الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية التكنولوجيا

قسم: الإعلام الآلي

Master Thesis

Speciality : Computer Networks and Distributed Systems (RISR)

Theme

Scientific Workflow Optimization based on Discrete Symbiotic Organism Search (DSOS) in Cloud Computing

Presented by :

CHAREF Chahinaz

MOUEDDENE Mohamed Hassane

Directed by :

Dr. KOUIDRI Siham

Année universitaire 2022-2023

ملخص

أصبحت الحوسبة السحابية نموذ ًجا رائدًا لالستخدام الفعال لخدمات الحوسبة والتخزين والشبكات عند الطلب. مع الطلب المتزايد على الموارد السحابية، أصبح تخطيط المهام موضو ًعا بحثيًا مهمًا في هذا المجال. يتعلق التخطيط الفعال للوظائف بتكليف اآلالت االفتراضية بمهام لتقليل وقت تشغيل سير العمل. ومع ذلك، من المعروف أن تخطيط سير العمل يمثل مشكلة NP كاملة، مما يشكل تحديات حسابية. تقدم هذه الوثيقة استعرا ًضا لألدبيات لألعمال الحالية في هذا المجال ونهجنا المقترح لمعالجة مسألة جدولة المهام المستندة إلى السحابة.

<u>الكلمات الدالة :</u> الحوسبة السحابية ، جدولة سير العمل ، البحث عن الكائنات التكافلية المنفصلة ، التبادلية ، التعايش ، التطفل ، النظام البيئي ، .CloudSim.

Abstract

Cloud Computing has emerged as a prominent paradigm for utilizing on-demand computing, storage, and network services in an efficient manner. With the increasing demand for cloud resources, task scheduling has become a significant research topic in this domain. Efficient task scheduling aims to assign tasks to virtual machines in order to minimize workflow execution time. However, the scheduling of workflows is known to be an NP-complete problem, posing computational challenges. This paper presents a literature review of existing work in this field and introduces our proposed approach to address the task scheduling problem in cloud computing.

<u>Key words:</u> cloud computing, workflow scheduling, makespan, Discrete symbiotic organism search,mutualism, commensalism, parasitism, ecosystem, CloudSim.

Résumé

L’informatique en nuage est devenue un paradigme de premier plan pour l’utilisation efficace des services d’informatique, de stockage et de réseautage à la demande. Avec la demande croissante de ressources en nuage, la planification des tâches est devenue un sujet de recherche important dans ce domaine. Une planification efficace des tâches consiste à attribuer des tâches à des machines virtuelles pour réduire le temps d’exécution du flux de travail. Cependant, la planification du flux de travail est connue pour être un problème NP complet, posant des défis informatiques. Le présent document présente une recension des écrits sur les travaux existants dans ce domaine et l’approche que nous proposons pour aborder la question de la planification des tâches en nuage.

<u>Mots clés:</u> cloud computing, workflow scheduling, makespan, Discrete symbiotic organism search, mutualism, commensalism, parasitism, ecosystème, CloudSim.

Gratitude

First and foremost, we would like to express our gratitude to Allah the almighty that has granted us the strength, willpower, and patience to complete this endeavor.

Our sincere appreciation goes to our mentor, Dr. KOUIDRI Siham, whose guidance, trust, and patience have made an invaluable contribution without which this work could not have reached its destination. May this work serve as a profound tribute to their exceptional character.

We would also like to extend our heartfelt thanks to the members of the jury, who graciously dedicated their time to evaluate this humble piece of work.

Our profound gratitude is also extended to all of our teachers. It is through their teachings and shared knowledge that this project has come to fruition. We owe them a great debt of gratitude for their contributions to our growth and the realization of this brief.

Dedication Charef Chahinaz

To my cherished parents, to whom I owe an immeasurable debt of gratitude.

To my beloved sisters and my nephew, whose unwavering support has been a constant source of strength and inspiration.

To my brother and family and many friends who were there throughout the entire journey.

To my good friend and source of strength Nada for always being there.

To that lobster that once walked by.

A very heartfelt gratitude to Dr. **Kouidri Siham** for her sacrifice with us during this thesis. As well as my partner **Moueddene Mohamed hassane** Thank you for always being there for me.

Dedication Moueddene hassane

I dedicate this work to:

To my family, whose unconditional love, encouragement, and sacrifices have been my driving force.

To my friends who have been a great support throughout the academic years.

A big appreciation to Mrs **KOUIDRI Siham** for the best guidance in our thesis, also to my partner **Charef Chahinaz** who accompanied me in this bizarre journey.

To those who have supported and believed in me, this thesis is dedicated with deep appreciation.

Contents

List of acronyms

NIST: The National Institute of Standards and Technology.

IaaS : Infrastructure as a service.

SaaS: Software as a service.

PaaS: Platform as a service.

VM: Virtual Machine.

FCFS: First Come First Served.

FIFO: First in first out algorithm.

SJF: Shortest Job First.

RR : Round Robin algorithme.

CPM: Critical Path Method.

SOS : Symbiotic Organisms Search.

DSOS : Discrete Symbiotic Organisms Search Algorithm.

PT: Processing time.

DT: Data transfer.

CB: Cloud broker.

List of Figures

List of Tables

General Introduction

Cloud computing has revolutionized the way businesses and individuals access and utilize computing resources. It is a model that provides on-demand access to a shared pool of computing resources, including networks, servers, storage, applications, and services. These resources can be rapidly provisioned and released with minimal management effort.
In traditional computing models, organizations would have to invest in and maintain their own physical infrastructure, such as servers and data centers. This required significant upfront costs, ongoing maintenance, and limited scalability. Cloud computing, on the other hand, offers a flexible and cost-effective alternative.
This plateform provides a scalable and flexible platform for executing scientific workflows by leveraging the computational power and resources available in the cloud. It allows researchers and scientists to access virtualized resources, such as virtual machines, storage, and networks, on-demand and pay for what they use.
Optimizing the scheduling of scientific workflows is crucial for achieving optimal performance, reducing execution time, and maximizing resource utilization. The goal is to allocate tasks to available resources in a way that minimizes the overall makespan (total time to complete the workflow) and optimizes other performance metrics such as cost, energy consumption, or reliability.
This problem is considered as an NP-hard problem. NP-hardness refers to a class of computational problems that are considered difficult to solve efficiently, meaning there is no known polynomial-time algorithm to solve them, in this case, the problem complexity arises due to various factors, including task dependencies, resource constraints, data transfert, and optimization objectives. The goal is to find an optimal schedule that minimizes the overall makespan, maximizes resource utilization, minimizes data transfert, and considers other performance metrics.

Finding an optimal solution for scientific workflow scheduling is often impractical in real-world scenarios, especially for large-scale workflows. Instead, heuristic and approximation algorithms are commonly employed to find near-optimal or satisfactory solutions within a reasonable amount of time. This thesis explores the potential of utilizing the Discrete Symbiotic Organism Search (DSOS) algorithm for scientific workflow optimization in cloud computing. DSOS, inspired by symbiotic relationships observed in nature, is a metaheuristic algorithm capable of efficiently exploring solution spaces to find near-optimal solutions for combinatorial optimization problems. By simulating the symbiotic interactions between organisms, DSOS offers a promising approach to address the optimization challenges within scientific workflows.
The primary objective of this research is to investigate the effectiveness of DSOS in optimizing various aspects of scientific workflows in cloud computing environments. Key optimization areas include task scheduling, resource allocation, data transfer. This research contributes to the advancement of scientific workflow optimization, providing valuable insights and techniques to improve research efficiency and accelerate scientific discoveries in cloud computing environments.

Organization of the manuscript

The work that we have carried out in the context of the problem is summarized in this document, which is structured in four chapters:

Chapter 1 : The first chapter serves as an introduction to cloud computing and its significance in modern computing environments. It provides a comprehensive overview of the key concepts, architectures, and benefits associated with cloud computing.

Chapter 2 : This chapter focuses on the scientific workflow scheduling strategies in cloud computing.

Chapter 3 : Third chapter is dedicated to the description of the approach proposed , detailing the algorithm used " Discrete Symbiotic organism search DSOS".

Chapter 4 : The last chapter is a discussion of the application developped in this concept and the results optained.

Chapter 1

Cloud computing

1.1. Introduction

The rapid advancement of information and communication technologies has enabled the development of new computing paradigms, where processing, storage, communication, sharing and dissemination techniques information have changed dramatically. Individuals and organizations are increasing use of external servers for storage and distribution efficient and reliable information.

Nowadays all that is required to use services on a remote server, using computing cycles of a pile of servers that are in different locations, share private and confidential information and complete tasks simply with a web page that will allow the user to have access to different resources from all around the globe. In this chapter we will be addressing this technology called Cloud Computing, the latter has attracted so much attention recently. According to a Gartner press release from June 2008, Cloud Computing will be no less influential than e-business (Gartner 2008a). Cloud computing is also expected to be a fundamental approach towards Green IT which aims to minimize the negative effect caused by IT operations on the environment. To understand this concept, we dedicated this chapter to defining it and mentioning different terminologies and elements related to it. [1]

1.2. Cloud computing

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or provider interaction this definition is commonly used and it was formulated by the National Institute of Standard and Technology (NIST). [2]

The main idea of these technologies is to offer computing data, applications, resources as a public utility in the form of distributed services on the network in a payment system (pay as you go), more plainly, cloud computing is a model that can obtain resources such as processors, memory, storage, and applications quickly from the Internet in real time and based on demand. Instead of having to make major investments to buy equipment, train staff, and provide ongoing maintenance, all the above can be handled by a cloud service provider, reducing the financial cost of it, the latter in turn depend on the data center industry, with more than 500,000 server farms set up around the globe. The functioning of such broadly disseminated data centers, in any case, requires a lot of energy for processing and cooling purposes which adds more expenses.

This is the case for example of Google App Engine, Amazon EC2 or Microsoft Azure, offers allowing the use of computing and communication infrastructures from Google, Amazon or Microsoft as utility services.

1.3. History

The concept of cloud computing is not a new invention, since mainframes were already a step towards the latter in the 1950s by giving users access to the central computer through several terminals within the and use its capabilities. At first, however, it was time-sharing (users had to reserve computing time, and were allowed to use the performance of the mainframe for their computations during that time).

The notion of multiple people sharing the same computer resource requires a technology called virtualization, this allowed that the computation instances could be built in an abstract way, these virtualized environments were finally accessible to everyone with the invention of the Internet in the 1990s.

In 1997, Professor Ramnath Chellapa of Emory University defined cloud computing as the new computing paradigm, where the boundaries of computing will be determined by economic rationale, rather than technical limits alone. Followed by that in 1999, Salesforce became the first company to offer applications over the internet, heralding the arrival of Software as a Service. [3]

In 2002, Amazon introduced its web-based retail services providing services like storage, computation and even human intelligence, On August 25, 2006, Amazon Web Services launched Elastic Compute Cloud (EC2), enabling people to rent virtual computers and use their own programs and applications online. [3]

In 2009, Google Apps also started to provide cloud computing enterprise applications, as well as Windows Azure which was launched by Microsoft, and many other companies joined the field like Oracle and HP. [3]

Cloud computing has become part of everyday life for many people. Most smartphones, or more broadly, the Internet of Things, are in constant contact with the cloud.

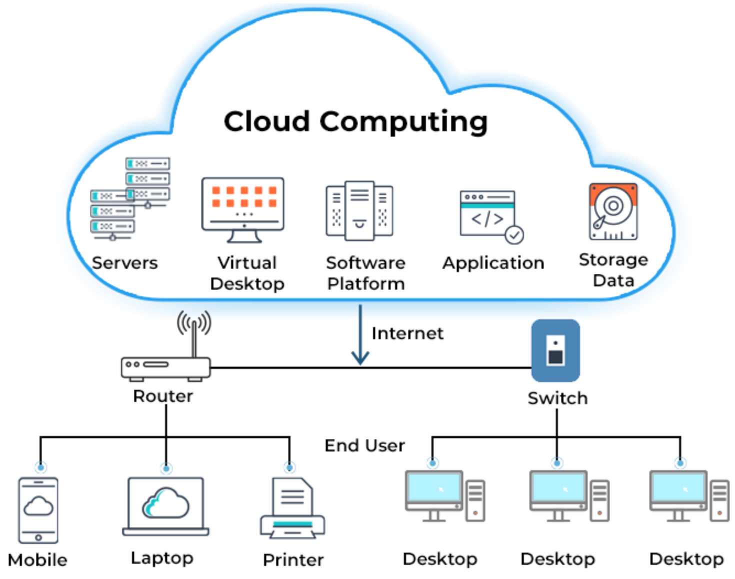


Figure 1.1: *Cloud computing*

[4]

1.4. Cloud computing services

Cloud computing providers use many service models. These services are organized into three successive levels: the infrastructure level (IaaS), the platform level (PaaS) and the application level (SaaS), referred to as SPI.

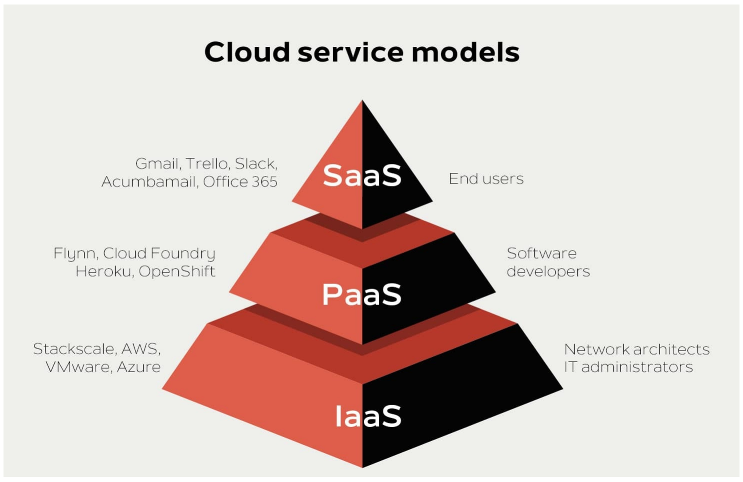

Cloud service models
Gmail, Trello, Slack, Acumbamail, Office 365
SaaS
End users
Flynn, Cloud Foundry Heroku, OpenShift
PaaS
Software developers
Stackscale, AWS, VMware, Azure
IaaS
Network architects IT administrators

Figure 1.2: *Cloud service models*
[5]

1.4.1. Infrastructure as a service (IaaS)

Infrastructure as a service (IaaS) is the most basic category of cloud computing services, it is a form of cloud computing that provides virtualized computing resources (network, storage, operating systems) over the internet, Users have most of the time almost complete control over the VMs they rent, they can choose pictures of preconfigured operating systems, or machine images custom apps containing their own apps, libraries, and configuration settings.

There are many commercial and open-source IaaS providers. From commercial platforms, the most common are: Amazon EC2, Microsoft Azure. [6]

1.4.2. Platform as a Service (PaaS)

PaaS as a services have specialized application development environments that include the tools and modules needed for this type of work letting to the developer manage the hardware or software necessary (develop, design, implementation, test, provide), including also a solution stack.
PaaS offers great flexibility, allowing in particular to quickly test a prototype or provide an IT service over a period of short duration.
Google App Engine, Microsoft Azure are examples of PaaS services. The Microsoft platform offers the possibility of modifying applications directly online thanks to remote desktop techniques. [7]

1.4.3. Software as a service (SaaS)

Software as a service makes it possible to provide users with ready-to-use applications. Unlike ordinary web applications, it is characterized by a high level of abstraction that allows the application to be adapted to a particular use case.
There are different types of application ranging from CRM (Customer Relationship Management), human resources management, collaborative tools, messaging, BI (Business Intelligence) and other business applications. Users connect to the application via the Internet, usually through a web browser on their phone, tablet or PC. [7]

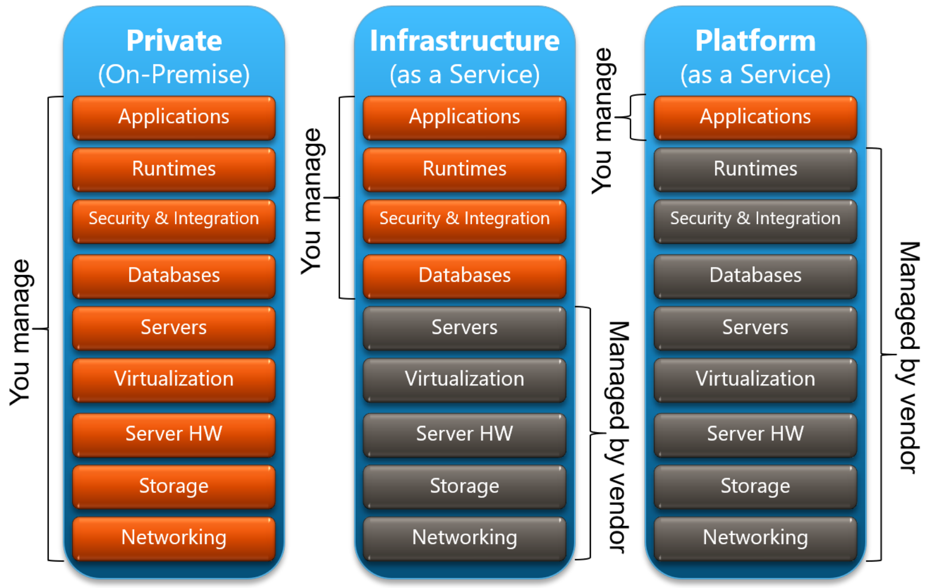


Figure 1.3: *Cloud computing services*
[8]

1.5. Cloud Computing Deployment Models

1.5.1. Public cloud computing

Public cloud (also known as external cloud), is when services are provided and used in a so-called pay-per-use manner where the cloud provider is external and its infrastructures and sources are accessible to everyone.

This infrastructure can be owned, managed and operated by a business, an educational institution or a public administration, or a combination of the three. It is accessible on the provider's website.

1.5.2. Private cloud computing

This cloud consists on the hosting of private applications, storage, or computation in a specific group or organization and restricts access to that entire group.

Unlike public cloud, private cloud computing requires resources and significant upfront development costs, data center costs, ongoing maintenance, hardware, software, and software. in-house expertise.

These resources are the property of the company, which manages and shares them. That is the reason it is used by large organizations and government agencies mostly. [7]

1.5.3. Community cloud computing

In a community cloud, infrastructure is deployed for exclusive use by a group of organizations or companies that share the same interests (missions, security requirements, policies, compliance rules, etc.), in such an architecture, system administration can be performed by one or more of the organizations sharing the cloud resources.

An example of this is OpenCirrus formed by HP, Intel, Yahoo, and others.

1.5.4. Hybrid cloud computing

Hybrid cloud computing is a combination of both public and private cloud, linked together by standardized or proprietary technologies that allow the portability of applications. Organizations can have parts of their services in their own infrastructures but also in public cloud. Or can use the public just when its needed.

1.6. Virtualization

Virtualization is a method of running multiple independent virtual operating systems on a single physical computer, the main use of this technology is to provide applications with standard versions to their cloud users.

Virtualization is the most essential part of cloud computing, in other words, it is one of the main cost-effective, hardware-reducing, and energy-saving techniques used by cloud providers. It consists of a software layer that allows the abstraction between the hardware and the operating system. Thus, several systems can be installed on the same physical machine. [8]

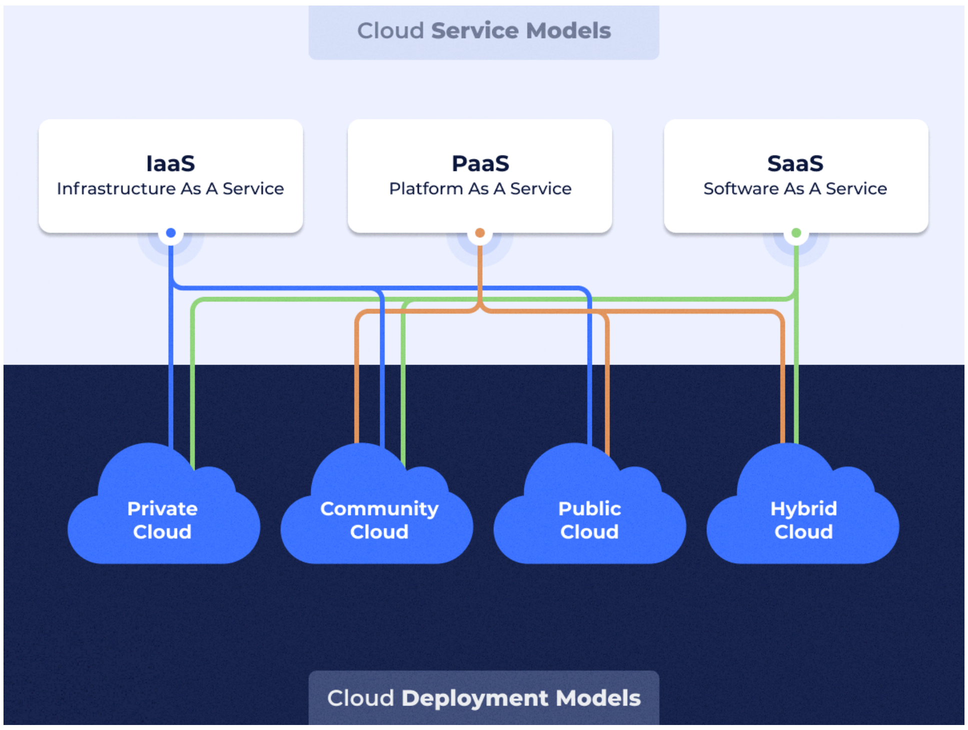


Figure 1.4: *CS models*
[9]

1.6.1. Types of virtualization

- **Server virtualization:** Typically, each physical server is dedicated to one specific application or task, so to solve the inefficiency problem, server virtualization came to light allowing an administrator to convert a server into multiple virtual machines, So, each system can operate its own operating systems in an isolated manner.

Its beneficial in virtual migration, reducing energy consumption, reduce infrastructural cost, etc.

- **Storage virtualization:** Storage virtualization uses all of the physical data storage of a machine and creates one large virtual storage unit that can be assigned and controlled using management software. It allows to manage and use storage from multiple sources as a single repository.
- **Application virtualization:** Application virtualization encapsulates the application and separates it from the underlying operating system. As an example, without changing the machine configuration, users can run a Microsoft Windows application on a Linux machine. It gives you access to the application without installing it onto the native device, in other words it helps a user to have remote access to an application from a server.
- **Network virtualization:** Network virtualization provides a facility to create and provision virtual networks-logical switches, routers, firewalls, load balancer, Virtual Private Network (VPN), and workload security within a short period of time.
- **Desktop virtualization:** Desktop virtualization grants the users OS to be remotely stored on a server in the data center. It allows the user to access their desktop virtually, from any location by a different machine through a thin client (such as a web browser), essentially creating a portable workstation. [8]

1.7. Grid Computing

Grid computing, also called "distributed computing." is a group of networked computers that work together as a virtual supercomputer to perform large tasks that would be difficult for a single machine. Splitting tasks over multiple machines helps reduce processing time and increase efficiency and minimize wasted resources. A grid computing network consists of a control node, a provider and a user. The control node gives the user access to the resources when they are idle. [10]

1.8. Utility computing

Utility computing is a model in which computing resources are provided to the user based on specific demand charging exactly for the services that has been provided which reduces the cost.

Utility computing helps eliminate data redundancy; as huge volumes of data are distributed across multiple servers or backend systems. And the client can have access anytime. [7]

1.9. Service-Oriented Architecture (SOA)

Service-oriented architecture (SOA) is a type of software design that makes software components and services reusable, it can work with or without cloud computing.

Each service offers a trade capability, and the services can also communicate with each other across platforms and languages. Developers use SOA to reuse services in different systems or combine multiple independent services to perform complex tasks. [10]

1.10. Load Balancing

Load balancing helps distribute traffic and workloads evenly between two computers or more to ensure that no single server or machine is under-loaded, overloaded, or idle.

The load balancer is in charge of managing traffic, it sits between servers and client devices, it is able to deal with different amount of work capacity by adapting its distribution decisions according to the moments a request is made.

1.11. Data centers

Data centers are a physical destination or more specifically warehouses of networked computers, storage systems, and computing infrastructure that organizations use to organize, process, store a large amount of data. Businesses can enhance their performance, scalability, and security by utilizing a data center as a cloud computing strategy because they offer enterprises processing and storage resources to execute their applications. It includes:

- Systems for storing, accessing and processing data across the organization.
- Physical infrastructure for data processing and data communication.
- Utilities such as cooling, electricity, network access, and uninterruptible power supplies (UPS).

1.12. Cloud computing characteristics

On demand self-service: Users are able to provision, monitor and manage computing resources such as server time and network storage, as needed without a human administrators required.

Multi tenancy and resource pooling: It is the software architecture that allow a single program instance to issue services to multiple users.

Measured and reporting service: Cloud systems automatically monitor and optimize resource usage, this technology will provide both the user and the resource provider with an account of what has been used.

Rapid elasticity: The Computing services should have IT resources that are able to rapidly scale out and in based on demand. For the user, the capacities available for supply often appear to be unlimited and can be appropriated in any quantity at any time.

Broad network access: Cloud computing is so versatile that it enables its users to access cloud services, and upload data. The users can be thin or thick heterogeneous client platforms (ex: mobile phones, laptops and workstations) [11]

1.13. Cloud computing architecture

Every organization weather its small or large uses cloud computing services for storing data and having access to it from anywhere, anytime with the help of internet. Every cloud infrastructure should provide scalability, transparency, security and intelligent monitoring. [12]

Cloud computing architecture consists of two parts, the frontend and backend as represented in figure 1.4

1.13.1. Front-end

In cloud computing the clients side is the front-end, it includes all the users interfaces and applications, which are used by the client to have access to the cloud computing services and resources. It consists of:

- User interface: it is the interface made by cloud where the users can complete tasks without the need to install any software on their machines.
- Software: in charge of the browser or the software the end user uses.
- Client device: the device and input devices used by the end user, it is a simple device that doesnt require any super abilities. [12]

1.13.2. Back-end

The back-end is the cloud environment itself, it includes all the resources to provide cloud services, it consists of hardware and storage located on a remote server controlled by the cloud provider.

It is comprised of:

- Application: this layer deals with the users requests, it has access to the data of clients by offering back-end services.
- Service: referring to the three types of services SaaS, IaaS and PaaS
- Cloud runtime: provides the execution and runtime environment for the virtual machine.
- Storage: a big portion of cloud is dedicated to store data like solid-state drives (SSDs), hard disk drives (HDDs), Intel Optane DC Persistent Memory
- Infrastructure: the various technologies such as CPU, Motherboard, Graphics Processing Unit (GPU), network cards, accelerator cards
- Management: it makes sure to allocate different resources to different tasks with every task getting its share of attention. [12]

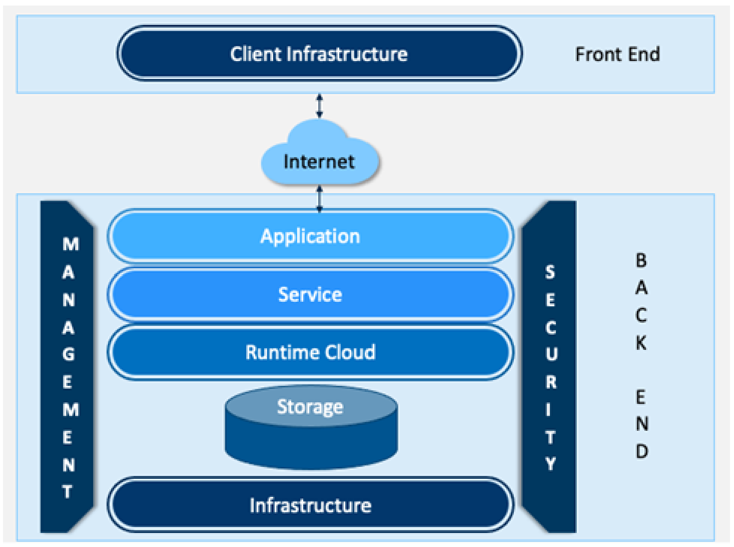


Figure 1.5: *Cloud computing architecture* [13]

1.14. Cloud computing security

To protect the cloud-based systems a set of policies, technologies, controls and procedures were set, it is called Cloud computing security, in order to preserve the cloud data and the users privacy.

The providers security responsibilities are related to the safeguarding of the infrastructure itself, as well as access to, patching, and configuration of the physical hosts and the physical network on which the compute instances run and the storage and other resources reside. [7]

Several points must be addressed to ensure that all security measures are implemented:

1.14.1. The location of data centers

Data centers should be in a secure location, an area not susceptible to natural disasters like floods, earthquakes or fires with barriers to prevent forced entry. Some hosts have several data centers; the client company must therefore be able to know where its data is located geographically. [7]

1.14.2. Guarantees of audits and contracts

the chosen service provider must be able to demonstrate through various audits that all the security rules are optimal and up-to-date.[7]

1.14.3. Constant data availability

the data should be available to the user at all times and will not be lost even in the event of technical problems. Providing a Disaster Recovery Plan (DRP) that prevents data loss by duplicating it and cloud infrastructure and services so that they remain available even if the initial servers encounter a problem.[7]

1.14.4. Securing data in moving

Securing data in motion: data security therefore depends on the quality of the relationship and administrative transparency with its service provider, but also on the purely technical quality of its offer. Data is vulnerable when it is being moved. This is why the streams are highly secured by trusted hosts.[7]

1.15. ADVANTAGES AND DISADVANTAGES OF CLOUD COMPUTING

1.15.1. ADVANTAGES

- Scalability: Cloud computing enables businesses to easily adjust their computing resources without significant upfront investments in hardware, allowing for quick additions or removals of resources.

- Cost-efficiency: By eliminating the need for businesses to invest in and maintain expensive infrastructure, cloud computing allows them to pay for resources on a subscription or pay-as-you-go basis, reducing capital expenses.

- Flexibility and accessibility: Cloud computing provides users with the ability to access data and applications from any location with an internet connection, facilitating remote work, collaboration, and multi-device accessibility.

- Reliability and uptime: Cloud service providers ensure high levels of reliability and uptime through redundant systems and backup procedures, ensuring constant availability of data and applications.

- Security: Cloud service providers prioritize security, employing advanced technologies and adhering to strict compliance standards, often providing superior security compared to individual businesses.
- Back-up and recovery: a robust disaster recovery and data backup capabilities is offered. Cloud service providers often have redundant systems and data replication mechanisms in place, ensuring that data is backed up and can be quickly restored in a case of a natural disaster

1.15.2. DISADVANTAGES

- Vendor lock-in: Adopting a particular cloud service provider and infrastructure can create challenges in switching providers or migrating to a different architecture, potentially limiting flexibility and increasing costs.

- Internet dependency: Cloud computing heavily relies on a stable and fast internet connection, making it susceptible to difficulties in data and application access during internet downtime or instability.

- Data security concerns: Although cloud service providers implement strong security measures, storing data in the cloud can raise concerns about unauthorized access, data breaches, or loss, particularly for businesses with regulatory restrictions on data storage.

- Potential downtime: Despite the generally high reliability of cloud service providers, there is still a possibility of service disruptions or downtime, which can disrupt business operations for those heavily reliant on cloud services.

1.16. Example of clouds

1.16.1. Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) offers the user unparalleled computing power and means of controlling his account, it can be auto-scaled to meet demand; it facilitates creating and starting virtual machines and simplified commands via a web interface.

Today EC2 provides complete control over a customer's computing resources, so new sample servers can be installed and booted in minutes, and their capacity can be quickly measured by platform utilities (Amazon Cloud Watch). Amazon EC2 provides so many features like virtual computing environments, known as instances as well as various configurations of CPU, memory, storage, and networking capacity for the instances, known as instance types. It also provides private and public keys to secure the login information for the instance; and storage volumes for the temporary data; plus, the Persistent storage volumes with multiple physical locations for the resources [14].

1.16.2. Google App Engine

Google App Engine (GAE) is a platform-as-a-service, it is mostly used to run Web applications to meet demand of the community of users of Google services, which always demand more efficiency and security on web platforms, GAE requires applications to be written in Java or Python and use the Google query language after storing them in Googles Bigtable.

The remarkable interest in GAE is providing more infrastructure than other hosting services such as EC2, in addition of eliminating some system administration and development tasks to make writing scalable applications easier. [14]

1.16.3. Oracle cloud

Oracle offers all types of infrastructure services; The global data centers in Oracle Cloud Infrastructure (OCI) provide servers, storage, network, applications, data management and other services that support dedicated cloud, multi-cloud, hybrid cloud and on-premises environments.

Oracle has an entire range of cloud solutions; it is up to the user to buy parts of the solutions or all of it in a form of an engineered system. [14]

1.17. Conclusion

Cloud computing has transformed the way businesses operate by providing scalable resources, seamless collaboration, and global expansion opportunities. Organizations can leverage cloud technology to streamline operations, access advanced technologies, enhance data security, and drive innovation. With cloud computing, businesses can focus on core competencies, reduce IT complexities, and accelerate time-to-market. Cloud computing has become an essential tool for businesses to thrive in the digital era.

Chapter 2

Scientific workflow scheduling strategies in cloud computing

2.1. Introduction

In recent years, the use of cloud computing has become increasingly popular in scientific research. Cloud computing provides on-demand access to a shared pool of computing resources, which can be used to perform scientific workflows more efficiently and cost-effectively than traditional on-premises computing environments.

With the growth of cloud computing, scientific workflows can be executed on cloud resources, which provide on-demand access to a large pool of computing, storage, and networking resources. Cloud computing offers several benefits for executing scientific workflows, including scalability, flexibility, and cost-effectiveness. However, executing scientific workflows in the cloud also presents significant challenges, such as resource allocation, data management, and task scheduling.

However, managing and scheduling scientific workflows in the cloud can be a complex task. Workflow scheduling involves determining the order in which tasks should be executed and allocating resources such as CPU, memory, and storage to each task. The goal is to optimize performance, minimize execution time, and reduce costs while meeting the workflow's requirements.

2.2. Overview of scientific workflows

Scientific workflows are a series of interconnected computational and data processing tasks that are used to solve a scientific problem or perform a scientific experiment. These workflows can be used in a variety of domains, including bioinformatics, physics, astronomy, and environmental science, among others.

2.2.1. Definition and characteristics of scientific workflows

a scientific workflow is a series of computational and data processing tasks that are executed on a cloud-based infrastructure. These tasks can be executed on a variety of virtualized resources, including virtual machines, containers, and server less computing platforms, among others. Scientific workflows in the cloud typically leverage the benefits of cloud computing, such as scalability, elasticity, and cost-effectiveness, to perform data-intensive and computationally intensive. [15]

- **Data-intensive:** Scientific workflows involve large amounts of data, which are processed and analyzed using various computational tools.
- **Iterative:** Scientific workflows are often iterative in nature, as scientists repeat the same process several times with different parameters or inputs to test their hypotheses.
- **Collaborative:**Scientific workflows involve collaboration among multiple researchers and often require the use of shared data and computing resources.
- **Reproducible:** Scientific workflows must be reproducible, meaning that other researchers should be able to reproduce the results of a given workflow using the same data and tools.
- **Transparent:** Scientific workflows should be transparent, meaning that the methods and techniques used to generate results should be clearly documented and available for scrutiny.
- **Flexible:** Scientific workflows should be flexible, allowing researchers to modify the workflow to accommodate changes in data or research goals.
- **Automated:** Scientific workflows often involve the use of automation tools to streamline the process of data processing and analysis.
- **Modular:** Scientific workflows are often composed of multiple, interdependent modules that can be combined to create more complex workflows.
- **Scalable:** Scientific workflows should be scalable, meaning that they can handle increasing amounts of data and computing resources as needed.
- **Domain-specific:** Scientific workflows are often tailored to specific scientific domains and their associated computational tools and techniques. [15]

Of the many possible ways to distinguish workflow computations, one is to consider a simple complexity scale. At the most basic level one can consider linear workflows, in which a sequence of tasks must be performed in a specified linear order. The first task transforms an initial data object into new data object that is used as input in the next data-transformation task.

At the next level of complexity, one can consider workflows that can be represented by a DAG, where nodes of the graph represent tasks to be performed and edges represent dependencies between tasks. Two main types of dependencies can be considered: data dependencies (where the output of a task is used as input by the next tasks) and control dependencies (where before to start one or a set of tasks some tasks must be completed).

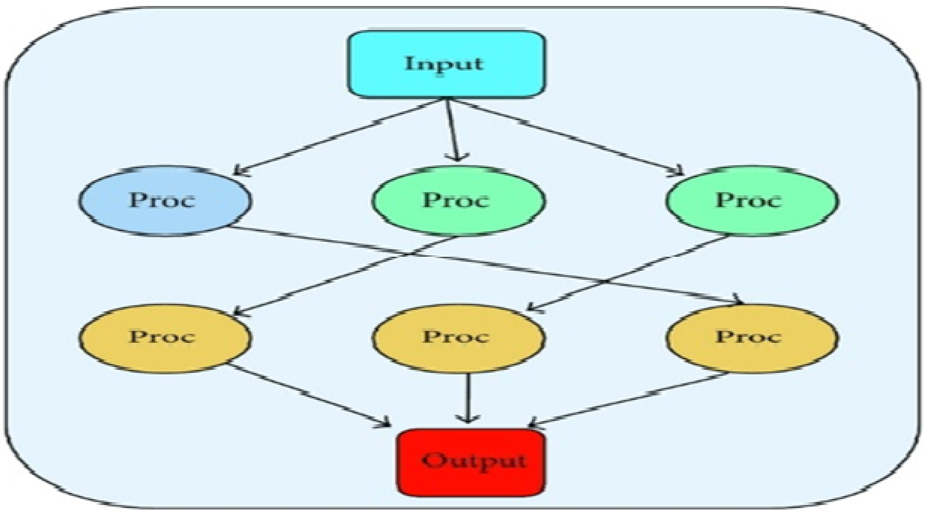


Figure 2.1: *A simple DAG including data and control nodes.* [16]

2.3. Components and structure of a scientific workflow in cloud computing

2.3.0.1. Cloud resources:

Cloud computing provides access to a variety of computing resources, such as virtual machines, storage, and databases, which can be used to run scientific workflows. These resources are typically provisioned dynamically based on the needs of the workflow. [17]

2.3.1. Workflow management system:

A workflow management system is used to manage the execution of scientific workflows on cloud resources. It provides tools for creating, deploying, and monitoring workflows, as well as managing the resources used by the workflows.

2.3.2. Data transfer:

Data transfer is a key component of scientific workflows on cloud computing. Large datasets may need to be transferred between cloud resources, and efficient data transfer mechanisms are essential for minimizing the time and cost of running workflows.

2.3.3. Scaling:

Cloud computing allows scientific workflows to be scaled up or down dynamically based on the needs of the workflow. This requires the workflow to be designed to take advantage of cloud scaling capabilities, such as parallelization and load balancing.

2.3.4. Security:

Cloud computing introduces new security considerations for scientific workflows, such as protecting data during transfer and ensuring the security of cloud resources. Security measures, such as encryption and access control, must be implemented to protect sensitive data. [17]

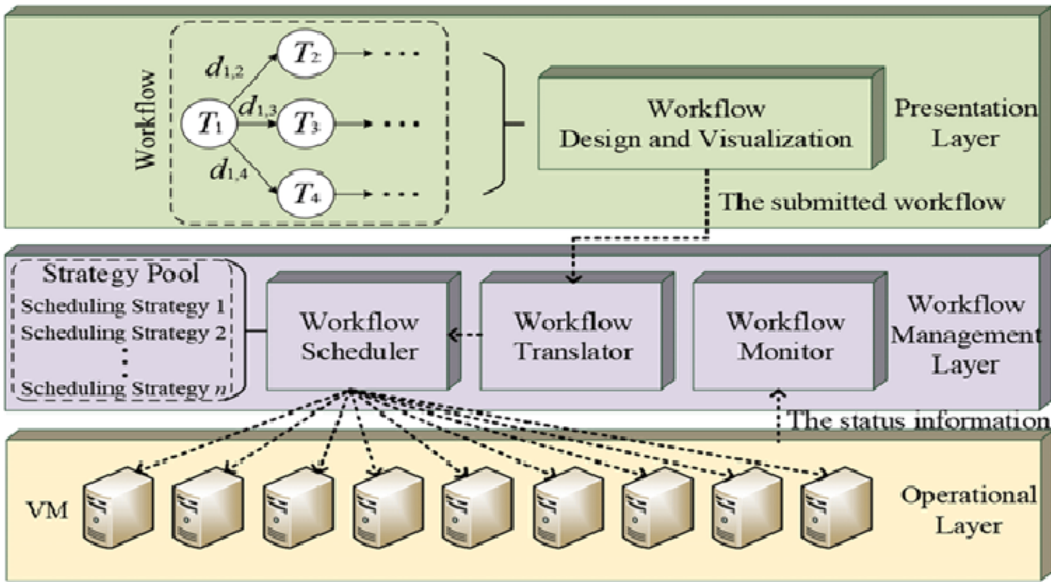


Figure 2.2: *Structure of a typical scientific workflow* [18]

2.4. Scheduling concept and definition

2.4.1. scheduling:

A scheduling problem consists in ordering in time an en-set of tasks contributing to the realization of the same project. The objective is to minimize the duration of the project taking into account the constraints anteriority depending on the different tasks. Scheduling in scientific workflow refers to the process of assigning computational resources, such as computing nodes or virtual machines, to specific tasks or stages in the workflow.

2.4.2. TASKS:

a task refers to a specific unit of work that performs a well-defined computational operation on data. A task can be a single command, script, program, or a combination of these that performs a specific analysis or transformation on the data. Tasks are organized in a sequence to form a workflow that represents the entire data processing or analysis pipeline.

2.4.3. DATA:

refers to any information that can be stored and processed by a computer system. Data is often the primary input for scientific workflows, and it may be processed, analyzed, and transformed by different tasks in the workflow to achieve a specific scientific goal. Data can be stored in various formats and structures, and it may be distributed across different storage systems or computing resources.

2.5. WORKFLOW SCHEDULING STRATEGIES IN CLOUD COMPUTING

2.5.1. DEFINITION AND IMPORTANCE OF WORKFLOW SCHEDULING

Workflow scheduling is the process of assigning and allocating computing resources, such as CPUs, memory, and storage, to specific tasks or stages in a workflow. The goal of workflow scheduling is to optimize the execution of the workflow by minimizing the overall execution time and maximizing the utilization of resources. [19]

Workflow scheduling is important in scientific computing and data-intensive applications because these workflows typically involve multiple stages or tasks that must be executed in a specific order and often have dependencies between them. In addition, these workflows often require significant computing resources, which can be expensive or limited. [19]

Effective workflow scheduling can help to improve the performance and efficiency of scientific computing and data-intensive applications by ensuring that tasks are executed in the correct order and on the appropriate computing resources. By minimizing the overall execution time and maximizing the utilization of resources, workflow scheduling can also help to reduce the cost of executing these workflows. [19]

2.5.2. Types of scheduling strategies

2.5.2.1. Independent task scheduling

Independent task scheduling refers to scheduling tasks in a workflow that do not have any dependencies on each other, meaning they can be executed in any order without affecting the final outcome of the workflow.

In scientific workflows, tasks that are independent of each other can be scheduled concurrently, thereby reducing the overall execution time. For example, if a workflow contains two tasks, A and B, and task A does not depend on the output of task B and vice versa, then both tasks can be scheduled to run concurrently on different computing resources.

Scheduling independent tasks can be done using various scheduling algorithms, such as First-Come, First-Served (FCFS) or Round-Robin (RR) scheduling. These algorithms can be used to allocate resources to tasks in an efficient and fair manner. [20]

2.5.2.2. dependent task scheduling

In such workflows, the order in which the tasks are executed is critical to the overall outcome of the workflow. Scheduling dependent tasks requires careful consideration of the dependencies between tasks to ensure that they are executed in the correct order. Scheduling dependent tasks in a scientific workflow can be a complex task, and there are many factors to consider, such as the available computing resources, the complexity of the workflow, and the desired performance goals. It is important to choose the right scheduling algorithm and to carefully analyze the dependencies between tasks to ensure that the workflow is executed efficiently and correctly. [20]

2.5.3. Algorithms of scheduling

2.5.3.1. First in first out algorithm (FIFO)

The First-In, First-Out (FIFO) algorithm is a simple scheduling algorithm that schedules tasks in the order they arrive. It is also known as the First-Come, First-Served (FCFS) algorithm. In the context of scientific workflows, the FIFO algorithm can be used to schedule independent tasks that do not have any dependencies on other tasks. [18]

2.5.3.2. Shortest Job First (SJF)

It resembles (FIFO) algorithm but (SJF) is a scheduling algorithm used to schedule tasks in order of their execution time. The advantage of the SJF algorithm is that it minimizes the average waiting time and turnaround time of the tasks. This can lead to a more efficient utilization of the computing resources and a shorter overall execution time of the workflow. [23]

2.5.3.3. Round-Robin (RR)

This algorithm assigns a fixed time slice to each task in the queue. When the time slice for a task is up, the next task in the queue is executed, and the process continues in a circular fashion. When a new task is submitted to the queue, it is added to the end of the queue. The task at the front of the queue is then assigned a fixed time quantum for execution. If the task completes execution within the allotted time quantum, it is removed from the queue. If the task does not complete within the allotted time quantum, it is moved to the end of the queue and the next task in the queue is assigned a time quantum for execution. This process continues until all tasks have been completed. The advantage of the RR algorithm is that it provides a fair allocation of computing resources to all tasks, regardless of their execution time. [21]

2.5.3.4. Critical Path Method (CPM)

The CPM algorithm works by creating a directed acyclic graph (DAG) of the workflow, where each node represents a task and each directed edge represents a dependency between two tasks. The algorithm then calculates the earliest and latest start times and earliest and latest finish times for each task in the workflow. This information is used to calculate the slack time for each task, which is the amount of time by which a task can be delayed without affecting the completion time of the workflow. The critical path is identified by finding the longest path through the DAG, where the length of a path is defined as the sum of the expected completion times of the tasks on the path. The expected completion time of the workflow is then calculated as the sum of the expected completion times of the tasks on the critical path.

2.6. Conclusion

In conclusion, scientific workflow scheduling is an important aspect of cloud computing that can significantly impact the performance and efficiency of scientific workflows. A well-designed scheduling strategy can help minimize the overall execution time and maximize the utilization of resources, resulting in faster completion times and reduced costs. It is important to consider the characteristics of the scientific workflow, such as the task dependencies and resource requirements, when designing a scheduling strategy. Additionally, the availability and performance of the cloud computing resources should also be taken into account.

Chapter 3

Description of the proposed approach

3.1. Introduction

The cloud computing platform is characterized by the use of a large pool of computing resources accessible on demand through a service provider on the Internet. A variety of resources is given, such as network, storage, to users adopted by all three cloud computing as services.
In data-intensive or compute-intensive scientific workflows, cloudlets require the execution of multiple datasets. The minimization of the makespan of the workflow, which is the time taken to complete all the cloudlets, by finding the optimal assignment of cloudlets to resources and cloudletss scheduling can be accomplished with the Discrete Symbiotic Organisms Search (DSOS) [24] .
The DSOS algorithm can be applied to various optimization problems in cloud computing and scientific workflows, where the goal is to allocate resources and schedule cloudlets in an optimal manner, leading to improved performance and efficiency of the system.
In this chapter, we will introduce the DSOS algorithm and provide a detailed adaptation of its working principles. We will also discuss the key features of the algorithm, such as its symbiotic relationships, reproduction, and selection mechanisms.

3.2. Representation of Scientific Workflow

In the context of cloudlet scheduling, a scientific workflow refers to a series of cloudlets that need to be executed using particular datasets and in a specific order to achieve a specific result. A scientific workflow can be represented with a directed graph with no directed cycles (acyclic) DAG represented in the next figure, noting G = (T,E) with:

1. T = T1, , Tn the finite set of cloudlets which compose the scientific workflow, that are performed independently.
2. E : is the set of its arcs representing the data constraints between tasks T.

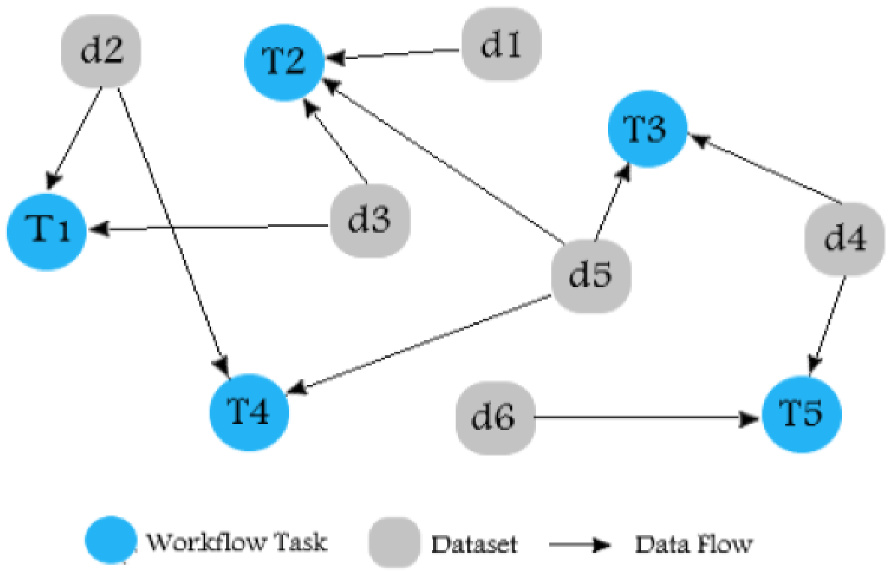
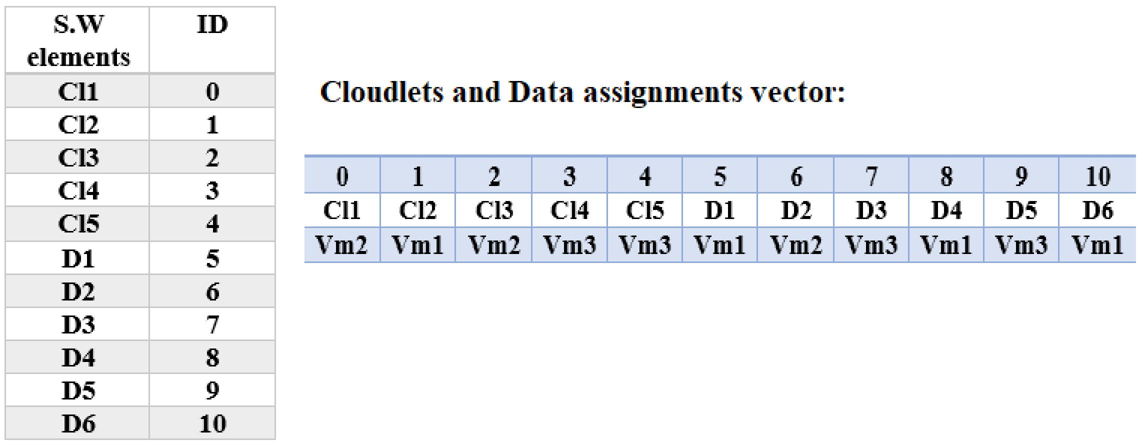


Figure 3.1: *An example of DAG*

3.3. Adaptation description

To solve optimization problem in large complex workflows, an algorithm inspired by the symbiotic relationships between different species of organisms in nature called Discrete Symbiotic Search Organisms (DSOS) was applied.
This algorithm is a metaheuristic optimization technique that can be used to schedule the execution of computational tasks in a workflow to minimize the overall runtime or resource utilization and the selection of the optimal set of experiments that can provide the information for a given scientific question, improving the efficiency and effectiveness of a scientific research. Our approach toward this algorithm is to include both cloudlets and data each cloudlet represents a computational operation that needs to be performed, and each data represents the input of the cloudlet, the cloudlets and data and their allocation to different virtual machines, in a discrete optimization problem, are represented as organisms in the DSOS algorithm. Each organism represents a possible solution to the problem, and the symbiotic relationships between the organisms are used to improve the quality of the solutions in the ecosystem according to the defined fitness function (see equation (4)).
The DSOS algorithm works by simulating the symbiotic relationship between different organisms, such as mutualism, commensalism, and parasitism. The organisms in the algorithm represent potential solutions to the optimization

problem, and they interact with each other to improve their fitness. Discrete Symbiotic Organisms Search (DSOS) algorithm is a discrete approach of Symbiotic Organisms Search (SOS).



S.W elements	ID
Cl1	0
Cl2	1
Cl3	2
Cl4	3
Cl5	4
D1	5
D2	6
D3	7
D4	8
D5	9
D6	10

Cloudlets and Data assignments vector:

0	1	2	3	4	5	6	7	8	9	10
Cl1	Cl2	Cl3	Cl4	Cl5	D1	D2	D3	D4	D5	D6
Vm2	Vm1	Vm2	Vm3	Vm3	Vm1	Vm2	Vm3	Vm1	Vm3	Vm1

Figure 3.2: *An example of an organism (our adaptation).*

3.4. Symbiotic Organisms Search (SOS)

3.4.1. In nature

The Symbiotic Organisms Search Algorithm (SOS) is a nature-inspired optimization algorithm that is based on the concept of symbiosis,

3.4.1.1. Mutualism

In nature, mutualistic symbiosis occurs when two different species of organisms interact with each other in a way that benefits both parties. For example, humans and certain strains of Escherichia E coli (gut bacteria), as illustrated in figure 3.3. It relies on intestinal contents for nutrients, and humans derive certain vitamins from E. coli, particularly vitamin K, which is required for the formation of blood clotting factors. [24]

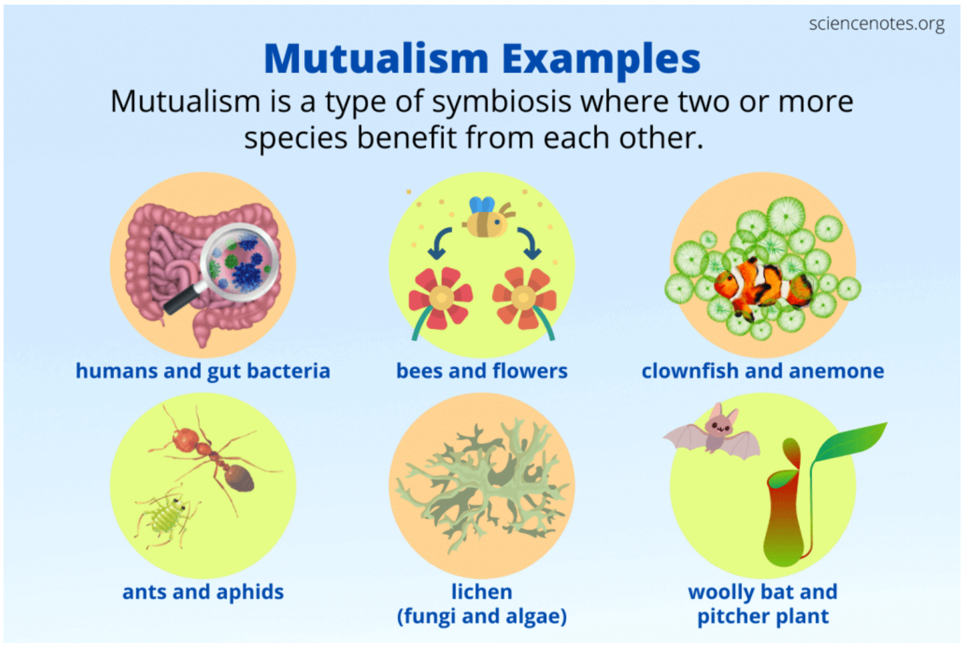


Figure 3.3: *Mutualism examples*
[24]

3.4.1.2. Commensalism

Commensalism is a symbiotic relationship where one species benefits while the other remains unaffected. It's a harmonious interaction where the benefiting species gains advantages without causing harm or receiving benefits in return from its partner. [24]

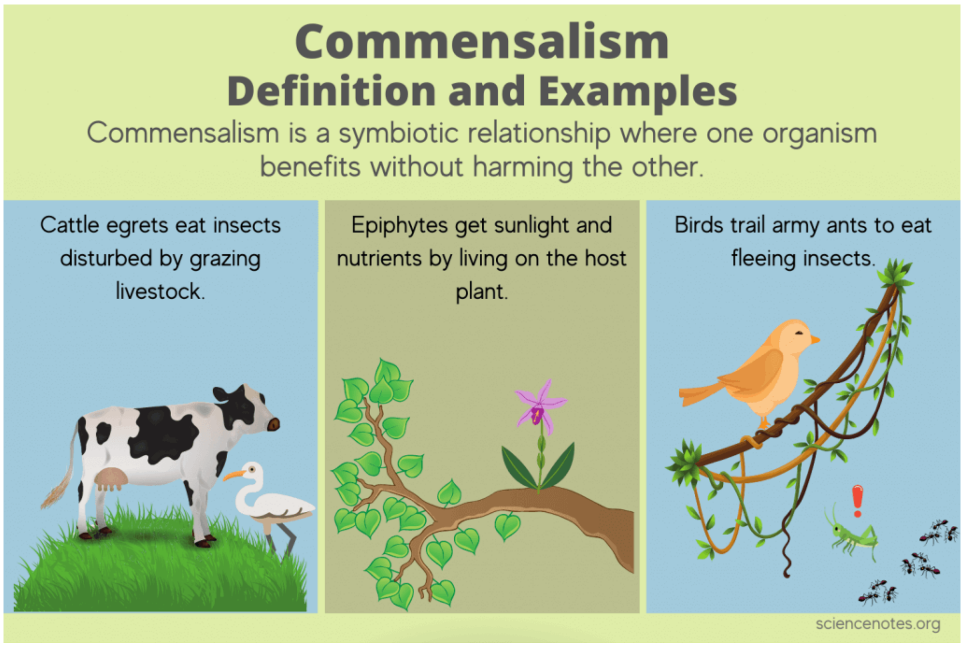

Commensalism
Definition and Examples
Commensalism is a symbiotic relationship where one organism benefits without harming the other.
Cattle egrets eat insects disturbed by grazing livestock.
Epiphytes get sunlight and nutrients by living on the host plant.
Birds trail army ants to eat fleeing insects.
sciencenotes.org

Figure 3.4: *Commensalism examples* [24]

3.4.1.3. Parasitism

Parasitism is a type of symbiotic relationship where one species, known as the parasite, benefits at the expense of another species called the host. In this relationship, the parasite gains advantages while the host experiences harm or negative effects. few example are illustrated in the next figure. [24]

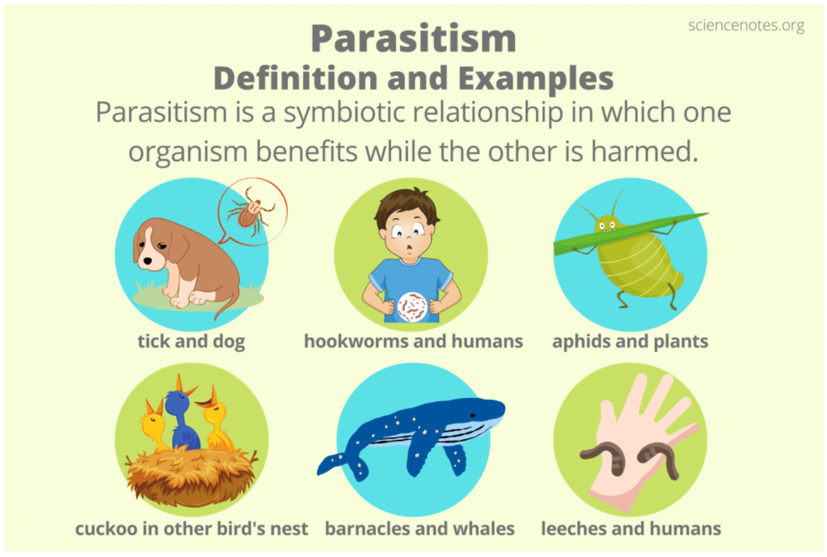


Figure 3.5: *Parasitism examples*
[24]

This algorithm has been successfully applied to a wide range of optimization problems, including engineering design, data mining, and image processing. It is a robust and efficient algorithm that can find optimal solutions in complex problem spaces.

Overall, the Symbiotic Organisms Search Algorithm is an example of how nature can inspire innovative solutions to complex problems. By simulating the mutualistic relationship between different organisms, the SOS algorithm can find optimal solutions in a variety of problem domains.

3.4.2. Formal

The SOS algorithm is a new metaheuristic algorithm for solving numerical optimization problems on a continuous real space, it uses an ecosystem of organisms, which consist of many candidate solutions, examined by it step by step, to eventually find an optimal solution.

Formally, the SOS algorithm works by maintaining the ecosystem, represented as a set of organisms. At each iteration, the algorithm simulates three types of symbiotic relationships between the organisms in the population: mutualism, commensalism, and parasitism.
The SOS starts its process by first randomly generating n number of organisms to populate the ecosystem. Each organism in this case represents a candidate

solution to the corresponding problem with a specific objective function. The search process begins immediately after the creation of the initial ecosystem. Subsequently, the candidate solution or new organism is updated in each evaluation phases of the algorithm, following a common symbiosis strategies adopted by organisms to increase their fitness and survival advantage in the ecosystem. The three phases of symbiosis considered here include mutualism phase, commensalism phase, and parasitism phase respectively. However, an update in each of the phase is only accepted if there is an improvement in the quality of the new solution. The optimization steps are performed iteratively until the algorithm termination condition is met. The three SOS evaluation phases adapted from the most common symbiotic relationships that exist among organisms in nature are briefly described as follows: Consider two organisms x_i and x_j to be coexisting in the ecosystem, where and are the optimization iterative values (indexes) with $\imath = 1, 2, d$ and $\imath \neq j$,in this case, d is the dimension of the problem. [26]

3.4.2.1. Mutualism phase:

in this phase, the organism x_j is randomly selected from the ecosystem to mutually interact with the organism x_i with the sole aim of increasing their mutual survival advantage in the ecosystem. The resulting new solutions x_i' and x_j' which is as a consequence of this interaction is calculated based on equations (1) and (2)

$$x_i' = x_i + r'(x^{\text{best}} - \frac{x_i + x_j}{2})f_1 \tag{3.1}$$

$$x_j' = x_j + r''(x^{\text{best}} - \frac{x_i + x_j}{2})f_2 \tag{3.2}$$

where the function r' generates a vector of random numbers between 0 and 1. The term $\frac{x_i+x_j}{2}$ equations (1) and (2) represents the relationship between the two organisms x_i and x_j . the term x^{best} denotes the highest degree of adaptation for the organisms. The terms f_1 and f_2 denote the mutual benefit factors, which represent the level of benefit that both x_i and x_j derive from the mutual association, since either of the organism can get partial or full benefit from the interaction. Both f_1 and f_2 are determined randomly using the expression:

$\mathrm{f}_1 = f_2 = 1 + \text{round}[r(0, 1)]$

The new candidate solutions x_i' and x_j' are however, accepted only if they give better fitness values than the previous solutions. [25]

3.4.2.2. Commensalism phase:

Similar to the mutualism phase, an organism x_j is randomly selected from the ecosystems population and made to interact with the organism x_i. The relationship interaction is such that only one organism benefits from the interaction. For example, the organism drives benefit from its interaction with x_j, while x_i does not benefit and is neither harmed as a result of the interaction.

$$x_i' = x_i + \text{rand}(-1, 1) \cdot (x^{\text{best}} - x_j) \tag{3.3}$$

where the term $(x^{\text{best}} - x_j)$ represents the benefit provided by the organism x_j to assist to x_i increase its level of survival advantage in the ecosystem. [25]

3.4.2.3. Parasitism phase:

in this phase, an artificial parasite vector denoted by x_{pv} created in the problem search space by mutating the organism $x_i$then modifying its randomly selected dimensions using a random number. The organism x_j with $i \neq j$ is selected randomly from the ecosystems population to serve as a host to the x_{pv} . The evaluation is carried out such that, if the fitness value of the x_{pv} is better than that of the organism x_j, then x_{pv} will replace the position of x_j in the population, otherwise, if the fitness value of x_j is better, then x_j will build an immunity against x_{pv} after which x_{pv} is removed from the population. [25]

3.5. Discrete Symbiotic Organisms Search Algorithm

3.5.1. SOS vs DSOS

Discrete Symbiotic Organisms Search (DSOS) algorithm is a discrete approach towards Symbiotic Organisms Search (SOS) algorithm, proposed by Abdullahi et al [27]. The main difference between these two algorithms lies in their search mechanisms. DSOS is designed for discrete optimization problems, where the decision variables can only take on discrete values. On the other hand, SOS is designed for continuous optimization problems, where the decision variables can take on any real value within a specified range.

3.5.2. Fitness function

A fitness function is used to evaluate the quality of each solution in the population, and to pick the optimal one, solutions can be compared to choose which is fitter, in this case it is the minimum of the makespan time, that is calculated by scheduling the cloudlets to the given VMs, each cloudlet has a processing time, in addition to the Data transfer time, in case the data are not located in the same virtual machine as the cloudlet, represented in equation (4).

$$\text{fitness} = \min\left(\sum_{i=1}^{n}\left(\text{Cloudlets processing time}_i + \text{Data transfer time}\right)\right)$$

$$f = \min\left(\sum_i PT_i + DT_i\right) \tag{3.4}$$

- Cloudlets processing time:

$$\text{PT}_i = \frac{\text{Cloudlet length}_i}{\text{Processing speed}}$$

- Data transfer time:

$$\text{DT}_i = \frac{\sum_{j=1}^{n} \text{Data Transfer time}}{\text{Bandwidth}}$$

3.5.3. Adaptation description

When cloudlets that are going to be scheduled and the data that is going to be used are received by the cloud broker (CB), cloud information service (CIS) has to identify the services required to execute the received cloudlets and data from the user and then schedule the tasks on the proper services, for instance there are tasks T1, T2, T3,...,Tn and data D1, D2, D3,..., Dt may be submitted to CB in a given time interval; the virtual machines that process the tasks are heterogeneous, each one has a different processing speed and memory, so the cost of executing a task varies from a VM to another. Suppose the virtual machines V1, V2, V3, ..., Vm, the cloudlets received by the CB will be executed in a First Come First Serve supposing that the machines are available. However, our goal is to schedule cloudlets and data on VMs in order to achieve higher utilization of VMs with minimal makespan, by finding the best group of tasks to be executed on VMs. Let $C_{ij}(i \in \{1, 2, 3, \ldots, m\}, j \in \{1, 2, 3, \ldots, n\})$ be the execution time of executing jth cloudlets on ith VM where m is the number of VMs is and n is the number of tasks. The fitness value of each organism can be determined using (4), which determines the strength of the level of adaptation of the organism to the ecosystem. [28]

In DSOS, the movement and position of organisms in the continuous space are mapped into developed discrete functions. DSOS consists of three phases: initialization phase, repetition phase, and termination. The initialization phase generates the initial population of organisms. Each organism consists of D elements indicating candidate solutions and a fitness function to determine the extent of optimality of solutions. Therefore, each organism corresponds to a choice for cloudlet schedule encoded in a vector of dimension $1 \times n$, with n being the number of cloudlets. The elements of the vector are natural numbers in the range $[1, m]$, where m is the number of VMs. Suppose x_k is the position of the k-th organism in the solution space; x_j signifies the virtual machine where task j is assigned by scheduler in the organism. The iterative phase mimics the mutualism, commensalism, and parasitism kinds of association to update the positions of the organisms. In mutualism and commensalism stages, x^{best} forms part of the update variables which act as the memory of the procedure. x^{best} is the best point an organism and its neighbors have visited so far. (5) through (8) are used to create modified positions of the selected organisms at mutualism phase.

$$s_1(p) \leftarrow x_i + r_1 \left(x^{\text{best}} - \frac{x_i + x_j}{2} \right) \tag{3.5}$$

$$s_2(p) \leftarrow x_j + r_2 \left(x^{\text{best}} - \frac{x_i + x_j}{2} \right) \tag{3.6}$$

$$x_i'(q) \leftarrow |s_1(P)| \bmod m \tag{3.7}$$

$$x_j'(q) \leftarrow |s_2(P)| \bmod m \tag{3.8}$$

$$\forall p \in \{1, 2, 3, \ldots, n\} \forall q \in \{1, 2, 3, \ldots, m\}$$

We use (9) and (10) to obtain the modified position of the organism xi in the commensalism phase.

$$s_3(p) \leftarrow r_3 \left(x^{\text{best}} - x_j \right) \tag{3.9}$$

$$x_i'(q) \leftarrow |s_3(p)| \bmod m \tag{3.10}$$

$$\forall p \in \{1, 2, 3, \ldots, n\} \forall q \in \{1, 2, 3, \ldots, m\} \quad \text{and } j \neq i$$

Where r_3 is a uniformly generated random number between 0 and 1.

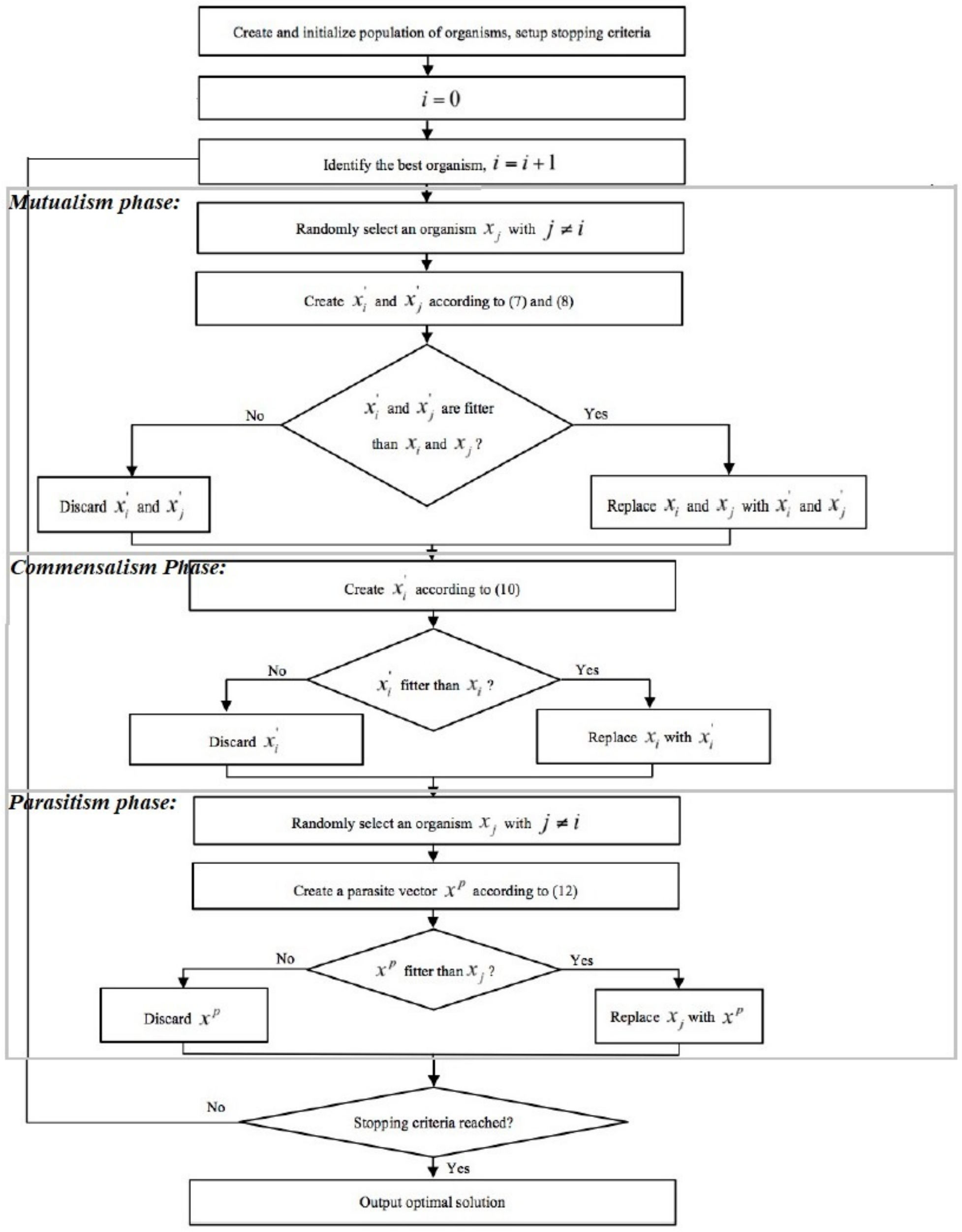

Create and initialize population of organisms, setup stopping criteria
$i = 0$
Identify the best organism, $i = i + 1$
Mutualism phase:
Randomly select an organism X_j with $j \neq i$
Create X_i' and X_j' according to (7) and (8)
X_i' and X_j' are fitter than X_i and X_j?
No
Yes
Discard X_i' and X_j'
Replace X_i and X_j with X_i' and X_j'
Commensalism Phase:
Create X_i' according to (10)
X_i' fitter than X_i?
No
Yes
Discard X_i'
Replace X_i with X_i'
Parasitism phase:
Randomly select an organism X_j with $j \neq i$
Create a parasite vector X^P according to (12)
X^P fitter than X_j?
No
Yes
Discard X^P
Replace X_j with X^P
No
Stopping criteria reached?
Yes
Output optimal solution

Figure 3.6: *Organogram of the DSOS.*

Algorithm 2 Discrete Symbiotic Organism Search Procedure

```
Create and Initialize the population of organisms in ecosystem $X = \{x_1, x_2, x_3, ..., x_N\}$
Set up stopping criteria
$iteration\_number \leftarrow 0$
$x^{best} \leftarrow x_i$
Do
    $iteration\_number \leftarrow iteration\_number + 1$
    $i \leftarrow 0$
        Do
            $i \leftarrow i + 1$
            For $j = 1$ to $N$
                If $f(x_j) < f(x^{best})$ Then // $f(x)$ is the fitness function
                    $x^{best} \leftarrow x_j$
                End if
            End for
            //mutualism phase
            Randomly select $x_j$ with $i \neq j$
            Update $x_i^{'}$ and $x_j^{'}$ according to equation (1) and (2)

            If $f(x_i^{'}) < f(x_i)$ Then
                $x_i \leftarrow x_i^{'}$
            End if

            If $f(x_j^{'}) < f(x_j)$ Then
                $x_j \leftarrow x_j^{'}$
            End if
            //commensalism phase
            Randomly select $x_j$ with $i \neq j$
            Update $x_i^{'}$ according to equation (3)
            If $f(x_i^{'}) < f(x_i)$ Then
                $x_i \leftarrow x_i^{'}$
            End if
            //parasitism phase
            Randomly select $x_j$ with $i \neq j$
            Create a parasite vector $x^p$ according to equation(12)
            If $f(x^p) < f(x_j)$ Then
                $x_j \leftarrow x^p$
            End if
        While $i <= N$
While stopping condition is not true
```

Figure 3.7: *DSOS algorithm.*

3.6. Conclusion

DSOS has been successfully applied to various optimization problems in scientific workflows, it has shown promising results in solving resource allocation problems, has a strong exploration and exploitation capability, which enables it to search a large solution space and identify high-quality solutions. This is particularly important in our field of research, which is scientific workflows that involve complex and diverse tasks and constraints.
Overall, DSOS is a promising optimization algorithm that offers a powerful tool for solving real-world optimization problems in various domains.

Chapter 4

Discussion of the experimental results

4.1. Introduction

In this chapter we will delve in the details of the practical side regarding our work, the scientific workflow optimization based on Discrete Symbiotic Organism Search (DSOS) in cloud computing, we have made a simulator designed in the given approach following the algorithm we introduced in the previous chapter, next we will define our work environment, the language used and present a series of simulations and their interpretations to highlight our proposals.

4.2. Java programming language

Java is a widely used, object-oriented, class based programming language; it is a computing platform for application development. Java offers a high level of reliability, scalability, versatility and most importantly security and it is easy to learn and use.
Java programming language was created by a team of developers led by James Gosling at Sun Microsystems (now a part of Oracle Corporation) in the early 1990s; it was initially called "Oak". In 1995, Sun changed the name to Java and modified the language to take advantage of the development business burgeoning by the www (World Wide Web). [29]
Java programming language is accompanied by several tools and components that are used for developing, running, and managing Java applications, such as:

- Java Development Kit (JDK): The JDK is a software development kit that includes all the necessary tools to develop and run Java applications, such as a compiler, debugger, and runtime environment.
- Java Runtime Environment (JRE): The JRE is a software package that provides the runtime environment necessary to run Java applications.
- Java Virtual Machine (JVM): The JVM is a software component that provides an execution environment for Java applications. It translates Java bytecode into native processor instructions and allows indirect OS or platform program execution.
- Software Development Kit (SDK): The SDK is a collection of software development tools that are used to create, test, and deploy software applications. [30]

4.3. Development environments

4.3.1. Hardware environment

The computer used to implement and test the proposed approach is equipped with an Intel Core i5-5200U processor operating at a speed of 2.20 GHz, and it has 8GB of memory capacity.

4.3.2. Software environment

4.3.2.1. Eclipse

In 2004, IBM established the Eclipse Foundation as an independent non-profit organization to manage the Eclipse project and promote its adoption among the software development community, its membership is available to anyone who wants to contribute, therefore, it is an open-source integrated development environment (IDE); it is free, extensible, universal and versatile integrated known for its plugins that allow developers to develop and test code written in other programming languages. It has the ability to support a wide range of programming languages and frameworks. It can be used from any machine since it's designed to work on a variety of platforms, including Windows, Linux, and macOS.
It provides a comprehensive set of tools for software development in various programming languages. It is cross-platform and includes a rich set of features, such as a powerful code editor with syntax highlighting, code completion and refactoring tools, as well as debugging tools. In the course of time, Eclipse has become widely vast used by software development teams of all sizes, from small startups to large enterprises with a large and active community of developers contributing to the project, helping it evolve. [31]

4.3.2.2. CloudSim Simulator

Cloudsim is also an open-source project, developed by a team of researchers under the guidance of Dr. Raj Kumar Buyya at Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. It is a common simulation tool for modeling and simulating different cloud computing scenarios, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).
It has many features such as the ability to simulate different cloud infrastructures, such as data centers, hosts, and virtual machines; and being a self-contained platform for modeling Clouds, service brokers, provisioning, and allocation poli-

cies. CloudSim supports different scheduling algorithms, such as static, dynamic, and hybrid scheduling. Users can simulate these algorithms to analyze their performance and compare them against each other. This allows users to optimize the scheduling algorithm based on their specific needs and requirements as in our field of study. [32]

4.3.2.3. CloudSim Architecture

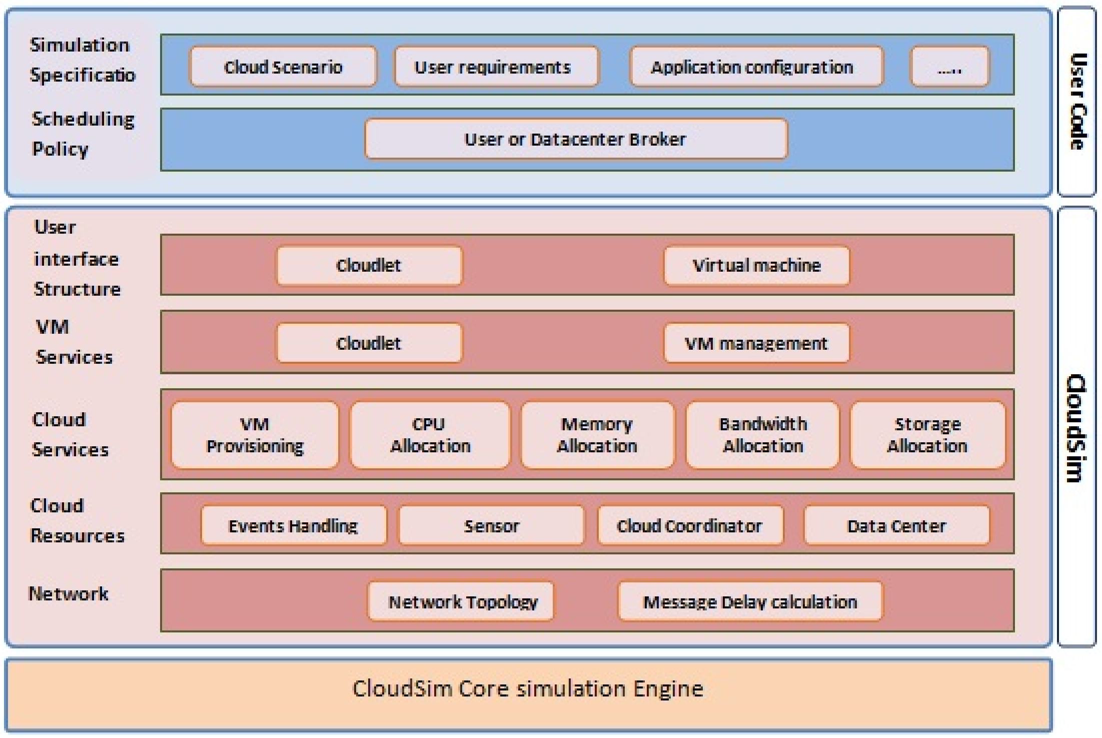


Figure 4.1: *CloudSim Architecture* [33]

- The CloudSim Core simulation engine: provides a modular architecture that allows for the addition of custom models and algorithms, making it highly flexible and extensible. Its comprehensive set of features includes modeling virtual machines, hosts, data centers, and various types of cloud services.
- The CloudSim layer: provides a high-level interface for defining cloud services and workloads, automatically creating the necessary virtual machines, hosts, and data centers. The CloudSim layer abstracts away many

of the details of the simulation engine, making it easier for users to create simulations without needing expertise in programming.

- The User Code layer: provides a flexible and extensible interface that enables researchers to conduct a wide range of cloud computing experiments and simulations.

4.3.2.4. Cloudsim classes

Here are few definitions of the major classes of CloudSim Simulation Toolkit (Version 3.0.3):

- CloudSim: This is the main class of the CloudSim framework, responsible for initializing and starting the simulation. It provides methods for setting up the simulation environment, specifying simulation parameters, and running the simulation.

- Cloudlet: A Cloudlet represents a task or job that runs on a virtual machine in the cloud. It contains information about the resource requirements, execution time, and deadline of the task.

- CloudletScheduler: This class determines the scheduling policy for Cloudlets on a virtual machine. It defines methods for adding, deleting, and querying Cloudlets, as well as allocating resources to them.

- Datacenter: A Datacenter represents a physical data center that hosts virtual machines and provides cloud services. It contains information about the hosts, virtual machines, and data center resources, such as storage and network bandwidth.

- DatacenterBroker: This class acts as a mediator between cloud consumers and data centers, managing the allocation of resources. It contains methods for creating and submitting Cloudlets to data centers, as well as monitoring the progress of the tasks.

- Host: A Host represents a physical machine that can host one or more virtual machines. It contains information about the processing capacity, memory, and storage of the machine.

- Pe: A Pe (processing element) represents a processing core in a physical machine. It contains information about the processing power and utilization of the core.

- Vm: A Vm (virtual machine) represents a virtual machine that runs on a host in the cloud. It contains information about the resource requirements, such as processing power, memory, and storage.

- VmAllocationPolicy: This class determines how virtual machines are allocated to hosts in a data center. It defines methods for selecting hosts for virtual machines, as well as migrating virtual machines between hosts.

- VmScheduler: This class determines the scheduling policy for virtual machines on a host. It defines methods for adding, deleting, and querying virtual machines, as well as allocating resources to them. [34]

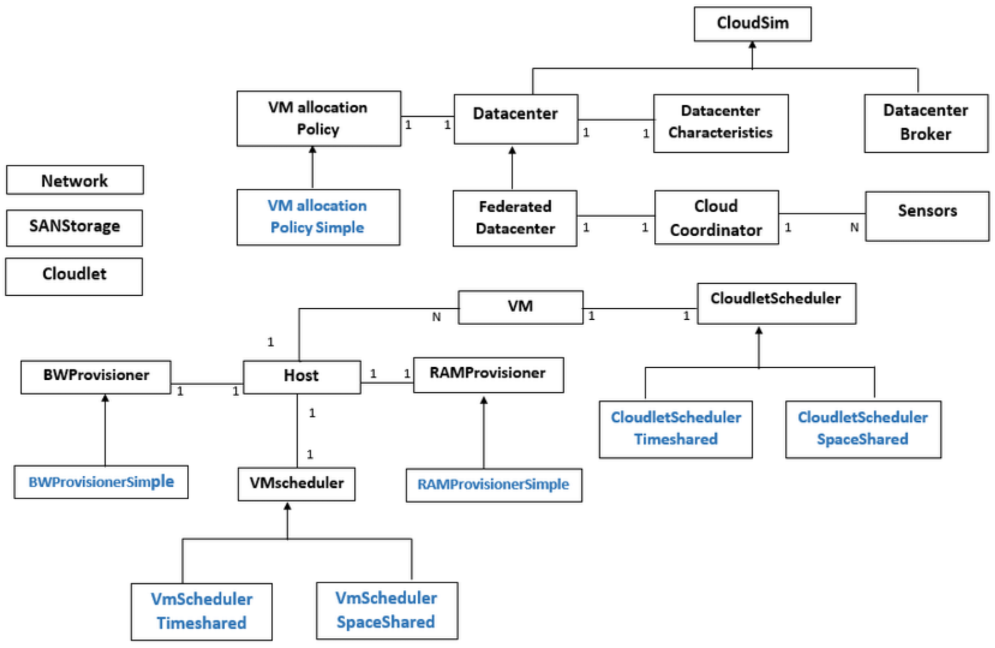


Figure 4.2: *CloudSim classes diagram* [35]

4.4. Main interface

To facilitate the user's access to CloudSim we created an application, considering CloudSim simulator does not have a graphic interface. The first interface gives a way in to the simulator by clicking the button "Launch" (Fig 4.2) the application begins and it transfers the user to the next interface.

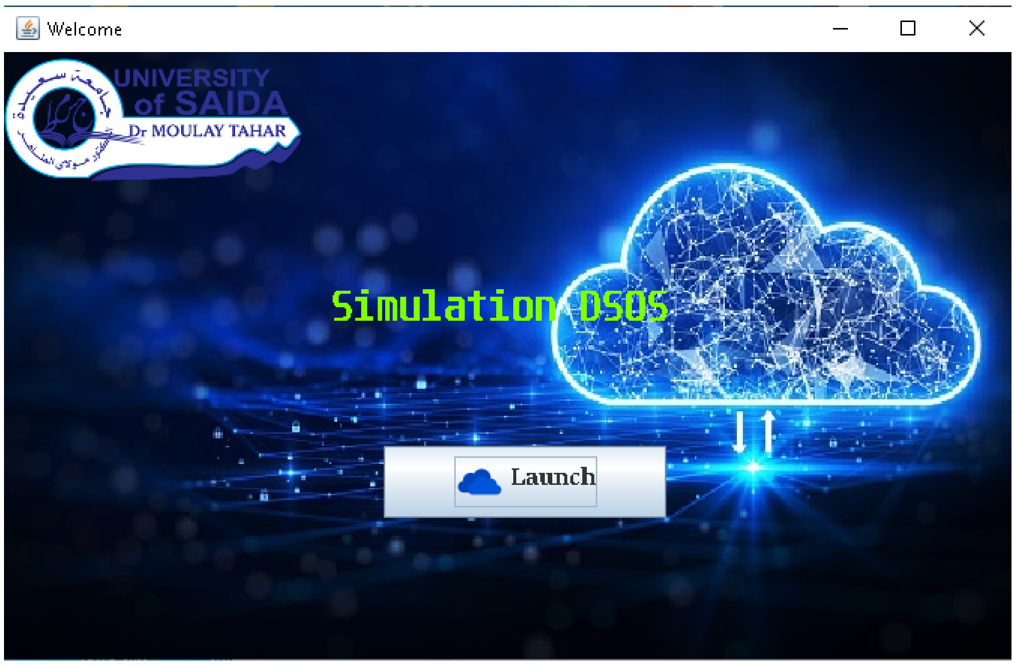


Figure 4.3: *Interface 1*

By launching the application the user

4.5. Simulation setup

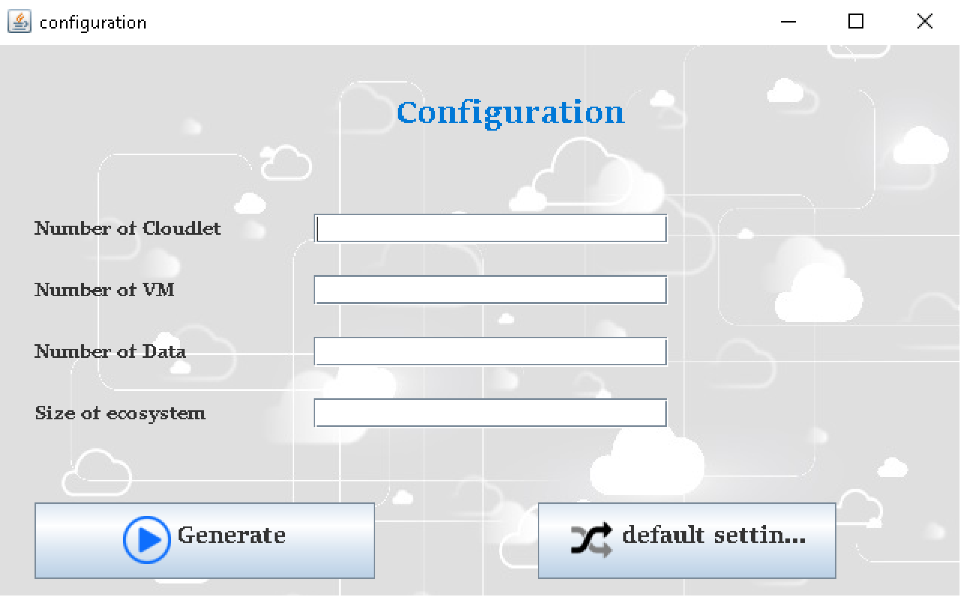


Figure 4.4: *Interface 2*

In the second interface (Figure 4.4), the user will encounter four empty parameters that need to be populated in order to create the ecosystem. These parameters are initially empty, allowing the user the flexibility to either generate them automatically using default settings or manually input specific values as per their requirements.
This empowers the user to have full control over the creation process, ensuring that the ecosystem is tailored precisely to their needs and preferences. Whether the user chooses to rely on the default settings or provide their own inputs, this stage plays a crucial role in defining the characteristics and composition of the ecosystem.

4.5.1. Cloudlets setup

This section provides you with the ability to configure the number of Cloudlets in the ecosystem. By adjusting this parameter, you can define the desired quantity of Cloudlets to be generated.
Moreover, the interface offers a convenient feature: it can automatically generate the parameters of each Cloudlet, including its size, length, and processing speed.

This random generation ensures a diverse range of Cloudlets, each with unique characteristics, contributing to the overall richness and variety of the ecosystem.

4.5.2. Virtual machines setup

the user is allowed you to define the number of virtual machines in the ecosystem. By specifying this parameter, you can establish the desired quantity of virtual machines to be included. Additionally, the interface incorporates a convenient feature where the parameters of each virtual machine are set randomly. This randomization process ensures a diverse set of virtual machines, each with its own unique characteristics. Parameters such as processing power, memory capacity, storage capacity, and network bandwidth are assigned in a randomized manner.

4.5.3. Data setup

insert the number of Data which will be needed in the cloudlet execution.

4.5.4. Ecosystem setup

To customize the size of the ecosystem, the user can make a selection. He has the option to use the "Default settings" button, which will generate all the previous attributes randomly, eliminating the need for manual input.
Alternatively, the user can click the "Generate" button to proceed to the next interface, where he will be prompted to choose the desired iteration number, as depicted in (Fig 4.5).
This iterative selection process allows the user to refine and specify the ecosystem according to their preferences.

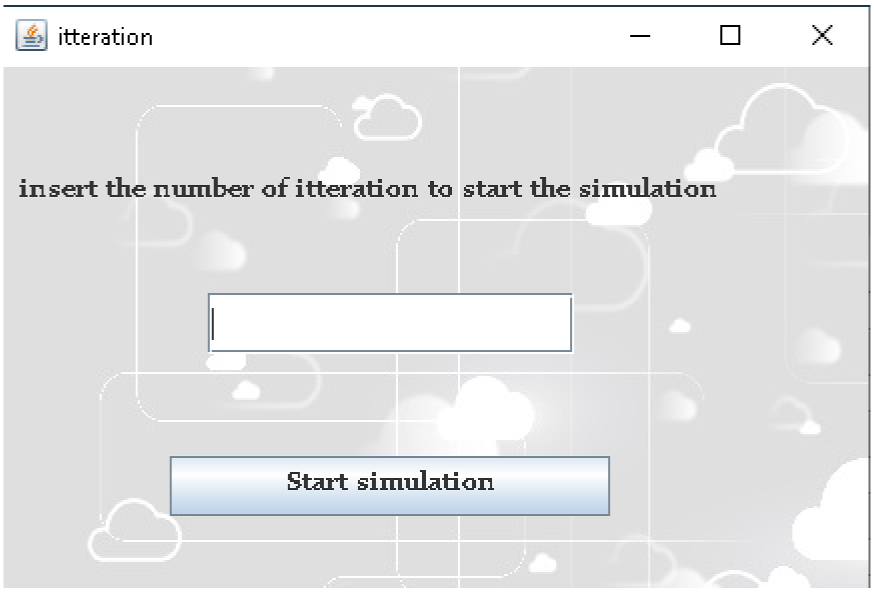


Figure 4.5: *Interface 3*

After picking the iteration number, the user can initiate the simulation by clicking the button in (Fig 4.5) in order to get the results.

4.6. Simulation

A new interface appears that consists of with an empty text field, each button obtain specific results:

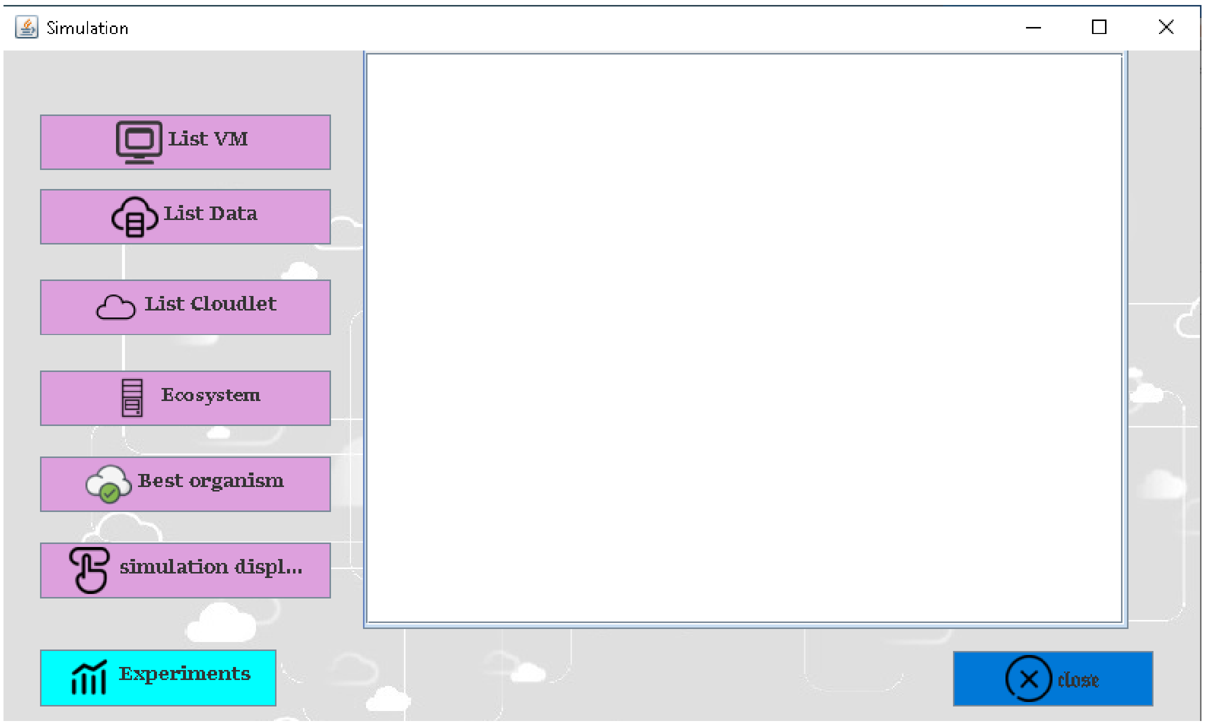
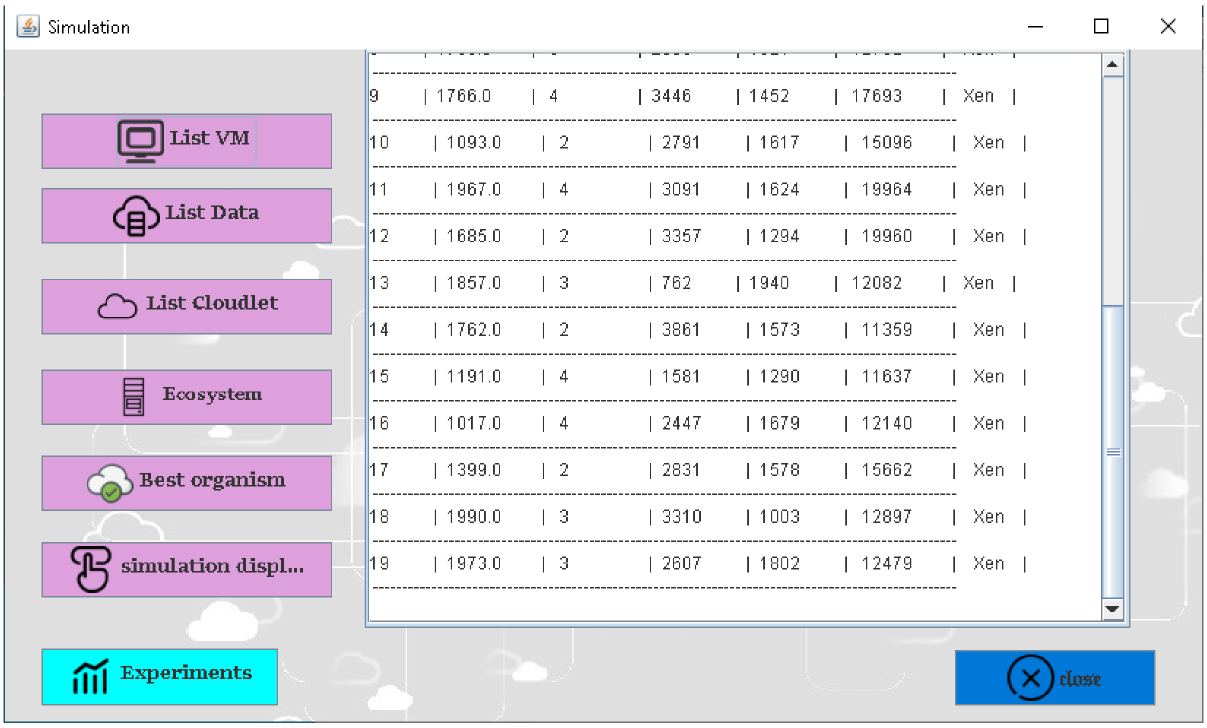


Figure 4.6: *Simulation interface*

4.6.1. Virtual machines list

The list of virtual machines (VMs) appears, showcasing their respective details such as VM ID, MIPS, number of processing elements (PEs), bandwidth, size, and VM name for each virtual machine we have got.

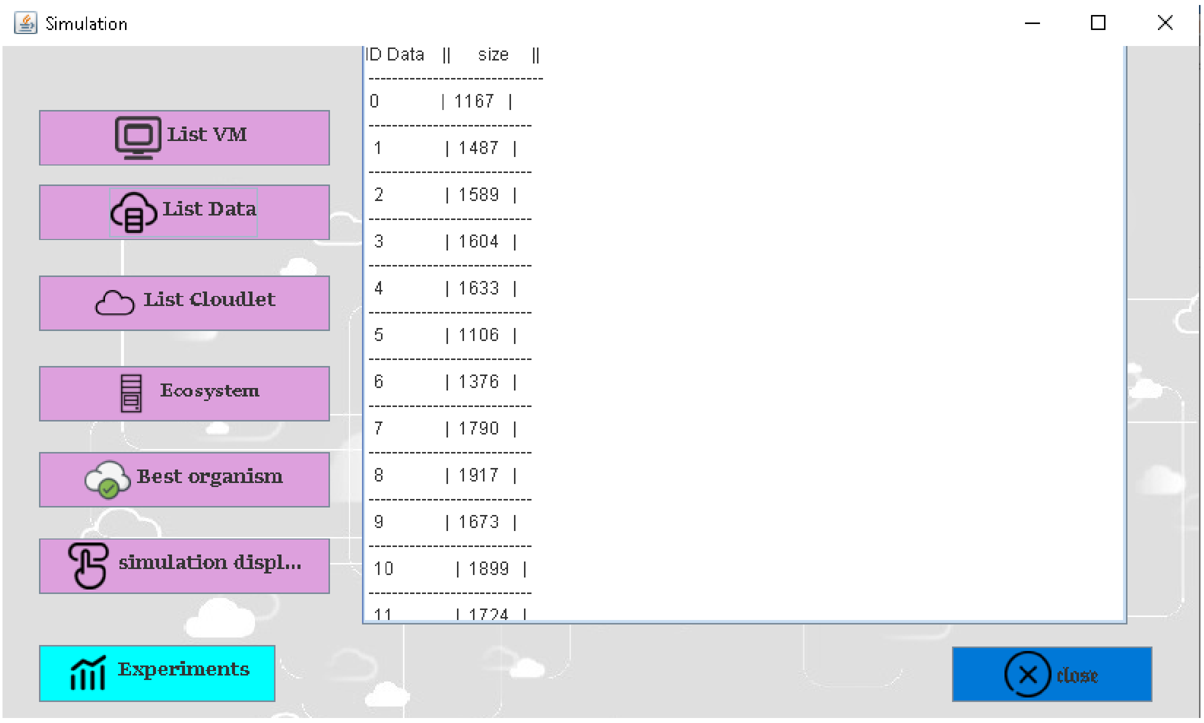


```
--------------------------------------------------------------------------------------------
9     | 1766.0    | 4          | 3446     | 1452     | 17693     |  Xen  |
--------------------------------------------------------------------------------------------
10    | 1093.0    | 2          | 2791     | 1617     | 15096     |  Xen  |
--------------------------------------------------------------------------------------------
11    | 1967.0    | 4          | 3091     | 1624     | 19964     |  Xen  |
--------------------------------------------------------------------------------------------
12    | 1685.0    | 2          | 3357     | 1294     | 19960     |  Xen  |
--------------------------------------------------------------------------------------------
13    | 1857.0    | 3          | 762      | 1940     | 12082     |  Xen  |
--------------------------------------------------------------------------------------------
14    | 1762.0    | 2          | 3861     | 1573     | 11359     |  Xen  |
--------------------------------------------------------------------------------------------
15    | 1191.0    | 4          | 1581     | 1290     | 11637     |  Xen  |
--------------------------------------------------------------------------------------------
16    | 1017.0    | 4          | 2447     | 1679     | 12140     |  Xen  |
--------------------------------------------------------------------------------------------
17    | 1399.0    | 2          | 2831     | 1578     | 15662     |  Xen  |
--------------------------------------------------------------------------------------------
18    | 1990.0    | 3          | 3310     | 1003     | 12897     |  Xen  |
--------------------------------------------------------------------------------------------
19    | 1973.0    | 3          | 2607     | 1802     | 12479     |  Xen  |
--------------------------------------------------------------------------------------------
```

Figure 4.7: *VM list*

4.6.2. Data list

The list of data used in our simulation, and the parameters are the Id and the size of each data.

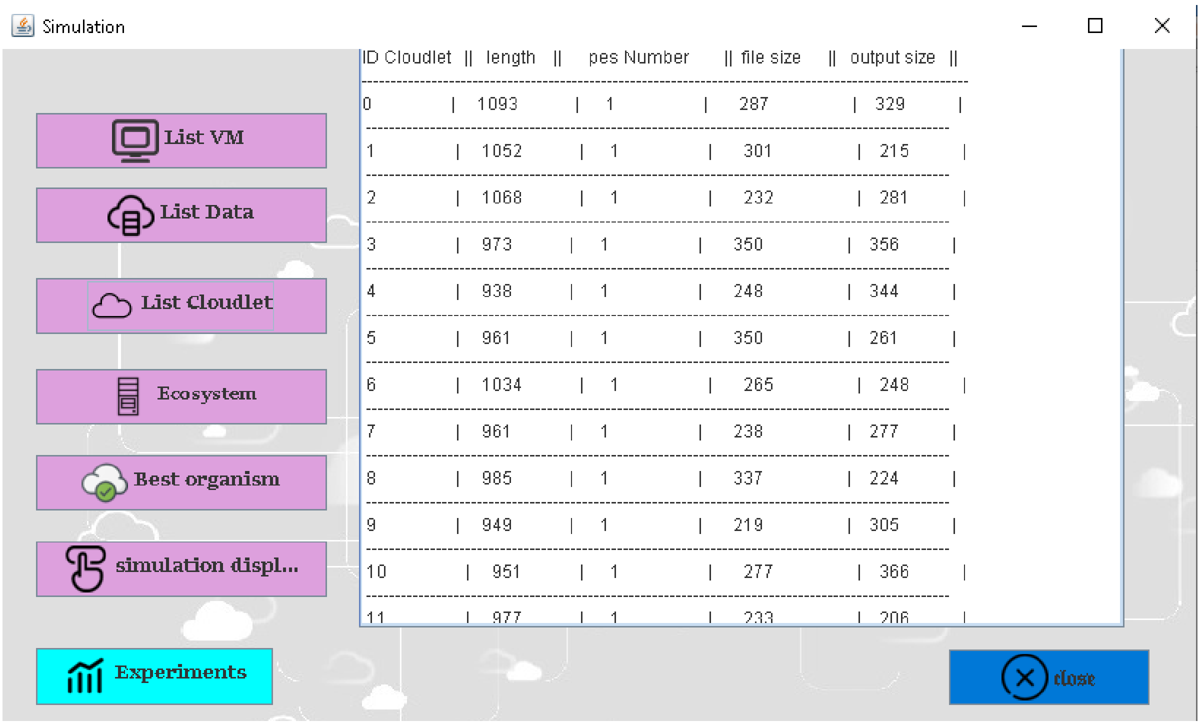


```
ID Data  ||   size   ||
------------------------------
0        | 1167  |
--------------------------
1        | 1487  |
--------------------------
2        | 1589  |
--------------------------
3        | 1604  |
--------------------------
4        | 1633  |
--------------------------
5        | 1106  |
--------------------------
6        | 1376  |
--------------------------
7        | 1790  |
--------------------------
8        | 1917  |
--------------------------
9        | 1673  |
--------------------------
10       | 1899  |
--------------------------
11       | 1724  |
```

Figure 4.8: *Data list*

4.6.3. Cloudlets list

The list of Cloudlets is displayed, providing essential information about each Cloudlet, including its ID, length, number of processing elements (PEs), file size, and output size (fig 4.8)

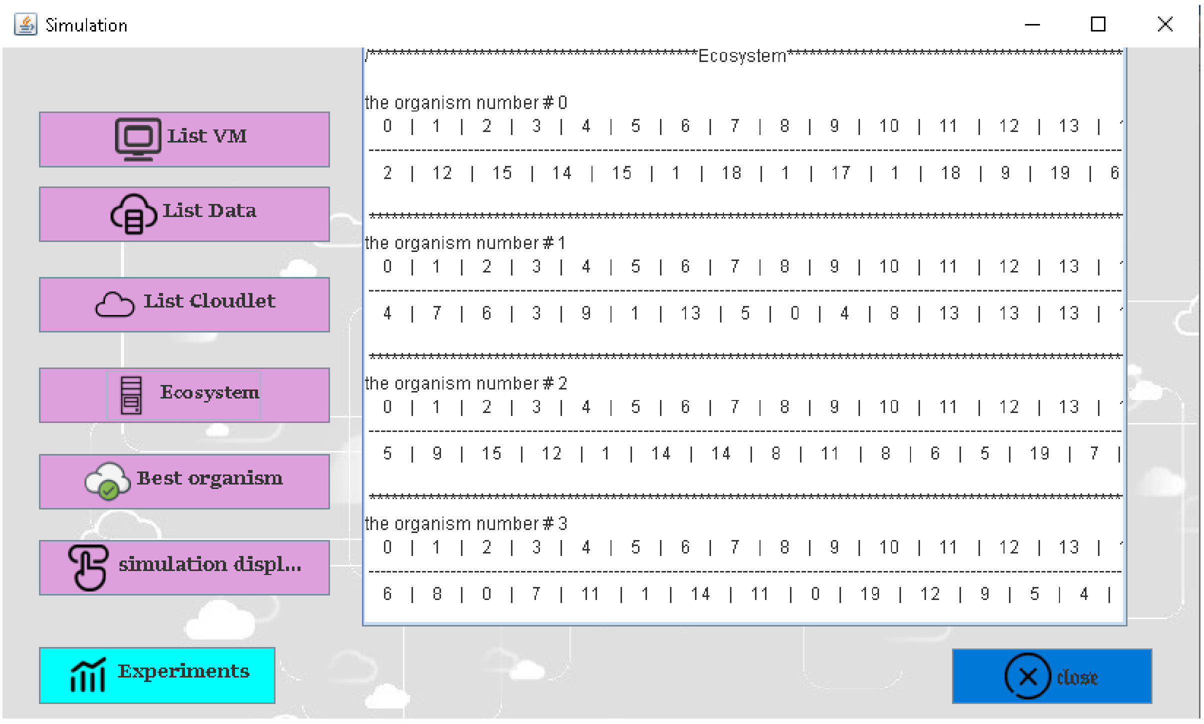


```
ID Cloudlet  ||  length  ||   pes Number    || file size   ||  output size  ||
--------------------------------------------------------------------------------
0        |  1093     |  1          |   287        |  329      |
------------------------------------------------------------------------------
1        |  1052     |  1          |   301        |  215      |
------------------------------------------------------------------------------
2        |  1068     |  1          |   232        |  281      |
------------------------------------------------------------------------------
3        |  973      |  1          |   350        |  356      |
------------------------------------------------------------------------------
4        |  938      |  1          |   248        |  344      |
------------------------------------------------------------------------------
5        |  961      |  1          |   350        |  261      |
------------------------------------------------------------------------------
6        |  1034     |  1          |   265        |  248      |
------------------------------------------------------------------------------
7        |  961      |  1          |   238        |  277      |
------------------------------------------------------------------------------
8        |  985      |  1          |   337        |  224      |
------------------------------------------------------------------------------
9        |  949      |  1          |   219        |  305      |
------------------------------------------------------------------------------
10       |  951      |  1          |   277        |  366      |
------------------------------------------------------------------------------
11       |  977      |  1          |   233        |  206      |
```

Figure 4.9: *Cloudlets list*

4.6.4. List of the ecosystem

By leveraging all the aforementioned attributes, a diverse ecosystem of organisms is created. Clicking the "List of the ecosystem" button reveals a comprehensive view of this ecosystem, showcasing each organism along with their associated elements, including cloudlets, data, and the corresponding virtual machines to which they are assigned.

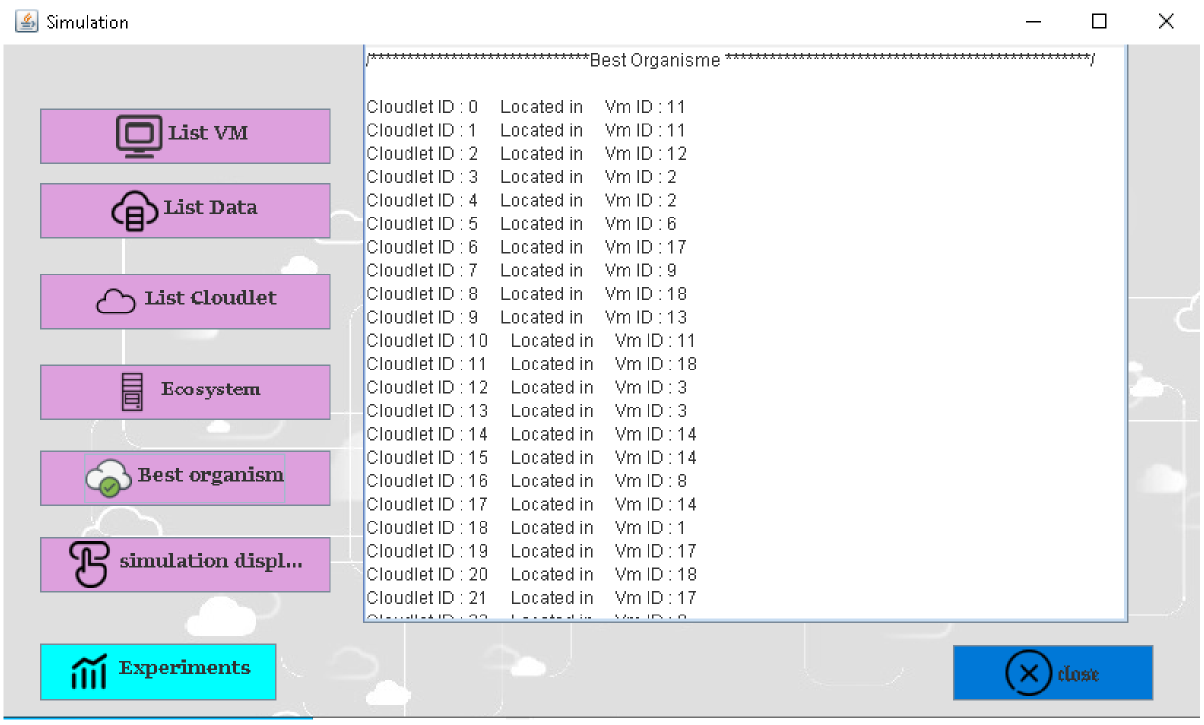


```
/*******************************************Ecosystem*******************************************

the organism number # 0
  0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |  10  |  11  |  12  |  13  |
-------------------------------------------------------------------------------------------------
  2  |  12  |  15  |  14  |  15  |  1  |  18  |  1  |  17  |  1  |  18  |  9  |  19  |  6

*************************************************************************************************
the organism number # 1
  0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |  10  |  11  |  12  |  13  |
-------------------------------------------------------------------------------------------------
  4  |  7  |  6  |  3  |  9  |  1  |  13  |  5  |  0  |  4  |  8  |  13  |  13  |  13  |

*************************************************************************************************
the organism number # 2
  0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |  10  |  11  |  12  |  13  |
-------------------------------------------------------------------------------------------------
  5  |  9  |  15  |  12  |  1  |  14  |  14  |  8  |  11  |  8  |  6  |  5  |  19  |  7  |

*************************************************************************************************
the organism number # 3
  0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |  10  |  11  |  12  |  13  |
-------------------------------------------------------------------------------------------------
  6  |  8  |  0  |  7  |  11  |  1  |  14  |  11  |  0  |  19  |  12  |  9  |  5  |  4  |
```

Figure 4.10: *List of the ecosystem*

4.7. Best organism

Upon completing all the phases and steps of the selected DSOS algorithm, thorough calculations were performed, leading us to identify the optimal organism (X best) . On the screen, the details of this organism are showcased, including the allocation of each cloudlet and data, indicating their respective locations. the parameters inserted are 40 cloudlets, 20 VMs, 20 data with the size of ecosystem equals 200 and 1000 iterations.

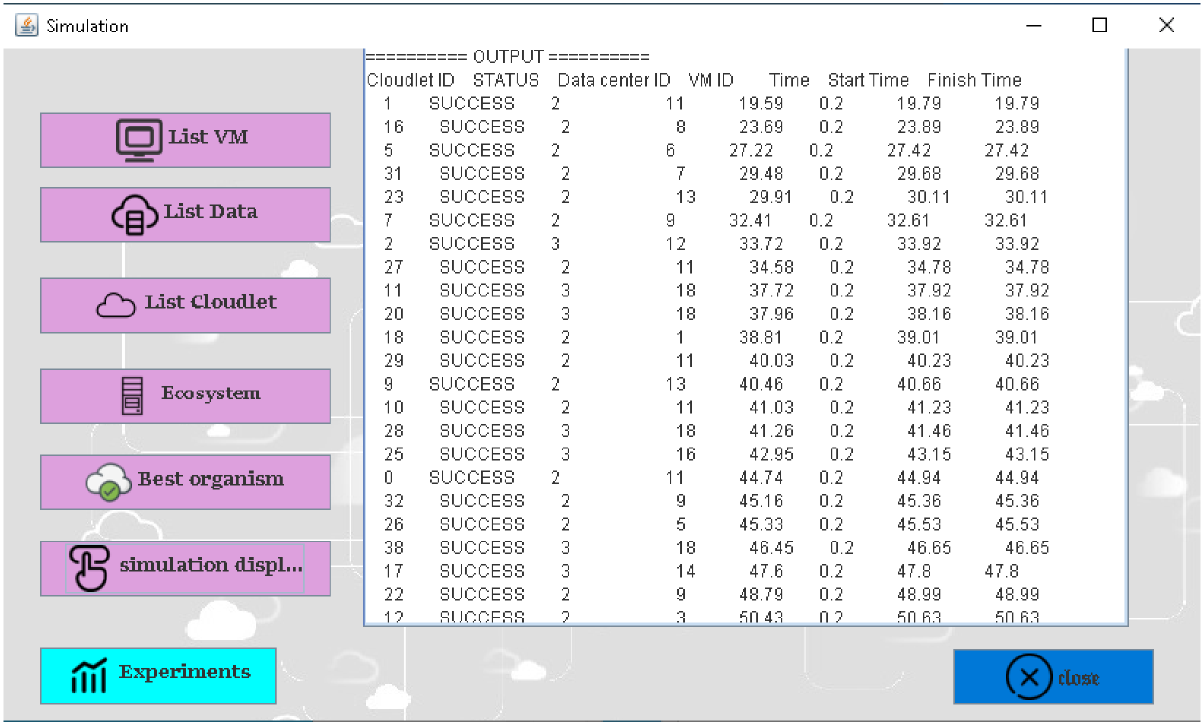


```
/*******************************Best Organisme *****************************************************/

Cloudlet ID : 0     Located in     Vm ID : 11
Cloudlet ID : 1     Located in     Vm ID : 11
Cloudlet ID : 2     Located in     Vm ID : 12
Cloudlet ID : 3     Located in     Vm ID : 2
Cloudlet ID : 4     Located in     Vm ID : 2
Cloudlet ID : 5     Located in     Vm ID : 6
Cloudlet ID : 6     Located in     Vm ID : 17
Cloudlet ID : 7     Located in     Vm ID : 9
Cloudlet ID : 8     Located in     Vm ID : 18
Cloudlet ID : 9     Located in     Vm ID : 13
Cloudlet ID : 10     Located in     Vm ID : 11
Cloudlet ID : 11     Located in     Vm ID : 18
Cloudlet ID : 12     Located in     Vm ID : 3
Cloudlet ID : 13     Located in     Vm ID : 3
Cloudlet ID : 14     Located in     Vm ID : 14
Cloudlet ID : 15     Located in     Vm ID : 14
Cloudlet ID : 16     Located in     Vm ID : 8
Cloudlet ID : 17     Located in     Vm ID : 14
Cloudlet ID : 18     Located in     Vm ID : 1
Cloudlet ID : 19     Located in     Vm ID : 17
Cloudlet ID : 20     Located in     Vm ID : 18
Cloudlet ID : 21     Located in     Vm ID : 17
```

Figure 4.11: *Organism best*

4.8. Simulation display

The following figure presents the results of our approach and random placement approach , depicting each cloudlet's status (success or failure), along with their corresponding data center and assigned virtual machine (VM). Additionally, it displays the execution time for each cloudlet, defining the start and finish times.

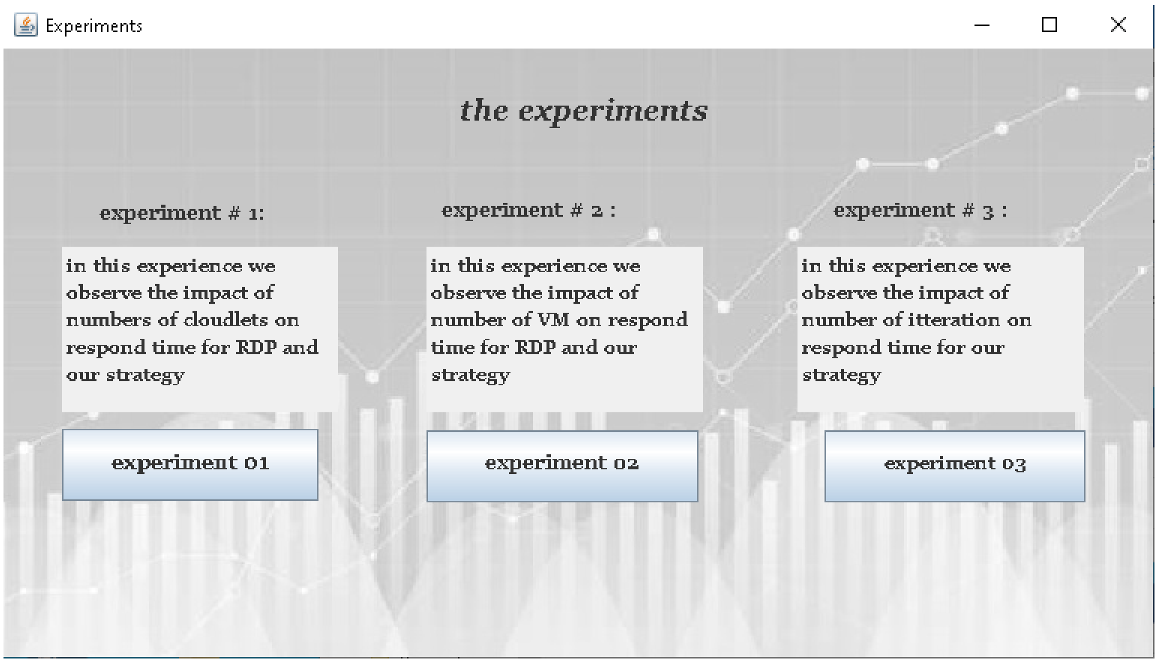


```
========== OUTPUT ==========
Cloudlet ID   STATUS   Data center ID   VM ID     Time   Start Time   Finish Time
  1     SUCCESS     2               11       19.59     0.2        19.79        19.79
  16     SUCCESS     2               8       23.69     0.2        23.89        23.89
  5     SUCCESS     2               6       27.22     0.2        27.42        27.42
  31     SUCCESS     2               7       29.48     0.2        29.68        29.68
  23     SUCCESS     2               13       29.91     0.2        30.11        30.11
  7     SUCCESS     2               9       32.41     0.2        32.61        32.61
  2     SUCCESS     3               12       33.72     0.2        33.92        33.92
  27     SUCCESS     2               11       34.58     0.2        34.78        34.78
  11     SUCCESS     3               18       37.72     0.2        37.92        37.92
  20     SUCCESS     3               18       37.96     0.2        38.16        38.16
  18     SUCCESS     2               1       38.81     0.2        39.01        39.01
  29     SUCCESS     2               11       40.03     0.2        40.23        40.23
  9     SUCCESS     2               13       40.46     0.2        40.66        40.66
  10     SUCCESS     2               11       41.03     0.2        41.23        41.23
  28     SUCCESS     3               18       41.26     0.2        41.46        41.46
  25     SUCCESS     3               16       42.95     0.2        43.15        43.15
  0     SUCCESS     2               11       44.74     0.2        44.94        44.94
  32     SUCCESS     2               9       45.16     0.2        45.36        45.36
  26     SUCCESS     2               5       45.33     0.2        45.53        45.53
  38     SUCCESS     3               18       46.45     0.2        46.65        46.65
  17     SUCCESS     3               14       47.6     0.2        47.8        47.8
  22     SUCCESS     2               9       48.79     0.2        48.99        48.99
  12     SUCCESS     2               3       50.43     0.2        50.63        50.63
```

Figure 4.12: *Simulation display*

4.9. Experimental results

In order to showcase the merits of our approach, we will primarily emphasize the response time metric. To comprehensively study the behavior of our proposal and analyze its simulation results, we will compare them with the random placement approach. We have conducted three sets of simulations to facilitate this comparative analysis.



in this experience we observe the impact of numbers of cloudlets on respond time for RDP and our strategy

in this experience we observe the impact of number of VM on respond time for RDP and our strategy

in this experience we observe the impact of number of itteration on respond time for our strategy

Figure 4.13: *Experiments interface*

4.9.1. Experiment 1

Impact of cloudlet number on the respond time:
In this simulation, we have designed three Data Centers comprising of 2 diverse Hosts. Each Host is equipped with a single processor with varying speeds in MIPS ranging from 20,000 to 200,000. The bandwidth of each Host ranges from 10,000 to 200,000. The size of the generated data is randomly generated between 900 MB and 1,100 MB. The purpose of this simulation is to investigate the effect of varying the number of Cloudlets on the response time.
We fixed 30 virtual machines, 10 data, 200 organisms and 1000 itteration.

Number of Cloudlets	Space shared + Random placement	Our strategie (DSOS)
10	76.63	40.12
40	182.84	92.81
80	581.31	315.25
200	869.73	669.9
400	1153.92	862.39

Table 4.1: *Impact of cloudlets number on the response time (ms)*

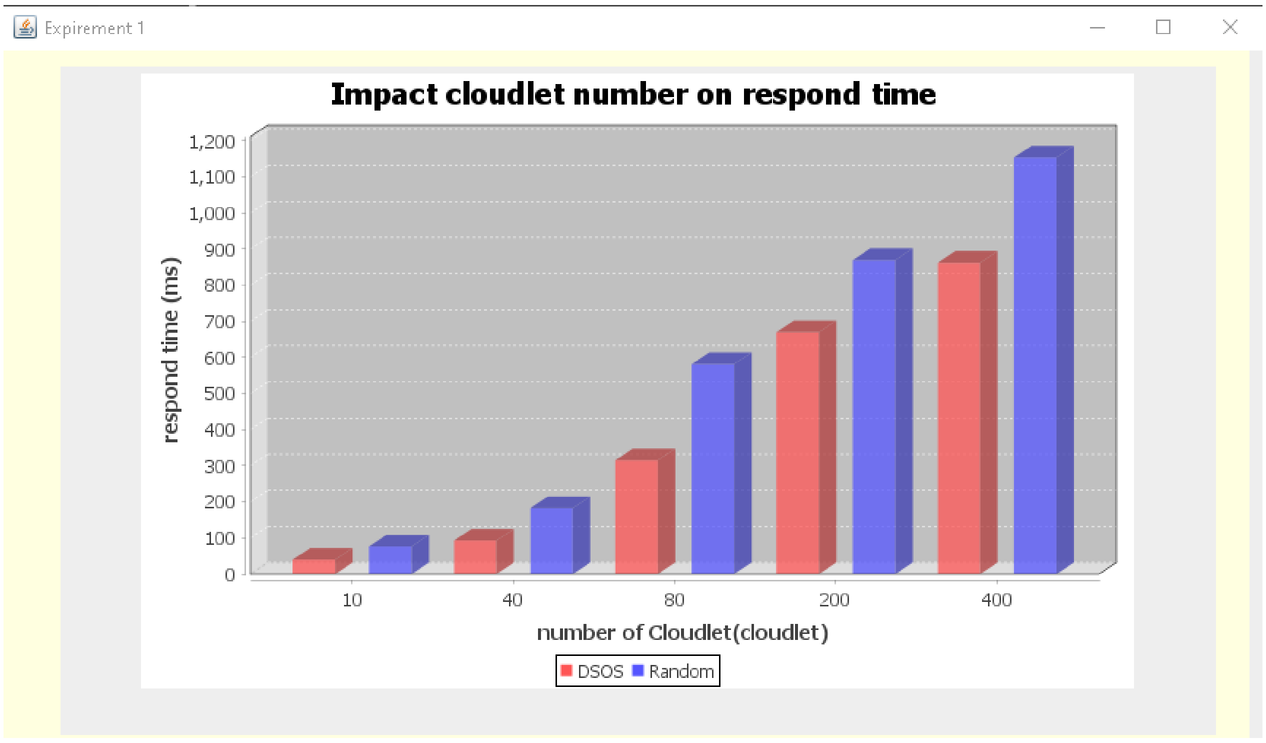


Figure 4.14: *Impact of cloudlets number on the response time*

Table 4.1 reveals the outcomes of a simulation conducted to examine the impact of varying the number of Cloudlets, comparing the results of our proposed approach (DSOS) to the space shared and random placement strategy we note that our approach provide reduced response time.

4.9.2. Experiment 2

Impact of VMs number on the response time:

In this experiment, we set up three Data Centers comprising 2 diverse Hosts. Each Host is equipped with a single processor, with speeds varying between 20,000 and 200,000 MIPS, and bandwidth ranging from 10,000 to 20,000, bandwidth (ranging from 2,000 to 20,000).

The objective of this experiment is to analyze the impact of the VMs number on the response time, we fixed 100 cloudlets, 40 data, 200 organisms with 1000 iteration.

Number of VMs	Space shared + Random placement	Our strategie (DSOS)
10	862.27	713.84
20	466.76	198.03
30	316.26	180.21
50	241	96.94
80	199.94	80.13

Table 4.2: *Impact of VMs number on the response time (ms)*

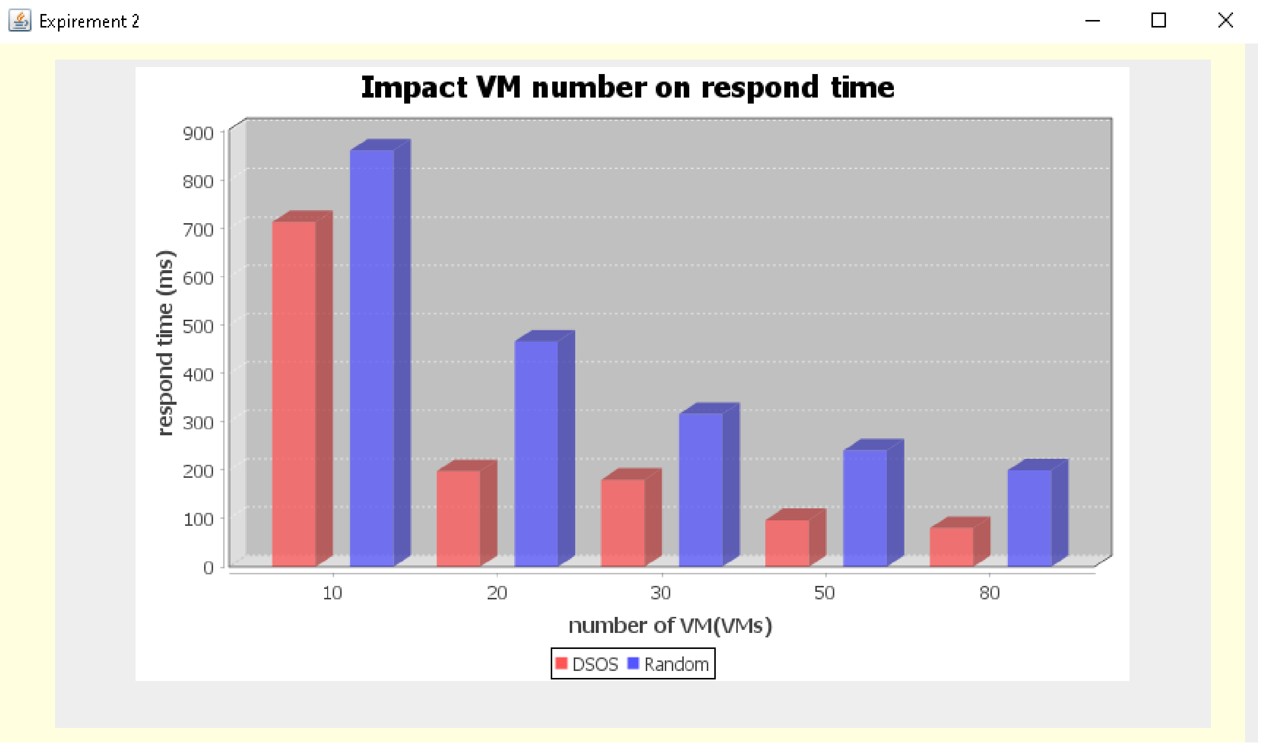


Figure 4.15: *Impact of the number of VMs on the response time*

In table 4.2 we conducted a comparison between the results of our chosen approach (DSOS) and thus of the space shared and random placement strategy. Concluding that our method offers less response time, which proves the reliability of our strategy.

Number of iterations	Our strategie (DSOS)
200	341.81
400	339.25
600	339.25
800	335.51
1000	335.51

Table 4.3: *Impact of VMs number on the response time (ms)*

4.9.3. Experiment 3

Impact of the iteration number on the response time:

We conducted an experiment fixing 100 cloudlets, 30 virtual machines, 40 data and 200 organisms to observe the impact of the iteration number on the response time, the results are shown in the next table and figure.

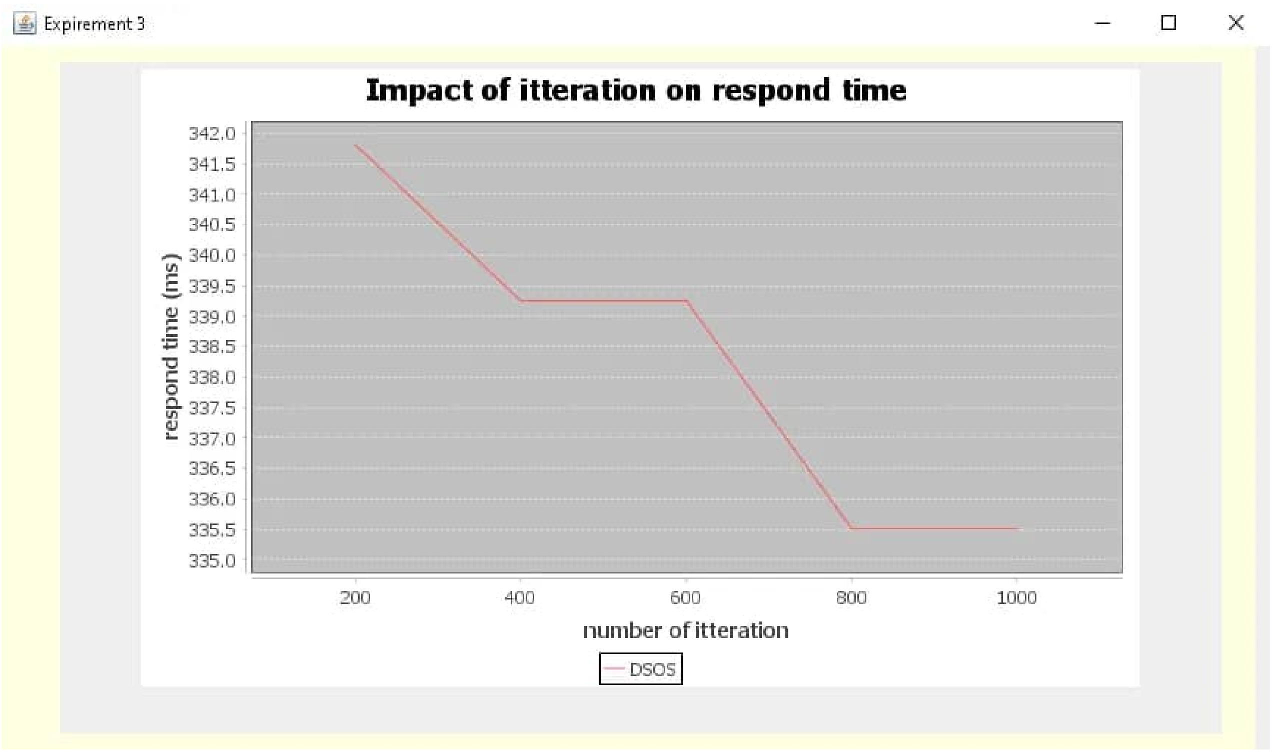


Figure 4.16: *Impact of the iteration on the response time*

As the number of iterations increases, the response time decreases noticeably.

This indicates the significant impact of iteration on improving system efficiency and reducing response time.
It is believed that the improvement in performance through DSOS is attributed to three operators: Mutualism, Commensalism, Parasitism. These operators have a higher probability of obtaining progressive solutions. DSOS is capable of converging towards an almost optimal solution

4.10. Results discussion

we explain this improvement in performance by DSOS to be originated from the mechanisms of mutual benefit and parasite vector that are unique to DSOS.

The mutual benefit factor mechanism in the mutualism phase gives the search process exploitative power by allowing it to explore the best solution regions. The commensalism mechanism allows for exploring new solutions without negatively affecting the current solution. This means that the algorithm can continue to search for better solutions while retaining a good solution at each iteration. The commensalism mechanism also helps diversify the search space, which can help avoid getting stuck in local optima.

Overall, the commensalism mechanism balances exploration and exploitation, resulting in better performance and more efficient optimization. The parasite vector technique in the parasitism phase is capable of preventing premature convergence by eliminating inactive solutions and introducing a more active solution that moves the search process away from local optima.

The parasitism phase gives the search process the ability to explore by not solely focusing on the best solution regions that could potentially trap the search in a certain area of exploration. These mechanisms play a vital role in the exploration and exploitation in the search process.
DSOS has fewer parameters and is easier to implement, which is considered an advantage in addition to its explorability and exploitability. The method is capable of improving the quality of the search process, which means that DSOS has a higher probability of obtaining a nearly optimal solution.

General conclusion

Cloud computing has fundamentally transformed the IT industry by providing scalable and cost-effective solutions for data storage and access. Its flexibility and convenience have revolutionized how businesses and individuals operate. Meanwhile, task scheduling is a critical aspect of cloud computing that ensures efficient utilization of computational resources. By intelligently assigning tasks to available resources, scheduling algorithms optimize performance, minimize response time, and reduce costs. It has a vital role in scientific workflows, optimizing the execution of complex computations by managing task dependencies and allocating resources efficiently. In this context the DSOS algorithm offers valuable assistance by optimizing cloudlet's scheduling, minimizing the makespan time by selecting the best organism after going through the three phases of the algorithm.

Through the development of our CloudSim simulator using the DSOS algorithm, we conducted a series of experiments to showcase the notable advantages of the approach we employed. The results validate the effectiveness and value of the approach we implemented, confirming its success in achieving desirable results.

the Discrete Symbiotic Organism Search (DSOS) algorithm stands as a promising metaheuristic approach for solving combinatorial optimization problems with discrete variables.

As a future perspective for us:

- Explore hybrid optimization approaches by combining DSOS with other algorithms to enhance scientific workflow optimization in the cloud.

- Integrate cost optimization techniques into the DSOS framework to minimize expenses while optimizing performance.

- Validate DSOS performance in large-scale cloud environments with diverse datasets to ensure robustness and efficiency.

- Extend DSOS to handle dynamic changes in cloud environments for adaptive task scheduling and resource allocation.

Bibliography

[1] Stanoevska-Slabeva, K., Wozniak, T. (2010). Cloud Basics An Introduction to Cloud Computing. In: Stanoevska-Slabeva, K., Wozniak, T., Ristol, S. (eds) Grid and Cloud Computing. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-05193-7$_4$

[2] Mell, P., Grance, T. (September 2011). National Institute of Standards and Technology Special Publication 800-145

[3] Rajinder, S., Inderveer, C. (2013). Cloud Computing Standardization Initiatives: State of Play, 10.11591/closer.v2i5.4894. International Journal of Cloud Computing and Services Science (IJ-CLOSER)

[4] Basics - Cloud computing. https://www.geeksforgeeks.org/cloud-computing/

[5] Main cloud service models: IaaS, PaaS and SaaS, (2023), https://www.stackscale.com/blog/cloud-service-models/

[6] A.T.Velte, T.J.Velte,Ph.D, R. Elsenpeter, (2010), Cloud Computing: A Practical Approach, a book.

[7] Implementation of a private IaaS cloud and analysis of big data within a distributed system at ACM (Master's degree, 2019-2020).

[8] David Chou, (2018), Cloud Service Models (IaaS, PaaS, SaaS) Diagram, https://dachou.github.io/2018/09/28/cloud-service-models.html

[9] D. Koshkin, Cloud Deployment Models: Advantages and Disadvantages, https://sam-solutions.us/advantages-and-disadvantages-of-cloud-deployment-models/

[10] C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter, (2014), Cloud Computing Patterns (Fundamentals to Design, Build, and Manage Cloud Applications) (a book)

[11] Bhatia,V., Essential Cloud Computing Characteristics. https://www.synopsys.com/cloud/insights/essential-cloud-computing-characteristics.html /doi/10.5555/1983741.1983763

[12] Vicat-Blanc P, Figuerola S, Chen X, Landi G, Escalona E, Develder C, Tzanakaki A, Demchenko Y, Espín J, Ferrer J, López E, Soudan S, Buysse J, Jukan A, Ciulli N, Brogle M, van Laarhoven L, Belter B, Anhalt F, Nejabati R, Simeonidou D, Ngo C, de Laat C, Biancani M, Roth M, Donadio P, Jiménez J, Antoniak-Lewandowska M and Gumaste A. Bringing optical networks to the cloud. The future internet. (307-320).

[13] https://www.sketchbubble.com/en/presentation-cloud-architecture.html

[14] J.W. Rittinghouse, J.F. Ransome, (2010), Cloud Computing Implementation, Management, and Security (a book)

[15] Rodriguez, M. A., Buyya, R. (2014). Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. IEEE transactions on cloud computing, 2(2), 222-235.

[16] Thramboulidis, K. , Lai, B. C., Cao, J., Talia, Domenico, (2013), Workflow Systems for Science: Concepts and Tools, 10.1155/2013/404525, ISRN Software Engineering https://doi.org/10.1155/2013/404525

[17] K. Ganga and S. Karthik, (2013), A fault tolerent approach in scientific workflow systems based on cloud computing, International Conference on Pattern Recognition, Informatics and Mobile Engineering, Salem, India, 2013, pp. 387-390, doi: 10.1109/ICPRIME.2013.6496507.

[18] Wang, Yawen , Guo, Yunfei, Guo, Zehua, Liu, Wenyan, Yang, Chao (2020) , Protecting Scientific Workflows in Clouds with an Intrusion Tolerant System 10.1049/iet-ifs.2018.5279

[19] Abrishami, S., Naghibzadeh, M. (2012). Deadline-constrained workflow scheduling in software as a service cloud. Scientia Iranica, 19(3), 680-689.

[20] Bessai, K. Gestion optimale de lallocation des ressources pour lexécution des processus dans le cadre du Cloud (thèse de Doctorat, Université Paris1 Panthéon-Sorbonne), 2014.

[21] S., Mohapatra, S. Mohanty, and K.S. Rekha, Analysis of different variants in round robin algorithms for load balancing in cloud computing, International Journal of Computer Applications, vol. 69, no. 22, 2013.

[22] Rahul,M. A Brief Review of Scheduling Algorithms in Cloud Computing. Asian Journal of Technology Management Research, vol.05, 2015.

[23] Akhtar, Muhammad AND Hamid, Bushra AND Ur-Rehman, Inayat AND Humayun, Mamoona and Hamayun, Maryam Khurshid, Hira. An Optimized Shortest job first Scheduling Algorithm for CPU Scheduling. J. Appl. Environ. Biol. Sci , 5(12)42-46, 2015. 5. 42-46. 2015.

[24] B. D. Martin, E. Schwab, (2012), Current Usage of Symbiosis and Associated Terminology, School of Allied Health Professions, Loma Linda University, Loma Linda, CA 92350, USA http://dx.doi.org/10.5539/ijb.v5n1p32

[25] Absalom El-Shamir Ezugwu , Aderemi Oluyinka Adewumi , Discrete Sym-biotic Organisms Search Algorithm for Travelling Salesman Problem, Expert Systems With Applications (2017), doi: 10.1016/j.eswa.2017.06.007

[26] M. Abdullahi, M.A. Ngadi, S.M. Abdulhamid, Symbiotic Organism Search optimization based task scheduling in cloud computing environment, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.08.006

[27] Megha Sharma, Amandeep Verma, Energyaware Discrete Symbiotic Organism Search Optimization algorithm for task scheduling in a cloud environment, (2017)

[28] SongIl Choe, Bo Li, IlNam Ri, ChangSu Paek, JuSong Rim, (2018), Improved Hybrid Symbiotic Organism Search Task-Scheduling Algorithm for Cloud Computing, College of Information Science, Kim Il Sung University,

[29] Gosling, J., Holmes, D. C., Arnold, K. (2005). The Java programming language.

[30] Haggar, P. (2000). Practical Java: programming language guide. Addison-Wesley Professional.

[31] Merks, E., Eliersick, R., Grose, T., Budinsky, F., Steinberg, D. (2003). The eclipse modeling framework. retrieved from, total, 37.

[32] Sundas, A., Panda, S. N. (2020, March). An introduction of CloudSim simulation tool for modelling and scheduling. In 2020 international conference on emerging smart computing and informatics (ESCI) (pp. 263-268). IEEE.

[33] Oussama, Simite and Afdel, Karim, Impact Live Migration on Cloud Performance (May 28, 2018). Smart Application and Data Analysis for Smart Cities (SADASC'18), Available at SSRN: https://ssrn.com/abstract=3186348 or http://dx.doi.org/10.2139/ssrn.3186348

[34] What is CloudSim? https://www.geeksforgeeks.org/what-is-cloudsim/

[35] Mishra, Suchintan, Sahoo, Manmath (2017) , On using CloudSim as a Cloud Simulator: The Manual 10.13140/RG.2.2.30215.91041