



**UNIVERSITY
OF PIRAEUS**

Ant Colony Optimization in Network Function
Virtualization – Resource Allocation

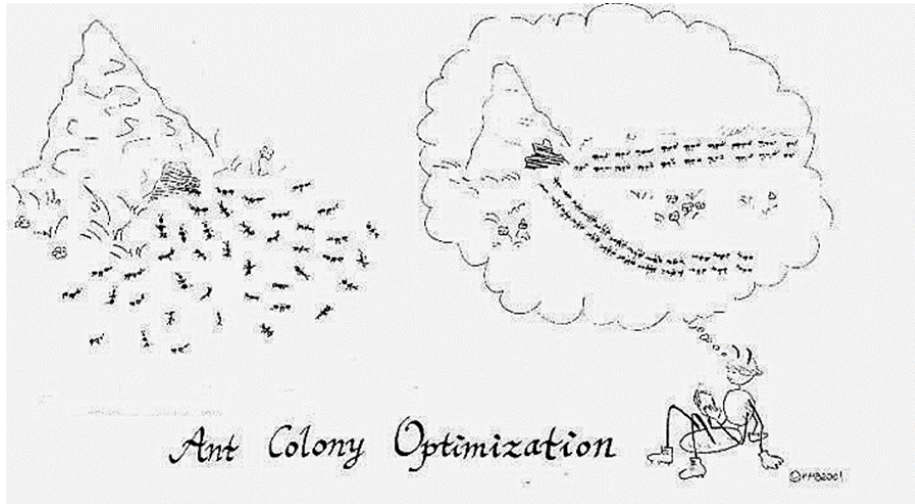
Tsoumanis George

21/12/2018

Final Project for the Master of "Digital Communications and Networks"

Supervisor Teacher : Rouskas Aggelos

Prologue



There are many concerns about how we should make use of the new technologies in terms of Telecommunications. The purpose of this Project is to pore over the Network Function Virtualization and to find solutions as far as the Resource Allocation is concerned. A specific metaheuristic algorithm, Ant Colony Optimization, has been deployed to give almost the optimize solution fora part of this problem.

Thanks

I would like to thank my supervisor teacher K.Tsagkaris for finding this really interesting project which will give me the opportunity to sail on new horizons. Furthermore, I would like to thank my friends for helping me and having edifying discussions on the top of this subject.

Contents

1	Abstract	vii
2	Chapter 1: Scope and Structure of the Project	1
2.1	Definition of meanings	1
2.1.1	Network	1
2.1.2	Node	1
2.1.3	Network Functions	2
2.1.4	Network Function Virtualization	2
2.1.5	Virtual Network Function	2
2.1.6	Services	2
2.1.7	Network Function Virtualization Infrastructure	3
2.1.8	Network Function Virtualization Management and Orchestration	3
2.1.9	Software Defined Networking	5
2.1.10	Network Service	5
2.2	Define the Problem	5
2.2.1	VNE Problem	5
2.2.2	Resource Allocation Three Phases	6
2.2.3	VNFs – Chain composition (VNFs-CC)	7
2.2.4	VNF - Forwarding Graph Embedding (VNF-FGE)	9
2.2.5	VNFs - Scheduling (VNFs-SCH)	10
2.3	11
2.4	Describe the important of the problem	11
2.5	The scope of the analysis and targets of this studying	12
2.5.1	Structure of the Project	12
3	Chapter 2	13
3.1	Problem Statement	13
3.1.1	Definition	13
3.1.2	Mathematic Formulation of VNF-FGE	13
4	Chapter 3	14
4.1	Related Works	14
4.1.1	Exact solutions	14
4.1.2	Heuristic solutions	16
4.1.3	Metaheuristic solutions	16
4.1.4	Tabu Search (TS)	17
4.1.5	Proposed TS Algorithm	17
5	Chapter 4	19
5.1	Solution Algorithm	19
5.1.1	General Info	19
5.1.2	The origins of ant colony optimization	20
5.1.3	The ACO Metaheuristic	21

5.1.4 Environment Description	22
6 Chapter 5	26
6.1 Simulation/Experimentation	26
7 Conclusions	41

Table 1: Catalog of terminology, symbols and acronyms

Abbreviation/Term	Description
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFV-RA	Network Function Virtualization Resource Allocation
NFV-MANO	Network Function Virtualization Management and Orchestration
NFVI-PoPs	NFV infrastructure points-of-presence
VNF	Virtual Network Function
VNFs-CC	VNFs Chain composition
VNF-FGE	VNF Forwarding Graph Embedding
VNFs-CH	VNFs Scheduling
VNFR	Virtual Network Functions Request
VIMs	Virtual Infrastructure Managers
HVSs	High Volume Servers
SFCs	Service Function Chains
EMS	Element Management System
OSS	Operations Support System
TSP	Travel Salesman Problem
NS	Network Service
NF	Network Function
SN	Subtracted Network
CAPEX	Capital Expenditures
OPEX	Operational Expenditures
NAT	Network Address Translation
IDSs	Intrusion Detection Systems
FW	Fire Wall
DHCP	Dynamic Host Configuration Protocol
MWD	Malware Detection
LI	Lawful Interception
ETSI	European Telecommunications Standards
ETSI ISG NFV	European Telecommunications Standards Institute Industry Specification Group

List of Tables

1	Catalog of terminology, symbols and acronyms	v
2	Mathematical Notations	14
3	The hole exploring Environment	22
4	Exploring the Environment of "1"	22
5	Exploring Environment according to the Service	23
6	Parameters of 1 st experiment	32
7	Parameters of 2 nd experiment	33
8	Parameters of 3 rd experiment	34
9	Parameters of 4 th experiment	35
10	Parameters of 5 th experiment	36
11	Parameters of 6 th experiment	37
12	Parameters of 7 th experiment	38
13	Parameters of 8 th experiment	39
14	Parameters of 9 th experiment	39
15	Parameters of 10 th experiment	40
16	Parameters of 11 th experiment	40
17	Parameters of 12 th experiment	40

List of Figures

1	Service Chain	7
2	NFV Architecture	7
3	NFV Management and Orchestration	8
4	VNF Chanin Composition	9
5	VNF Forward Graph Embedding	10
6	Services are being mapped	15
7	Tabu Search	19
8	Ant Colony Optimization	21
9	Define Parameters	26
10	All Nodes and Functions	27
11	Services and Nodes	28
12	Selection randomly of Nodes	29
13	Produced Solutions	30
14	The two final Solutions	31
15	The first experiment	32
16	The second experiment	33
17	The thirtd experiment	34
18	The fourth experiment	35
19	The fifth experiment	36
20	The sixth experiment	37
21	The seventh experiment	38
22	The eighth experiment	39

1 Abstract

This project has been proposed from my teacher in order to provide a really state of the art solution in the fields of Telecommunications. We studied about Network Function Virtualization and we found that a part of the Resource Allocation (many have involved in this subject) can be tackled with an algorithm that was first invented to solve graphical problems (such as the Travel Salesman Problem [TSP]) the Ant Colony Optimization. We have analyzed one of the three phases of the Resource Allocation, and we have decided to tackle the second one. To be more precise, this part takes for granted the other two phases, and it tries to find which combination between links and nodes is the best according to the cost function that someone defines. Our scope was to set as cost functions the minimization of the delay. The results of this project are more than desirable. This is because we have used a metaheuristic algorithm that can find about 90% of the optimal solution without searching the whole environment.

KEY WORDS: Network Function Virtualization, Resource Allocation, Ant Colony Optimization, algorithm, metaheuristic, optimal, delay

2 Chapter 1: Scope and Structure of the Project

In this Chapter we will give definition of very important meanings so as to make clear about what this subject is about and what exactly we will counter. Every Chapter has intro and conclusions so as to describe how it is needed and how it is linked with the other chapters

2.1 Definition of meanings

2.1.1 Network

Network is any and all connections between computing devices. If we think of human anatomy, the network is like the intricate nervous system that takes messages from the brain to the rest of the body and takes feedback from the body back to the brain. It is made up of a series of interconnected nodes that are communicating with one other via predetermined protocols, such as Transmission Control protocol (TCP) / Internet Protocol (IP), that dictate how information will be transported and handled by the devices (switches, routers, security gateways) that forward the traffic from one node to another. Networks can be characterized based on the proximity of all the nodes they connect, for example, local area network (LAN) or wide area networks (WANs), or whether they are public or private. They can also be characterized based on the type of traffic they transport, such as voice or data, e.g. wireless local area network (wLANs), or whether they are made up of physical or virtual devices, such as virtual local area networks (vLANs). They can also be described based on the way in which they transport communications; for example, packet-switched versus circuit switched networks. An operator's network consists of many intermediate Network Functions (NFs). Network address Translators (NATs), load balancers, firewalls, and Intrusion Detection Systems (IDSs) are examples of such functions. Traditionally, these functions are implemented on physical middle-boxes, which are network appliances that perform functions other than standard path discovery or routing decisions for forwarding packets. With the fast development of Internet and network services, more and more middleboxes are deployed in networks for technical reasons, value-add reasons, etc. However, middleboxes means high Capital Expenditures (CAPEX) and Operational Expenditures (OPEX), moreover, deployment or re-deployment of middleboxes needs expertise which increases OPEX and decreases flexibility. Generally, middleboxes are referred as service functions, network functions, or functions. Middle-boxes are based on special-purpose hardware platforms that are expensive and difficult to maintain and upgrade.

2.1.2 Node

Node is anything connected to a network. It may be equipment that is part of the network infrastructure, such as a hub, bridge, or switch, or it may be a host providing information, applications or services to other hosts or nodes on the network

2.1.3 Network Functions

Network Function (NF) – a functional building block within a network infrastructure, which has well-defined external interfaces and a well-defined functional behavior. In practical terms, a Network Function is today often a network node or physical appliance.

2.1.4 Network Function Virtualization

Network Functions Virtualization is an approach to telecommunications networking where the network entities that traditionally used dedicated hardware items which are now replaced with computers on which software runs to provide the same functionality. By running a network based around NFV, it is easier to expand and modify the network, and it is able to provide considerably more flexibility as well as being able to standardize on much of the hardware as it consists of additional computing power. Traditional physical network hardware has always been difficult to change and upgrade. Introducing a software patch or rolling out a new service on a physical network can take months to complete. This is both time consuming and costly. NFV, network functions virtualization started in with mobile telecommunications networks with the promise of making networks more flexible and far easier to upgrade and change. The NFV architecture comprises major components – including virtualized network functions (VNFs), NFV management and orchestration (MANO), and NFV Infrastructure (NFVI) – that work with traditional network components like OSS/BSS. Furthermore, NFV promotes virtualizing network functions, which are responsible for a specific treatment of data flows, which were carried out by specialized hardware devices, and migrating them to software-based appliances. By migrating NFs from dedicated hardware to virtualization platform, NFV can effectively improve the flexibility to deploy and manage service function chains (SFCs). SFC is defined as a sequence of (NF) that should be traversed by a given service flow in a predefined order.

2.1.5 Virtual Network Function

The internet base version of network functions is the Virtual Network Functions. The network functions that were developed as middleboxes can be taken the place of VNFs. With this virtualization trend, many and different technical network elements such as DHCP, NAT, FW etc., have the ability to be deployed as VMs. A network has many different VNFs which need different kind and quantity of resources. Due to the fact that software and hardware are separated in the virtual environment, is given the opportunity to make changes to the software and be aware of not alike service functions. A VNF can take action at many network layers and one or more VNFs can incorporated in the same physical network element.

2.1.6 Services

Service is a set of VNFs that can be implemented in one or multiple virtual machines. In some situations, VNFs can run in virtual machines installed in operating systems

or on the hardware directly; they are managed by native hypervisors or virtual machine monitors. A VNF is usually administered by an Element Management System (EMS), responsible of its creation, configuration, monitoring, performance and security. An EMS provides the essential information required by the Operations Support System (OSS) in a TSP's environment.

2.1.7 Network Function Virtualization Infrastructure

Network Functions Virtualization Infrastructure encompasses all of the networking hardware and software needed to support and connect virtual network functions in carrier networks. NFV defines standards for compute, storage, and networking resources that can be used to build virtualized network functions. NFVI is a key component of the NFV architecture that describes the hardware and software components on which virtual networks are built. NFV leverages the economies of scale of the IT industry. NFVI is based on widely available and low-cost, standardized computing components. Also, it works with servers – virtual, bare metal, or other – and the software, hypervisors, virtual machines, and virtual infrastructure managers to enable the physical and virtual layer of the network. Furthermore, it standards help increase the interoperability of the components of the virtual network functions and aim to enable multivendor environments. NFVI is composed of NFVI-PoPs which are where the VNFs, including resources for computation, storage, and networking, are deployed by a network operator. NFVI networks interconnect the computing and storage resources contained in an NFVI-PoP. This may include specific switching and routing devices to allow external connectivity. NFVI works directly with VNFs and VIMs and in concert with the NFV orchestrator. NFV services are instantiated at the directive of the NFV orchestrator, which calls on VIMs that manage the resources from the underlying infrastructure (NFVI). NFVI is as critical to realizing the business benefits outlined by the NFV architecture as any other functional block. It delivers the actual physical resources and corresponding software on which VNFs can be deployed. NFVI creates a virtualization layer that sits right above the hardware and abstracts the HW resources, so they can be logically partitioned and provided to the VNF to perform their functions.

2.1.8 Network Function Virtualization Management and Orchestration

Network Function Virtualization Management and Orchestration is the manager of the NFV so as to work properly from its early stages. NFV-MANO, management of NFV is now addressed by the MANO stream. NFV MANO is a working group (WG) of the ETSI ISG NFV. It is the ETSI-defined framework for the management and orchestration of all resources in the cloud data center. This includes computing, networking, storage, and virtual machine (VM) resources. The main focus of NFV MANO is to allow flexible on-boarding and sidestep the chaos that can be associated with rapid spin up of network components. NFV MANO is broken up into three functional blocks:

NFV Orchestrator(NFVO): provides management of the NFV services, which is responsible for on-boarding of new NS and VNF packages; NS lifecycle manage-

ment; global resource management; validation and authorization of NFVI resource requests. Some of the functions that are typically required by NFV orchestration include the following:

- Service coordination and instantiation: The orchestration software must communicate with the underlying NFV platform to instantiate a service, which means it creates the virtual instance of a service on the platform.
- Service chaining: Enables a service to be cloned and multiplied to scale for either a single customer or many customers.
- Scaling services: When more services are added, finding and managing sufficient resources to deliver the service.
- Service monitoring: Tracks the performance of the platform and resources to make sure they are adequate to provide for good service.

VNF Manager(VNFM): The VNFM is a key component of the NFV-MANO that helps standardize the functions of virtual networking and increases interoperability of software-defined networking elements. The VNFM is responsible for the lifecycle management of VNFs under the control of the NFVO, which it achieves by instructing the VIM. VNFM operations include:

- Instantiation of VNFs
- Scaling of VNFs
- Updating and/or upgrading VNFs
- Termination of VNFs

All VNF instances are assumed to have an associated VNF manager. A VNFM may be assigned the management of a single VNF instance or multiple VNF instances. The managed VNFs can be of the same or different types. VNF manager functions are assumed to be generic and can be applied to any VNF.

Virtualized Infrastructure Manager (VIM): Controls and manages the NFVI compute, storage, and network resources usually within one operator's infrastructure domain. These functional blocks help standardize the functions of virtual networking to increase interoperability of software -defined networking elements. VIMs can also handle hardware in a multidomain environment or may be optimized for a specific NFVI environment. The VIM is responsible for managing the virtualized infrastructure of an NFV-based solution. VIM operations include:

- It keeps an inventory of the allocation of virtual resources to physical resources. This allows the VIM to orchestrate the allocation, upgrade, release, and reclamation of NFVI resources and optimize their use.
- It supports the management of VNF forwarding graphs by organizing virtual links, networks, subnets, and ports. The VIM also manages security group policies to ensure access control.

- It manages a repository of NFVI hardware resources (compute, storage, networking) and software resources (hypervisors), along with the discovery of the capabilities and features to optimize the use of such resources.

The VIM performs other functions as well – such as collecting performance and fault information via notifications; managing software images (add, delete, update, query, copy) as requested by other NFV-MANO functional blocks; and managing catalogues of virtualized resources that can be consumed from the NFVI. In summary, the VIM is the management glue between hardware and software in the NFV world.

2.1.9 Software Defined Networking

Software Defined Networking encompasses multiple kinds of network technologies designed to make the network more flexible and agile to support the virtualized server and storage infrastructure of the modern data center. The goal of SDN is to enable cloud computing and network engineers and administrators to respond quickly to changing business requirements via a centralized control console. SDN originally defined an approach to designing, building, and managing networks that separates the network's control or SDN network policy (brains) and forwarding (muscle) planes thus enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services for applications as SDN cloud computing or mobile networks.

2.1.10 Network Service

Network Service is distributed with one or more network functions and is a full completed functionality. It is also a set of chained VNFs in which the packets must get through is so as to become part of the network service. The NFV deploys a network service when it defines the: 1) numbers of the VNFs, 2) the order of the VNFs and 3) how the VNFs are distributed in the Network Functions Virtualization Infrastructure (NFVI) or Substrate Network (SN).

2.2 Define the Problem

2.2.1 VNE Problem

Embedding virtual networks in a Substrate Network (SN) is the main resource allocation challenge in network virtualization and is usually referred to as the VNE problem[1][2]. VNE deals with the allocation of virtual resources both in nodes (mapped to substrate nodes) and links (mapped to substrate paths)[3]. It is mainly concerned with the efficient mapping Virtual Network Requests onto a shared substrate network. The VNE problem can be either offline or online. In offline problems, all the virtual network requests are known and scheduled in advance while for the online problem, such requests arrive dynamically and can stay in the network for an arbitrary duration[4][5][6]. VNE is known to be NP-hard[7] ; therefore, most of the work done in this area has focused on the design of heuristic or metaheuristic

algorithms and the use of networks with minimal complexity when solving mixed Integer Linear Programming (ILP) models. Embeddings can be optimized with regard to several parameters, such as: embedding cost, link bandwidth, QoS, economical profit, network survivability, energy efficiency[8][9], security[10][11], etc.

2.2.2 Resource Allocation Thee Phases

It is believed that the NFV framework will settle many of the network difficulties that exist nowadays because of the wide use of specific hardware equipments. Furthermore, network optimization as well as cost reduction are likelihood to be achieved. Also, it gives the chance to set up hybrid scenarios such as the co-existence of functions being executed on virtualized and on physical resources[12]. Such hybrid scenarios may be important in the transition towards NFV. In order to deploy a NS the data traffic needs to cross a set of middleboxes with specific order so as to create a processing in accordance with the function they perform. Middleboxes orchestration is the task which decides which middleboxes are needed and guides the traffic in between them.[13]. This procedure is being performed by hand until now, and is set at the forwarding table entries of routers; the above is a cumbersome and error prone process[14]. But the procedure of placing the physical middleboxes tends to be inefficient due to the fact that it is inconvenient and costly to change the location of the middleboxes with respect to the network conditions. In the NFV ecosystem, an NS is a set of chained VNFs as shown in Fig. 1. The resource allocation of demanded network services is one of the basic challenges in NFV-based infrastructure. The name of this challenge has been named NFV Resource Allocation (NFV-RA) problem. Productive algorithms are required for the resource allocation in NFV so as to decide on which HVSs the VNFs will be placed and to have the ability to migrate functions from one server to another in order to achieve load balancing, decrease of CAPEX and OPEX, energy saving , etc. [15]. This, give us the opportunity to allocate dynamically the virtualized resources such as CPU, memory, storage, to each VNF in accordance with the required service traffic and also to deploy the targeted service function chains according to geography or customer sets. Standardization organizations have already been working on introducing NFV into 5G networks.

The component that is responsible for the resource allocation in the NFV architecture framework is the orchestrator. Fig. 3 shows a scenario where VNFs are managed by the orchestrator through the VNF as well as the virtualized infrastructure manager. (see Fig. 2). All the conditions are evaluated by the orchestrator so as to execute the task of VNFs chains on the physical resources which it leans on the VNF and the virtualized infrastructre managers. There are three stages in which the resource allocation is performed: 1) VNFs Chain composition (VNFs-CC) which handles problems that have to do with how many of each VNF to deploy and what's their order. 2) VNF Forwarding Graph Embedding(VNF-FGE) which handles problems that have to do with where to place the VNFs in the network infrastructure in a appropriate way so as to streamline the network cost and assert the services' requirements and 3) VNFs Scheduling (VNFs-CH) which handles problems that have to do with scheduling of VNFs' execution time so as to minimize the total execution

time of services.

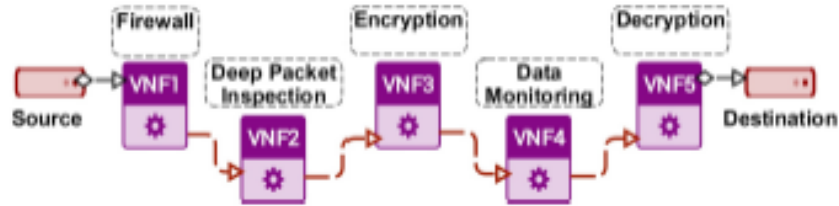


Figure 1: Service Chain

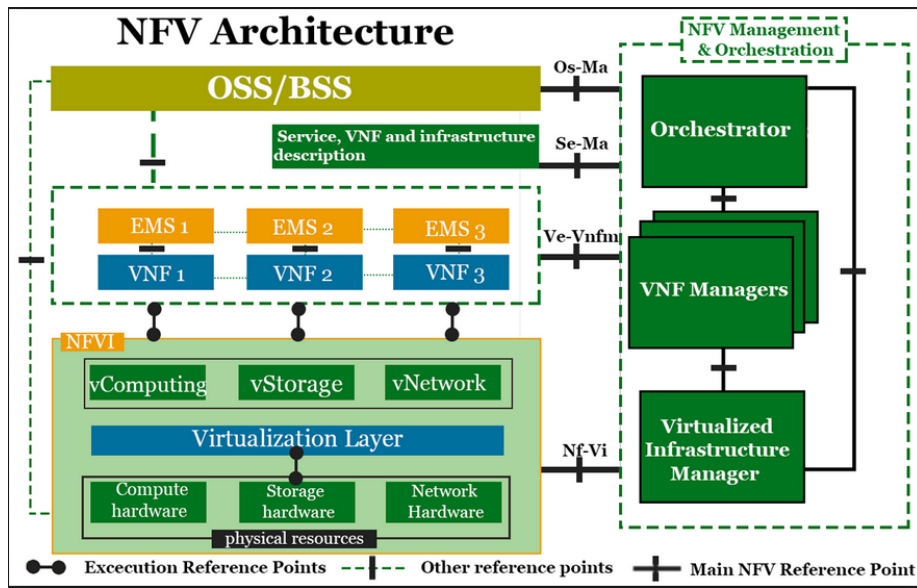


Figure 2: NFV Architecture

2.2.3 VNFs – Chain composition (VNFs-CC)

When a NS is given to allocate its resources, the service providers get a chain of VNFs so as to place and link the received functions on the physical substratum. Even if this procedure is automated it is stiff both for clients as well as providers. The first one have to specify services with complexity function dependencies, while the latter have not the ability to build the chains in a way to be applied to their infrastructure properly. Suboptimum allocation of chains is the aftermath which means that more VNF instances may be required and the costs will be increased. The Structure of the chain most of the times is flexible even if some dependencies and connections

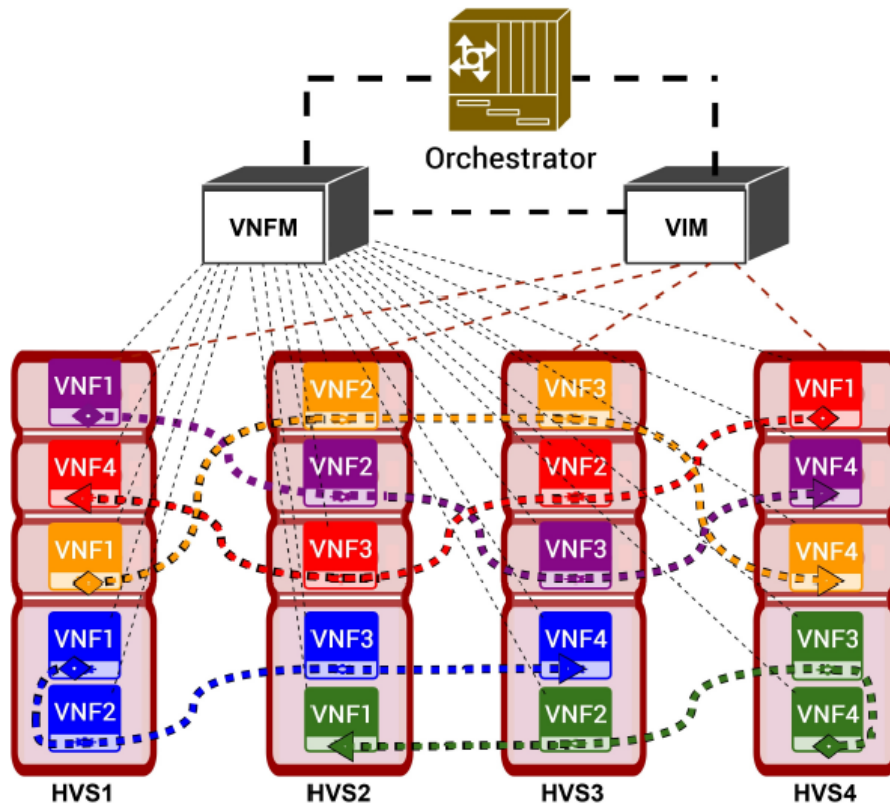


Figure 3: NFV Management and Orchestration

in between VNFs must be considered. There is no properly dependency between leakage prevention system and a traffic shaper or between a proxy server and a WAN optimizer. As an aftermath, many different chains can serve the same service. SFC composition problem is called the problem of searching the most suitable VNF chain given the specifications of the network service and the constraints of the resource. Fig. 4(a) gives us the image of the specification of a network. In place of providing the chain structure as a sum, the necessary information have to be informed by the clients so as to allow the network service providers to obtain their best chain with respect to a predetermined goal for example, lessen the network functions as much as possible. Basically, a Virtual Network Functions Request is contained by five parts:

- the pace of the data in the network,
- the chained VNFs with its own processing needs,
- the end-to-end VNFs flow,

- the outbound connections,
- the required reliance.

In Figure 4 (b), is depicted a scenario with two potential chains for the service delineated in Fig. 4(a). We can see that VNFs 2 and VNFs 3 are appeared in the sub-flows VNF1, whereas VNF4 is appeared in the two sub-flows.

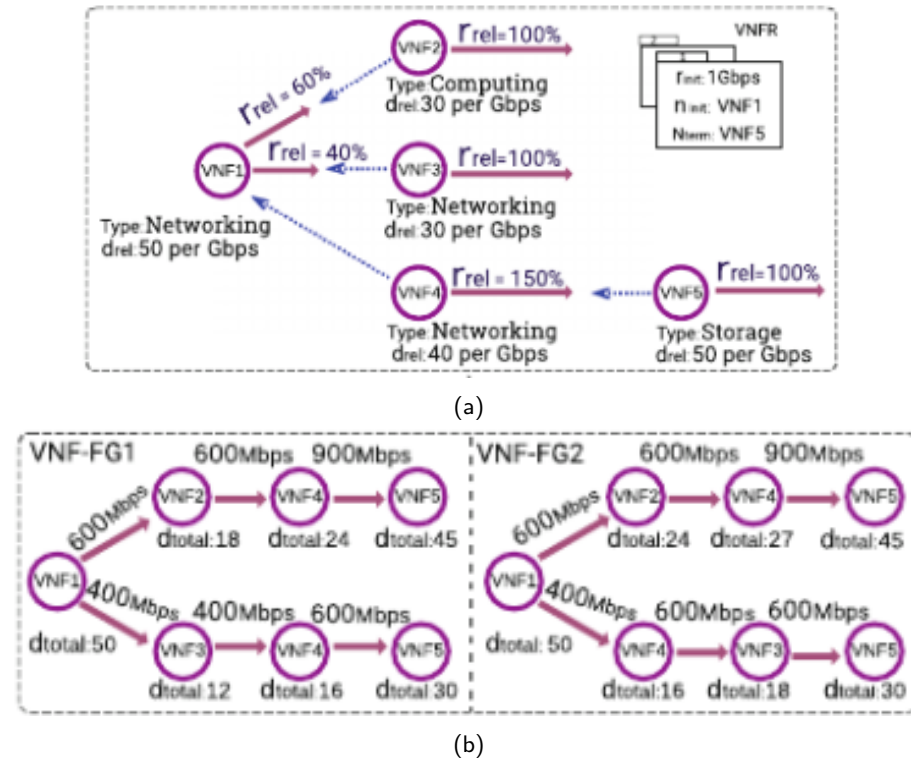


Figure 4: VNF Chain Composition

2.2.4 VNF - Forwarding Graph Embedding (VNF-FGE)

The chain of VNFs which are composing an end-to-end network service, as it has been claimed, is called VNF-FG. "This resulting graph in the first stage is given as the input of the embedding stage". The NS runs an ordered set of VNFs which VNF-FG is composed by so as to meet the service features. VNF-FGE search to find where to distribute the VNFs in the network in a appropriate way. Additionally, the optimization of the resource must be met with success in regard to a specific objective (e.g. minimization of delay). VNF-FGE can be assumed that it is NP-hard due to the fact that it can be seen as a generalization of VNE which is a

NP-hard. The problem comprises the matching of virtual resources to substrate resources. Substrate resources can be spent only when all the virtual resources can be matched and the entire network is embedded. A specific composition of VNFs which composes a service, provides a functionality. The physical resources are removed by the virtualized layer and fasten the VNFs to the virtualized infrastructure. A HVS is part of the NFV based network architecture with the characteristic of a physical node. Hypervisor is a tool which is used by the HVS so as to control the VMs with respect to the available resources. The VMs can serve more than one VNFs with the same type e.g. computing storage and network. In the VNF-FGE there are two phases of mapping. The virtual node mapping in which the VNFs should be served by HVSs and the virtual link mapping in which the algorithm tries to map all the virtual links between VNFs to the path in SN.

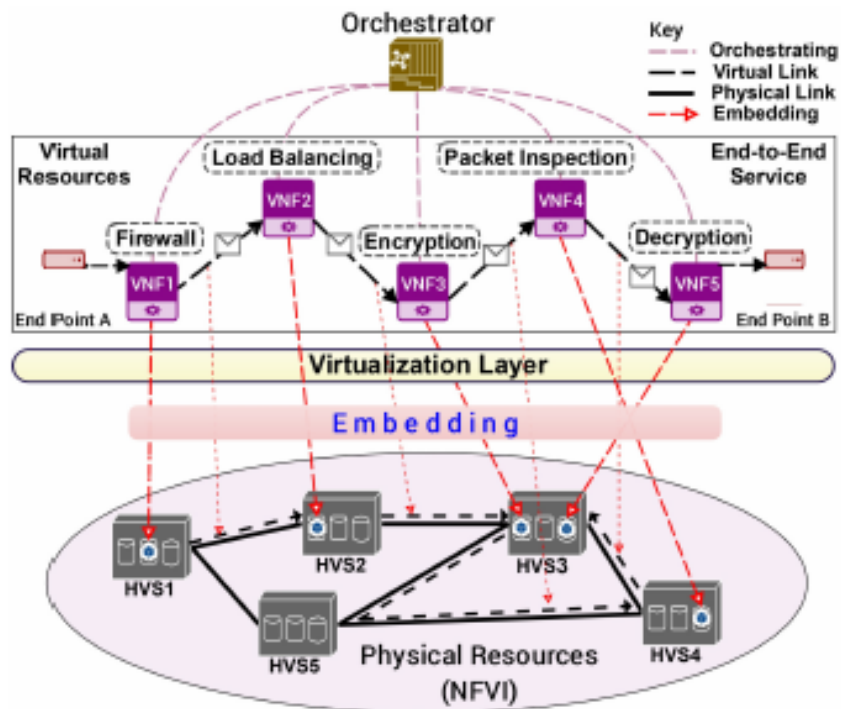


Figure 5: VNF Forward Graph Embedding

2.2.5 VNFs - Scheduling (VNFs-SCH)

This stage attempts to give an answer to the following question: what is better, to execute each function in order to minimize the total execution time or execute each function with degrading the service performance and respecting all the precedence and dependencies between the VNFs composing the NS? The NFVI consists of

many and several different HVSs, thus in order to minimize the execution time of the network service we have to make a proper scheduling plan. The effectiveness, performance, and efficiency of the scheduling process can be defined in terms of:

- number of available HVSs in the NFV infrastructure so they can process the functions composing the services
- the computing capacity of each server to process all the assigned functions
- the complexity of the different network services, i.e., the number of functions composing each service.

2.3

VNE and NFV-RA are in the same problem domain; in the end, the outcome of both problems is the efficient allocation of virtual requests on top of the physical network infrastructure. However, they present the following differences:

- (i) VNE has static virtual network topologies where nodes are arranged in a fixed, predetermined order as input. In contrast, NFV-RA's input is a network service request composed of a set of VNFs with precedence constraints and resource demands that can be denoted by several Virtual Network Function Forwarding Graphs (VNF-FGs); the task of the first stage of the problem (VNFs-CC) is to efficiently build a suitable VNF-FG with regard to the operator's goals.
- (ii) Resource demands may change depending of the traffic load directed to them: e.g., computing resource demands of a transcoding VNF vary depending on how many multimedia data have to be transcoded. Also, bandwidth demands change depending on the ordering of VNF instances[16], whereas resource demands are mostly static in VNE.
- (iii) VNE shares several similarities with the second stage of NFV-RA: VNF-FGE. In fact, we think of VNF-FGE as a generalization of VNE because, while the latter considers only one type of physical (networking) device, a much wider number of different network functions coexist in NFV that can be mapped in different kind of hardware (networking, computation and storage) devices.

2.4 Describe the important of the problem

Telecommunication Service Providers (TSPs) rely on having physical devices and equipments to their possession for each function composing a given service. In order to cover the incensed requirements of users for new services, TSPs need to buy, store and handle new physical components and as an aftermath there is a increased CAPEX and OPEX. Furthermore, because of the hard competition in between them, it is difficult for the TSPs to reply to the increased expenses with increased subscriber prices. This situation has steered to declining average revenue per user, and enforced them to build more dynamic and agile networks with ease in installation and managing in order to remain lucrative. The proposal of the

NFV [17],[18] was to give a hand of help to the TSPs to meet with success these goals. The basic idea behind of the NFV is to disconnect Network Functions (NFs) from the physical network component on which they run. Advanced virtualization technologies can play crucial role in pooling various NFs onto high volume servers, switches and storage, which could be located in datacenters, network nodes or user premises so as this to be achieved. VNFs have the possibility to be composed to form a service and then relocated to different network without buying and installing new hardware. With NFV TSPs will decrease their CAPEX and OPEX by leading scalable and dynamic NFs to services. Despite this, when a number of VNFs is required for a given service, two main questions aroused in NFV. Firstly, how to put into effect and determine network functions and services. Secondly, how to map and schedule the service's functions to the physical network in an efficient way. The first question is being covering by the ETSI [19]. While, the second one is not of a big interest, with the TSPs doing it by hand. Challenges are increased when we are talking about a large scale. Furthermore, the manually procedure of TSPs will not be feasible as service requirements and networks are growing dynamically. As noted in [20], In order to carry out the mapping of VNFs onto the physical nodes an automatic procedure is required. This automation will be cover of course by smart algorithms that will be able to face efficiently the services and the resources [21].

2.5 The scope of the analysis and targets of this studying

In this project, we formulate the VNFs mapping problem and then suggest different algorithms for solution. In particular, we propose one algorithms that perform the mapping of VNFs based on a metaheuristic criterion such as available buffer capacity for the node and the processing time of a given VNF on the possible nodes. In addition, the proposed metehuristic algorithm starts by creating an initial solution randomly, which is iteratively improved by searching for better solutions in its environment. Finally, in order to have a comparable evaluation of the experiment, we find the best solution as far as the minimization of the processing time in Nodes for a given service is concern. Then we compare it with the results that our algorithm produces.

2.5.1 Structure of the Project

The present Project for finding solution of the second phase of Resource Allocation in the Network Function Virtualization will be propounded in a total of 5 Chapters. The structure of the project is described below:

- (i) In Chapter2, is given the definitions of the problem while in (2.2) is given the mathematic formulation of VNF-FGE
- (ii) In Chapter3, is given the related works that have been made so as to tackle the same problem.
- (iii) In Chapter4, is given the general information about the Ant Colony Optimization as well as its origins.

- (iv) In Chapter 5, is given the simulation of the developed program through screenshots while plotted graphs are displayed to give a better image for the produced solutions.

3 Chapter 2

3.1 Problem Statement

As far as the problem that we will tackle on this study is concerned, we give emphasis in the second stage of the Resource Allocation, the Virtual Network Function – Forward Graph Embedding (VNF-FGE) and we give the mathematic formulation and the constraints of it.

3.1.1 Definition

Definition VNF-FGE in ACO. Given a network graph $G(N,A)$, where N is the set of HVSs, A the functions, $Au \leq A$ functions that are in the service. Given a set of edge demands D each demand k & D being characterized by source, destination, nominal buffer, processing time, the VNF-FGE problem is to find:

- (i) the optimal placement of VNF nodes over NFVI clusters.
- (ii) NFVI cluster capacity constraints

In our network model we propose a minimization of the processing time of the NFVI cluster depended on the network service. Furthermore, we assume that:

- (i) Every service has at least one function
- (ii) Every function is linked with at least one node
- (iii) Every node can serve at least one function
- (iv) Every node can serve as much functions as his buffer can

3.1.2 Mathematic Formulation of VNF-FGE

Table I reports the mathematical notations used in the Ant Colony Optimization formulation of the VNF-FGE. We work in an extended graph in which we try to place the functions i in the nodes j (Actually to place the VNFs to HVSs). We use two binary variables: $map_{i,j}$ represents whether the function i is placed in the node j or not; $service_m$ represents whether a $function_m$ is selected to be part of a specific service.

Now we present the constraints:

- $$\sum_{i=1} map_{i,j} \geq 1 \forall j \in [1, n]$$

This means that every Node must serve at least one Functon

Table 2: Mathematical Notations

Parameters	Description
$1 \leq k \leq m$	number of Functions
$1 \leq i \leq l$	number of Functions in Service
$1 \leq j \leq n$	number of Nodes
$20 \leq fb_i \leq 30$	$Function_i$ buffer
$75 \leq nb_j \leq 100$	$Node_j$ buffer
$map_{i,j}$	$map_{i,j} = \{1, \text{ if } Node_j \text{ can serve the } Function_i \text{ and } 0, \text{ otherwise}\}$
$service_m$	$service_m = \{1, \text{ if } Function_m \text{ is selected for the service and } 0, \text{ otherwise}\}$
$processing_{i,j}$	$p_{i,j} = \{\geq 15 \text{ and } \leq 30, \text{ if } map_{i,j} = 1 \text{ and } 0, \text{ if } map_{i,j} = 0\}$
$pheromone_{i,j}$	$pheromone_{i,j} = \{\geq 1, \text{ if } map_{i,j} = 1 \text{ and } 0, \text{ otherwise}\}$

- $nb_j \geq 0 \forall j \in [1, n]$
This means that the buffer of each node must be greater than zero
- $l \leq 0.7 * m$
This means that the service must be contained by maximum the 70% of the number of the Functions

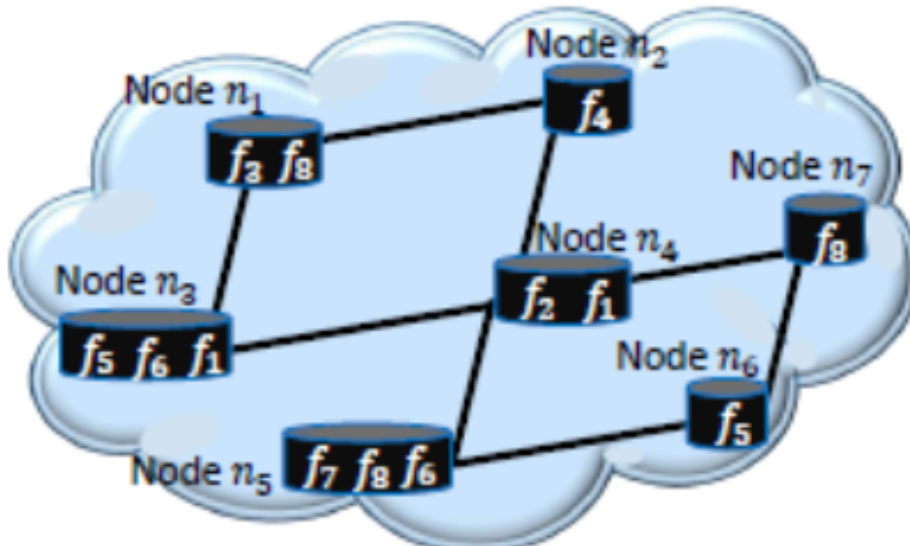
Function Mapping In Fig. 6(a) we can see a virtual network which is consisted of Nodes ($n1, n2, \dots, n7$) and every Node is able to serve a specific number of functions ($f1, f2, \dots, f8$). Afterwards, a Service Request with functions ($(f8 \rightarrow f2 \rightarrow f3 \rightarrow f6 \rightarrow f5)$) arrives so as to be mapped. A likelihood mapping could be the Fig. 6(b) in which the functions $f3$ and $f8$ are served by the Node $n1$.

4 Chapter 3

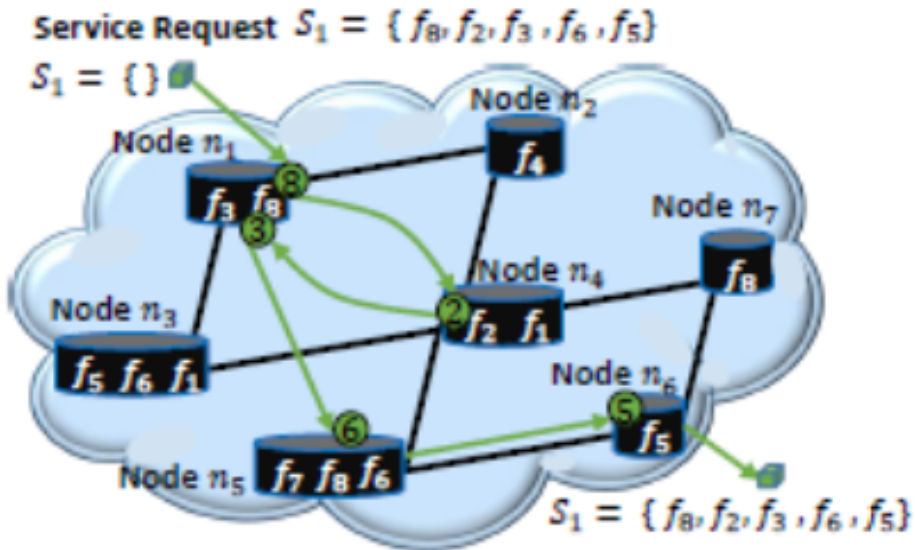
4.1 Related Works

4.1.1 Exact solutions

Linear programming algorithms can find optimal solutions. ILP can specifically be used to optimize the VNF-FGE problem. "Although ILPs are in many practical situations NP-complete, there are exact algorithms (e.g. branch and bound or branch and price) that solve small instances of the problem in reasonable time". The tools that execute these algorithms, which are called solvers, can be found s open-source are proprietary. The VNE-FGE problem was proposed to be solved with the help of ILP where the main goal was to minimize the usage of the physical nodes. The evaluation of the performance has happened under different traffic loads and in a small provider scenario. "Another exact strategy is presented in where a network-enabled cloud is considered the NFVI". Another proposal for ILP was to minimize the usage of the bandwidth in the physical network. When all the physical



(a) Network node capabilities



(b) After mapping of service 1

Figure 6: Services are being mapped

nodes are NFV-enabled, the results show that the savings in the bandwidth are bigger than in a hybrid physical network scenarios.

4.1.2 Heuristic solutions

In NFV-RA the execution time is very important. Due to the fact that NFV has to do with dynamic online environments the time that the service requests arrive are not known. So, the execution time when we are trying to solve the NFV-RA has to be minimized so as to avoid delay. For this reason, solutions that are based in heuristic are proposed. Such an approach is presented in [22] where a repetitive greedy algorithm is put into effect so as to map VNFs to physical node and to make the total network load balanced. Algorithms that are trying to find the shortest path are implemented for link mapping. There is another approach based on heuristic trying to solve the VNF-FGE problem presented in [23]. In this presentation, is proposed to divide the VNF-FGE into two known NP-hard problems. Firstly to facility location and secondly to generalized assignment problems. It is introduced then rounding-based heuristics so as to find solution for the problem and get results that are better as far as the OPEX is concerned than with solutions that are greedy based.

The VNF-FGE is the second stage and it has been studied to large extent. "VNF-FGE can also be referred as middlebox/network function placement". It has been proved that the VNF-FGE problem is NP-hard. To find solution to this kind of problems, many works develop it as a mixed ILP and implement a method which is heuristic-based. In [7], It is studied two heuristic-based algorithms so as to make the middlebox placement problem optimal. The first one is a greedy algorithm, while the second one is ,VOLUME 4, 2016 8085 L. Wang et al.: Joint Optimization of Service Function Chaining and Resource Allocation , a Simulated Annealing algorithm. The results of the emulation have shown that the Simulated Annealing algorithm does better than the greedy ones. References [19]–[21] in order to find solution for the placement problem in different use cases such as SDN and data center networks, use heuristic algorithms. Studies have also done in some context-aware placement problems. "T. Taleb et al. in [18] study VNF placement algorithms in virtual 5G network, with the goal of minimizing the path length and optimize the sessions mobility". VNF scheduling can be mentioned to execution as well as traffic scheduling in between VNF instances. R. Mijumbi et al. Suggest approached that are based in greedy and metaheuristic so as to face the VNF execution time scheduling in online VNF-FGE and VNF-SCH [11].

4.1.3 Metaheuristic solutions

NFV-RA can be considered as a tendency to combine optimization problem where through an appreciable search-environment an optimal solution can be found. Because it is difficult to find an optimal solution through large instances, metaheuristic algorithms are used in order to approach a near-optimal solution by repetitive improvement of solutions. [24]. "To the best of our knowledge, the only metaheuristic-based NFV-RA approach is presented in [68]". Tabu search algorithm here is used in order to find solution to the VNF-FGE and VNFs-SCH problems at the same time targeting at decreasing the flow execution time. The results from the simulation have shown that the relation is highly accepted, the average flow execution time is

low as well as the embedding cost.

4.1.4 Tabu Search (TS)

Tabu search was used for mathematical optimization and it is a metaheuristic search method that is based on neighborhood(local) search methods[25] . The local search [26] method takes a possible solution Z and it checks the direct neighbors $N(Z)$ with the anticipation of finding a better solution Z' . The solutions that can be found in the $N(Z)$ are looked for to be almost the same to Z apart from few details. Nevertheless, the local search methods have the tension to be hooked in regions where the solutions are sub-optimal or are almost equal. Tabu search can improve the efficiency of local search by making the basic rules more relaxed. At every step, moves that are worsen have the chance to be accepted only if an improved moved is not available (for example, when a move is hooked in a local minimum). Furthermore, Tabu search gives description of the visited solutions or the sets of the rules provided by the user with the help of memory structure. The algorithm points out some solutions as 'tabu' when a possible solution has been re-visited the previous short period or the one or more of the rules have been infringed so as not moving there constantly. This kind of move is named tabu move. Nevertheless, when this king of move has a adequately appealing evaluation where it could give a better solution than has ever been visited before, the classification that tabu has done can by overturned. A situation that gives space to a overturn to be happened is called aspiration criterion[27].

4.1.5 Proposed TS Algorithm

For the purpose of designing a Tabu search algorithm, five considerable parts must be defined: the initial solution, the neighborhood solutions, tabu list, aspiration criterion and stopping condition. Now, we give description of these aspects.

- (i) Initial solution: The first step is to define randomly an initial solution Z_0 . This can be done by selecting randomly from the set $N(i) \leq N$ a virtual machine j for each function i in the service. After that, the mapped functions are scheduled into the VMs. The solution Z that was found is set to Z_0 .
- (ii) Neighborhood Solutions: For purpose of finding better than Z' we have to assess solutions $N(Z)$ in the neighborhood Z . It is needed to be determined the $N(Z)$. In accordance to the ideal, the solutions that indicate a movement of each function from one VM to another could create different solution. Though, this situation would lead to make the search environment much bigger. As a consequence, the neighborhood is restricted according to the changes of the mapping in the functions with the biggest previous time gap. The function f' that has very big time gap is chosen to be migrated. According to that, all likelihood solutions which would help the migration of f' from its current position to another virtual node which has the ability to process it , are involved in the $N(Z)$. If none of the VM has the capability to process the f' , as next candidate for migration is chosen the next function with the biggest

time gap. As the migrations is completed, the scheduling of all the proceeding functions is assessed so as to guarantee that the flow time is minimum.

- (iii) "Tabu List: If a function i has been moved from virtual machine j_1 to virtual machine j_2 , we declare it as tabu to move this function back to j_1 during the next $m - 1$ iterations." The usage of $m-1$ is to give the opportunity to the left-over $m-1$ functions to move before the function that is candidate can return back to its VM. Tabu is also the selection of a solution with higher flow time than the until now best solution.
- (iv) Aspiration criterion: Allow for exceptions from the Tabu list, if such moves lead to promising solutions. Furthermore, the algorithm selects the least tabu move from the tabu moves when there is no other available moves.
- (v) Stopping condition: There are determined two different use cases that define when the algorithms has to be stopped:
 - (1) If there is not any improvement in the flow time after m repetitions in a row,
 - (2) If there is no possible solution for all the function in the neighborhood of solution Z . Apart form the above, the algorithm will be stopped also when the number of the repetitions in ended.

Algorithm 2 Tabu Search-based NFMS(S, N)

```
1: Determine Initial Solution  $Z_0$ 
2: if  $Z_0$  notPossible then
3:   Mapping and Scheduling Failed
4:   return
5: end if
6: Initialize:  $Z = Z_0, Z^* = Z_0, T_l = \emptyset$ 
7: Determine function  $f$  to move and neighborhood  $N(Z)$ 
8: while StopConditionNotMet() do
9:   solutionSet =  $\emptyset$ 
10:  for sCandidate  $\in N(Z)$  do
11:    if sCandidate doesNotViolate  $T_l$  then
12:      solutionSet = solutionSet  $\uplus$  sCandidate
13:    end if
14:  end for
15:  sCandidate = BestFlowTime(solutionSet)
16:   $Z =$  sCandidate
17:  updateTabuList( $T_l$ )
18:  if FlowTime(sCandidate) < FlowTime( $S^*$ ) then
19:    addNewTabu( $T_l, f, \text{getNode}(f), \text{getSize}(S)$ )
20:     $Z^* =$  sCandidate
21:  end if
22: end while
23: return  $Z^*$ 
```

Figure 7: Tabu Search

5 Chapter 4

5.1 Solution Algorithm

5.1.1 General Info

Social insects have possessed an important role in the eco system due to the evolution. What supports this concept is their social organization. These insect societies show us that the actions of the individuals but of the society as well are not led by a centralized control. Through to the obey to the common good and to specific communication skills they generate complex patterns. Ants can be characterized as the most prosperous species as there are almost 9000 different species and they can

live everywhere and in enormous numbers. [28]. Beyond entomologist's interest of observation of ants, nowadays computer scientists and engineers have shown curiosity too. "Ant societies feature, among other things, autonomy of the individual, fully distributed control, fault-tolerance, direct and environment mediated communication, emergence of complex behaviors with respect to the repertoire of the single ant, collective and cooperative strategies, and self-organization". These characteristics are appeared at the same time As a consequence, ant societies are shown as new models for building algorithms for optimization problems such as scheduling, network design, bioinformatics etc ." The different simulations and implementations described in this research go under the general name of ant algorithms (e.g., [29] , [30], [31], [32], [33], [34], [35])". A new state is being created by the researchers which combines implementations and theoretical aspects . It has been used as the solid base to use the ant way to solve with success many real problems. "Genetic algorithms [[36], [37], [38]] and neural networks [[39], [40], [41]] are other remarkable and well-known examples of Nature-inspired systems/algorithms".

5.1.2 The origins of ant colony optimization

"Marco Dorigo and colleagues introduced the first ACO algorithms in the early 1990's [[42],[43],[44]". Ants belong to the social insects and their behavior was the inspiration of creating these algorithms. They give priority to survival of their colony rather than of themselves. It is true that ants can find shortest paths starting from their nest to the food source. The way that ants try to find for food is described below. Firstly, they start from their nest and are exploring it in a circle without a specific order. Ants while are moving are secreting a chemical called pheromone on the ground which creates a trail. Ants have the ability to smell and to communicate through the pheromone. When it is time to select their way, it is likelihood to choose paths that are marked by strong pheromone. When ants find food, they take some of it back to their nest and the amount of the pheromone that they leave is proportionally to the quantity and to the quality of the food that was found. "It has been shown in [45] that the indirect communication between the ants via pheromone trails—known as stigmergy[46]—enables them to find shortest paths between their nest and food sources". We have a graph $G = (V, E)$ which is consisted of two nodes v_s and v_d and two links e_1 and e_2 respectively. The length of each link is l_1 and l_2 accordingly where $l_2 > l_1$. As real ants leave on the ground pheromone while they are moving, trails are created. For each link e_i , $i=1,2$ we introduce an artificial pheromone value τ_i , $i=1,2$ which indicates the strength of the pheromone. The behavior of the artificial ants is described below: They start from the point v_s (their nest) and they have to choose one of the two paths with probability $\pi_i = \tau_i / (\tau_1 + \tau_2)$, $i = 1, 2$ so as to go the food. From the equation we can understand that is $\tau_1 > \tau_2$ the probability to choose the first path is higher. Ants then choose the same path to turn back and the pheromone changes $\tau_i \leftarrow \tau_i + (Q / l_i)$ (2), where Q is the constant value of the model. The procedure that described above is repeatedly simulated. In the real world the pheromone evaporated over time. The evaporation is simulated as: $\tau_i \leftarrow (1 - \rho) \cdot \tau_i$, $i = 1, 2$. (3) the parameter $\rho \in (0, 1]$ regulates the evaporation. For the simulation of the system we set the

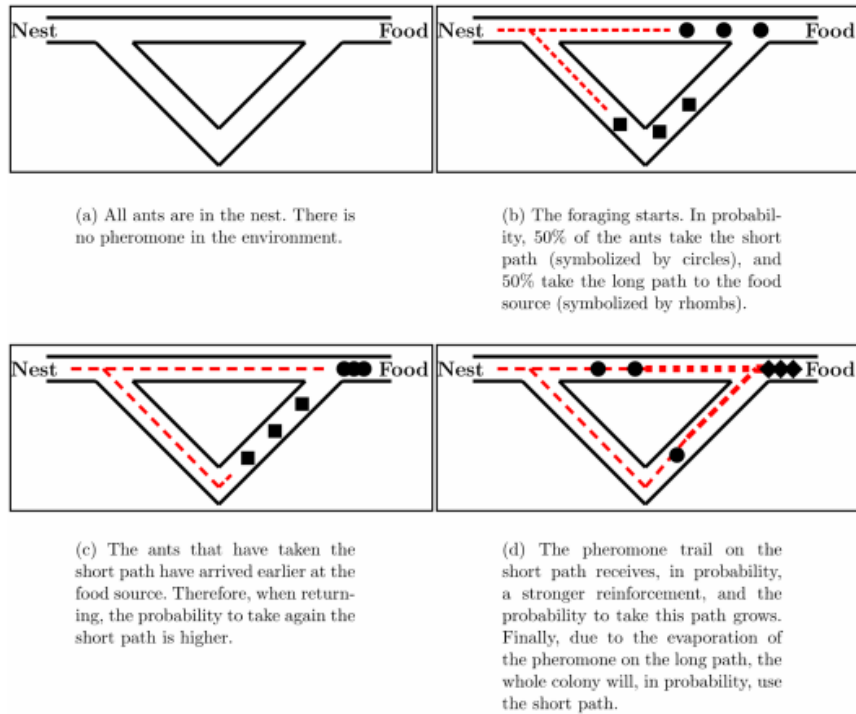


Figure 8: Ant Colony Optimization

values $l_1=1$, $l_2=2$ and $Q=1$ for the parameters. The values of the pheromone are set to 0.5 each because they cannot take the value 0. The simulation has shown that over time the ants tend to find and use the shortest path. There are three main differences between the artificial and the real ants. Firstly, real ants move through the environment asynchronously while the artificial ones move from their nest to the food and back again. Secondly, real ants deposit pheromone with every move that they do, while artificial ones only when they return back to their nest. Thirdly, real ants' searching for food behaviour is grounded on evaluating a solution indirectly, while artificial ones' evaluation of a solution is based on a quality measure which defines the strength of pheromone.

5.1.3 The ACO Metaheuristic

Artificial ants are looking for new stochastic solutions based on probability considering i) the pheromone in the paths that is changing in the course of time dynamically and ii) the heuristic information about the problem. In ACO ants have the possibility to search for a bigger amount of solution comparing with a greedy heuristic. The heuristic information plays a crucial role due to the fact that leads ants to find more optimistic solutions which can affect on the further repetitions [47]. ACO is

an algorithm that is being used in order to solve any optimization problem.

5.1.4 Environment Description

We will create an Environment in which our artificial ants will try to search for almost the optimal solution. This environment is well defined and there are restrictions as well. The searched frame is actually a big array which is set by both variables nodes(HVSs) as well as functions. The nodes represent the columns and functions the rows. This array is dynamic. How big the array will be it depends on the numbers which the user will select for each variable. For easiness, we will set the nodes as "X" and the functions as "Y". So, we have that the environment is equal to X*Y. For example, if we set that there will be 10 nodes and 4 functions, our environment will be like the Table 3 bellow.

Table 3: The hole exploring Environment

1.1[0]	1.2[1]	1.3[1]	1.4[0]
2.1[0]	2.2[0]	2.3[1]	2.4[1]
3.1[1]	3.2[0]	3.3[0]	3.4[1]
4.1[1]	4.2[0]	4.3[0]	4.4[0]
5.1[0]	5.2[1]	5.3[0]	5.4[0]
6.1[0]	6.2[1]	6.3[1]	6.4[0]
7.1[1]	7.2[0]	7.3[0]	7.4[1]
8.1[0]	8.2[0]	8.3[1]	8.4[0]
9.1[0]	9.2[1]	9.3[0]	9.4[1]
10.1[1]	10.2[0]	10.3[1]	10.4[0]

If a node(HVS) can support a function, the respective cell will be replaced by 1, otherwise by 0. To be more precise, let's say that the node 1 can support the functions 2 and 3 this will have as an aftermath of turning the cells 1.2 as well as the 1.3 to 1 and the cells 1.1 and 1.4 to 0. With this logic, an array of 1 and 0 will be created and it will indicate to the artificial ants from where they can pass through. We are interested only for the cells that contain 1s. Based on the previous example, our new environment for exploring is the Table 4 below.

Table 4: Exploring the Environment of "1"

3.1[1]	1.2[1]	1.3[1]	2.4[1]
4.1[1]	5.2[1]	2.3[1]	3.4[1]
7.1[1]	6.2[1]	6.3[1]	7.4[1]
10.1[1]	9.2[1]	8.3[1]	9.4[1]
-	-	10.3[1]	-

Now it's time to set which functions our service needs to maintain in order to be completed. We take that step for granted so we choose randomly which these functions are. Let's assume that our service needs the functions 1, 3 and 4. As a

consequence, our environment is lessened by 1 row and the new one array is like the Table 5 below.

Table 5: Exploring Environment according to the Service

3.1[1]	1.3[1]	2.4[1]
4.1[1]	2.3[1]	3.4[1]
7.1[1]	6.3[1]	7.4[1]
10.1[1]	8.3[1]	9.4[1]
-	10.3[1]	-

For these cells we create randomly values for the processing time, node buffer capacity and for the function buffer demand. The values of the processing time will indicate how much time does a node take in order to execute a function. The node buffer capacity will indicate how many “resources” will the node have in order to process the functions. The function buffer demand will indicate how many “resources” does a function needs in order to be processed by a node. The values of the processing time in milliseconds (ms) are selected randomly in a range of [15,30]. The values for the function buffer demand are selected randomly in a range of [20,30] and the values of the buffer capacity in nodes are selected randomly in a range of [75,100]. For the cells of the Table 5 we create an array in which we will save the values of the pheromone and at the beginning of the experiment all cells will have the value 1.0. In this example, there are $4*5*4 = 80$ possible paths for the ants. Now that we have created the environment, the artificial ants are ready to explore it and find solutions. The procedure that the first ant is taking in order to find a solution is described in steps below:

- (i) It starts from the nest.
- (ii) It will select one of the four possible cells in the first row randomly.
- (iii) Having selected the first cell, it will select one of the five possible cells in the second row randomly.
- (iv) Having selected the second cell, it will select one of the four possible cells in the third row randomly.
- (v) It calculates, from the respective array, the total processing time from the cells that has passed.
- (vi) Saves the value of the total processing time in a variable such as “bestSolution”
- (vii) Update the pheromone values from the cells that the ant passed.

Now that we have defined the “bestSolution”, the following ants are taking the same steps but with different action in step 6. If the new calculated value for the total processing time is less than the “bestSolution” then the value of the variable “bestSolution” is replaced by it and then the step 7 is executed. Otherwise the value

of the “bestSolution” remains the same and no update in pheromone is happen until a new better solution is found.

When a cell is selected, the node buffer capacity of this cell is reduced according to the function buffer demand of this cell. So, there are constraints in the selection of the cells. In order to a node to support a function, the remained node buffer capacity must be bigger than the function buffer demand. If for example there is left 15 of buffer capacity in a node, this node cannot support a function with buffer demand 20. This is a well-defined environment in which our artificial ants will try to find almost the optimal path which is the path with the minimum processing time.

For our experiment we set 100 ants which will start to explore the environment one by one. Each of one will follow the aforementioned procedure. We repeat this over 100 times. For the first 3 repetitions of the experiment we use the first procedure for selecting nodes which is totally random. For the next 97 repetitions, ants are using another procedure which is described below:

- (i) It starts from the nest.
- (ii) It will select one of the four possible cells in the first row according both to the pheromone values as well as to randomness.
- (iii) Having selected the first cell, it will select on the five possible cells in the second row according both to the pheromone values as well as to randomness.
- (iv) Having selected the second cell, it will select on the four possible cells in the third row according both to the pheromone values as well as to randomness.
- (v) It calculates, from the respective array, the total processing time from the cells that has passed.
- (vi) If the value of the total processing time is better than the “bestSolution”, then it is replaced by last calculated value.
- (vii) Update the pheromone values from the cells that the ant passed.

We decided to do that in order to create at the begging more possible paths with pheromone so as to give the chance to the next ants to have more possible paths with pheromone to search.

The mathematical equation of the update of the pheromone is given below:

$$pheromone' = pheromone + \frac{q}{processing}$$

This indicates that the new value of the pheromone (pheromone') in a node is the previous one (pheromone) adding the value of

$$q/processing$$

where q is a constant value and processing is the processing time which a specific function needs in a specific node. Having update all the pheromone values of the cells, we sort the pheromone array so as to put on the top the cells with the highest

values of pheromone. This is because ants are trying to pass firstly from nodes with high pheromone. We say try because as aforementioned above, the selection of a node is depended also in randomness. In order to achieve this randomness, we produce a number between (0,1) which will be compared with

$$\frac{\textit{pheromone}}{\sum \textit{pheromone}}$$

which is the value of the pheromone in a the first shorted cell for a specific function divided by the total amount of pheromone in cells that can support this specific function. If the random number is bigger than calculated value, then the ant will select this node only if the node can support this function with respect to the buffer restrictions. Otherwise, a new value of

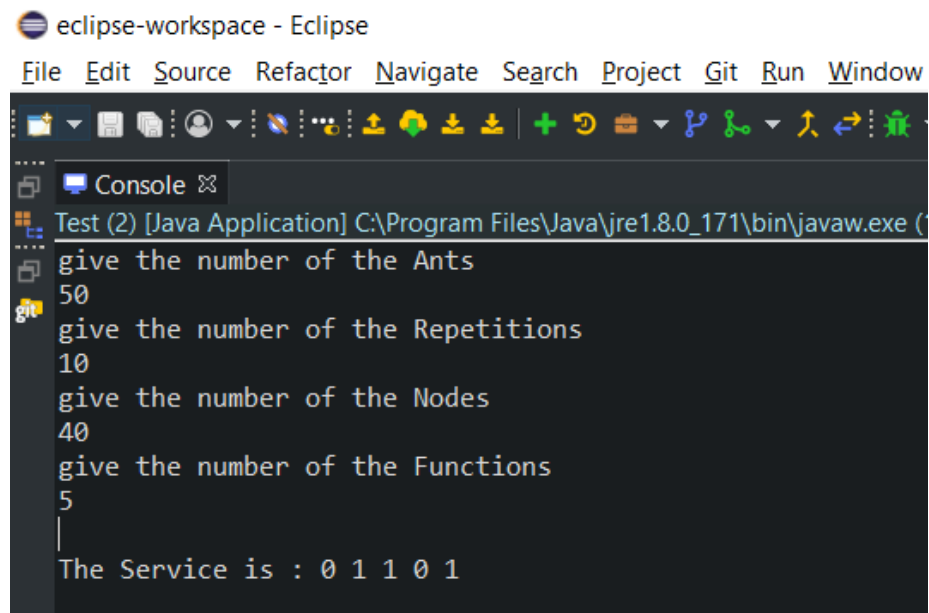
$$\frac{\textit{pheromone}}{\sum \textit{pheromone}}$$

will be calculated based on next shorted cell and a new random number will be produced. If none of these calculated values in the hole row is bigger than the random generated number, the selection of the cellpasses to randomness. Every ant passes through this procedure. The pheromone values in the cells are being updated only if a better solution than the best one is found; the other solution are not taken into consideration. If a node is selected for a function, the buffer capacity of it is subtracted depending on the buffer demand of the function.

6 Chapter 5

6.1 Simulation/Experimentation

In Fig. 9 we can see that when we execute our program, a few parameters are need to be defined. These are the number of the Ants that will search for possible solutions, the number of Repetitions which claims how many times the search for solution will be executed. Furthermore, both the number of Nodes as well as the Functions has to be determined so as to create the search area in which the solutions will be found. The final line of this screenshot defines which Functions the Service will use e.g. from the total five Functions, the Service will use the second the third and the fifth one.



```
eclipse-workspace - Eclipse
File Edit Source Refactor Navigate Search Project Git Run Window
Test (2) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (
give the number of the Ants
50
give the number of the Repetitions
10
give the number of the Nodes
40
give the number of the Functions
5
|
The Service is : 0 1 1 0 1
```

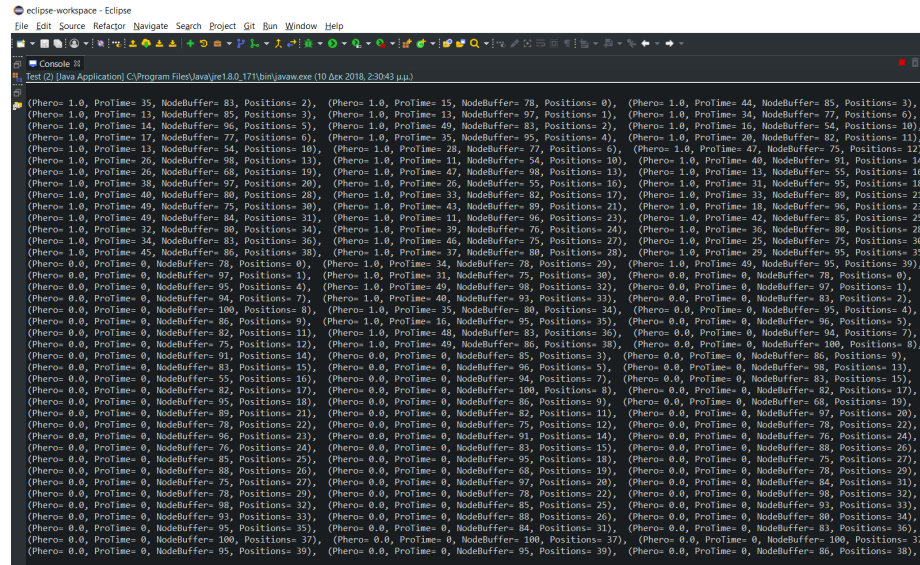
Figure 9: Define Parameters

In Fig. 10 we see a frame with six columns and fifty rows. The first five columns are the total Functions of our problem and the sixth column describes the Buffer that each Node has. Each row is each Node and if this Node can serve a Function the number “1” is displayed with the first “()” describing the processing time of executing this function and with “{ }” describing the buffer needed for the function to be executed. Otherwise, the number “0” is displayed.

```
eclipse-workspace - Eclipse
File Edit Source Refactor Navigate Search Project Git Run Window Help
Test (2) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (10 Δεκ 2018, 2:30:43 μ.μ.)
The Service is : 0 1 1 0 1
0 (0){24}, 0 (0){21}, 1 (15){30}, 1 (18){25}, 0 (0){22}, NodeBuffer= 78
0 (0){24}, 0 (0){21}, 1 (13){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 97
0 (0){24}, 1 (35){21}, 1 (49){30}, 1 (37){25}, 0 (0){22}, NodeBuffer= 83
0 (0){24}, 1 (13){21}, 0 (0){30}, 1 (33){25}, 1 (44){22}, NodeBuffer= 85
0 (0){24}, 0 (0){21}, 1 (35){30}, 1 (44){25}, 0 (0){22}, NodeBuffer= 95
1 (24){24}, 1 (14){21}, 0 (0){30}, 1 (38){25}, 0 (0){22}, NodeBuffer= 96
0 (0){24}, 1 (17){21}, 1 (28){30}, 0 (0){25}, 1 (34){22}, NodeBuffer= 77
0 (0){24}, 0 (0){21}, 0 (0){30}, 1 (41){25}, 0 (0){22}, NodeBuffer= 94
0 (0){24}, 0 (0){21}, 0 (0){30}, 1 (19){25}, 0 (0){22}, NodeBuffer= 100
1 (22){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 86
0 (0){24}, 1 (13){21}, 1 (11){30}, 0 (0){25}, 1 (16){22}, NodeBuffer= 76
0 (0){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 1 (20){22}, NodeBuffer= 82
0 (0){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 1 (47){22}, NodeBuffer= 75
1 (26){24}, 1 (26){21}, 1 (47){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 98
0 (0){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 1 (40){22}, NodeBuffer= 91
1 (15){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 83
1 (23){24}, 0 (0){21}, 1 (26){30}, 0 (0){25}, 1 (13){22}, NodeBuffer= 85
0 (0){24}, 0 (0){21}, 1 (33){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 82
0 (0){24}, 0 (0){21}, 0 (0){30}, 1 (14){25}, 1 (31){22}, NodeBuffer= 95
0 (0){24}, 1 (26){21}, 0 (0){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 89
1 (43){24}, 1 (38){21}, 0 (0){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 97
0 (0){24}, 0 (0){21}, 1 (43){30}, 0 (0){25}, 1 (33){22}, NodeBuffer= 89
0 (0){24}, 0 (0){21}, 0 (0){30}, 1 (38){25}, 0 (0){22}, NodeBuffer= 78
1 (50){24}, 0 (0){21}, 1 (11){30}, 0 (0){25}, 1 (18){22}, NodeBuffer= 96
1 (21){24}, 0 (0){21}, 1 (39){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 76
1 (30){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 1 (42){22}, NodeBuffer= 85
1 (46){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 88
0 (0){24}, 0 (0){21}, 1 (46){30}, 1 (50){25}, 0 (0){22}, NodeBuffer= 75
0 (0){24}, 1 (40){21}, 1 (37){30}, 0 (0){25}, 1 (36){22}, NodeBuffer= 80
0 (0){24}, 0 (0){21}, 1 (34){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 78
0 (0){24}, 1 (49){21}, 1 (31){30}, 0 (0){25}, 1 (25){22}, NodeBuffer= 75
1 (11){24}, 1 (49){21}, 0 (0){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 84
1 (31){24}, 0 (0){21}, 1 (49){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 98
1 (36){24}, 0 (0){21}, 1 (40){30}, 1 (28){25}, 0 (0){22}, NodeBuffer= 93
1 (21){24}, 1 (32){21}, 1 (35){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 80
0 (0){24}, 0 (0){21}, 1 (16){30}, 1 (35){25}, 1 (29){22}, NodeBuffer= 95
0 (0){24}, 1 (34){21}, 1 (48){30}, 1 (43){25}, 0 (0){22}, NodeBuffer= 83
1 (42){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 100
1 (17){24}, 1 (45){21}, 1 (49){30}, 0 (0){25}, 0 (0){22}, NodeBuffer= 86
1 (24){24}, 0 (0){21}, 0 (0){30}, 0 (0){25}, 1 (49){22}, NodeBuffer= 95
```

Figure 10: All Nodes and Functions

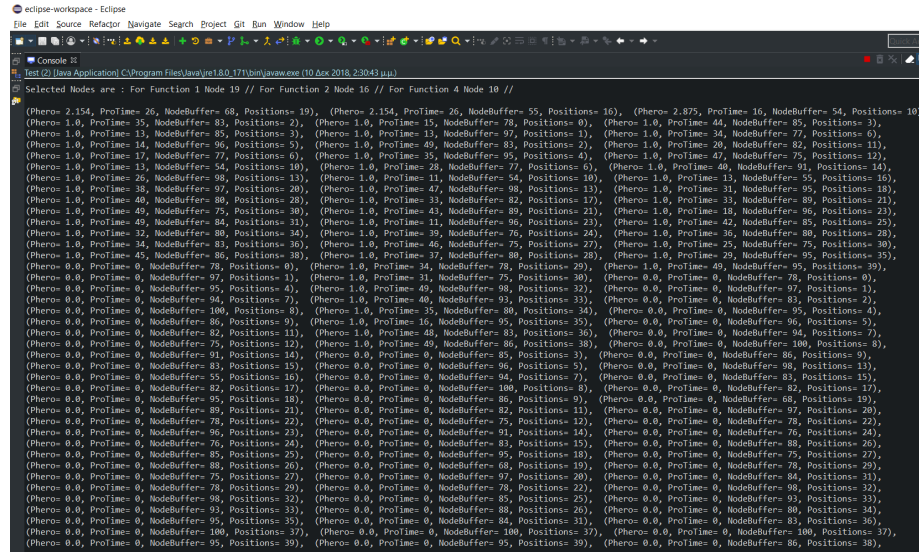
In Fig. 11 we see only the three Functions that the Service needs. Furthermore, we have put the Nodes that can serve Functions at the beginning of the list and then the Nodes that cannot.



```
eclipse-workspace - Eclipse
File Edit Source Refactor Navigate Search Project Git Run Window Help
Test (2) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (10.0x2018.2.2043 μμ)
(Phero- 1.0, ProTime= 35, NodeBuffer= 83, Positions= 2), (Phero- 1.0, ProTime= 15, NodeBuffer= 78, Positions= 0), (Phero- 1.0, ProTime= 44, NodeBuffer= 85, Positions= 3),
(Phero- 1.0, ProTime= 13, NodeBuffer= 85, Positions= 3), (Phero- 1.0, ProTime= 13, NodeBuffer= 97, Positions= 1), (Phero- 1.0, ProTime= 34, NodeBuffer= 77, Positions= 6),
(Phero- 1.0, ProTime= 14, NodeBuffer= 96, Positions= 5), (Phero- 1.0, ProTime= 49, NodeBuffer= 83, Positions= 2), (Phero- 1.0, ProTime= 16, NodeBuffer= 54, Positions= 10),
(Phero- 1.0, ProTime= 17, NodeBuffer= 77, Positions= 6), (Phero- 1.0, ProTime= 35, NodeBuffer= 95, Positions= 4), (Phero- 1.0, ProTime= 49, NodeBuffer= 82, Positions= 11),
(Phero- 1.0, ProTime= 13, NodeBuffer= 54, Positions= 10), (Phero- 1.0, ProTime= 28, NodeBuffer= 77, Positions= 6), (Phero- 1.0, ProTime= 47, NodeBuffer= 75, Positions= 12),
(Phero- 1.0, ProTime= 26, NodeBuffer= 98, Positions= 13), (Phero- 1.0, ProTime= 11, NodeBuffer= 54, Positions= 10), (Phero- 1.0, ProTime= 40, NodeBuffer= 91, Positions= 14),
(Phero- 1.0, ProTime= 26, NodeBuffer= 68, Positions= 19), (Phero- 1.0, ProTime= 47, NodeBuffer= 90, Positions= 13), (Phero- 1.0, ProTime= 13, NodeBuffer= 55, Positions= 16),
(Phero- 1.0, ProTime= 38, NodeBuffer= 97, Positions= 20), (Phero- 1.0, ProTime= 20, NodeBuffer= 55, Positions= 10), (Phero- 1.0, ProTime= 34, NodeBuffer= 95, Positions= 10),
(Phero- 1.0, ProTime= 40, NodeBuffer= 80, Positions= 28), (Phero- 1.0, ProTime= 33, NodeBuffer= 82, Positions= 17), (Phero- 1.0, ProTime= 33, NodeBuffer= 89, Positions= 21),
(Phero- 1.0, ProTime= 49, NodeBuffer= 75, Positions= 30), (Phero- 1.0, ProTime= 43, NodeBuffer= 89, Positions= 21), (Phero- 1.0, ProTime= 18, NodeBuffer= 96, Positions= 23),
(Phero- 1.0, ProTime= 49, NodeBuffer= 84, Positions= 31), (Phero- 1.0, ProTime= 11, NodeBuffer= 96, Positions= 23), (Phero- 1.0, ProTime= 42, NodeBuffer= 85, Positions= 28),
(Phero- 1.0, ProTime= 32, NodeBuffer= 80, Positions= 34), (Phero- 1.0, ProTime= 39, NodeBuffer= 76, Positions= 24), (Phero- 1.0, ProTime= 36, NodeBuffer= 80, Positions= 28),
(Phero- 1.0, ProTime= 34, NodeBuffer= 83, Positions= 36), (Phero- 1.0, ProTime= 46, NodeBuffer= 75, Positions= 27), (Phero- 1.0, ProTime= 25, NodeBuffer= 75, Positions= 30),
(Phero- 1.0, ProTime= 45, NodeBuffer= 85, Positions= 38), (Phero- 1.0, ProTime= 37, NodeBuffer= 80, Positions= 20), (Phero- 1.0, ProTime= 29, NodeBuffer= 95, Positions= 35),
(Phero- 0.0, ProTime= 0, NodeBuffer= 78, Positions= 0), (Phero- 1.0, ProTime= 34, NodeBuffer= 78, Positions= 20), (Phero- 1.0, ProTime= 49, NodeBuffer= 95, Positions= 39),
(Phero- 0.0, ProTime= 0, NodeBuffer= 97, Positions= 1), (Phero- 1.0, ProTime= 31, NodeBuffer= 75, Positions= 30), (Phero- 0.0, ProTime= 0, NodeBuffer= 78, Positions= 0),
(Phero- 0.0, ProTime= 0, NodeBuffer= 95, Positions= 4), (Phero- 1.0, ProTime= 49, NodeBuffer= 98, Positions= 32), (Phero- 0.0, ProTime= 0, NodeBuffer= 97, Positions= 1),
(Phero- 0.0, ProTime= 0, NodeBuffer= 94, Positions= 7), (Phero- 1.0, ProTime= 40, NodeBuffer= 93, Positions= 32), (Phero- 0.0, ProTime= 0, NodeBuffer= 83, Positions= 2),
(Phero- 0.0, ProTime= 0, NodeBuffer= 86, Positions= 9), (Phero- 1.0, ProTime= 16, NodeBuffer= 95, Positions= 35), (Phero- 0.0, ProTime= 0, NodeBuffer= 96, Positions= 5),
(Phero- 0.0, ProTime= 0, NodeBuffer= 82, Positions= 11), (Phero- 1.0, ProTime= 48, NodeBuffer= 83, Positions= 36), (Phero- 0.0, ProTime= 0, NodeBuffer= 94, Positions= 7),
(Phero- 0.0, ProTime= 0, NodeBuffer= 75, Positions= 12), (Phero- 1.0, ProTime= 49, NodeBuffer= 86, Positions= 38), (Phero- 0.0, ProTime= 0, NodeBuffer= 100, Positions= 8),
(Phero- 0.0, ProTime= 0, NodeBuffer= 91, Positions= 14), (Phero- 0.0, ProTime= 0, NodeBuffer= 85, Positions= 3), (Phero- 0.0, ProTime= 0, NodeBuffer= 86, Positions= 9),
(Phero- 0.0, ProTime= 0, NodeBuffer= 83, Positions= 15), (Phero- 0.0, ProTime= 0, NodeBuffer= 96, Positions= 5), (Phero- 0.0, ProTime= 0, NodeBuffer= 98, Positions= 13),
(Phero- 0.0, ProTime= 0, NodeBuffer= 55, Positions= 16), (Phero- 0.0, ProTime= 0, NodeBuffer= 94, Positions= 7), (Phero- 0.0, ProTime= 0, NodeBuffer= 83, Positions= 15),
(Phero- 0.0, ProTime= 0, NodeBuffer= 82, Positions= 17), (Phero- 0.0, ProTime= 0, NodeBuffer= 100, Positions= 8), (Phero- 0.0, ProTime= 0, NodeBuffer= 82, Positions= 17),
(Phero- 0.0, ProTime= 0, NodeBuffer= 95, Positions= 18), (Phero- 0.0, ProTime= 0, NodeBuffer= 86, Positions= 9), (Phero- 0.0, ProTime= 0, NodeBuffer= 68, Positions= 19),
(Phero- 0.0, ProTime= 0, NodeBuffer= 89, Positions= 21), (Phero- 0.0, ProTime= 0, NodeBuffer= 82, Positions= 11), (Phero- 0.0, ProTime= 0, NodeBuffer= 97, Positions= 20),
(Phero- 0.0, ProTime= 0, NodeBuffer= 78, Positions= 22), (Phero- 0.0, ProTime= 0, NodeBuffer= 79, Positions= 12), (Phero- 0.0, ProTime= 0, NodeBuffer= 78, Positions= 22),
(Phero- 0.0, ProTime= 0, NodeBuffer= 96, Positions= 23), (Phero- 0.0, ProTime= 0, NodeBuffer= 91, Positions= 14), (Phero- 0.0, ProTime= 0, NodeBuffer= 76, Positions= 24),
(Phero- 0.0, ProTime= 0, NodeBuffer= 76, Positions= 24), (Phero- 0.0, ProTime= 0, NodeBuffer= 83, Positions= 15), (Phero- 0.0, ProTime= 0, NodeBuffer= 88, Positions= 26),
(Phero- 0.0, ProTime= 0, NodeBuffer= 85, Positions= 25), (Phero- 0.0, ProTime= 0, NodeBuffer= 95, Positions= 18), (Phero- 0.0, ProTime= 0, NodeBuffer= 75, Positions= 27),
(Phero- 0.0, ProTime= 0, NodeBuffer= 88, Positions= 26), (Phero- 0.0, ProTime= 0, NodeBuffer= 68, Positions= 19), (Phero- 0.0, ProTime= 0, NodeBuffer= 78, Positions= 20),
(Phero- 0.0, ProTime= 0, NodeBuffer= 75, Positions= 27), (Phero- 0.0, ProTime= 0, NodeBuffer= 97, Positions= 20), (Phero- 0.0, ProTime= 0, NodeBuffer= 84, Positions= 31),
(Phero- 0.0, ProTime= 0, NodeBuffer= 78, Positions= 29), (Phero- 0.0, ProTime= 0, NodeBuffer= 78, Positions= 22), (Phero- 0.0, ProTime= 0, NodeBuffer= 98, Positions= 32),
(Phero- 0.0, ProTime= 0, NodeBuffer= 98, Positions= 32), (Phero- 0.0, ProTime= 0, NodeBuffer= 85, Positions= 25), (Phero- 0.0, ProTime= 0, NodeBuffer= 93, Positions= 13),
(Phero- 0.0, ProTime= 0, NodeBuffer= 93, Positions= 33), (Phero- 0.0, ProTime= 0, NodeBuffer= 88, Positions= 26), (Phero- 0.0, ProTime= 0, NodeBuffer= 80, Positions= 34),
(Phero- 0.0, ProTime= 0, NodeBuffer= 95, Positions= 35), (Phero- 0.0, ProTime= 0, NodeBuffer= 84, Positions= 31), (Phero- 0.0, ProTime= 0, NodeBuffer= 83, Positions= 36),
(Phero- 0.0, ProTime= 0, NodeBuffer= 100, Positions= 37), (Phero- 0.0, ProTime= 0, NodeBuffer= 100, Positions= 37), (Phero- 0.0, ProTime= 0, NodeBuffer= 100, Positions= 37),
(Phero- 0.0, ProTime= 0, NodeBuffer= 95, Positions= 39), (Phero- 0.0, ProTime= 0, NodeBuffer= 95, Positions= 39), (Phero- 0.0, ProTime= 0, NodeBuffer= 86, Positions= 38),
```

Figure 11: Services and Nodes

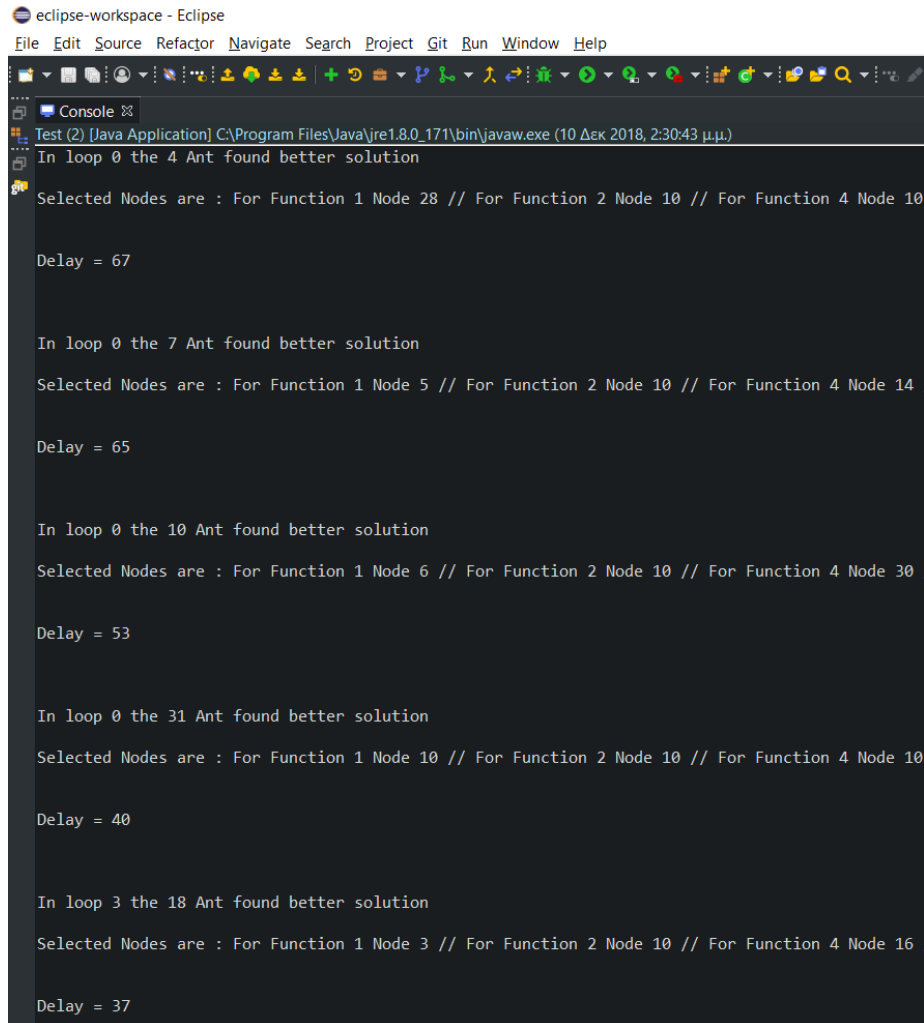
In Fig. 12 we can see that our program selects randomly a Node for each Function. After the selection, the frame has been sorted according to the pheromone values of each Node. In the first line are the Nodes that have been selected because their pheromone values are bigger than the other.



```
eclipse-workspace - Eclipse
File Edit Source Refactor Navigate Search Project Git Run Window Help
Console
Test(2) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (10 Dec 2018, 2:30:43 pm)
Selected Nodes are : For Function 1 Node 19 // For Function 2 Node 16 // For Function 4 Node 10 //
(Phero= 2.154, ProTime= 26, NodeBuffer= 68, Positions= 19), (Phero= 2.154, ProTime= 26, NodeBuffer= 55, Positions= 16), (Phero= 2.875, ProTime= 16, NodeBuffer= 54, Positions= 10),
(Phero= 1.0, ProTime= 35, NodeBuffer= 83, Positions= 2), (Phero= 1.0, ProTime= 15, NodeBuffer= 78, Positions= 0), (Phero= 1.0, ProTime= 44, NodeBuffer= 85, Positions= 3),
(Phero= 1.0, ProTime= 13, NodeBuffer= 85, Positions= 3), (Phero= 1.0, ProTime= 13, NodeBuffer= 97, Positions= 1), (Phero= 1.0, ProTime= 34, NodeBuffer= 77, Positions= 6),
(Phero= 1.0, ProTime= 14, NodeBuffer= 96, Positions= 5), (Phero= 1.0, ProTime= 49, NodeBuffer= 83, Positions= 2), (Phero= 1.0, ProTime= 20, NodeBuffer= 82, Positions= 11),
(Phero= 1.0, ProTime= 17, NodeBuffer= 77, Positions= 6), (Phero= 1.0, ProTime= 35, NodeBuffer= 95, Positions= 4), (Phero= 1.0, ProTime= 47, NodeBuffer= 75, Positions= 12),
(Phero= 1.0, ProTime= 13, NodeBuffer= 54, Positions= 10), (Phero= 1.0, ProTime= 28, NodeBuffer= 77, Positions= 6), (Phero= 1.0, ProTime= 40, NodeBuffer= 91, Positions= 14),
(Phero= 1.0, ProTime= 26, NodeBuffer= 98, Positions= 13), (Phero= 1.0, ProTime= 11, NodeBuffer= 54, Positions= 10), (Phero= 1.0, ProTime= 13, NodeBuffer= 55, Positions= 16),
(Phero= 1.0, ProTime= 38, NodeBuffer= 97, Positions= 20), (Phero= 1.0, ProTime= 47, NodeBuffer= 98, Positions= 13), (Phero= 1.0, ProTime= 31, NodeBuffer= 95, Positions= 18),
(Phero= 1.0, ProTime= 48, NodeBuffer= 80, Positions= 28), (Phero= 1.0, ProTime= 33, NodeBuffer= 82, Positions= 17), (Phero= 1.0, ProTime= 33, NodeBuffer= 89, Positions= 21),
(Phero= 1.0, ProTime= 49, NodeBuffer= 75, Positions= 20), (Phero= 1.0, ProTime= 41, NodeBuffer= 89, Positions= 23), (Phero= 1.0, ProTime= 18, NodeBuffer= 96, Positions= 23),
(Phero= 1.0, ProTime= 49, NodeBuffer= 84, Positions= 31), (Phero= 1.0, ProTime= 11, NodeBuffer= 96, Positions= 23), (Phero= 1.0, ProTime= 42, NodeBuffer= 85, Positions= 25),
(Phero= 1.0, ProTime= 32, NodeBuffer= 80, Positions= 34), (Phero= 1.0, ProTime= 39, NodeBuffer= 76, Positions= 24), (Phero= 1.0, ProTime= 36, NodeBuffer= 80, Positions= 28),
(Phero= 1.0, ProTime= 24, NodeBuffer= 83, Positions= 30), (Phero= 1.0, ProTime= 46, NodeBuffer= 75, Positions= 27), (Phero= 1.0, ProTime= 25, NodeBuffer= 75, Positions= 38),
(Phero= 1.0, ProTime= 45, NodeBuffer= 86, Positions= 38), (Phero= 1.0, ProTime= 37, NodeBuffer= 80, Positions= 28), (Phero= 1.0, ProTime= 29, NodeBuffer= 95, Positions= 35),
(Phero= 0.0, ProTime= 0, NodeBuffer= 78, Positions= 0), (Phero= 1.0, ProTime= 34, NodeBuffer= 78, Positions= 29), (Phero= 1.0, ProTime= 49, NodeBuffer= 95, Positions= 39),
(Phero= 0.0, ProTime= 0, NodeBuffer= 97, Positions= 1), (Phero= 1.0, ProTime= 31, NodeBuffer= 75, Positions= 30), (Phero= 0.0, ProTime= 0, NodeBuffer= 78, Positions= 0),
(Phero= 0.0, ProTime= 0, NodeBuffer= 95, Positions= 4), (Phero= 1.0, ProTime= 49, NodeBuffer= 98, Positions= 32), (Phero= 0.0, ProTime= 0, NodeBuffer= 97, Positions= 1),
(Phero= 0.0, ProTime= 0, NodeBuffer= 94, Positions= 7), (Phero= 1.0, ProTime= 40, NodeBuffer= 93, Positions= 33), (Phero= 0.0, ProTime= 0, NodeBuffer= 83, Positions= 2),
(Phero= 0.0, ProTime= 0, NodeBuffer= 100, Positions= 8), (Phero= 1.0, ProTime= 35, NodeBuffer= 80, Positions= 34), (Phero= 0.0, ProTime= 0, NodeBuffer= 95, Positions= 4),
(Phero= 0.0, ProTime= 0, NodeBuffer= 86, Positions= 9), (Phero= 1.0, ProTime= 16, NodeBuffer= 95, Positions= 39), (Phero= 0.0, ProTime= 0, NodeBuffer= 96, Positions= 5),
(Phero= 0.0, ProTime= 0, NodeBuffer= 82, Positions= 11), (Phero= 1.0, ProTime= 48, NodeBuffer= 83, Positions= 36), (Phero= 0.0, ProTime= 0, NodeBuffer= 94, Positions= 7),
(Phero= 0.0, ProTime= 0, NodeBuffer= 75, Positions= 12), (Phero= 1.0, ProTime= 49, NodeBuffer= 86, Positions= 38), (Phero= 0.0, ProTime= 0, NodeBuffer= 106, Positions= 8),
(Phero= 0.0, ProTime= 0, NodeBuffer= 91, Positions= 16), (Phero= 0.0, ProTime= 0, NodeBuffer= 85, Positions= 3), (Phero= 0.0, ProTime= 0, NodeBuffer= 86, Positions= 9),
(Phero= 0.0, ProTime= 0, NodeBuffer= 83, Positions= 15), (Phero= 0.0, ProTime= 0, NodeBuffer= 96, Positions= 5), (Phero= 0.0, ProTime= 0, NodeBuffer= 98, Positions= 13),
(Phero= 0.0, ProTime= 0, NodeBuffer= 55, Positions= 16), (Phero= 0.0, ProTime= 0, NodeBuffer= 94, Positions= 7), (Phero= 0.0, ProTime= 0, NodeBuffer= 83, Positions= 15),
(Phero= 0.0, ProTime= 0, NodeBuffer= 82, Positions= 17), (Phero= 0.0, ProTime= 0, NodeBuffer= 100, Positions= 8), (Phero= 0.0, ProTime= 0, NodeBuffer= 83, Positions= 15),
(Phero= 0.0, ProTime= 0, NodeBuffer= 89, Positions= 21), (Phero= 0.0, ProTime= 0, NodeBuffer= 82, Positions= 11), (Phero= 0.0, ProTime= 0, NodeBuffer= 97, Positions= 20),
(Phero= 0.0, ProTime= 0, NodeBuffer= 78, Positions= 22), (Phero= 0.0, ProTime= 0, NodeBuffer= 75, Positions= 12), (Phero= 0.0, ProTime= 0, NodeBuffer= 78, Positions= 22),
(Phero= 0.0, ProTime= 0, NodeBuffer= 96, Positions= 23), (Phero= 0.0, ProTime= 0, NodeBuffer= 91, Positions= 14), (Phero= 0.0, ProTime= 0, NodeBuffer= 76, Positions= 24),
(Phero= 0.0, ProTime= 0, NodeBuffer= 76, Positions= 24), (Phero= 0.0, ProTime= 0, NodeBuffer= 83, Positions= 15), (Phero= 0.0, ProTime= 0, NodeBuffer= 88, Positions= 26),
(Phero= 0.0, ProTime= 0, NodeBuffer= 85, Positions= 25), (Phero= 0.0, ProTime= 0, NodeBuffer= 95, Positions= 18), (Phero= 0.0, ProTime= 0, NodeBuffer= 75, Positions= 22),
(Phero= 0.0, ProTime= 0, NodeBuffer= 88, Positions= 26), (Phero= 0.0, ProTime= 0, NodeBuffer= 68, Positions= 19), (Phero= 0.0, ProTime= 0, NodeBuffer= 78, Positions= 20),
(Phero= 0.0, ProTime= 0, NodeBuffer= 75, Positions= 22), (Phero= 0.0, ProTime= 0, NodeBuffer= 97, Positions= 20), (Phero= 0.0, ProTime= 0, NodeBuffer= 84, Positions= 31),
(Phero= 0.0, ProTime= 0, NodeBuffer= 78, Positions= 29), (Phero= 0.0, ProTime= 0, NodeBuffer= 78, Positions= 22), (Phero= 0.0, ProTime= 0, NodeBuffer= 98, Positions= 32),
(Phero= 0.0, ProTime= 0, NodeBuffer= 98, Positions= 32), (Phero= 0.0, ProTime= 0, NodeBuffer= 85, Positions= 25), (Phero= 0.0, ProTime= 0, NodeBuffer= 83, Positions= 33),
(Phero= 0.0, ProTime= 0, NodeBuffer= 93, Positions= 33), (Phero= 0.0, ProTime= 0, NodeBuffer= 88, Positions= 26), (Phero= 0.0, ProTime= 0, NodeBuffer= 80, Positions= 34),
(Phero= 0.0, ProTime= 0, NodeBuffer= 95, Positions= 35), (Phero= 0.0, ProTime= 0, NodeBuffer= 84, Positions= 31), (Phero= 0.0, ProTime= 0, NodeBuffer= 83, Positions= 36),
(Phero= 0.0, ProTime= 0, NodeBuffer= 100, Positions= 37), (Phero= 0.0, ProTime= 0, NodeBuffer= 100, Positions= 37), (Phero= 0.0, ProTime= 0, NodeBuffer= 100, Positions= 37),
(Phero= 0.0, ProTime= 0, NodeBuffer= 95, Positions= 39), (Phero= 0.0, ProTime= 0, NodeBuffer= 95, Positions= 39), (Phero= 0.0, ProTime= 0, NodeBuffer= 86, Positions= 38),
```

Figure 12: Selection randomly of Nodes

In Fig. 13 we see the point where the solutions are being produced with details in the exact Ant and Repetition as well as in the Nodes that have been selected.



```
eclipse-workspace - Eclipse
File Edit Source Refactor Navigate Search Project Git Run Window Help
Test (2) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (10 Δεκ 2018, 2:30:43 μ.μ.)
In loop 0 the 4 Ant found better solution
Selected Nodes are : For Function 1 Node 28 // For Function 2 Node 10 // For Function 4 Node 10
Delay = 67
In loop 0 the 7 Ant found better solution
Selected Nodes are : For Function 1 Node 5 // For Function 2 Node 10 // For Function 4 Node 14
Delay = 65
In loop 0 the 10 Ant found better solution
Selected Nodes are : For Function 1 Node 6 // For Function 2 Node 10 // For Function 4 Node 30
Delay = 53
In loop 0 the 31 Ant found better solution
Selected Nodes are : For Function 1 Node 10 // For Function 2 Node 10 // For Function 4 Node 10
Delay = 40
In loop 3 the 18 Ant found better solution
Selected Nodes are : For Function 1 Node 3 // For Function 2 Node 10 // For Function 4 Node 16
Delay = 37
```

Figure 13: Produced Solutions

In Fig. 14 we see the final Nodes that have been selected to serve the Functions 2, 3 and 5 with their respective details. We display two different solutions as we have identified that the final best solution that the program found can be different from the total delay of the Nodes with the biggest Pheromone. In some cases the one solution is better than the other and vice versa.

```

There were 5 better solutions
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
[Display The Arrays With Respect To The Final Best Solution]
(Phero= 3.308, ProTime= 13, NodeBuffer= 64, Positions= 3), Minimum Proc.Time is :13
(Phero= 14.635, ProTime= 11, NodeBuffer= 46, Positions= 10), Minimum Proc.Time is :11
(Phero= 3.308, ProTime= 13, NodeBuffer= 63, Positions= 16), Minimum Proc.Time is :13
The Final Delay = 37
The minimum Delay = 37
Stats for 0 loops are 100.0%
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
[Display The Arrays With Respect To The Firsr Position Of It]
(Phero= 3.308, ProTime= 13, NodeBuffer= 3, Positions= 10), Minimum Proc.Time is :13
(Phero= 14.635, ProTime= 11, NodeBuffer= 3, Positions= 10), Minimum Proc.Time is :11
(Phero= 6.625, ProTime= 16, NodeBuffer= 3, Positions= 10), Minimum Proc.Time is :13
The Final Delay of Firsr Positions = 40
The minimum Delay = 37
Stats for 1 loop of 1st Positions are 92.5%
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
After 1 loops there was a 100.0% of perfections

```

Figure 14: The two final Solutions

And we achieved to find the 100% optimal solution in this experiment

At this point we will display the fluctuation of the delay according to different parameters of the program. For the sake of the diagram we will use one Ant and one hundred Repetitions. There will be four or in some cases three different parameters. The yellow line is the Minimum Delay which in the specific experiment can be found. The red line is the value of Delay in each Repetition that our program found. The green line is the best value for the Delay that was found until that Repetition. And lastly, the black line is the Delay of the Nodes with the highest pheromone.

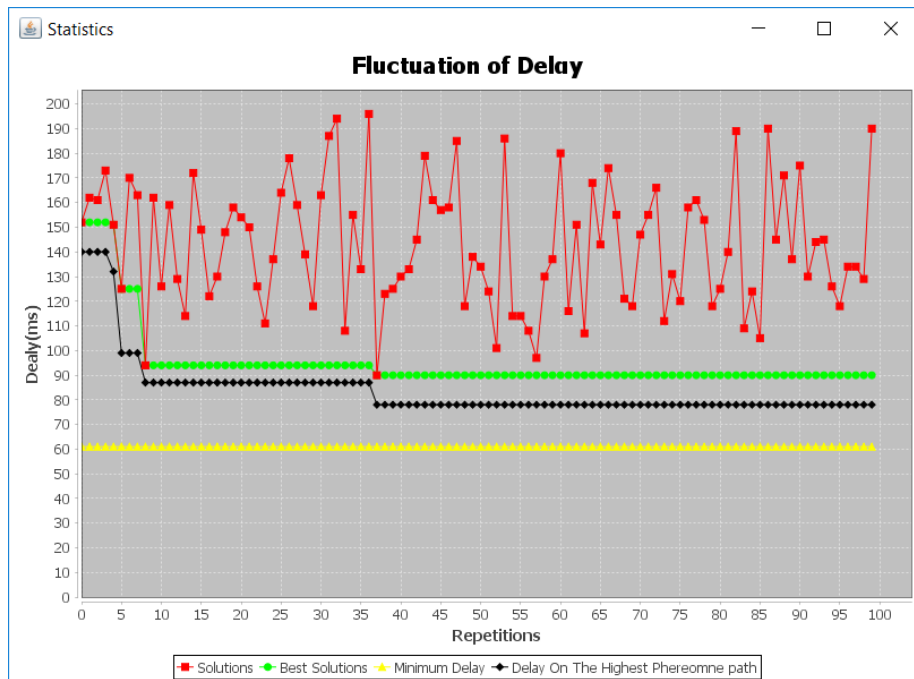


Figure 15: The first experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
50	5	30	40
Percentage of optimal approached delay according to Final Best Solution	67.7%	Percentage of optimal approached delay according to highest Pheromone values	78.20%

Table 6: Parameters of 1st experiment

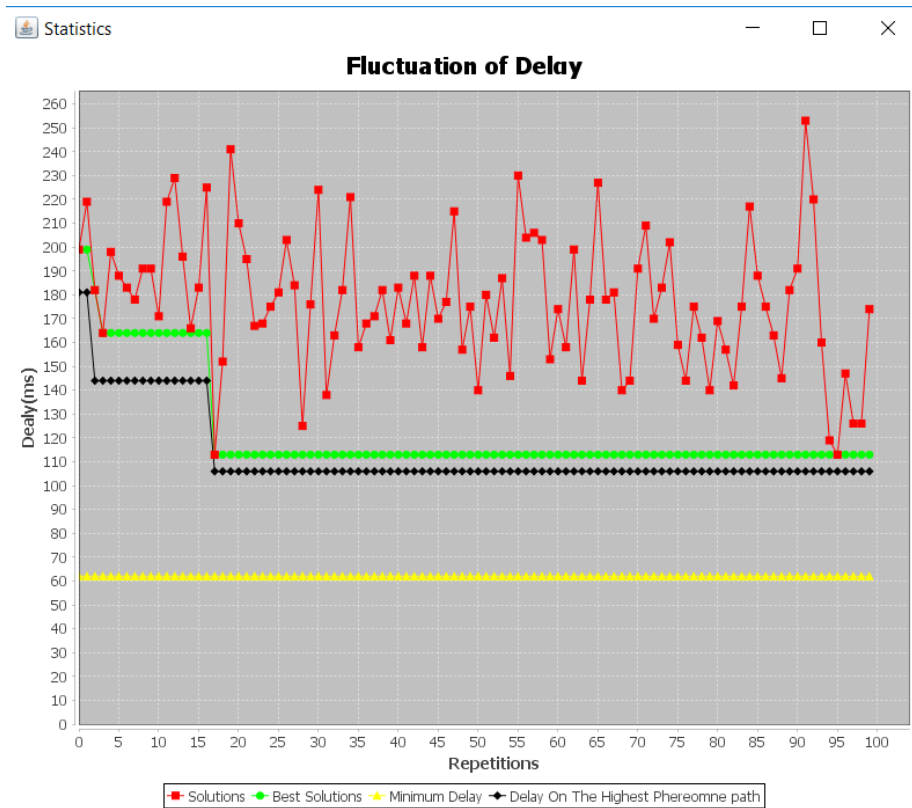


Figure 16: The second experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	30	40
Percentage of optimal approached delay according to Final Best Solution	54.86%	Percentage of optimal approached delay according to highest Pheromone values	58.49%

Table 7: Parameters of 2nd experiment

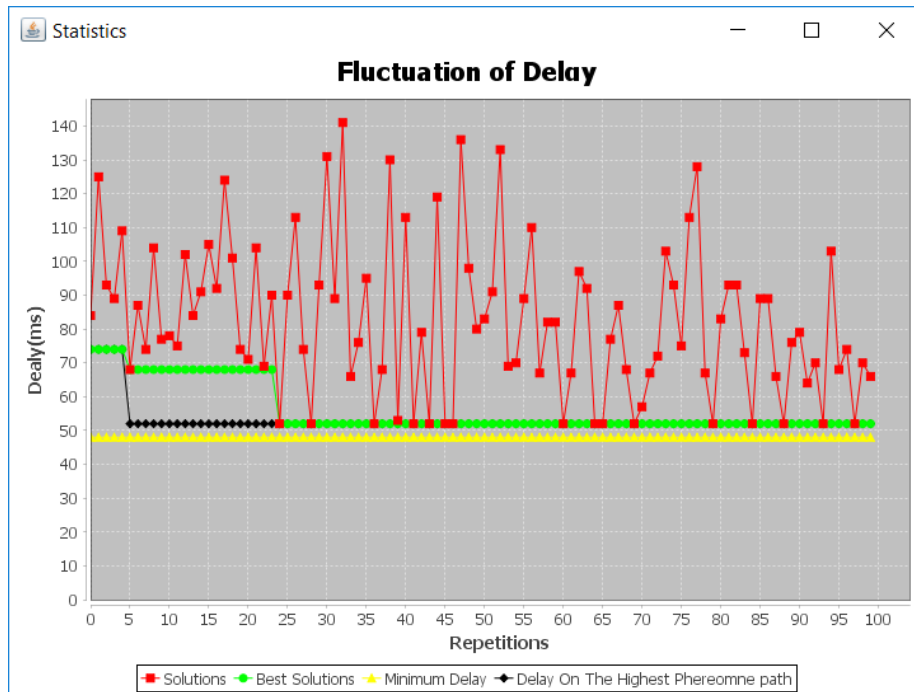


Figure 17: The third experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
50	5	120	40
Percentage of optimal approached delay according to Final Best Solution	92.3%	Percentage of optimal approached delay according to highest Pheromone values	92.3%

Table 8: Parameters of 3rd experiment

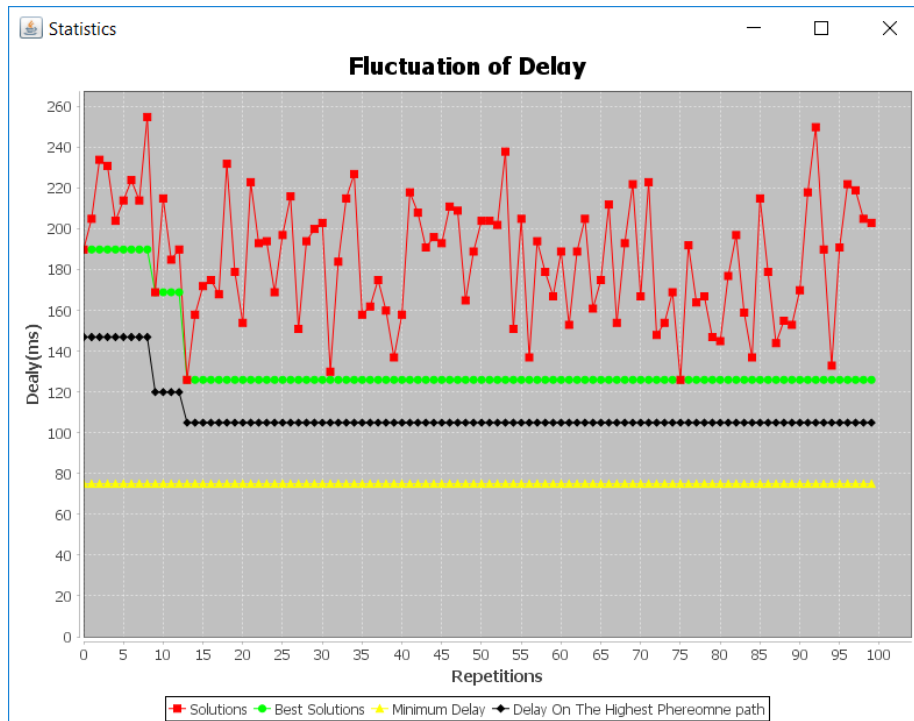


Figure 18: The fourth experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	120	40
Percentage of optimal approached delay according to Final Best Solution	959.52%	Percentage of optimal approached delay according to highest Pheromone values	71.42%

Table 9: Parameters of 4th experiment

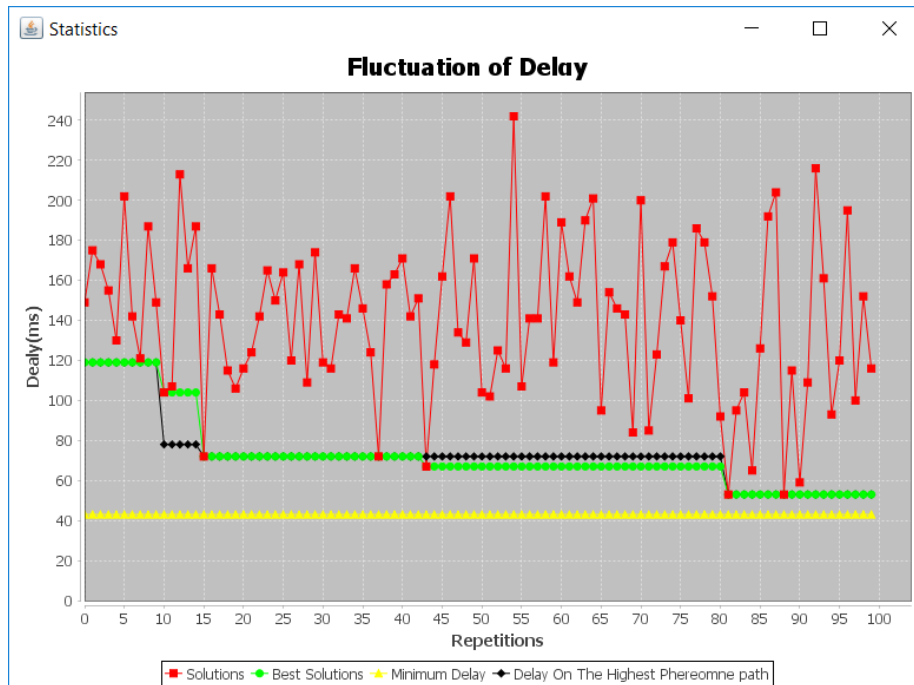


Figure 19: The fifth experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
50	5	30	80
Percentage of optimal approached delay according to Final Best Solution	81.13%	Percentage of optimal approached delay according to highest Pheromone values	81.13%

Table 10: Parameters of 5th experiment

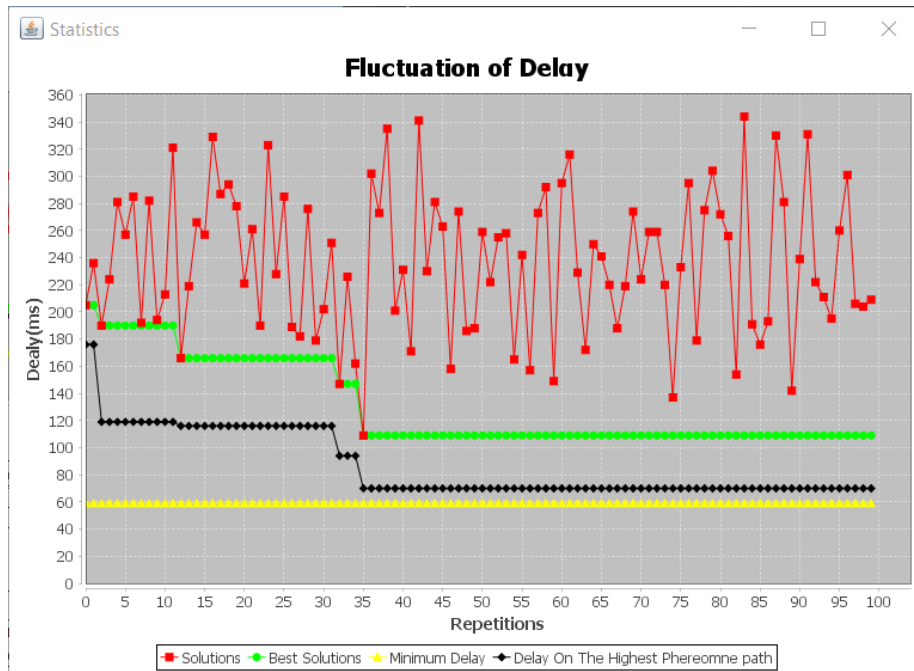


Figure 20: The sixth experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	30	80
Percentage of optimal approached delay according to Final Best Solution	54.14%	Percentage of optimal approached delay according to highest Pheromone values	84.28%

Table 11: Parameters of 6th experiment

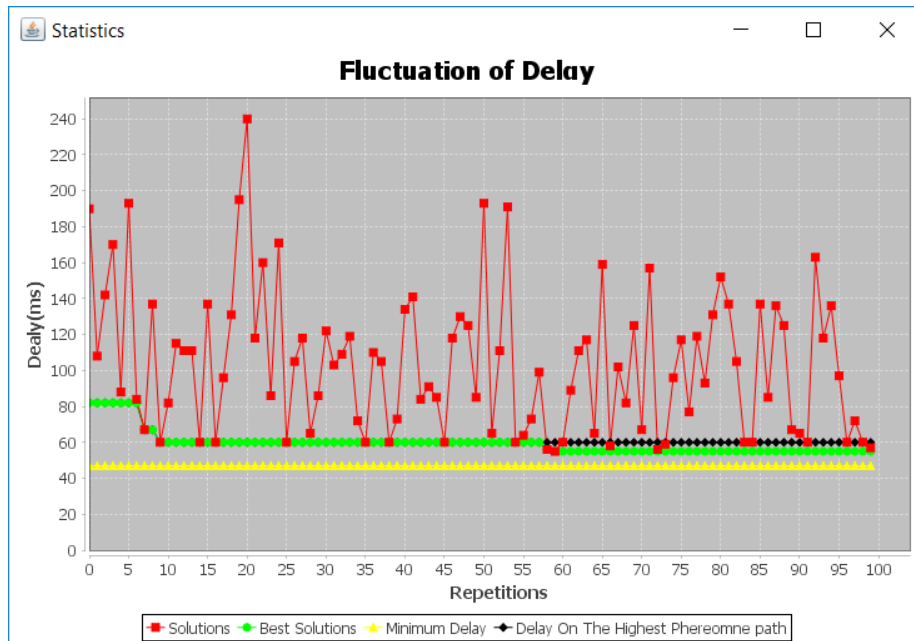


Figure 21: The seventh experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
50	5	120	80
Percentage of optimal approached delay according to Final Best Solution	85.45%	Percentage of optimal approached delay according to highest Pheromone values	78.33%

Table 12: Parameters of 7th experiment

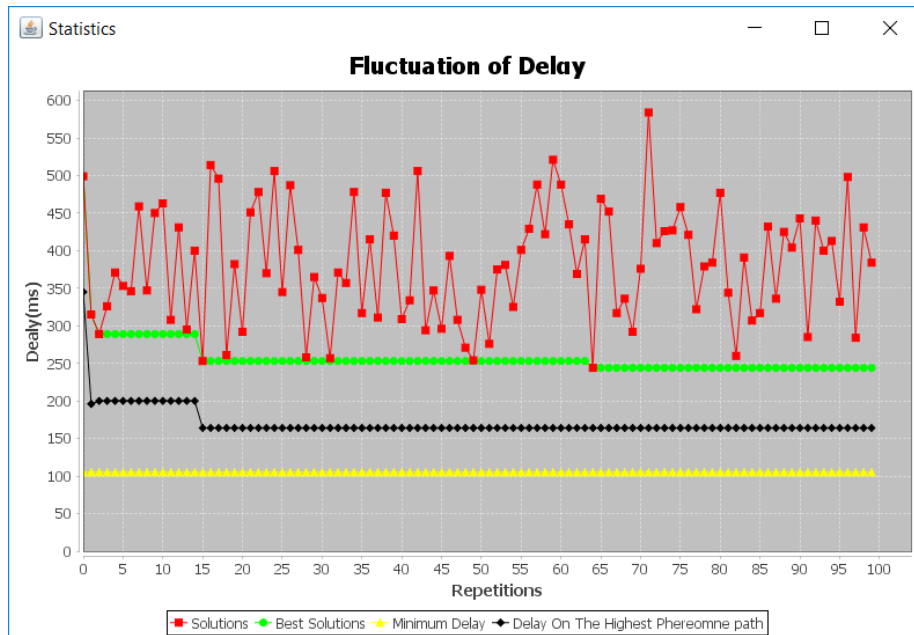


Figure 22: The eighth experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	120	80
Percentage of optimal approached delay according to Final Best Solution	43.03%	Percentage of optimal approached delay according to highest Pheromone values	64.02%

Table 13: Parameters of 8th experiment

Now we will present some extra statistics. We will generate 1000 different experiments with the same parameters and we will take the average percentage of the optimal approach. The number of the Ants as well as the Repetitions will be 100.

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	120	40
Percentage of optimal approached delay with 1000 different experiments	91.26%		

Table 14: Parameters of 9th experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	30	40
Percentage of optimal approached delay with 1000 different experiments	87.20%		

Table 15: Parameters of 10th experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	30	80
Percentage of optimal approached delay with 1000 different experiments	81.68%		

Table 16: Parameters of 11th experiment

# of Nodes	#of Functions	value of constant "q"	Difference in min and max of the processing delay
100	10	120	80
Percentage of optimal approached delay with 1000 different experiments	87.93%		

Table 17: Parameters of 12th experiment

7 Conclusions

A research has been made in the area of Network Function Virtualization and at the same time in the meta-heuristic algorithm Ant Colony Optimization for a potential usage in Telecommunications. By combining these two together we have achieved to solve one of the three main problems that arise in NFV, the Resource Allocation. To solve the RA we had to find how to allocate the resources so as to have the biggest advantage in the minimum delay in our case. In order to achieve this we had to find which Nodes will serve which Functions. For the easiness of the problem we generate a random array with 0 and 1. The 1 in a cell of the array says that the Node which is the column will serve the Function which is the row, otherwise it will not. In this environment we had to approach almost the optimal solution as far as the delay is concerned by using the ACO algorithm which is playing the crucial role, the role of the solution searcher. According to the parameters that the problem is defined with, the result of the solutions are changing. With some parameters that we found in a paper e.g. (#of Ants = 100, #of Repetitions = 100, #of Nodes = 100, #of Functions = 10, $q = 30$, $diff = 15$) our program finds a solution that approached the optimal by 90 %. Of course by altering the parameters the results are changing respectively. If we increase the #of Ants or the # of Repetitions, it is likelihood to have better results because the solution environment will be searched further. Additionally, if we increase the #of Nodes or the #of Functions it is likelihood to have worse results because the solution environment will be scaled up further. Furthermore, if we increase the difference between the min and the max of the value in processing time the results that we will get will be worse because it will be more difficult to identify the minimum values in the solution environment. Last but not least, the value q seems to play an important role as well. It is the value that adds up the value of the pheromone in a path when a better solution is found. From the experiments that have been made, it is shown that if we increase this value we get better result than when we decrease it.

References

- [1] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, p. 1114–1128, Fourth 2012.
- [2] A. Fischer, J.-F. Botero, M. T. Beck, H. de Meer, , and X. Hesselbach, "Virtual Network Embedding: A Survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, p. 1888–1906, Fourth 2013.
- [3] J. F. Botero, M. Molina, X. Hesselbach-Serra, and J. R. Amazonas, "A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems," *Journal of Network and Computer Applications*, no. 0, p. –, 2013, accessed: 2018-01-10. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1084804513000672>
- [4] B.Wang, X. Chang, J. Liu, and J. Muppala, "Reducing power consumption in embedding virtual infrastructures," *Globecom Workshops (GCWkshps), 2012 IEEE*, p. 714–718, Dec 2012.
- [5] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration," *SIGCOMM Comput. Commun. Rev*, vol. 38, no. 2, p. 17–29, 2008.
- [6] Y. Chen, J. Li, T. Wo, C. Hu, and W. Liu, "Resilient Virtual Network Service Provision in Network Virtualization Environments," *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference*, p. 51–58, Dec 2010.
- [7] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, p. 213–220, 2016.
- [8] J. F. Botero, X. Hesselbach, M. Duelli *et al.*, "Energy efficient virtual network embedding," *IEEE Communications Letters*, vol. 16, no. 5, p. 756–759, May 2012.
- [9] J. F. Botero and X. Hesselbach, "Greener networking in a network virtualization environment," *Computer Networks*, vol. 57, no. 9, p. 2021–2029, 2013.
- [10] S. L., Z. C., H. Xu, and M. Xu, "Towards security-aware virtual network embedding," *Computer Networks*, vol. 91, p. 151–163, 2015.
- [11] S. Liu, Z. Cai, H. Xu, and M. Xu, "Security-aware virtual network embedding," *Communications (ICC), 2014 IEEE International Conference*, p. 834–840, June 2014.
- [12] R. Mijumbi, J. Serrat, J. Gorricho, N. Boutenand *et al.*, "Network function virtualization: State-of-the-art and research challenges. (German) [On the electrodynamics of moving bodies]," *Communications Surveys Tutorials, IEEE*, vol. 18, no. 1, p. 236–262, Firstquarter 2016.

- [13] A. Gember-Jacobson, A. Krishnamurthy, S. S. John, R. Grandl *et al.*, "Stratos: A networkaware orchestration layer for middleboxes in the cloud," *CoRR*, vol. abs/1305.0209, 2013.
- [14] M. F. Bari, S. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions. [In in network and service management(cnsm)]," *11th International Conference*, p. 50–56, Nov 2015.
- [15] E. Hernandez-Valencia, S. Izzo, and B. Polonsky, "How will NFV/SDN transform service provider opex?" *Network, IEEE*, vol. 29, no. 3, p. 60–67, May 2015.
- [16] M. Beck and J. F. Botero, "Coordinated allocation of service function chains. (San Diego, USA) [In2015 ieee global communications conference: Selected areas in communications: Software defined networking and network functions]," Dec 2015.
- [17] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten *et al.*, "Network function virtualization: State-of-the-art and research challenges," 2015.
- [18] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latre *et al.*, "Management and orchestration challenges in network function virtualization," 2015.
- [19] E. N. ISG, "Etsi network functions virtualisation (nfv) industry standards (isg) group draft specifications," Dec 2014, accessed: 2018-01-14. [Online]. Available: <http://docbox.etsi.org/ISG/NFV/Open>
- [20] R. Guerzoni, "etwork Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action. Introductory white paper," *SDN and OpenFlow World Congress*, June 2012.
- [21] R. Mijumbi, J. Serrat, J.-L. Gorricho *et al.*, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions."
- [22] R. Riggio, T. Rasheed, and R. Narayanan, "Virtual network functions orchestration in enterprise wlans.[In integrated network management (im), 2015 ifip/ieee int. symposium]," p. 1220–1225, May 2015.
- [23] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions.[In 2015 ieee conference on computer communications (infocom)]," p. 1346–1354, April 2015.
- [24] E.-G. Talbi, "Metaheuristics: from design to implementation," *John Wiley & Sons*, vol. 74, 2009.
- [25] F. Glover, "Future paths for integer programming and links to artificial intelligencet," *Computers & Operations Research*, vol. 13, no. 5, p. 533–549, 1986.
- [26] W. Michiels, E. Aarts, and J. Korst, *Theoretical Aspects of Local Search (Monographs in Theoretical Computer Science. An EATCS Series)*, 2007.

- [27] F. Glover and M. Laguna, "Tabu Search," *Kluwer Academic Publishers*, 1997.
- [28] B. Holldobler and E. Wilson, "The ants," 1990.
- [29] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," 1999.
- [30] M. Dorigo, G. D. Caro, and T. Stutzle, "Ant algorithms," 2000.
- [31] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," 2000.
- [32] M. Dorigo, G. D. Caro, and M. Sampels, "Ant Algorithms - Proceedings of ANTS 2002 Third International Workshop on Ant Algorithms, Brussels Brussels, Belgium," vol. 2463 of *Lecture Notes in Computer Science*, 2002.
- [33] M. Dorigo and T. Stutzle, "Ant Colony Optimization MIT Press Cambridge," 2004.
- [34] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Swarm intelligence: From natural to artificial systems," 1999.
- [35] E. Bonabeau and G. Theraulaz, "Swarm smarts. scientific american," 2000.
- [36] J. Holland, "Adaptation in natural and artificial systems," 1975.
- [37] D. E. Goldberg, "Genetic algorithms in search, optimization and machine learning," 1989.
- [38] D. B. Fogel, "Evolutionary computation," 1995.
- [39] J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," 1975.
- [40] J. Hertz, A. Krogh, and R. G. Palmer, "Introduction to the theory of neural computation," 1991.
- [41] C. M. Bishop, "Neural networks for pattern recognition," 1995.
- [42] D. M., "Optimization, learning and natural algorithms," PhD thesis, Politecnico di Milano, Dipartimento di Elettronica, 1992.
- [43] D. M, M. V, and C. A, "Positive feedback as a search strategy," Politecnico di Milano, Dipartimento di Elettronica, Tech. Rep., 1991.
- [44] —, "Ant system: Optimization by a colony of cooperating agents," 1996.
- [45] D. J-L, A. S, G. S, and P. J-M, "The self-organizing exploratory pattern of the argentine ant," 1990.
- [46] G. P-P, "La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie: Essai d'interpretation des termites constructeurs," 1959.

- [47] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction.(Cambridge, MA)," 1998.
- [48] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and Issues. RFC 3234," 2002.
- [49] P. Quinn and T. Nadeau, "Service Function Chaining Problem Statement," *Active Internet-Draft, IETF Secretariat, Internet-Draft draft-ietf-sfcproblem-statement-05*, Apr. 2014.
- [50] D. Joseph and I. Stoica, "Modeling middleboxes," *Network, IEEE*, vol. 22, no. 5, p. 20–25, 2008.
- [51] B. Addis, D. B. M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization.[In] Cloud Networking (CloudNet) ," *2015 IEEE 4th International Conference*, p. 171–177, 2015.
- [52] H. Moens and F. D. Turck, "Vnf-p: A model for efficient placement of virtualized network functions.[In] Network and Service Management (CNSM) ," *2014 10th International Conference on. IEEE*, p. 418–423, 2014.
- [53] L. Xin and Q. Chen, "A survey of network function placement.[In] 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)," p. 948–953, 2016.
- [54] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping.[In] INFOCOM 2009," *IEEE*, p. 783–791, 2009.
- [55] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten *et al.*, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions.[In] IEEE Conference on Network Softwarization (NetSoft)," *University College London*, April. 2015.
- [56] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping.Network[On]IEEE/ACM Transactions," vol. 20, no. 1, p. 206–219, Feb 2012.
- [57] R. Mijumbi, J. Serrat, J.-L. Gorricho, and R. Boutaba, "A path generation approach to embedding of virtual networks.Network and Service Management[On]IEEE Transactions," vol. 12, no. 3, p. 334–348, 2015.
- [58] V. Roshanaei, A. Azab, and H. ElMaraghy, "Mathematical modelling and a meta-heuristic for flexible job shop scheduling," *International Journal of Production Research*, vol. 51, no. 20, p. 6247–6274, 2013.
- [59] J. F. Riera, E. Escalona, J. Batalle *et al.*, "Virtual network function scheduling: Concept and challenges[In] Smart Communications in Network Technologies (SaCoNeT[On]2014 International Conference," p. 1–5, June 2014.

- [60] J. G. Herrera and J. F. Botero, "Joint optimization of service function chaining and resource allocation in network function virtualization," 2016.
- [61] H. Moens and F. D. Turck, "Virtual network functions orchestration in enterprise wlangs.[In integrated network management (im), 2015 ifip/ieee int. symposium]," p. 1220–1225, May 2015.
- [62] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," 2016.
- [63] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," 2014.
- [64] A. Ocampo, J. G. Herrera, P. H. Isolani *et al.*, "Optimal service function chain composition in network functions virtualization," accessed: 2018-01-10. [Online]. Available: https://www.researchgate.net/publication/318176078_Optimal_Service_Function_Chain_Composition_in_Network_Functions_Virtualization
- [65] M. Dorigo, "Optimization, Learning and Natural Algorithms. (Italy)," *PhD Thesis, Politecnico*, 1992.
- [66] —, "From Ant Colonies to Artificial Ants : First International Workshop on Ant Colony Optimization. (Bruxelles, Belgique)," Oct. 1998.
- [67] Y. C. Liang and A. E. Smith, "An ant colony optimization algorithm for the redundancy allocation problem (RAP)," *IEEE Transactions on Reliability*, vol. 53, no. 3, p. 417–423, 2004.
- [68] "Network Functions Virtualisation— Introductory White Paper," *ETSI*, June 2013.
- [69] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV:A Comprehensive Survey."
- [70] A. F. Ocampo *et al.*, "Optimal Service Function Chain Composition in Network Functions Virtualization."
- [71] S. L. S. L. Young-Bo Sim, "Function-Oriented Networking and On-Demand Routing System in Network Using Ant Colony Optimization Algorithm."
- [72] F. Glover, "Tabu Search – Part 1," *ORSA Journal on Computing*, p. 190–206, 1989.
- [73] —, "Tabu Search – Part 2," *ORSA Journal on Computing*, p. 4–32, 1990.
- [74] J. Li, "Virtual Network Embedding (VNE): Current Research Issues and Challenges."
- [75] M. Luizelli, L. Bays, L. Buriol *et al.*, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions . [In ifip/ieee im 2015]," pp. 98–106, 2015.

- [76] H. Moens, "Vnf-p: A model for efficient placement of virtualized network functions," accessed: 2018-01-09. [Online]. Available: https://www.researchgate.net/publication/268800292_VNF-P_A_Model_for_Efficient_Placement_of_Virtualized_Network_Functions
- [77] L. WANG, Z. LU, X. WEN *et al.*, "Joint optimization of service function chaining and resource allocation in network function virtualization," 2016.
- [78] M. Dorigo and T. Stutzle, "Ant colony optimization: Overview and recent advances," Université Libre de Bruxelles, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Tech. Rep., 05 2009.
- [79] "Node," accessed: 2018-01-09. [Online]. Available: <https://www.sdxcentral.com/term/node>
- [80] *Optimal Service Function Chain Composition in Network Functions Virtualization*, ser. LNCS 10356. Springer, 2014.
- [81] ETSI, "Network functions virtualisation (nfv)," accessed: 2018-01-09. [Online]. Available: <https://www.etsi.org/technologies-clusters/technologies/nfv>
- [82] E. V3.1.1, "Network function virtualisation (nfv) release 3; charging; report on usage metering and charging use cases and architectural study," accessed: 2018-01-09. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/008/03.01.01_60/gr_nfv-eve008v030101p.pdf
- [83] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten *et al.*, "Placement and scheduling of functions in networkfunction virtualization," 2015, accessed: 2018-01-09. [Online]. Available: <https://arxiv.org/pdf/1512.00217.pdf>
- [84] C. Blum, "Ant colony optimization: Introduction and recent trends," 2005, accessed: 2018-01-09. [Online]. Available: <https://www.ics.uci.edu/welling/teaching/271fall09/antcolonyopt.pdf>
- [85] C. Blum and X. Li, "Swarm intelligence in optimization," accessed: 2018-01-09. [Online]. Available: <https://pdfs.semanticscholar.org/1e0c/c80e285203998fa9ddf848c4a9e9941bb925.pdf>
- [86] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten *et al.*, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," accessed: 2018-01-09. [Online]. Available: <http://www.maps.upc.edu/rashid/files/NetSoft2015.pdf>
- [87] A. F. Ocampo, J. Gil-Herrera, P. H. Isolani, M. C. Neves *et al.*, "Optimal service function chain composition in network functions virtualization," accessed: 2018-01-09. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-319-60774-0_5.pdf
- [88] M. Dorigo and T. Stutzle, "Ant colony optimization: Overview and recent advances," accessed: 2018-01-09. [Online]. Available: <https://pdfs.semanticscholar.org/7e47/f3e6972d9b7515d10f8cfbcd3b8f21e22b80.pdf>

- [89] G. D. Caro, "Ant colony optimization and its application to adaptive routing in telecommunication networks," Université Libre de Bruxelles, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Tech. Rep., 09 2004.
- [90] Y. Xie, Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," 2016, accessed: 2018-01-09. [Online]. Available: <https://arxiv.org/pdf/1608.00095.pdf>