable match approach to determine vital link to preserve shortest path length," International Conference on Computing, Communication & Automation, Noida, 2015, pp. 408-413. doi: 10.1109/CCAA.2015.7148410

[44] H.W. Kuhn, "The Hungarian method for the assignment problem," Naval Research Logistics, 2: 83-97, 1955. doi: 10.1002/nav.3800020109

[45] J. Munkres, "Algorithms for the Assignment and Transportation Problems", Journal of the Society for Industrial and Applied Mathematics, 5(1):3238, 1957 March. doi: 10.1137/0105003

[46] H.W. Kuhn, "Variants of the Hungarian method for assignment problems", Naval Research Logistics Quarterly," 3: 253258, 1956.

[47] "Finding perfect matchings in bipartite hypergraphs", https://epubs.siam.org/doi/abs/10.1137/1.9781611974331.ch126

[48] R. Su & L. Zhou, & J. Tang, "Locomotive Schedule Optimization for Da-qin Heavy Haul Railway," Mathematical Problems in Engineering, 2015, 1-14, doi: 10.1155/2015/607376.

[49] W. Zhu and C. Guo, "A Local Search Approximation Algorithm for Max-k-Cut of Graph and Hypergraph," 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming, Tianjin, 2011, pp. 236-240. doi: 10.1109/PAAP.2011.35

[50] S. Alqallaf, M. Almulla, L. Niepel and M. Newborn, "Hybrid local search approximation algorithm for solving the capacitated Max-K-cut problem," 2015 2nd World Symposium on Web Applications and Networking (WSWAN), Sousse, 2015, pp. 1-5. doi: 10.1109/WSWAN.2015.7210302

[51] P. Baran, "On Distributed Communications Networks," in IEEE Transactions on Communications Systems, vol. 12, no. 1, pp. 1-9, March 1964. doi: 10.1109/T-COM.1964.1088883

[52] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 98-106. doi: 10.1109/INM.2015.7140281

[53] University of Queensland Article https://people.smp.uq.edu.au/Infinity/Infinity%2014/Random_Walks.html

[54] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," 2012 20th IEEE International Conference on Network Protocols (ICNP), Austin, TX, 2012, pp. 1-6. doi: 10.1109/ICNP.2012.6459946

[55] M. Satyanarayanan, "The Emergence of Edge Computing," in Computer, vol. 50, no. 1, pp. 30-39, Jan. 2017. doi: 10.1109/MC.2017.9

[56] S. Wang, Y. Hou, F. Gao and X. Ji, "A novel IoT access architecture for vehicle monitoring system," 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, 2016, pp. 639-642. doi: 10.1109/WF-IoT.2016.7845396

[57] J. G. Andrews et al., "What Will 5G Be?," in IEEE Journal on Selected Areas in Communications, vol. 32, no. 6, pp. 1065-1082, June 2014. doi: 10.1109/JSAC.2014.2328098

[58] G. Dantzig., "Linear programming and extensions". Princeton University Press, 1963.

[59] C. H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization - Algorithms and Complexity". Prentice Hall, 1982.

[60] D. Raz, H. Levy, B. Itzhak, "A resource-allocation queueing fairness measure," SIGMETRICS Perform. Eval. Rev. 32, 1 (June 2004), 130-141. doi: https://doi.org/10.1145/1012888.1005704

[61] R. Cziva, C. Anagnostopoulos and D. P. Pezaros, "Dynamic, Latency-Optimal vNF Placement at the Network Edge," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 693-701. doi: 10.1109/INFOCOM.2018.8486021

[62] R. Jain., D.M. Chiu, W. Hawe,"A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," CoRR. cs.NI/9809099.

[63] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," in Computer, vol. 36, no. 1, pp. 41-50, Jan. 2003. doi: 10.1109/MC.2003.1160055

[64] L.-J. ZHANG, "EIC Editorial: Introduction to the Body of Knowledge Areas of Services Computing," IEEE Transactions on Services Computing, pp. 6274, June 2008.

[65] M. Milenkovic et al., "Toward Internet distributed computing," in Computer, vol. 36, no. 5, pp. 38-46, May 2003. doi: 10.1109/MC.2003.1198235

[66] L. KLEINROCK, "A vision for the internet," ST Journal of Research, pp. 45, Nov. 2005.

[67] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016. doi: 10.1109/JIOT.2016.2579198

[68] W. Shi and S. Dustdar, "The Promise of Edge Computing," in Computer, vol. 49, no. 5, pp. 78-81, May 2016. doi: 10.1109/MC.2016.145

[69] S. Pandi, S. Wunderlich and F. H. P. Fitzek, "Reliable low latency wireless mesh networks From Myth to reality," 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2018, pp. 1-2. doi: 10.1109/CCNC.2018.8319326

[70] P. Mell, and T. Grance, "The NIST definition of cloud computing," 2011.

[71] https://www.gurobi.com/

[72] https://www.ibm.com/analytics/cplex-optimizer

[73] T. Sttzle and H. Hoos, "Stochastic Local Search-Foundations and Applications," 2005. doi: 10.1016/B978-155860872-6/50021-4.

[74] J. Edmonds, R,M, Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," Journal of the ACM19, 248264 (1972).

[75] N. Tomizawa, "On some techniques useful for solution of transportation network problems," Networks, 1: 173-194, 1971. doi:10.1002/net.3230010206

[76] K. S. Ghai, S. Choudhury, A. Yassine, "A Stable Matching Based Algorithm to Minimize the End-to-End Latency of Edge NFV," Procedia Computer Science, Volume 151, 2019, Pages 377-384, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2019.04.052.

[77] J. Kleinberg, E. Tardos, "Algorithm Design," ISBN- 0321295358.