DYNAMIC RESOURCE ALLOCATION IN CLOUD COMPUTING USING A NEW HYBRID METAHEURISTIC ALGORITHM

Egyetemi doktori (PhD) értekezés

Seyed Majid Mousavi

Témavezető: Dr. Fazekas Gábor

DEBRECENI EGYETEM

Természettudományi Doktori Tanács

Informatikai Tudományok Doktori Iskola

Debrecen, 2017

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Hungary, Debrecen, June 2017

Seyed Majid Mousavi

This is to certify that the thesis entitled "Dynamic resource allocation in Cloud Computing using a new hybrid meta-heuristic algorithm" submitted by Seyed Majid Mousavi to University of Debrecen for the award of the degree of Doctor of Philosophy is a bona fide record of the research work carried out by him under my supervision and guidance. The content of the thesis, in full or parts have not been submitted to any other Institute or University for the award of any other degree or diploma.

Hungary, Debrecen, June 2017

Dr. Fazekas Gábor

DYNAMIC RESOURCE ALLOCATION IN CLOUD COMPUTING USING A NEW HYBRID METAHEURISTIC ALGORITHM

Értekezés a doktori (Ph.D.) fokozat megszerzése érdekében
a informatika tudományágban

Írta: Seyed Majid Mousavi okleveles informatikus

Készült a Debreceni Egyetem Informatikai Tudományok doktori iskolája
(Diszkrét matematika, adatfeldolgozás és vizualizáció programja) keretében

Témavezető: Dr. Fazekas Gábor

A doktori szigorlati bizottság:

elnök: Prof. Halász Gábor

tagok: Dr. Porkoláb Zoltán

Dr. Aszalós László

A doktori szigorlat időpontja: 2016. June 17.

Az értekezés bírálói:

Dr. ....

Dr. ....

Dr. ....

A bírálóbizottság:

elnök: Dr. ....

tagok: Dr. ....

Dr. ....

Dr. ....

Dr. ....

Az értekezés védésének időpontja: **2017**

Acknowledgment

Firstly, I would like to express my sincere gratitude to my advisor Dr. Fazekas Gábor for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. Besides my advisor, I would like to thank the rest of my thesis committee: Prof.Terdik György, Prof. Csink László, and Dr. Szathmáry László, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

Last but not the least, I would like to thank my family: my parents and to my brother and sisters for supporting me spiritually throughout writing this thesis and my life in general.

Abstract

Cloud computing is an emerging technology and new trend for computing based on the internet. In cloud computing, the dynamic resource allocation is an important process used for the purpose of effective distribution of loads among virtual machines. The shared use of resources by the consumers without any strategy brings a range of issues and challenges in the cloud environment such as scalability, fault tolerance, reliability, availability, and energy efficiency. Utilizing dynamic resource allocation for load balancing is considered as an important optimization process of task scheduling in cloud computing. Load balancing strives to balance the workload across virtual machines to achieve optimal machine utilization. An inefficient resource allocation strategy and load balancer may overload some virtual machines while other virtual machines are idle. Therefore, In order to achieve maximum resource efficiency and scalability, exploring efficient methods and techniques, as well as the development of novel algorithms, are highly desired. Meta-heuristic optimization techniques have had an exceptional growth over the last two decades. The remarkable ability of meta-heuristic techniques is motivated scientists from different fields to solve different problems. Furthermore, such techniques can often find optimal solutions with less computational effort than optimization algorithms, iterative methods, or simple heuristics. Accordingly, this thesis proposes a new meta-heuristic load balancing algorithm with a combination of two relatively new optimization algorithms. This algorithm can well contribute in maximizing the throughput in cloud computing using well-balanced load across virtual machines. moreover, the algorithm is able to overcome the problem of entrapping into local optimum. To evaluate the performance of the proposed algorithm, the algorithm is benchmarked on eleven test functions and a comparative study is conducted to verify the results with other existing algorithms. Also, a simulation experiment is conducted to evaluate the effectiveness of the algorithm in resource allocation problem. In the simulation, we have used uniformly and normal distribution workloads in two homogeneous and heterogeneous cloud environments. The results show the proposed algorithm is effective and outperforms than other existing algorithms. Also, our proposed algorithm illustrates that there is a significant improvement in cost of energy consumption, load balancing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Introduction

Cloud computing is an emerging technology and new trend for computing for providing IT infrastructure based on virtualization of resources and services on a pay-per-use basis [1-3]. In cloud environment the physical machines run multiple virtual machines (VM) which are presented to the clients as the computing resources. The architecture of a virtual machine is based on a physical computer with similar functionality [4]. In fact virtual machine is a guest program with software resources functioning similar to a physical computer. The popularity and economical aspect of this technology have been caused many organizations and companies adopt cloud services and virtualized infrastructures [5-6]. Daily increase of cloud adoption and migrating to virtualized technology has caused cloud providers establish of large-scale of infrastructures for cloud services.

Resource allocation technique is an important process to assign cloud resources based on users' application demands to achieve an optimal number of in-use servers in cloud environments [7]. Therefore, an efficient resource allocation technique can be increasingly important for cloud environments. Figure 1.1 shows resource allocation in cloud computing.

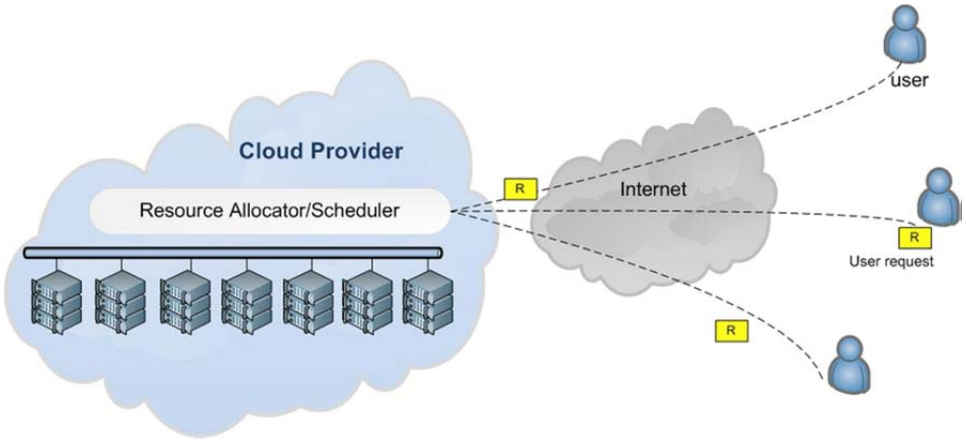


Figure 1-1: Resource allocation in Cloud Computing [8]

Resource allocation process operates dynamically for the purpose of load balancing of non-preemptive tasks. This technique strives to balance the workload across virtual machines, which aims to minimize response time in order to keep promises and quality of service in accordance with Service Level Agreements (SLA) between the clients and the provider [9].

Furthermore, this process has to be carried out regularly due to the time-variant nature of the loads of Application Environments (AE) [10].

In fact cloud clients are interested in having the shortest possible time to complete their jobs and requests at the minimum cost [11]. On the other hand, the cloud providers are interested in maximizing the utilization of their resources in order to lower overall cost to increase their profit. Obviously, these two objectives are in conflict and often none of the parties are not satisfied with the traditional methods of resource allocation and load balancing techniques [9,12]. Classical methods are very time consuming for achieving fully optimized solution and in some cases are impossible [13]. Also, traditional approximate methods are reported inconclusive and inaccurate and often trapped in local optimum [14].

In cloud computing, there are two technical restrictions. First, the capacity of the machines is physically limited; secondly, priorities for performing the jobs and requests should be in direction with maximizing the efficiency of resources [15-17]. Therefore, using of resource allocation and load balancing techniques are led to the reduction of the number of in-used servers which has the direct effect on resources efficiency and overall throughput of cloud computing. Virtual machines in distributed systems have different usage conditions including; utilization costs and also different processing power [18]. The users' jobs may also have the different amount of information. In addition, to allocate appropriate resources on any machine to the jobs, the response time[1] is also considered. The most important problem in this process is the ordering process and how placement the tasks on resources are conducted. In fact, by increasing the productivity of resources, the response time can be reduced and simultaneously, can improve the total cost of resource utilization and load balancing [11,19].

Therefore, In order to achieve maximum resource efficiency and scalability, exploring new methods and techniques as well as development of novel algorithms are highly desired.

[1] Response time is the time interval taken between submission of the user's request and the first response that is produced

The remarkable ability of meta-heuristic techniques is motivated scientists from different fields to solve variety of the problems. The meta-heuristic optimization techniques have had an exceptional growth over the last two decades [20]. Therefore, such techniques can often find optimal solutions with less computational effort than optimization algorithms, iterative methods, or simple heuristics [21-22]. The question that arises is "why meta-heuristic techniques are remarkably common?". The answer will be easily found in four main properties that characterize most meta-heuristics: simplicity, flexibility, derivation-free mechanism, and avoidance of entrapment in local optima [23].

The aim of this study is to provide a new meta-heuristic approximation algorithm for resource allocation in order to establish load balancing based on time and cost. A balance should be established between the three target assessment variables for evaluating the proposed approximation algorithm. These three target variables include:

- The time of completing the latest task among virtual machines.
- The average of the cost paid for use of resources by the user.
- The efficiency which is caused by the impact of load balancing based on completion time and cost [24].

Regarding to the foregoing of the distributed system, the question is raised that; "How shall we allocate resources to different users' jobs and applications to achieve maximum efficiency in cloud system?". Accordingly, in this study, a new hybrid meta-heuristic algorithm with combination of two relatively new optimization algorithms is proposed which can well contribute in maximizing the throughput using well balanced load across virtual resources.

1.2. Problem Statement and Motivation

With increasing use of distributed systems, and providing different on-demand Internet-based services, as well as the importance of task completion time, resource allocation problem in the purpose of load balancing is raised as one of the important topics in cloud systems. Load balancing problem solving is of great importance to increase productivity

through the allocation of resources. This has prompted researchers to use all their knowledge and create the ways to minimize costs and reduce Makespan. In distributed computing, the cost must be paid for use of any resource, so minimizing the time of the use of resources will be necessary[25]. On the other hand, resource allocation and load balancing are two important challenges in cloud systems [26]. The resource allocation aims to achieve well balanced load across virtual resources to increase resource efficiency in order to minimize the time for all tasks. To achieve this goal, load balancing must be considered fairly on resources. Distributed computing uses a variety of computational resources to facilitate doing tasks, so choosing the appropriate techniques to perform the tasks can increase the efficiency of large-scale cloud computing environments. Moreover, the load balancing is required in order to achieve green computing in clouds [27].

So we can say that time, cost and load balancing are three important parameters for solving the problem of resource allocation and scheduling requests from the user. These three parameters are raised at two different levels of scheduling, namely the user-level and system-level, which are usually at odds with each other. To solve the problem of load balancing, resource allocation can be done so that a balanced workload is considered for different resources. If the above steps are met, system performance will be increased and finally, the tined needed for implementation of tasks will be reduced.

Unfortunately, the dynamic nature of cloud resources as well as the various demands of users has led to the complexity of the resources allocation problem. In the cloud computing, dynamic flexibility in resources allocation is offered by virtual machines. In some circumstances, the situation may arise that the two applications at the same time try to access a shared resource. Resource allocation technique should be in such a way that manages optimization in order to avoid resource competition, piece-part resource, as well as resource allocation request more than the tasks need. The importance and the need for discussion on this research is related to resource allocation algorithms in cloud environment according to the environment assumptions.

1.3. Novelty and Contributions

The goal of this study is to establish a suitable mapping between jobs and resources in order to reach efficient load balancing in the cloud system, so as to satisfy the needs of both user and system levels simultaneously. This study contributes to propose a novel meta-heuristic algorithm for resource allocation in the purpose of load balancing. This combinatorial algorithm is proposed using newest efficient optimization methods. This thesis compares the performance of the proposed algorithm with the respect of the existing other solutions. The major contributions of this thesis are as follows:

- A novel algorithm for resource allocation in the purpose of load balancing of non-preemptive independent jobs in cloud computing in order to maximizing the throughput using well balanced load across virtual machines.
- Overcoming the problem of entrapped into local optimum during resource allocation process.
- One of the important features of the proposed algorithm is that the algorithm doesn't require any special controller, and only needs normal optimization parameters, such as population size and number of iterations which are involved in its implementation. Therefore, the algorithm has the least dependency on the parameters.
- In virtual systems, resources are dynamically changed. Thus, their behavior has different performance in time. The meta-heuristic algorithms are able to provide better decisions in solutions space. However, load balancing on virtual systems is influenced by various factors such as time and cost.
- A systematic literature survey about various resource allocation and load balancing methods, techniques and algorithms in cloud environment. Also, a comprehensive review is conducted regarding pros/cons and the merits/demerits of these methods.

- Performance analysis and evaluation of the proposed algorithm is presented with respect to other existing algorithms.

1.4. Research Hypotheses

It is hypothesized that the combination of two efficient optimization methods to cover each other's weaknesses can improve approximate solutions for resource allocation and load balancing in virtualized systems. It is hypothesized that each virtual machine based on its processor capacity can do one or more tasks because each virtual machine can have one or more processor core. The research divides the main hypothesis into small hypotheses. Accordingly, the following hypotheses are proposed:

- Resources and virtual machine are considered same.
- Tasks are independent.
- Distributed environment is heterogeneous and dynamic.
- All tasks must be done.
- Each task is performed only by a virtual machine.
- Allocation is done exactly once.
- Each task has a specific size and volume of data.
- Each virtual machine has different and specified processing speed.
- Each virtual machine has a special price that must be paid for the use of it in time.
- It is hypothesized that, In order to achieve maximum resource efficiency and scalability, exploring meta-heuristic algorithms as well as the development of new combinatorial meta-heuristic algorithms are highly desired.
- It is hypothesized that, approximate meta-heuristic algorithms find optimal solutions with less computational effort than optimization algorithms, iterative methods, or simple heuristics.
- It is hypothesized that, establishment of balance among completion time, cost of energy consumption, and increasing efficiency is done using optimization methods. This balance is calculated relative to the load balancing factor.

- It is hypothesized that, the proposed algorithm can be introduced in a particular framework to be applicable to OpenStack[1] software or in cloud computing system. The ultimate goal of this hypothesis is introducing a framework for communicating proposed method in a distributed system, so that the complexity of communications in terms of time in cloud computing systems to be justified.

- It is hypothesized that, the proposed algorithm for resource allocation can be performed in polynomial time and ensures convergence to the optimal solution. This means that it does not stick in local optimums.

1.5. Thesis Organization

This thesis is organized into six chapters. Chapter 2 provides concepts, advantages and applications of the cloud computing. Also, the different aspects of resource allocation, load balancing, and scheduling trends are explained. Chapter 3 presents background and a systematic state-of-the-arts in resource allocation, and related issues in cloud computing. Chapter 4 officially defines the problem and then presents a bin-packing based solution for resource allocation in cloud computing. We introduce a new hybrid meta-heuristic method using two of relatively new optimization methods to perform initial resource allocation and load balancing while minimizing load imbalance, and cost of energy consumption. Chapter 5 presents computational and simulation results along with a comparative study to verify the achieved results. Chapter 6 provides conclusions, summarizes the contributions of this work, perspectives, and discusses the future studies directions. At the end, used references in this thesis will be mentioned.

[1] OpenStack is an open-source software which is developed in order to create, manage and control of virtual servers and other resources in the data center.

Chapter 2

Cloud Computing and Theoretical Foundations

2.1. Introduction

The shared use of resources by the consumers without any strategy brings a range of issues and challenges in the cloud environment such as: scalability, fault tolerance, reliability, availability, and energy efficiency. These challenges appear when multiple concurrent requests to a single server led to the server malfunctioning due to overload, while other servers are idle. The main objective of the resource allocation is to reach optimal resource utilization, avoid overloading the system, maximizing throughput and minimize the response time, allocation the available resources with an effective strategy. Therefore, in such a large scale heterogeneous environment is a big challenge. Before addressing resource allocation challenge and solving the problem, a comprehensive study is needed to identify all aspect of cloud computing. This chapter provides the main concept of cloud computing and virtualization technique which serves as the soul of cloud computing.

2.2. Cloud Computing

Cloud computing is an emerging internet-based practice to provides computing as a utility service. This technology is a popular model for providing Information Technology (IT) resources as a network-based service in a cost-efficient and pay-per-use method. Since cloud computing is new trends in IT outsourcing, organizations adopt and migrate to this technology for their business processes. Although there are many definitions of cloud computing in literature, none of them give a clear abstraction of this paradigm. The most comprehensive definition is proposed by the National Institute of Standards and Technology (NIST)[28]. The NIST definition is: "*this technology is a network-based model to enable convenient, on-demand network access to a shared pool of configurable computing resources such as: networks, servers, storage, applications, and services. This model can be rapidly provisioned and released with the least management effort or service provider interaction*".

The most important key features of this technology are on-demand self-service, broad network access, resource pooling, rapid elastically, and measured service [30].

- *On-demand self-service:* Cloud resources are provisioned whenever they are required.
- *Broad network access:* Resources are accessible over the different kind of networks.
- *Resource pooling:* In order to allocate resources to cloud consumers, resources are pooled and dynamically allocated to users' jobs.
- *Rapid elastically:* The capability of scalability of resources in request peak times.

Measured service: A crucial feature in which used resources and services are monitored and measured by cloud providers in order to billing, access control, resource optimization, and other tasks.

Figure 2-1 shows main component of a cloud system. Cloud computing consists of five main specifications such as: On-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [29]. Moreover, cloud computing has different service models and four main deployment models.

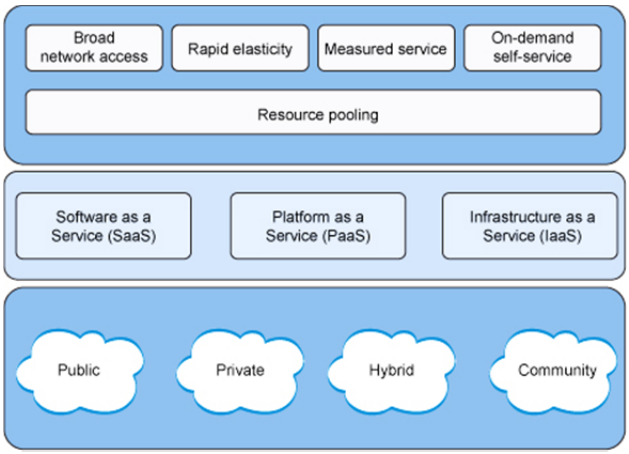


Figure 2-1: NIST visual model of Cloud Computing [29]

2.2.1. Cloud Deployment Models

A cloud deployment model is an important aspect that represents a specific instance of cloud infrastructure. There are four different cloud deployment models [31] as follows:

- *Private cloud:* This model is a particular deployment model of cloud that provides a distinct and secure cloud environment which is used by only one organization over the Internet or a private internal network.
- *Community cloud:* Community deployment model refers to a shared cloud environment among several organizations from a specific group such as banks or heads of trading firms.
- *Public cloud:* Public cloud model is the standard cloud computing model which is the most recognizable model of cloud computing that cloud provider makes virtualized resources using pooled shared physical resources.
- *Hybrid cloud:* This model of cloud is a combination of other deployment models on-premises with multiple providers. Because of complexity in the most of the enterprises, they prefer to use the hybrid cloud solution where the advantage of each model (the public, private or on-premises infrastructure) supports a single application.

2.2.2. Cloud Service Models

There are various cloud-based services. Many cloud providers deliver a various type of cloud emerged services. Three cloud service models have become widely used in small to medium businesses: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS)[32-33]. Figure 2-2 shows different cloud service models and their services content.

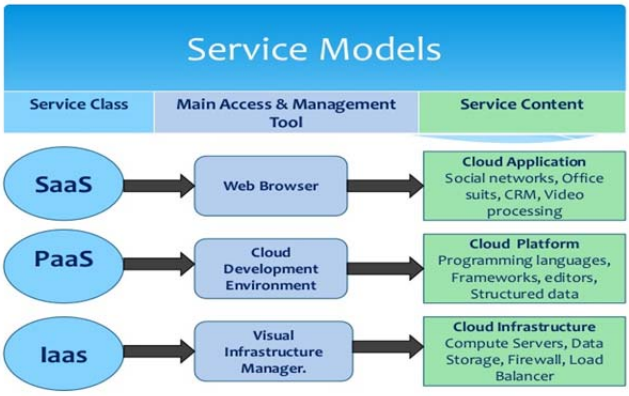


Figure 2-2: Different Cloud service models

- *Software as a Service (SaaS).* This service is a model for the distribution of applications over the Internet which is accessible for customers through a web browser or a program interface. Some of the well-known cloud software services are Google Apps, Salesforce, Office 365, Netflix, and etc.

- *Platform as a Service (PaaS)*. This service refers to a set of cloud computing services that provide a distributed platform to allow developers (Software developers, web developers and businesses) to build applications and services over the internet. Some of these services are AWS[1], Windows Azure, Google App Engine, and etc.

- *Infrastructure as a Service (IaaS).* This service is one of the main layers of cloud computing in order to provide virtualized computing resources over the Internet. Leading IaaS providers include AWS, Rackspace Open Cloud, and IBM SmartCloud Enterprise.

2.3. Virtualization and Cloud Computing

Virtualization is a technique which is known as the soul of the cloud computing. Virtualization technology hides the complexity of physical resources abstraction. This technique provides hardware independence, ease of duplication, flexibility, relocation of resources, isolation and creating protected environment, and green IT [34-35]. Figure 2-3 presents different type of virtualization in cloud environments.

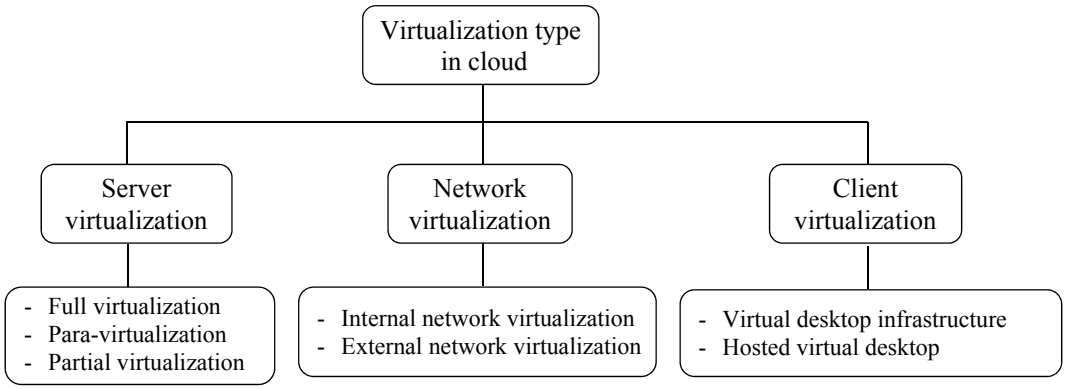


Figure 2-3: Virtualization type in Cloud environment

[1] Amazon Web Services (AWS)

Virtualization may be done in different cloud computing levels: Hardware virtualization, Network virtualization, and Client virtualization [36].

- *Server virtualization:* A physical server is partitioning into multiple smaller virtual servers based on virtualization techniques. The main advantage of this partitioning includes maximizing utilization of resources in the physical server, and cost savings. Figure 2-4 shows a physical server which is divided into multiple virtual machines (VMs). In fact the architecture of each VM is based on physical computers with similar functionality [37]. A cloud is created from numerous physical machines. Each physical machine runs multiple virtual machines which are presented to the end-users or so called clients as the computing resources. In cloud computing, a virtual machine is a guest program with software resources which works like a real physical computer.

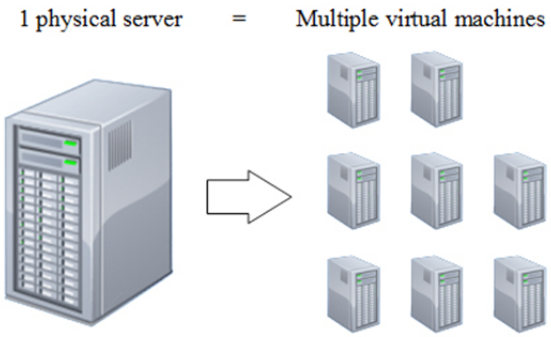


Figure 2-4: Virtualized server

Host operating system is the operating system which is installed directly on the physical server. Guest operating system is the operating system which is installed on the virtual machines in the physical server. Hypervisor or Virtual Machine Manager (VMM) is a software in order to host several different virtual machines on a single server [38]. In order to implementing a virtual machine we can use two different hypervisor as follow:

- *Hypervisor type-1:* This type-1 of hypervisor runs directly on hardware as operating system as shown in figure (left). It operates as a hardware virtualization engine. This

type of hypervisor has better performance and greater flexibility in comparison with another existing type.

- *Hypervisor type-2:* This type supports guest virtual machines on a host operating system as figure shows (right). IBM strongly recommends that this type of hypervisors be used mainly on client systems where efficiency is less critical.

Figure 2-5 depicts different levels of host/guest operation systems and hypervisors. Although both types of hypervisors are usable in different environments, reaching maximum efficiency in each type of hypervisor is dependent on the environment which has to be operated.

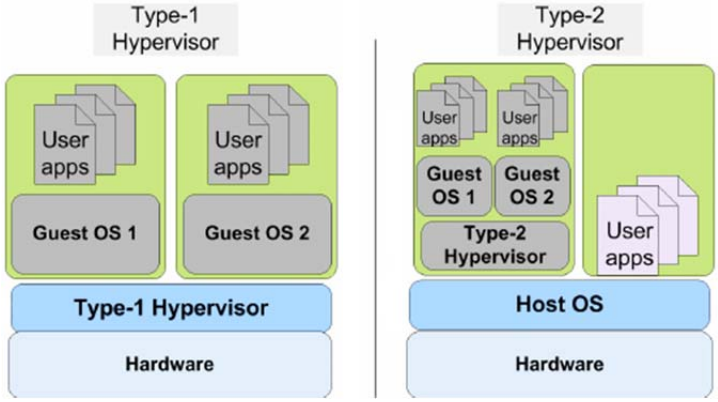


Figure 2-5: Different types of hypervisors

2.4. Resource Allocation

Resource allocation technique is an important process to allocate resources based on user's application demands to achieve an optimal number of servers in use. Nowadays, cloud environments are mainly heterogeneous; they have physical and virtualized servers from multiple generations and multiple vendors; which means that cloud consumers are geographically dispersed and utilize a diverse range of resources. Cloud computing provides a heterogeneous collection of parallel and distributed computing to deliver on-demand access to shared pool of resources [39]. These resources may include a computer, group of computers, network links, central processing units or disk drives [40-42]. The shared use of resources by the consumers without any strategy brings a range of issues and challenge in cloud environment such as: scalability, fault tolerance, reliability, availability and energy

consumption. These challenges appear when multiple concurrent requests to a single server lead to the server malfunctioning due to overload, while other servers are idle [43]. Thus, handling and delivering appropriate resource allocation is a major challenge where users' jobs are fluctuating frequently. Since the main objective of resource allocation is to reach optimal resource utilization, avoid overloading the system, maximizing throughput in such a large scale heterogeneous environment is a big challenge. Figure 2-6 depicts resource allocator place in cloud system.

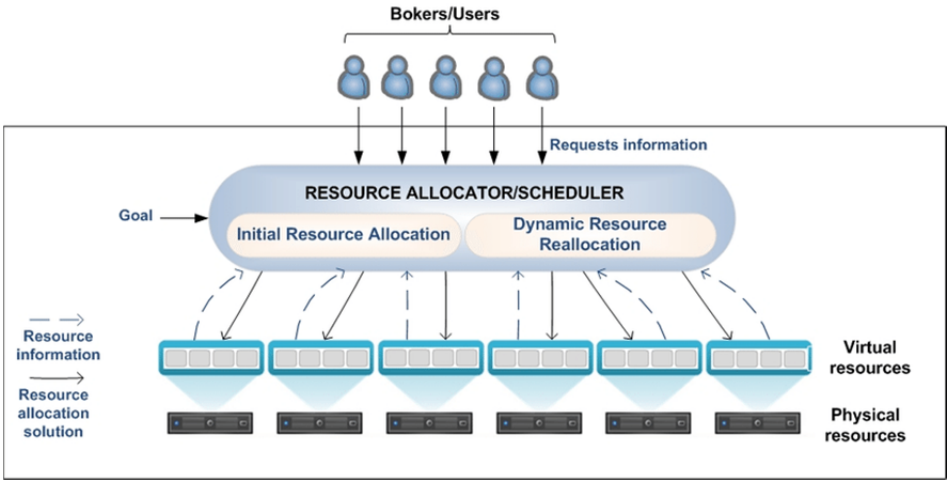


Figure 2-6: Resource allocator level in Cloud system [8]

An inefficient resource allocation decreases the quality of services in the cloud system if the allocation process not to be accurately managed. In cloud computing, there are two technical restrictions. First, the capacity of the machines is physically limited; secondly, priorities for the implementation of the jobs should be in direction with maximizing the efficiency of resources [44-45]. Therefore, an efficient resource allocator provides an efficient process by allowing the service provider to allocate resources based on a managed process for each individual module.

In the cloud computing, the resource can be considered as software, platform, and infrastructure. These three resources are known respectively as SaaS, PaaS, and IaaS. In cloud environments, resource allocation is done at two levels as follows:

- In the first level, when cloud consumer is uploaded to cloud. The load balancing algorithm allocates required instances to physical servers in order to balance the loads across virtual machines.

- At the second level, all of the received requests by an application should be assigned to a specific instance which is related to the application in the cloud to balance the load across instances of the same application

2.5. Goals of Resources Allocation in Cloud

The main purpose of the resources allocation in the cloud environment is to achieve customer satisfaction with minimal processing time, reducing fees for leasing resources and simultaneously ensure the quality of servicing and improve throughput for trust and satisfaction of the service provider. Special purposes of task scheduling include load balancing, the quality of service, economic principle, and system throughput, which each of them will be discussed in the system [46].

2.5.1. Load Balancing

Load balancing and tasks scheduling have a close relationship in the cloud environment. Scheduling mechanism is responsible for optimal matching of task and resources in time and cost. Load balancing in distributed environment is expressed at two levels, the first level is the load on the virtual machine and the second level is the resource layers [47].

2.5.2. Quality of Service

In fact, the goal of cloud system is to provide storage and distributed services. Resources are performed according to the demands and in the form of service quality of a cloud provider. When resource management is done for the allocation of the task, we should first ensure the quality of service [48].

2.5.3. Economic Principles

Cloud resources are widely distributed around the world. These resources may belong to a certain organization, which conducts its own management policy. This distributed business model provides services to suit different needs, which are related to the demands, therefore

paying for demand is reasonable [49]. It can be said that advance market leads to promoting job scheduling and resource management. Therefore we should make sure that both sides are satisfied and obtain their profits.

2.5.4. Best Time to Rrun

Mainly, for applications, the tasks can be divided into different categories according to user requirements, so the best time to run is set based on various targets for each task. This indirectly improves the quality of scheduling service in the distributed environment.

2.5.5. Good Throughput

Throughput in cloud environment is considered as measure to assess the optimal performance of the tasks. In addition, it is considered as a goal that must be considered in the development of the economic model. Increased throughput is useful for both the user and the service provider. Figure 2-7 shows classification of some target functions in distributed systems.

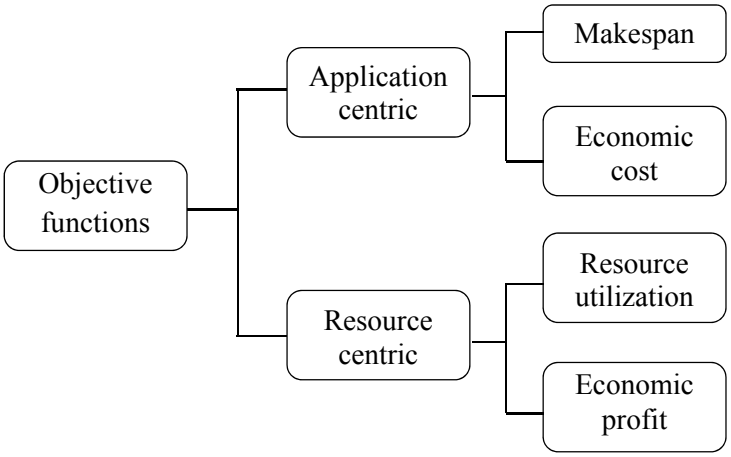


Figure 2-7: Some target functions in distributed systems

2.6. Resource Allocation Strategy (RAS)

Resource allocation strategy is a collection of activities for optimal utilization of resources within the limit of cloud computing system in order to meet the cloud application requirements [50]. In RAS process, the type and amount of required resources by each

application must be considered. In order to achieve an efficient RAS avoiding following criteria is very vital:

- *Resource contention:* The collision of two applications in order to access a resource at the same time.
- *Scarcity of resources:* Lack or limitation of enough resources for applications.
- *Resource fragmentation:* Lack of integration between required resources or isolation of resources in the system.
- *Over-provisioning:* Extra resource allocation for an application while other applications need the resource.
- *Under-provisioning:* Allocation of the resources to an application fewer than demand.

In order to avoid above problems in resource allocation, RAS should consider the important parameters (as input parameters) from both cloud providers and cloud consumers [50,51]. Some of these parameters are shown in table 2-1.

Table 2-1: Important parameters for RAS [50]

Parameter	Provider	Customer
Provider Offerings	√	-
Resource Status	√	-
Available Resources	√	-
Application Requirements	-	√
Agreed Contract Between Customer and provider	√	√

From cloud consumer's viewpoint, service level agreement and application requirement should be considered as major parameters to RAS. However, on the other hand, some parameters in table 2-1 such as provider offering, the status of resources and available resources are vital to be considered. But, from the cloud provider's point of view, predicting the dynamic need of users for each individual request, and also dynamic nature of application demands are impractical [50].

Virtual machines in distributed systems have different usage conditions including; cost of utilizing them and also different processing power. The jobs required by users may also have a different amount of information. The most important problem in this process is the order process and how the placement of jobs on resources is conducted. In fact by increasing the productivity of resources, the response time can be reduced and simultaneously, can improve the total cost for resource utilization and load balancing.

2.6.1. Resource Provisioning Process

Resource provisioning process provides the appropriate resource in order to allocate to the users' jobs. This process itself consists of three main processes: discovery of resources, allocation process, and monitoring process [50-52]. Figure 2-8 shows process of resource provisioning in different provider layers in the cloud environment.

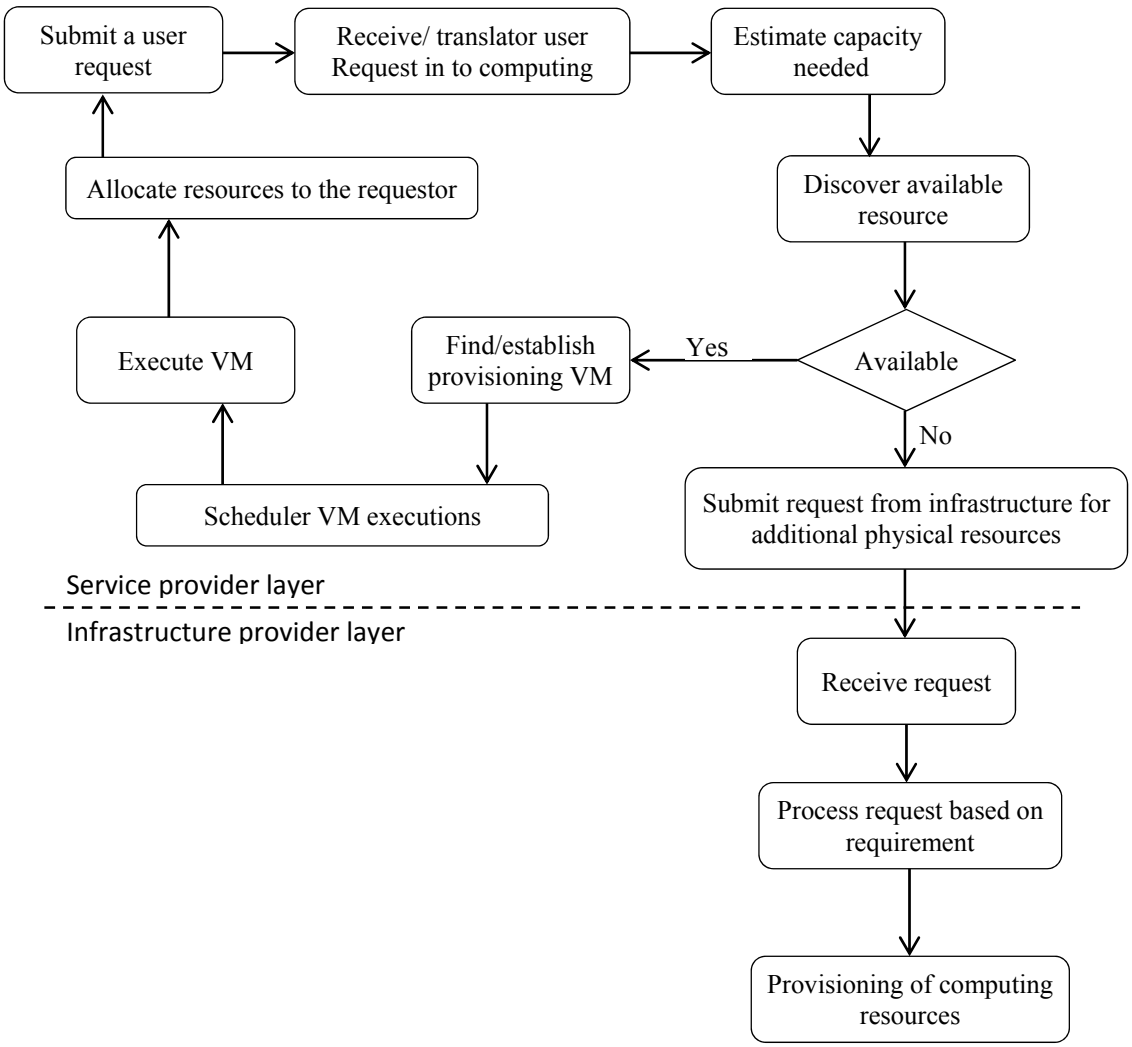


Figure 2-8: Resource provision process

As the figure illustrates discovery process and allocation process are accomplished at the service provider layer while Infrastructure provider performs the processing requirements of the physical resources. As the figure shows cloud consumers do not have any direct interaction with infrastructure provider. Resources are provisioned through service provider from the resource pool and delivered to consumers [52].

2.6.2. Resource Monitoring Process

The resource monitoring process is a methodology which can be performed by internal auditors. The main purpose of resource monitoring is to ensure whether resource provisioning and allocations comply with standards and service level agreements. Cloud resource monitoring is a key tool to identify risks as an assessment function to improve throughput and efficiency [53-54].

The main motivation for cloud resource monitoring is to provide an assurance engagement between the cloud provider and consumers to raise cloud consumer's confidence concerning the measurement of cloud services against criteria.

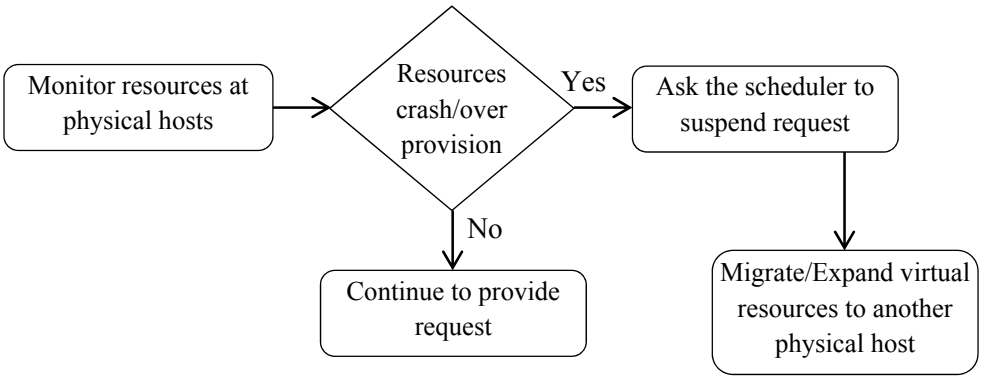


Figure 2-9: Resource monitoring process

Figure 2-9 depicts the monitoring process provided by both providers. The figure shows both providers have a direct role in order to monitor available resources to ensure the proper functionality and related processes. In the case of any crash or overloaded, the providers can change the process. But as it is shown in figure 2-10, only service provider plays a major role in the monitoring process. This process is done in order to improve resource utilization process. In such a circumstances service provider performs a load balancing process.

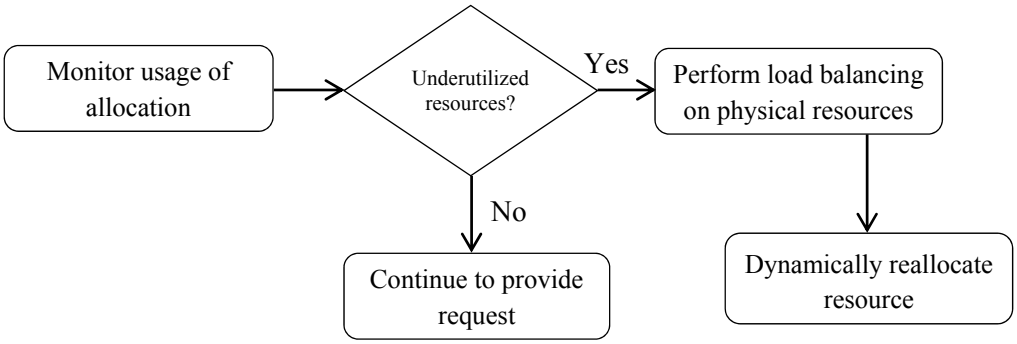


Figure 2-10: Monitor resources and load balancing

2.7. Load Balancing

Load balancing is a technique to distribute workloads equally and dynamically to all virtual machines in cloud computing [55]. Resource allocation is done dynamically for the purpose of load balancing of non-preemptive tasks. Load balancing is an *NP*-hard optimization problem in cloud computing [56]. This technique strives to balance the workload across VMs, which aims to minimize response time in order to keep promises and quality of service in accordance with service level agreements between the clients and the provider. Furthermore, as we mentioned earlier in previous chapter this process has to be carried out regularly due to the time-variant nature of the loads of application environments[10]. Regardless of advantages load balancing saves energy consumption which helps in a clean and green environment. The main advantages of load balancing in cloud computing are as follows [57]:

- No machine is overloaded.
- Maximizing throughput and overall performance of cloud.
- Optimal resource utilization.
- Avoiding bottlenecks.
- Reducing response time and completion time.
- Provision for a backup plan.
- Reaching the dream of green computing.
- Decreasing Energy Consumption and Carbon emission.

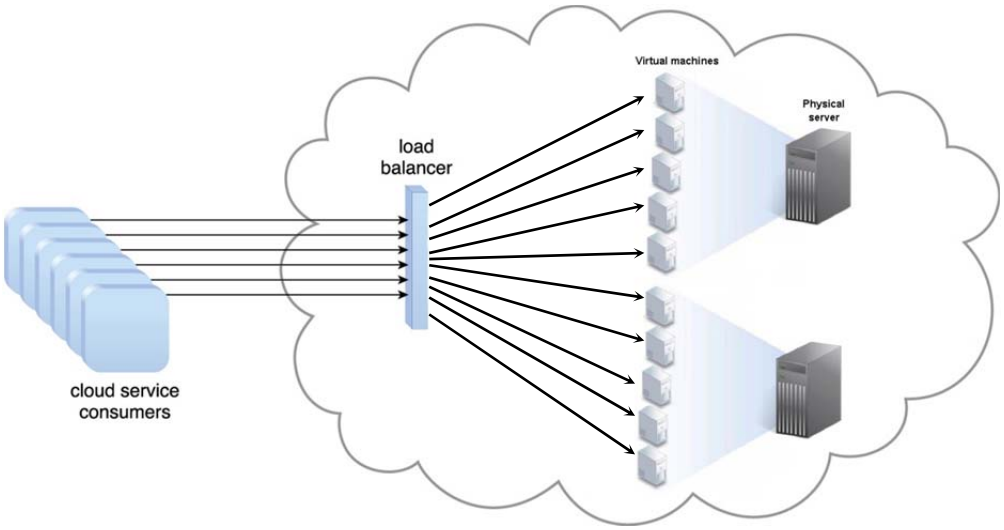


Figure 2-11: Load balancing in Cloud Computing

Figure 2-11 shows load balancer place in cloud environment. As the figure illustrates, load balancer distribute loads among virtual machines. This distribution is based on a specific algorithm or technique. Different types of load balancing techniques are as follows:

- *Static Load Balancing:* The implementation of this type of load balancing is easy but inefficient for the cloud environment, especially in high workloads [58]. This method needs prior information of resources such as capacity, availability, processing power, etc.

- *Dynamic Load Balancing:* The implementation of the algorithm is difficult but compatible with cloud environment [58-59]. This method is suitable for systems with heterogeneous resources. The advantages of these methods are as follows:
  - No need prior information.
  - Changeable at runtime by user.
  - Appropriate for heterogeneous resources.

- *Centralized Load Balancing:* In this method, there is a server to allocate resources and monitoring all other nodes. This server which is called coordinator, stores all information about allocations, requests, and requirements. This method is used in private or small cloud computing environments.

- *Distributed Load Balancing:* there are several coordinators in each domain. Each coordinator stores all information about its own domain such as allocations, requests, and requirements. This method is compatible for all distributed networks.

- *Hierarchical Load Balancing:* This algorithm divided cloud network to different levels for load balancing. This technique uses a hierarchical process based on a tree structure. In this process, all parents have the information of all own children and decision is made based on a hierarchical structure for each level.

2.8. Scheduling in Cloud Computing

Resource scheduling problem in cloud computing is also known as the resources allocation in cloud environment. Scheduling is a very important issue in the field of cloud computing. Figure 2-12 shows a view of a scheduling system in the cloud environment.

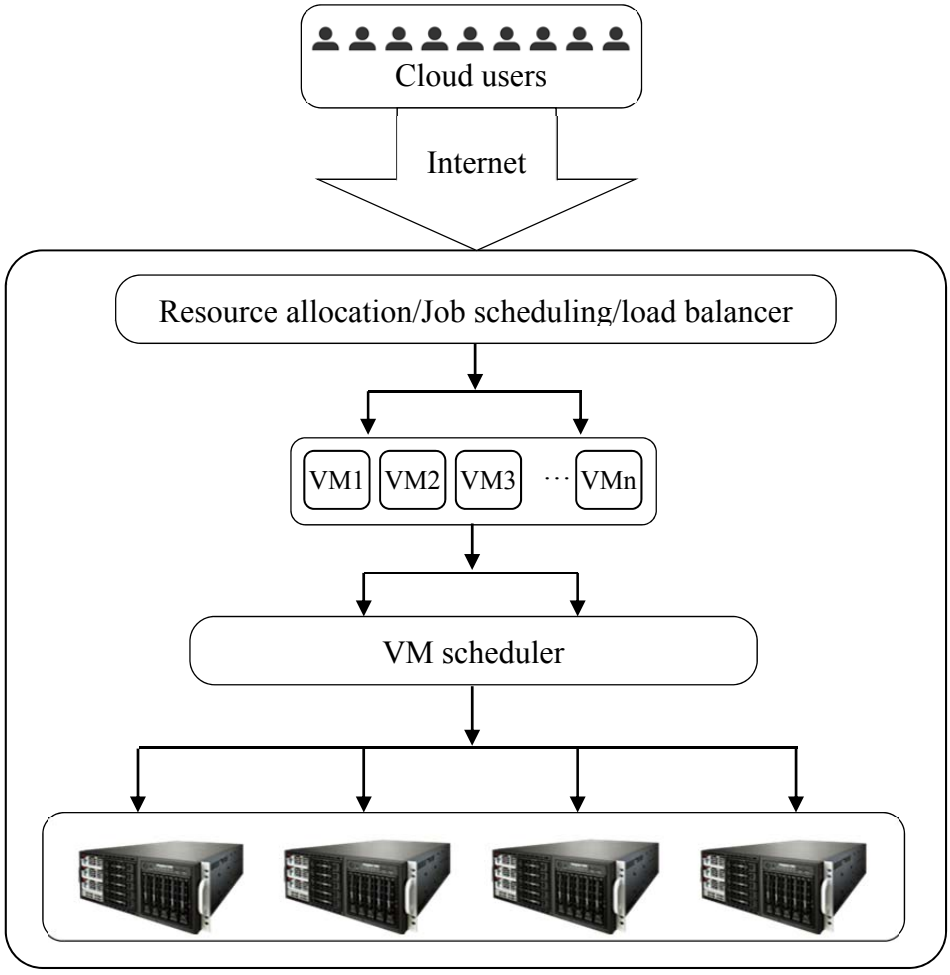


Figure 2-12: High-level view of task scheduling in Cloud Computing

In cloud environment, each user may face with hundreds of virtual resources to perform any task. In this case, the allocation of tasks to virtual resources by the user is impossible. Scheduling system controls different tasks in distributed systems to reduce response time and increases productivity of resources which causes to increase computing power [60].

The purpose of scheduling is distribution of resources and productivity of shared resources. In the sense of scheduling, we will be faced with challenges, including cost, time, security, reliability, inefficiency, and lack of control, which have been proposed in recent years to improve these factors. In this section, we first review the issues addressed in terms of scheduling, then the tasks done in this area will be discussed.

As figure 2-12 shows, it is obvious that, after sending the user's request via the Internet on the distribution system, and then categorizing them, scheduling is raised and used one of the evolutionary algorithms. It also will assign a set of tasks to available virtual machines.

2.9. Scheduling Process in Cloud

With the increasing popularity of distributed computing systems, scheduling theory draws more attention. Scheduling is mapping the tasks to resources based on characteristics of requests and tasks. In general, the scheduling process includes: Resource Discovering, Resource Filtering, Resource Selection and Task Submission [61-62].

- *Resource Discovering:* In this section, the data center server discovers the available resources.
- *Resource Filtering:* After the discovery of resources, information on their status is collected.
- *Resource Selection:* Source selection is done based on specified parameters of task and resource. This stage is decision-making.
- *Task Submission:* Task will be submitted to the selected resource in the previous step.

The scheduler carries out all scheduling activities and implemented to keep all resources busy. This program has a direct effect on maximizing throughput, minimizing response time and latency. Figure 2-13 illustrates the data flow process in the cloud environment.

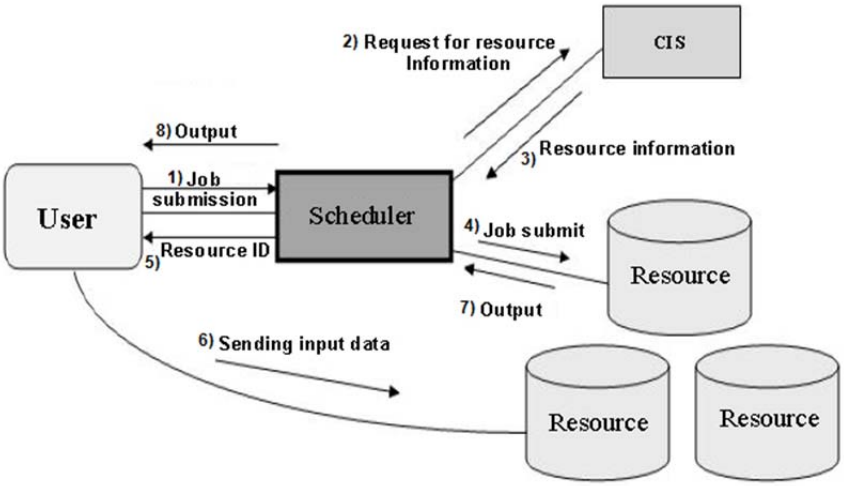


Figure 2-13: The process of scheduling in the Cloud [63]

1) Job submission: The requested job is submitted by the user.

2) Request for resource information: Cloud Information System (CIS) consists of all information about cloud resources. Needed information is requested for a job.

3) Resource information: Received information is sent to scheduler to make scheduling and appropriate scheduling process.

4) Job submission: job is submitted and needed resources are allocated to the job by resource allocation process.

5) Resource ID: Resources Id are sent. This resources id is used by user cloud interface to control the process of data flow between user and cloud.

6) Sending input data: Input data are sent to resources by the user based on specified scheduling.

7) Output to scheduler: scheduler receives real-time information from resource controller in order to the administration of scheduling program.

8) Output to user: scheduler sends appropriate specified information to user cloud interface.

Different classifications are listed for scheduling algorithms in distributed systems. In the following, a general example of the categories is shown in figure 2-14.

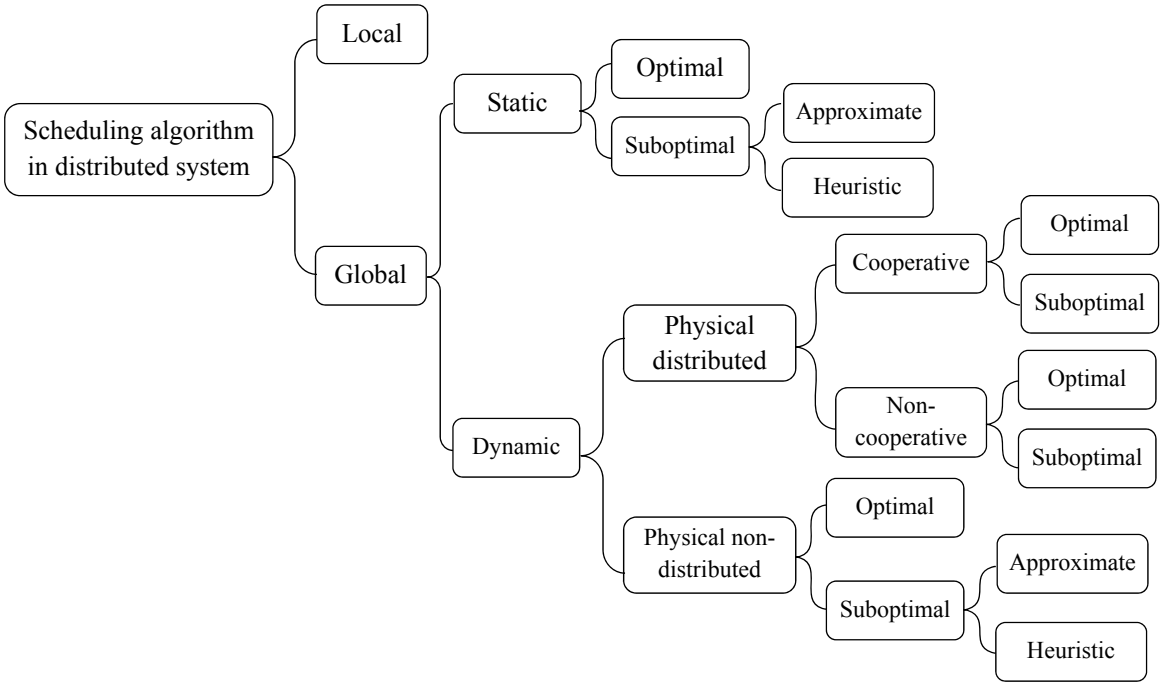


Figure 2-14: Classification of scheduling algorithms in distributed environment

Chapter 3

Background & Related Work

3.1. Introduction

The main aim of this literature survey is to identify different related issues in resource allocation, load balancing, and scheduling such as algorithms, methods, and techniques in cloud computing as well as identifying areas for future research. Although in this research, many solutions were identified, it has become apparent that much of the research being done only relates to the theoretical side of this issue. Thus this review shows that, however many of solutions and techniques have been identified, the future research should focus more on the practical implications.

3.1.1. Literature Search Method

The first step is implemented to identify relevant literature. This step was conducted with the help of provided online databases from different universities, publishers, and search engines. The focus was to find the state of the arts regarding resource allocation and load balancing issues in cloud environments. It is important to mention that review of all the found results in search engines and different databases is impossible from a practical standpoint. It is also important to note that based on the search way, search engines operate and order the results depend on the computer used for search and many other search settings and factors. Table 3-1 shows search keywords, online research databases and search engines which are used in this literature study.

Table 3-1: Online research resources and keywords

Search criterions	Name
Publishers	Taylor & Francis – Wiley- Springer – Elsevier
Online databases	Scopus – ScienceDirect – Web of Science – IEEEXplore – ACM Digital Library – DBLP – MathSciNet – arXiv
Search engines	Google Scholar – Academic Search – Microsoft Academic search – WorldWide Sience – CiteSeerX
Keywords relevance to : Resource allocation; Resource Scheduling; Resource Management Load balancing	

Figure 3-1 shows the inclusion and exclusion criterions in order to select related studies for this literature review.

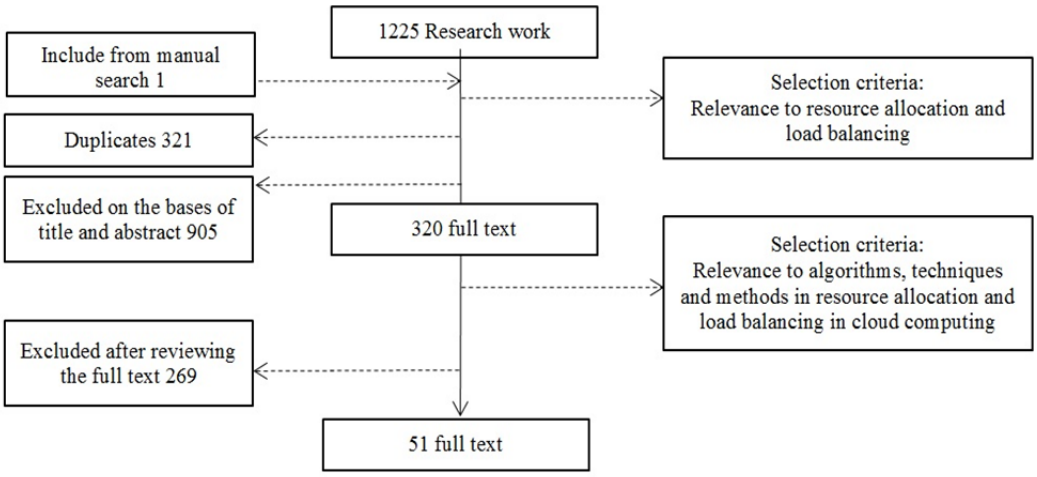


Figure 3-1: The process of selected articles

3.2. Related Works

Selecting and developing an appropriate algorithm to solve multi-objective problems is of utmost importance. In order to explore efficient solutions and addressing the related issues from different aspects and to handle the constraints of the resource allocation at different levels, existing state of the art needs to be studied and discussed. In this section, we provide a detailed overview of resource allocation and load balancing techniques, methods and algorithms at different dimensions and levels. Existing solutions are divided into three categories as follows:

3.2.1. Single Objective Algorithms

Wu et al. [64] developed a method for tasks scheduling in line with load balancing using RDPSO[1] algorithms [65] in distributed environment. In this algorithm, Candidate solutions are provided in task-resource solution pairs. Each particle not only learns about other particles, which moves in direction, but also shows the other pairs at the right direction. The goal of this method is to reduce the time of execution of the task.

[1] Revised Discrete Particle Swarm Optimization

Izakian et al.[66] developed a version of Particle Swarm Optimization(PSO)[67-69] algorithm for scheduling tasks as discrete in order to balance the load on the grid. In this model, each particle shows that each task is allocated to one resource available. In addition, solutions have been put in a $m \times n$ matrix, where *m* represents the number of machines and *n* represents the number of tasks. Each element of the matrix represents a particle, which has a value of 0 or 1, and in each column, only one element can have a value of 1. Columns represent tasks allocated, and rows represent tasks assigned to a resource. In the proposed method, the velocity of the particle, the best particle's personal position and the best global position of each of them is shown as a $m \times n$ matrix. The best personal position is a position that a particle has ever had, and the best global position is a position that all particles have ever had.

Banerjee et al.[70] introduced a method of resource allocation by using Ant Colony Optimizer(ACO)[71]. The purpose of this method is to maximize scheduling throughput to handle all users' requests due to heterogeneous resources in distributed environment, as well as minimizing the completion time of the last task by using load balancing. In this model, the distance between machines is shown as *(r, s)*, which has specified length or cost. The cost is shown as $\delta(r, s)$, as well as a pheromone concentration level that is shown as $\tau(r, s)$. Pheromone updating law is calculated in accordance with evaporation factor and the cost imposed $\Delta\tau k\ (r, s)$ on the ant *k* when is in the direction of route *(r, s)*. The concentration of pheromone is updated on all routes between machines, when the task is processed on a machine. This dynamic exploratory model is made in two ways, batch and online. When the online method is used, any request is assigned immediately after entering machine, and in batch mode, first of all requests are collected, then scheduling with respect to the approximate specified running time (time of event) maps the tasks on the resources.

Gao et al.[72] proposed a method that includes a mechanism to maintain load balancing using ant colony algorithm and the theory of complex networks. This type of scheduling has been redesigned only for Open Cloud Computing Federation (OCCF). OCCF is includes several cloud providers, which calls for the creation of a single resource interface of users. A complex network is a graph with certain characteristics which simple networks don't have

those features and this algorithm is consist of four steps. In the first phase, the ant is frequently sent to a machine that has a little load to load balance to be maintained in OCCF, as well as updating the pheromone on each machine. In the second stage, when the ant was unable to correctly handle the workload, it will be sent by the machine, then the previous step will be repeated, except when, the ant machine will convert to a machine with maximum workload. A load balance is done in the third stage, the ant moves backward along the route and the pheromone is updated on it. Each machine has a table, which contains the pheromone route and its neighbor machine, the table saves the values which represent the pheromone on that route. Updating pheromone is combined with increasing and evaporating. Finally, the complex network structure evolves to adapt changes in workload. A complex network with the specifications listed can be obtained through local behavior of ants. The profile is useful for load balancing process in proposed algorithm.

Ludwig et al.[73] developed a method to schedule tasks in grid computing. In the proposed algorithm, ants and tasks have strong relationship with each other. Whenever a task for allocation refers to the resource, an ant will arise to find the best machine to allocate that task. As soon as the task is allocated, the ant stores all information of relevant machines in the form of a sequence of pheromones in a Load information table. Load information table contains information on load in all machines. The ant meets machine and stores the load in the table to guide other ants to select the best possible route. Many stores are listed in the table. Authors added two rates to the proposed algorithm, Decay Rate (DR) and Mutation Rate (MR). These rates are used when the ant wants to go from one machine to another. There are two items to select, or go to a machine randomly due to the probability of mutation given. The second way is that, using Load information table in two machines, determines the next destination with the passage of time, due to higher decay probability, mutation probability is reduced. In this case, the ant relies on information in the table for routing, and does not do random selection.

Babu & Krishna [56] introduced a method to balance load inspired by Honey bees algorithm [74] in a distributed environment. Their aim is to provide the algorithm to achieve load balancing across virtual machines, and to minimize completion time of the last task in

the cloud infrastructure. In this method, virtual machines based on the amount of their load are placed in three groups of virtual machines with overload, virtual machines with under load and virtual machines with balance load. Each group includes a number of virtual machines, the tasks have been removed from the machines with overload and after making a decision will be placed on one of the virtual machines with under load. In this way, each task is considered as a bee, and virtual machines with under load are considered as the destination of the bees. Bees have to update the information including the amount of loads on each virtual machine, the total load on all virtual machines, the number of tasks of each virtual machine and the number of virtual machines in each group. When the process of task shifting on the machine is done, the virtual machine that is balanced will be added to load-balanced virtual machines group. The load balancing process ends, when all virtual machines are placed in this balanced group. In [75], Zhao introduced a method of assigning tasks scheduling using particle swarm optimization algorithm in order to minimize processing time of the tasks, and the cost of using resources in a distributed environment.

Karthick et al.[76] developed a method of scheduling for allocation of tasks in distributed systems using ant algorithm. The aim of offering this procedure is to create load balancing and reduce completion time of tasks scheduling. In this method, the tasks are selected to perform by artificial ants. Every ant select a task to perform acts based on pheromones table and heuristic values that have been assigned to each task in the preparation stage of algorithm. After selecting any task by the ant, it's time to select the processor for that task among the available processors. A processor is selected to run, which gives the earliest completion time and the best solutions by using the algorithm. Each ant stores in its memory information about the tasks performed, such as task completion time, number of processors that the task is executed on them, as well as the status of each processor in different moments to speed up computing the next time.

In [77], Abdullah and Osman developed a method of tasks scheduling using simulated annealing algorithm for distributed system, which has less execution time than the genetic algorithm. In this method, there are n and m machines, which the tasks should be allocated to

existing machines. Generally, the aim of algorithm is to reduce the execution time considering the deadline specified, so that the load balancing to be maintained.

Xu et al. [78], in order to find suitable solution for mapping a set of requests to the available resources in the system, introduced a method according to the distributed system conditions. In this way, the initial population is done purposefully, which it speeds up the process of finding the solution. The purpose of this algorithm is to reduce the implementation time and create load balancing.

3.2.2. Multi Objective Algorithms

The multi-objective algorithm is an area of multiple criteria decision making. The multi-objective optimizations involve several conflicting objectives, which are concerned with mathematical optimization problems to be optimized simultaneously [79]. This sort of optimization methods has been used in many fields of science, including engineering, economics, and logistics where optimal decisions need [79].

3.2.2.1. Non-dominated Sorting Genetic algorithm-based Algorithm (NSGA)

Xue et al.[80] presented a Non-dominated Sorting Genetic Algorithm-based(NSGA)[81] multi-objective method for resource allocation in distributed environment scheduling. Their aim was to minimize the time and cost in load balancing using resources and achieve Pareto optimal front, so that crowding distance is met. For this purpose, they used Self-adaptive Crowding Distance (SCD), in addition, in the proposed method, a mutation operator was added to the traditional algorithm NSGA-II to avoid premature convergence.

In this method, the tasks are shown by a Directed Acyclic Graph (DAG), the equation of $E = \{<T>, <E>, <V>\}$ is considered for a graph, where $<T>$ is the set of tasks and T_i represents a task, the amount of T_i represents the amount calculated for that task, $<E>$ is a set of edges, each edge as E_{ij} has a relation with two tasks *i* and *j*, and represents that the task *j* can't be done until the task *i* is finished.

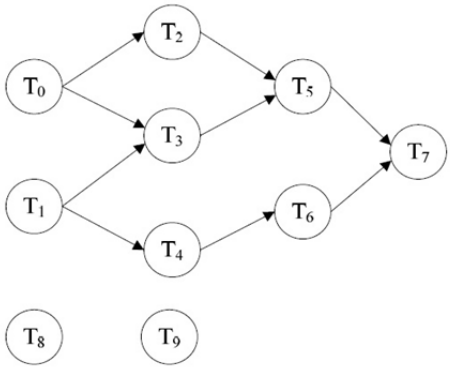


Figure 3-2: An example of a workflow [80]

$<V>$ is a set of virtual machines in a cluster, V_i means i^{th} machine of this set, the cost V_i represents the cost of using the i^{th} machine per unit of time. The ability of V_i is used to demonstrate the processing power for the machine *i*. Figure 3-2 shows a workflow of T_0 to T_7, the tasks T_9 and T_8 are independent.

$$\begin{cases} 0, & Pre(T_i) = \emptyset, Col(T_i) = \emptyset \\ \max\limits_{1 \le i \le m} t_e(T_k), & Pre(T_i) \neq \emptyset, Col(T_i) = \emptyset \\ \sum_{j=0}^{n-1} t_e(T_j), & Pre(T_i) = \emptyset, Col(T_i) \neq \emptyset \\ \max\limits_{1 \le i \le m} t_e(T_k) + \sum_{j=0}^{n-1} t_e(T_j), & Pre(T_i) \neq \emptyset, Col(T_i) \neq \emptyset \end{cases} \quad (3\text{-}1)$$

The purpose of the $Pre(T_i)$ is a direct parent of node T_i, $Col\ (T_i)$ is a task has priority over the task *i*, and should be allocated to the machine before that, the start time for processing is shown as T_s, and end time of the processing is displayed by T_e. *m* represents the number of independent tasks and *n* is the number of tasks dependent on T_i. Equation (3-1) shows this process. According to equation (3-1) we realize that as long as $Pre(T_i)$ and $Col\ (T_i)$ are empty, T_i is not be processed, the processing value for T_i consists of three parts:

- The first section is the time spent by T_i when it is allocated to a specified machine.
- Waiting time spent by T_i, when is waiting for the completion of task processing of T_i.

- Waiting time spent by T_i, when is waiting for completion of the task T_i. Now, we can obtain T_e using the equation (3-2).

$$T_e(T_i) = \frac{amount(T_i)}{ability(V_i)} + T_s(T_i) \quad (3\text{-}2)$$

Assumptions considered by authors to create the model were as follows:

- Task can be performed on several machines.
- When several tasks are allocated to a machine, the task which is received earlier has higher priority.
- The more is the processing power of machine, the more the price will be.

Considering these assumptions and using start time for the first processing and the completion time of final processing, the total time spent using equation (3-3) is obtained. Financial cost is obtained with the use of equation (3-4).

$$t_{total} = T_e(T_{last}) - T_s(T_{first}) \quad (3\text{-}3)$$

$$c_{total} = \sum_{i=1}^{P} c_i * \left(T_e(T_i) - T_s(T_i)\right) \quad (3\text{-}4)$$

Two intended aims by Xu et al. is obtained using equation (3-5).

$$\begin{cases} min(t_{total}) = min(T_e(T_{last}) - T_s(T_{first})) \\ mmin(c_{total}) = min(\sum_{i=1}^{P} c_i * (T_e(T_i) - T_s(T_i))) \end{cases} \quad (3\text{-}5)$$

To create a scheduling mode, the matrix $a = p \times q$ is considered where p represents the working group and q is the number of virtual machines. The elements of the matrix a_{ij} have values of 0 or 1, if $a_{ij} = 1$ (i.e. T_i is assigned to V_j, otherwise $a_{ij} = 0$).

According to the assumption 2, a task can't be run on several machines and only one element in each column can have a value of 1. Based on the assumption 3, more than one element in each row can have a value of 1.

Table 3-2: An example of machine coding [80]

T_i / V_j	T_0	T_1	T_2	…	T_n
V_0	0	1	0	…	1
V_1	1	0	0	…	0
V_2	0	0	0	…	0
⋮	⋮	⋮	⋮		⋮
V_j	0	0	1	…	0

Table 3-2 shows an example of this matrix coding method. To overcome the problem of low convergence speed and failure to comply with congestion, authors at the intersection, mutation and fourth step of standard algorithms NSGA-II made some changes; the changes have created as follows:

To improve the performance of the intersections, a strategy is used called STOX[1] [82] that comes from SJOX[2] [82]. However this strategy trapped the algorithms to the local optimums. To resolve this problem, Xu et al. have made the changes in mutation part. This means that in the matrix position, two columns randomly are selected as the starting and ending point, and gene segments can move between the two points. Figure 3-3 shows the mutation operation.

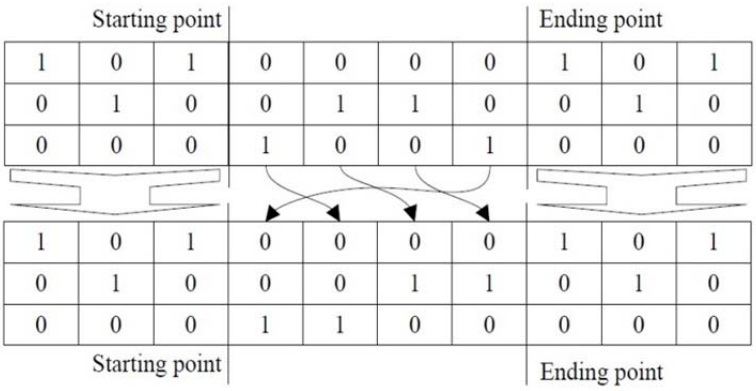


Figure 3-3: Mutation operation in the NSGA-II algorithm[80]

Standard NSGA-II algorithm has the following steps:

- Setting the population size, the maximum evolutionary generation *max(Gen)* and initializing the population *P(Gen)*.

[1] Similar Task Order Crossover
[2] Similar Job Order Crossover

- Selecting the chromosomes in *P(Gen)* to perform crossover and mutation as well as generating new population *Q(Gen)*.
- Integrating *P(Gen)* and *Q(Gen)* as *R(Gen)* and performing sorting and ranking chromosome in *R(Gen)*, chromosomal are placed in few ranks.
- Calculating the crowding distance of chromosomes in every rank and sorting them in ascending at congestion distance.
- Choosing m high chromosomes of *R(Gen)* and transferring them to *P(Gen + 1)* based on ranking and crowding distance of chromosomes *R(Gen)*.
- Set *Gen = Gen + 1* and, if *Gen <max(Gen)* go step 2, otherwise go to step 7.
- Inserting chromosomes *P(Gem)* in the Pareto front.

SCD[1] method is used to adjust the distance of condensation, which is defined as the quotient of the two operands. One of them is a maximum distance of two targets in the cost dimension, and the other is maximum distance of two targets in the time dimension. A variable is also considered as a counter that counts the time when SCDP[2] does not change during the period. The evolution of generation ends, when the count reaches a certain level. According to the authors, the algorithm might get stuck in local optimum. To fix the problem, the probability of crossover and mutation increases to 0.01 to reach a certain level. SCDP algorithm has 6 steps as follows:

- Initializing the *maxPc, maxPm* and *maxcount* variables, represent the probability of crossover, mutation and upper limit of counter. For the first generation, SCDP amount is equal to 1, otherwise, it is equal to the previous generation value.
- Calculating new SCDP in the current generation, that if it is not equivalent to the previous generation, variables of the stage 1 are updated otherwise one unit is added to SCDP.
- If the counter reaches the upper limit, go to the step 6, otherwise go to step 4.
- If the probability of crossover reaches to its upper limit, then go to step 5, and otherwise, add 0.01 to the probability of crossover.

[1] Self-adapting Crowding Distance
[2] Self-adapting Crowding Distance Parameter

- If the probability of mutation reaches to its upper limit, go to step 6, otherwise add 0.01 to the probability of mutation, otherwise it ends.

All steps are considered in the stage 4 of traditional NSGA-II algorithm. Evaluation results show improving performance of the algorithm compared with its old version.

Salimi et al.[83] introduced a multi-objective task scheduling using fuzzy systems and standard NSGA-II algorithms for distributed computing systems. The goal of this method was to minimize implementation of task time, and the costs paid by the user for the use of resources and increase the productivity of resources. This study was associated with the load balancing in the distributed system. They used the indirect method and fuzzy systems and ignored the implementation of the third objective function to solve this problem.

In [84], Cheng provided an optimized hierarchical resource allocation algorithm for workflows using general heuristic algorithm. In this model, the main objective of co-ordination between the tasks and duties assigned to the service, in accordance with the operational needs is to perform the tasks properly and observing the priority between them. This model accomplishes workflow tasks scheduling aimed at load balancing and divide the tasks to different levels and mapping (allocation) of each level of tasks to resources that they have the processing power. Figure 3-4 shows division of the tasks to different.

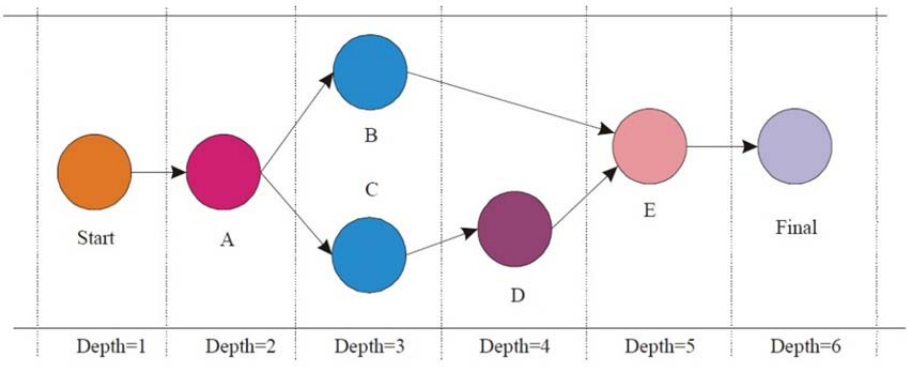


Figure 3-4: Dividing the tasks to parallel levels to properly allocate tasks to the available resources [84]

3.2.2.2. Grey Wolf Optimizer (GWO)

Grey Wolf algorithm has recently been introduced by Mir Jalali [85-86] that is based on the behavior of wolves hunting and their rule hierarchies. Hierarchical structure and social behavior of wolves is modeled during hunting process in the form of mathematical models and is used for the design of optimization algorithm. Wolves' leader is called Alpha which is responsible for hunting. The second level of wolves which helps header is called Beta. The third level of wolves is called Delta which is designed to support alpha and beta. The lowest level is called Omega.

In general, the algorithm steps can be summarized as follows:

- The fitness of all solution levels is computed and three top solutions are selected as alpha, beta, and delta until the end of the algorithm. The alpha level solution is the best fitness one. After alpha, beta and delta are the best solutions respectively. And the next better solution is delta.

- In each iteration, the three top solutions (alpha, beta, and delta wolves) have the ability to estimate the hunt, and do it in each iteration using the following equations:

$$\vec{D} = \left| \vec{C}.\vec{X_p}(t) - \vec{X}(t) \right| \qquad (3\text{-}6)$$

$$\vec{X}(t+1) = \vec{X_p}(t) - \vec{A}.\vec{D} \qquad (3\text{-}7)$$

The wolves encircle around the hunt. *Xp*, is hunting position vector. *A* and *C* are hunting vector coefficients. *X* is the wolves' position and *t* shows the stage of each iteration. *D* shows behavior of encircling around the hunt. The vectors *A* and *C* are calculated as follows:

$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a} \qquad (3\text{-}8)$$

$$\vec{C} = 2.\vec{r_2} \qquad (3\text{-}9)$$

- In each iteration, after determining of the position of alpha, beta, and delta wolves' positions, other solutions are updated in compliance with them. Hunting information

is done by alpha, beta and delta. And the rest update their *X* position with them. As figure 3-5 shows, wolves are able to change their position based on location of prey.

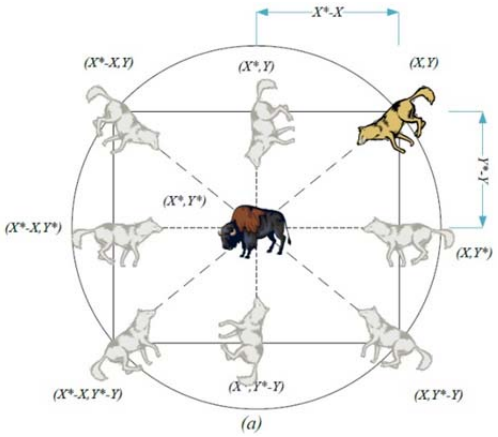


Figure 3-5: View of the gray wolves motion in haunting [85]

$$\vec{D}_{\alpha}=\left|\vec{C}_{1}.\vec{X}_{\alpha}\text{-}\vec{X}\right|,\quad \vec{D}_{\beta}=\left|\vec{C}_{2}.\vec{X}_{\beta}\text{-}\vec{X}\right|,\quad \vec{D}_{\delta}=\left|\vec{C}_{3}.\vec{X}_{\delta}\text{-}\vec{X}\right| \qquad (3\text{-}10)$$

$$\vec{X}(t+1)=\frac{\vec{X}_{1}+\vec{X}_{2}+\vec{X}_{3}}{3} \quad \textit{where:} \quad \left\{\begin{array}{l}\vec{X}_{1}=\vec{X}_{\alpha}-\vec{A}_{1}.(\vec{D}_{\alpha})\\ \vec{X}_{2}=\vec{X}_{\alpha}-\vec{A}_{2}.(\vec{D}_{\alpha})\\ \vec{X}_{3}=\vec{X}_{\alpha}-\vec{A}_{3}.(\vec{D}_{\alpha})\end{array}\right\} \qquad (3\text{-}11)$$

- In each iteration, vector and consequently vectors *b* and *c* are updated (figure 3-6).

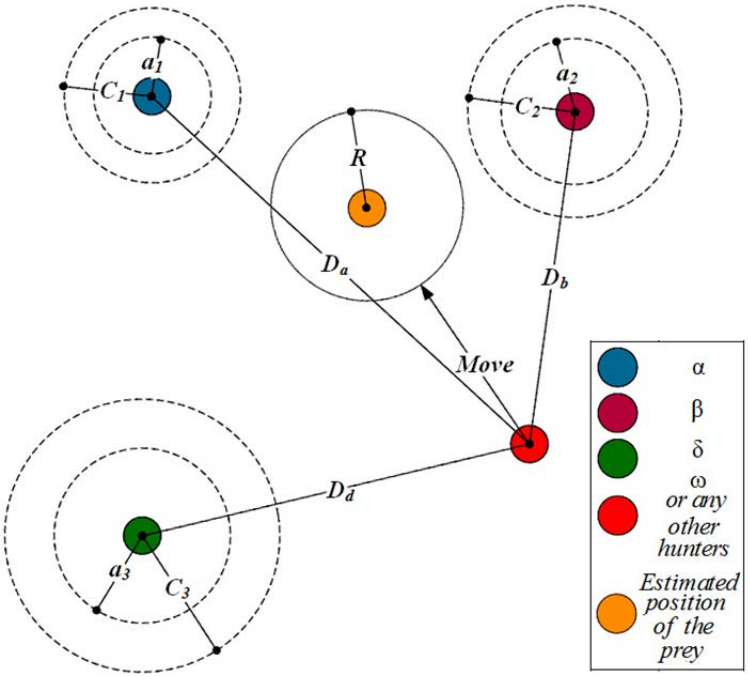


Figure 3-6: Updating wolves' position [85]

- At the end of iteration, alpha wolf position is considered as the optimal point. This value is *A*. The value of *A* is an option value which is between [-2a, 2a]. The absolute value of *A* is less than 1, so the wolves attack when they are at the *A* distance from the prey, while it is necessary to converge toward each other's at the distances of more than one.

The flowchart of gray wolves is introduced by Guha et al.[87]. In the gray wolf algorithm, parameters such as the initial population size, vector coefficients and the number of iterations and the number of wolves' level are to be determined. Then, the cost function of optimization which is minimized in this study is introduced then. Afterward, the initial population is formed randomly then, the fitness function is introduced. Then, in the loop on a regular basis, the position of the wolves' levels is determined and the fitness function is calculated, and using them, the new positions are calculated again. Iteration of this loop is specified according to the initial parameters. After repeating fitness value loop, the value of the optimal function will be shown. Figure 3-7 shows flowchart of the gray wolf algorithm.

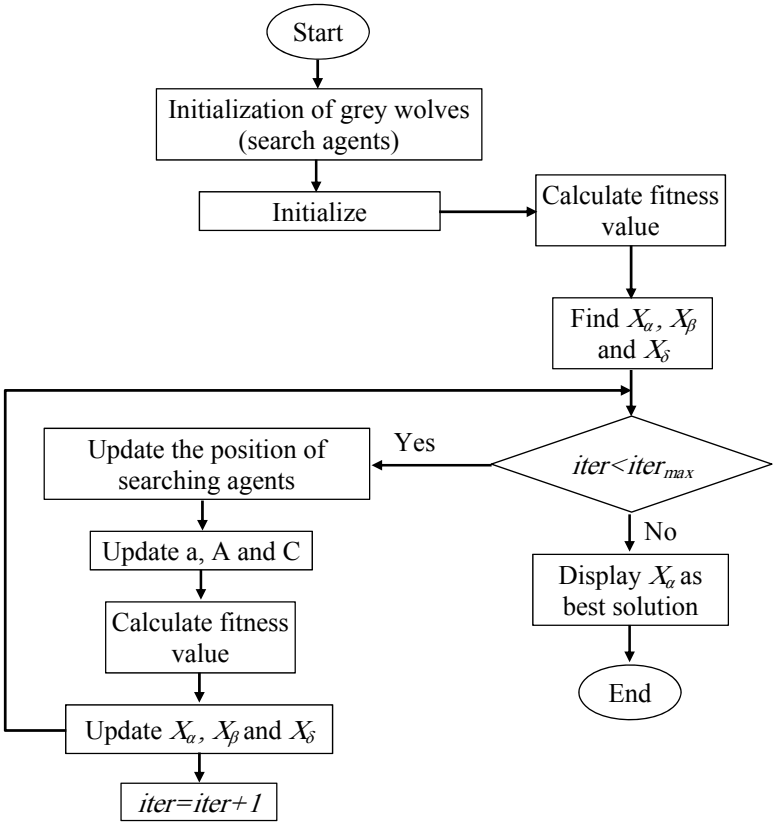


Figure 3-7: Flowchart of Gray wolf [87]

The results of the gray wolves algorithm are as follows [88-89]:

- Avoiding local optimum and the convergence of this method in problems with restrictions and no limits has been approved.
- Rotational movements of wolves can be modeled in multidimensional spaces. So many multi-objective problems with the correct values such as load balancing in distributed systems can be modeled.
- Determining and adjusting the values of A and C should be discussed to solve different optimization problems. It is very convenient for multi-objective optimization.

3.2.2.3. Teaching–Learning-Based Optimization (TLBO)

Teaching–Learning-Based Optimization algorithm is a way to explore the space of a problem to find the settings or parameters to maximize a specific purpose. The algorithm was introduced by Rao et al.[90] Similar to other evolutionary optimization techniques, Teaching–Learning-Based Optimization algorithm is an algorithm derived from nature, and works based on teacher teaching in a classroom. This algorithm is inspired from modeling the teaching and learning problem mathematically and presents a new model for solving optimization problems. Teaching-Learning-Based Optimization algorithm is based on the teaching of a teacher in the classroom. A teacher in the classroom by expressing material plays an important role in student learning and if the teaching is effective, the students will learn the material better. In addition to the teacher factor, review of lessons by students would lead to better learning. This algorithm uses a total population of solutions to achieve the overall solution. Population is considered as a group of learners or students in a class.

A teacher tries to increase the level of knowledge by teaching and learning, so the students can achieve a good score. In fact, a good teacher makes students closer to the level of his knowledge. The teacher is a person with high knowledge in the class that shares his knowledge with students in class, so that the best solution (the best student of the population class) in the same iteration acts as a teacher. It should be considered that the students acquire knowledge based on the quality of teaching by the teacher and students status (the average

class scores). In addition, students increase their knowledge through interaction between themselves. This idea is the basis of Teaching–Learning-Based Optimization algorithm for solving optimization problems. The algorithm operates in two phases, the first phase is the teacher who shares his knowledge to develop the class and the second phase is the review of courses by students in the same class. In the following, we describe the process of teaching and learning:

Training process:

The first stage of Teaching-Learning-Based Optimization algorithm is training phase. At this stage, a teacher tries to improve the scores of a classroom.

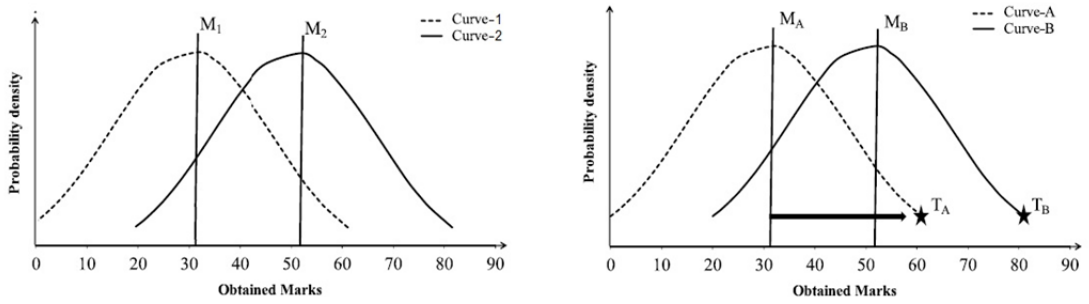


Figure 3-8: Distribution of scores by students: by two teacher (left)
by through for a group of students(right) [90]

In Figure 3-8, the Gaussian distribution function is used and the average scores obtained by students in classroom are shown as M_1 *and* M_2. In this figure, M_1 and M_2, respectively, show the average scores of two separate classrooms with the same students. As it is shown in the figure 3-8 (left), the second teacher with the average scores of M_2 has acted better than the first teacher with average score of M_1. Gaussian probability function is as follows:

$$f(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \qquad (3\text{-}12)$$

In this formula, μ is the average scores of students which are shown as M_1 and M_2. The only point of evaluating a teacher is the students' scores, and when a teacher wants to improve the status of a class, he/she should focuses on the average scores.

In figure 3-8(right), T_A is the best student in the class which is mimicked as a teacher. T_A will try to increase average M_A towards their own level according to his or her capability, thereby increasing the student's level to a new average M_B. It means that the academic level of students is approaching to their teacher, or exactly equal with him/her. This creates a new population of the classroom which has shown an average of M_B and teacher T_B. In fact, the knowledge level of students does not reach the teacher's level. It is just close to it, which is also depends on the level of classroom ability (average scores).

Learning process:

The second stage of Teaching-Learning-Based Optimization algorithm is interacting and learning process among students in a classroom. In a classroom, students can discuss and learn the issues. It is a mutual interaction phase between students, which transfers the knowledge among students. Figure 3-9 shows interactive learning among students that cause the student with low knowledge moves toward the student with more knowledge.

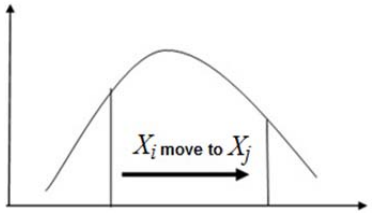


Figure 3-9: Student learning in algorithm TLBO

Based on the figure 3-10, X_j has a better score than X_i. So according to the learning phase, X_i must move toward X_j.

$$X_{i,new} = X_j = X_i + (X_j - X_i) \qquad (3\text{-}13)$$

Figure 3-10 shows an exact state of the weak student toward the strong one, which is the best motion. To reach this goal Teaching-Learning-Based Optimization algorithm should use a random parameter r to move parameter X_i to X_j.

$$X_{i,new} = X_j + r\ (X_j - X_i) \qquad X_j > X_i \qquad (3\text{-}14)$$

If, X_j wants to move to a better position on the problem space, should be located at distance of $X_j + (X_j - X_i)$ from the X_i.

$$X_{i,new} = X_j + r\ (X_i - X_j) \qquad X_j < X_i \qquad (3\text{-}15)$$

Parameter r is a random number in problem, which leads to increase power search for algorithm. According to the formula, students moving step is equal to the result of subtracting good student with a bad student. Figure 3-10 shows the flowchart of Teaching-Learning-Based Optimization algorithm.

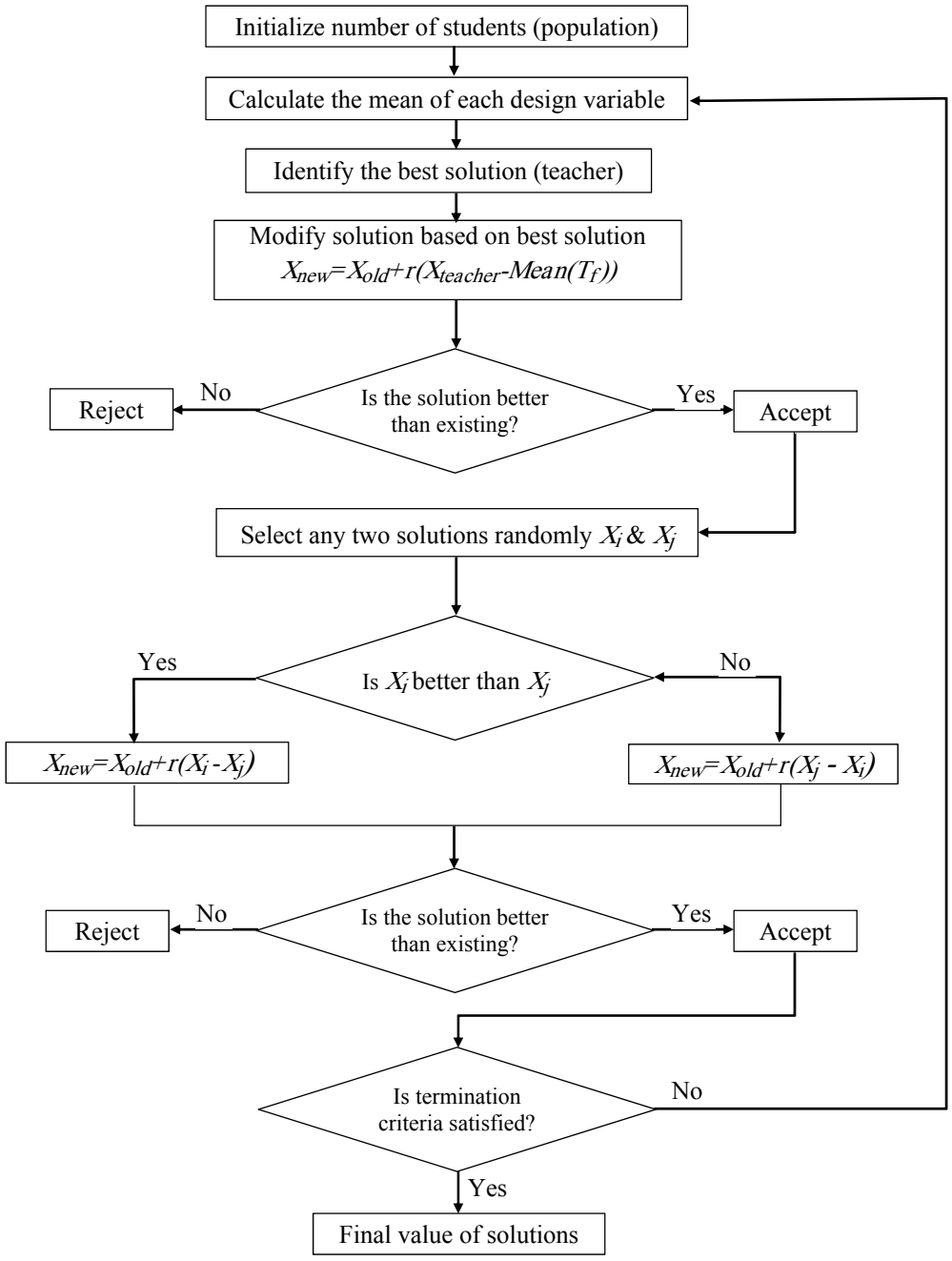


Figure 3-10: Flowchart of Teaching-Learning-Based Optimization algorithm [90]

According to the defined pseudo-code, students increase their knowledge in two ways. One method is participating in classroom and benefitting from the teacher knowledge, and the other one is reviewing of lessons between the students. It is assumed for modeling that each student exchanges his idea with another student randomly. After calculating the new member of the population, the cost function value is compared with the value of the cost function of the same member in the previous iteration. If the new value was lower, a new member will be replaced. This process is repeated. Pseudo-code of learning phase is as follows:

For $i = 1 : P_n$
  Randomly select two learners X_i *and* X_j, *where* $i \neq j$
  If $f(X_i) < f(X_j)$
    $X_{i,\,new} = X_{old} + r_i (X_i - X_j)$
  Else
    $X_{i,\,new} = X_{old} + r_i (X_j - X_i)$
  End If
End For
Accept X_{new} *if it gives a better function value.*

Steps of Teaching–Learning-Based Optimization algorithm:

- Preparation: Setting parameters values and creating the initial population.
- Calculating the average members of the population.
- Choose the best member of the population as a teacher.

Teacher is considered as T that is X_{best}. You can consider the teacher as the best member of population, which makes moving average scores towards itself $M_{new} = T$.

The new class mean must approach to the best student in the classroom. In fact, all class members should learn the new average, and make themselves close to it, and move toward it, which is shown in the following formula:

$$X_{i,new} = X_i + r\left(M_{new} - T_f \times M\right) \qquad (3\text{-}16)$$

T_f is a random number between the numbers {1, 2}, which is selected with equal probability and is multiplied in the previous average. If the training factor is equal to 1, the

new mean motion is moved normally. But if the factor is equal to two, moving step of average will be twice and the difference increases, therefore the probability of learning and improving solutions will be increased.

1. Better solutions replace the old solution (worse).
2. Learning phase:

   At this stage, for each solution X_i, we select a random solution such as X_j.

   If Xi is better:

   $$X_{i,new}=X_i+r(X_i-X_j)$$

   If X_j is better:

   $$X_{i,new}=X_i+r(X_j-X_i)$$
3. Better solutions replace the old solution (worse).
4. Termination conditions are checked, and if these conditions are not met, the implementation of the algorithm goes to the second phase, and otherwise, the loop is stopped, and the algorithm ends.

The results of using education-based learning algorithm are as follows:

1. According to learning, the basis of the training, improves the solution.
2. The specific parameter for setting is required.
3. Single-objective optimization is very convenient.

Since training is based on local operations, always convergence may not be guaranteed.

3.2.3. Hybrid Algorithms

Gomathi and Karthikey[91] introduced a method for assigning tasks in a distributed environment using Hybrid Particle Swarm Optimization(HPSO) algorithm, so they can meet the user needs and increase the amount of load balancing with productivity. The aim of the authors is to minimize the longest completion of task time among processors and create load balancing. In the proposed method, the resources are heterogeneous in distributed

environments. For this method, there are *n* independent tasks which are shown as T_i where *i={1,2,3,...,n}* and m different processors which are shown as R_j where *j={1,2,3,...,n}*.

The time of implementation of task *i* on the resource *j* is shown as *p (i, j)*, and the resource utilization for each processor is shown as $R_{i(utilisation)}$. Task allocation to resources is in form of a permutation matrix, in the permutation matrix if *x(i, j)*=1, that is, the task *i* is assigned to the resource *j*, otherwise $x_{i,j} = 0$. This method assures us that each task exactly is assigned to a processor. The method of calculating targets is in form of equations (3-17), (3-18), and (3-19). As mentioned in Chapter 1, there are two general methods for solving multi-objective scheduling problems. In fact we have two purposes, using traditional methods to a target or directly using multi-objective optimization algorithms, in case of attempting to solve the problem.

$$MS = \max_{1 \le i \le m} \sum_{j=1}^{n} P_{ij} * x_{ij} \quad (3\text{-}17)$$

$$R\bar{\imath}(utilisation) = \left(\sum_{i=1}^{m} Ri(utilisation)\right)/m \quad (3\text{-}18)$$

$$fv = \min MS / \max R\bar{\imath}(utilisation) \quad (3\text{-}19)$$

S.t.

$$\sum_{i=1}^{m} x_{ij} = 1 , \forall j \in T$$

$$x_{ij} = \{0,1\}, \forall i \in R, \forall j \in T$$

$$R_i(utilisation) = \frac{\sum_{j=1}^{n} P_{ij}}{MS}, \forall i \in R$$

In this method, each solution is shown as a particle in the population; each particle is a vector with n dimension which is defined for scheduling an independent task. Figure 3-11 shows the particles in the proposed model.

Task0	*Task1*	*Task2*	*Task3*	*Task4*
Resource2	*Resource0*	*Resource1*	*Resource0*	*Resource1*

Figure 3-11: The mapping of tasks to resources in the HPSO algorithm [91]

The position and velocity of particles in the first generation can be obtained in accordance with equations (3-20) and (3-21).

$$X_0^k = X_{min} + (X_{max} - X_{min}) * r \qquad (3\text{-}20)$$

$$V_0^k = V_{min} + (V_{max} - V_{min}) * r \qquad (3\text{-}21)$$

r is a random number between zero and one, due to the binding property of particle swarm optimization algorithm, the particles, position is continuously calculated. In this method, the continuous amount has become the discrete amount using the small position value, and the mapping is performed in accordance with equation (3-22). In this mapping, and continuous amounts $x_i^k = \{x_i^1\ x_i^2\ \dots\ x_i^n\}$ have become the discrete values $s_i^k = \{s_i^1\ s_i^2\ \dots\ s_i^n\}, now$ using equation (3-23), the allocation is done. Table 3-3 shows 6-tasks mapping between three resources.

Table 3-3: Mapping of tasks to resources [91]

Dimension	X_i^k	S_i^k	R_i^k
0	0.89	2	2
1	-0.11	1	1
2	3.15	7	1
3	-0.39	0	0
4	3.41	8	2
5	2.64	5	2

We have both social and cognitive factors, social factor represents collaboration between the particles to move toward the best goal, and the cognitive component represents the personal experience of the particle. In this hybrid algorithm, cognitive component is defined as the difference of position between two different particles which are randomly chosen and are substituted in the velocity vector; the equation (3-22) illustrates this process.

$$\delta = X^k - X^j \quad V_{k+1}^i = W_k V_k^i + \beta\delta + c_2 r_2 \left(P_k^g - X_k^i\right) \quad if\ rand\ (0,1) <=\ V_k^i \qquad (3\text{-}22)$$

In this equation, δ is the vector dimension, and β is a scaling coefficient in the range of (0, 1). T_i position which is related to the i^{th} particle is defined as equation (3-23).

As it is shown in equation (3-23) and (3-24), the particles move to a new location only when the new location has much better fitness according to [91], this prevents premature convergence of problem solution in HPSO and helps to solve the problem.

$$X_{k+1}^{i} = T_i \quad if \quad fv(T_i) < fv\left(X_k^i\right) \qquad (3\text{-}23)$$

$$T_i = X_k^i + V_{k+1}^i \qquad (3\text{-}24)$$

The algorithm which is including 100 tasks and 5 resources in 100 replications with 10 particles per generation was compared with the standard particle swarm optimizer algorithm and the comparison results demonstrate the effectiveness of this method in reducing makespan and especially increasing the efficiency of the system.

In [92] introduced a heuristic method based on particle swarm algorithm for tasks scheduling on distributed environment resources, which considers the computational cost and the cost of data transfer. This algorithm optimizes dynamic mapping tasks to resources using particle swarm optimization classical algorithm and ultimately balance the system load balancing. This optimization method is composed of two components, one of them is tasks scheduling operations and the other one is particle swarm algorithm particles steps to obtain an optimal mix of the tasks to resources mapping. In this method, each particle represents a mapping of tasks to available resources. The first step in this exploration method is to calculate the mapping of all tasks, which is possible when there will also be a dependency between tasks and this algorithm takes into account the dependencies between tasks as the allocation of ready tasks to resources based on output pairs obtained from particle swarm optimization algorithm. The purpose of the ready task is something that its implementation has ended and input data is obtained to perform child task by doing relevant calculations. Upon completion of the task, ready list will be updated. After that, the average delay time (to start the task) and bandwidth to transfer data will be updated between resources based on network utilization. In other words, since the communication cost varies over time, particle swarm algorithm mapping operations is calculated again, and this creates a scheduling heuristic method to handle runtime for tasks mapping. The scheduling process has been as online and is being repeated until all tasks are scheduled [93].

3.3. Overview and Comparative Study

In this section, using the table 3-4, we have an overview on the works done in the field of tasks scheduling which were mentioned in the previous section. This table includes the objectives of tasks scheduling, the algorithms used in these methods, the simulation environment of the algorithms and the presented year. Grey wolf algorithm and teaching-learning-based algorithm still have not been evaluated in the context of resource allocation, load balancing, and scheduling in cloud computing. Table 3-4 presents a comparative study between different resource allocation techniques based on their strengths and limitations.

Table 3-4: A summary of the works done in the field of resources allocation with scheduling and load balancing and cost

Author	*Evolutionary algorithm*	*Environment*	*Targets*	*Year*	*Simulation tool*
Xue et al [80]	multi-target genetic	Cloud	• Reduce the longest termination time among resources • Reduce the resources cost • Load balancing	2014	Matlab
Salimi et al [83]	multi-target genetic	Grid	• Reduce the longest termination time among resources • Reduce the resources cost • Load balancing	2014	GridSim
Cheng [84]	genetic	Cloud	• Reduce the longest termination time among resources • Load balancing	2012	Java environment
Gomathi & Karthikey [91]	swarm optimization	Cloud	• Reduce the longest termination time among resources • Load balancing	2013	Java environment
Pandey et al [92]	swarm optimization	Cloud	• Reduce the longest termination time among resources • Load balancing	2010	Amazon EC2
Wu et al [64]	swarm optimization	Grid	• Reduce the longest termination time among resources • Reduce the workflow time • Load balancing	2012	Ad-hoc VC++ toolkit
Izakian et al [66]	swarm optimization	Cloud	• Reduce the longest termination time among resources • Reduce the workflow time • Load balancing	2010	Java environment

Banerjee et al [70]	Ant colony	Cloud	• Reduce the longest termination time among resources • Load balancing	2009	Simulated cloud
Mousavi & Fazekas [19]	Ant colony	Cloud	• Reduce the longest termination time among resources • Load balancing	2016	CloudSim
Ludwig & Moallem [73]	Ant colony	Grid	• Reduce the longest termination time among resources • Load balancing	2011	GridSim
Babu & Krishna [56]	Bee colony	Cloud	• Reduce the longest termination time among resources • Load balancing	2013	CloudSim
Zhao [75]	swarm optimization	Cloud	• Reduce the longest termination time among resources • Reduce the resources cost	2015	CloudSim
Abdullah & Othman [77]	Simulated Annealing	Cloud	• Reduce the longest termination time among resources • Load balancing	2014	CloudSim

Clients want that their work to be completed in the shortest possible time and at minimal cost which cloud servers should receive. On the other hand, the cloud providers are interested in maximizing the use of their resources and also to increase their profits. The two are in conflict with each other.

The literature review shows that traditional methods which are used for optimization may be definitive and accurate, yet they are often trapped in local optimum. In fact due to the dynamic nature of distributed environment and heterogeneous resources, in such a system, the scheduling process must be done automatically and very quickly. That is why the scheduling process is recognized as *NP*-complete problem [26]. Traditional approaches are not dynamic and suitable to solve such scheduling problem. These approaches contain a large search space; facing a large number of possible solutions and a tedious process to find the optimal solution.

There is currently no efficient method available to solve these problems. In such circumstances, the traditional approach has been set to find a fully optimized solution instead

of finding the semi-optimal solution, but in a shorter time. In this context, IT professionals are focused on exploratory methods. Therefore, meta-heuristic algorithms which have a global overview, as they ensure convergence to solution and do not fall into the trap in local optimum, are of importance. Consequently, the GW algorithm is chosen for this purpose. In addition, the TLBO algorithm is used in a hybrid form with GWO to improve local optimization and increase accuracy.

Chapter 4

Proposed Method

4.1. Elementary algorithms for solving resource allocation

In the default for resource allocation problem, the users' jobs have to be allocated to virtual resources in the cloud servers (e.g. virtual server) and each server has a limited capacity [94]. The question is "how to allocate appropriate resources to the jobs in order to achieve well-balanced load across virtual servers with the least number of servers?".

An elementary algorithm which was raised for this problem is the First Fit (FF) algorithm[95]. This algorithm is a greedy algorithm and used as the basis for comparing the methods in the investigation. In this algorithm, the resources are allocated to the jobs, with the order of first available empty resource [95]. In this thesis, the resource allocation problem is considered as an example of Bin Packing Problem (BPP)[96] with several objectives of response time, load balancing, and cost.

4.2. Proposed method

Dynamic resource allocation and load balancing on virtualized systems like cloud computing are influenced by various factors such as time and cost. Since our aim is maximum utilization of cloud resources with respect different factors, using classical methods are inefficient. In order to optimize resource allocation and load balancing process, we need multi-objective optimization and approximation methods because these problems have to be solved in multi-dimensional spaces. Therefore, due to the multi-objective and dynamic nature of the resource allocation and load balancing problems and also difficulties in dealing with the local optimum, traditional methods need to improve and major advancement.

4.2.1. An elementary method

We have several jobs which need the amount of 0.5, 0.3, 0.4, 0.8, 0.2, 0.2 and 0.2% of cloud resources (servers' processor). These resources in the cloud have the same capacity of 1. The aim is to perform allocation with the least resources. In other words, we have the best load balancing aimed to the fewest resources, so that, time and cost is not worse from the desired limit. This optimization problem is raised aimed to use the least resources. Resource allocation with at least resources causes the maximum capacity of the servers to be used and

response time and load balancing to be improved and energy consumption is also reduced by minimizing the number of virtual servers. In an example, the applicant has raised his requests in the cloud system as follows:

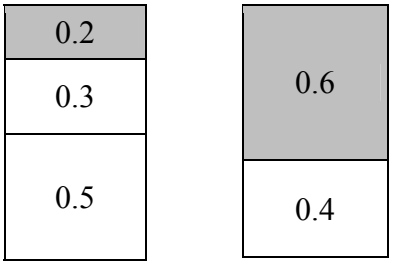
$Items \rightarrow 0.5\ ,\ 0.3\ ,\ 0.4\ ,\ 0.8\ ,\ 0.2\ ,\ 0.2\ ,\ 0.2$ $(The\ amount\ of\ requested\ resource\ for\ each\ job\ by\ the\ applicant)$

$Resources \rightarrow 1,1,1,\ldots,1$ $(Resources\ capacity)$

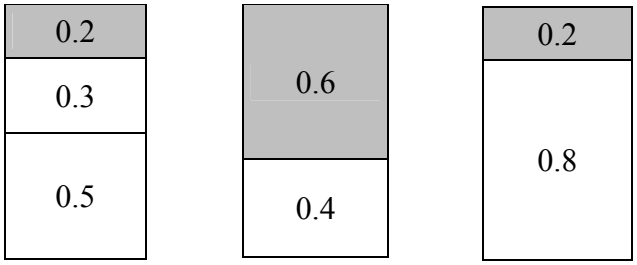
Assumes, first 0.5, then 0.3 and 0.4 are entered to the cloud environment and we have a series of resources with the capacity of 1. We tried to achieve the best load balancing for the entire system with minimal resources. We also want to find an optimal solution for resource allocation in a greedy manner. The greedy Index is that the first resource with empty space to be used and resources must not be wasted and also available resources should be used as much as possible. Allocation steps are as follows:

The first step: The order of jobs 0.5, 0.3 can be considered for the first resource, but there is no space for 0.4 because its value increases from 1 and is more than the resource capacity. The capacity of 0.2 (empty space) is left from the first resource after the allocation of jobs 0.5 and 0.3.

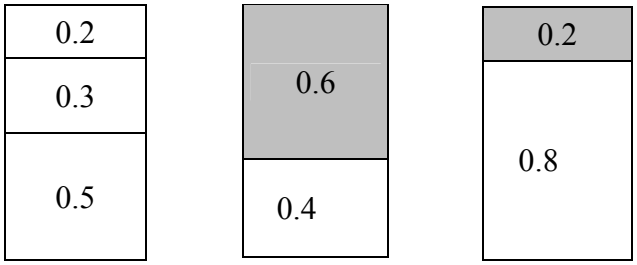
The second step: In the second phase, job 0.4 can be placed on the resource 2, and there is no space for the next resource 0.8.



The third step: we can place job 0.8 on the third resource.



The fourth step: The job with 0.2 of processing is placed on the first resource, and the first resource should be filled.



The fifth step: The jobs with 2.0 and 2.0 can be placed in the second resource and 2.0 of the second resource remains empty.

At the end of this method, three resources were allocated, and in total 0.4 = (0.2 + 0.2) of the resources is unallocated. The problem in optimal mode should use a maximum of three resources because:

The total required resources is (0.2 + 0.2 + 0.2 + 0.8 + 0.4 + 0.3 + 0.5) = 2.6, and because resources are used properly, 2.6 ≈ 3 resources are needed. The main concern issue about these solutions is that, with the increasing number of requests, the amount of empty space of resources will be increased. Consequently, the total empty spaces will be plenty, in a way that the waste resources are increased in cloud resources.

4.2.2. Combinatorial multi objective method

There are many algorithms can be used for multi-dimensional problem-solving. Today, meta-heuristic algorithms play a pivotal role to solve multi-dimensional problems. Grey wolf algorithm is a multi-dimensional meta-heuristic algorithm which is completely explored in the previous chapter. The hierarchical structure of the grey wolves is mathematically modelled and used for the design of optimization techniques. In this algorithm, the best solutions are calculated based on a multi-dimensional space and the optimal solution is selected into a solution space. The grey wolf algorithm can be used to solve the multi-dimensional problem but usually trapped in the local optimum.

We also investigated the different aspect of teaching–learning-based algorithm in the previous chapter. This meta-heuristic algorithm is an optimization method to explore the space of a problem in order to find the settings or parameters to maximize a specific purpose. In virtualized systems, the performance of virtualized resources is dynamically changed based on their workloads in time. The use of teaching-learning-base algorithm provides better decisions on future choices. Unlike the grey wolf algorithm, teaching–learning-based algorithm avoids to entrapping into the local optimum.

Our proposed method is a combination of these two multi-dimensional optimization algorithms to eliminate their weaknesses in order to make a new hybrid robust meta-heuristic algorithm. Proposed method integrates the abilities of exploitation and exploration in the grey wolf algorithm with the abilities of the convergence and avoiding local optimum in teaching–learning-based algorithm.

One of the most important features of hybrid algorithm is that doesn't require any special controller, and only normal optimization parameters, such as initial population size, the number of iterations and so on are involved in its implementation, and this has led that it has the least dependency on the parameters. Proposed algorithm can improve approximate solutions into solution space for the resources allocation and load balancing in cloud computing.

4.2.3. Mathematical model

Load balancing index is calculated as the evaluation parameters as follows:

- L is the load balancing parameter.
- 1-L is lack of load balancing parameter.

$$B = a*(1-L)+b*C+c*T \qquad (3\text{-}1)$$

The index shows the amount of load balancing between task completion time (T), and cost of energy consumption (C) to perform the task and resource productivity (B). Therefore, load balancing index in the cloud computing system is defined as above, in which their coefficients according to the cloud computing system (a, b, c) are subject to change. According to the importance of load balancing, L usually has the higher value, and coefficients b and c according to the type of system and the importance of the cost and response time will change. Finally, the aim is to minimize the index B.

When the number of requested jobs increases, the complexity rises and continues to extent that resources are wasted, while load imbalance reaches to its maximum and a lot of resources will be wasted. Consequently, completion time T and the cost of energy consumption C increases. In order to overcome these incensements, with reducing the number of resources by load balancing technique across virtual servers, energy consumption will be reduced in cloud computing. Load balancing causes the reduction in response time than existing resources as much as possible. Therefore, in principle, we are looking for load balancing by reducing the number of servers in our resource allocation.

There is a distributed network in a cloud environment with resource systems S_1, S_2,S_3,...,S_n. The resources are ready to service in the distributed network to various nodes. Different jobs are sent to the resource systems by nodes. The overall goal of the system is an agreed task scheduling for resource allocation and performing the jobs in order to achieve load balancing in the system.

Here the scheduler is responsible for allocating one or more Jobs to one or more resource system in the distributed system [97]. In other words, the agreement on job scheduling is done by the scheduler. The scheduler provides a scheduling for resource allocation [98]. Several jobs are allocated at time t in parallel and processed in distributed system. The number of variables T_k is permutation between jobs and resources. This variable is called p, and its value is calculated as follows:

$$P = n^m \qquad \textit{(n is number of tasks and m is number of resources)} \qquad (3\text{-}2)$$

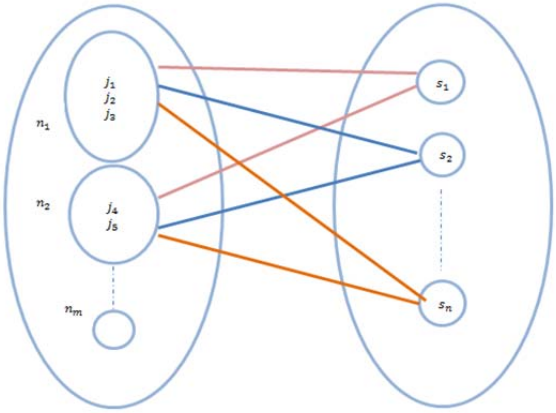


Figure 4-1: View of resource allocation in a distributed environment

As it is described in figure 4-1 each node includes several jobs. Each job requires a series of specific resources. The problem can be introduced as follows:

$Job \rightarrow j_1, j_2, \ldots, j_n$

$Resource \rightarrow R_1, R_2, \ \ldots \ , R_m$

If in particular example, the resources $R_1, R_2, \ \ldots \ , R_m$ have the same capacity and the processing power of all is the same, and $j_1, j_2, \ldots, j_n$ needs 1% of the processor processing, the professional model can be defined in such a way that, what jobs use which resources to achieve maximum load balancing, minimum response time and minimum cost. For the exact solution of the problem, all possible allocation modes must be calculated to choose the best mode. Due to a large number of the modes (exponential), the problem is an example of bin packing problem which is *NP*-complete type.

An objective function is defined for resource i and the jobs j (eq.3-3). yi is the number of resources (Bins). Therefore, the objective function and the mathematical programming model of optimization is as follows:

$$Min\ \left(B = a*(1-L_{(y_j)})+b*C_{(y_j)}+c*T_{(y_j)}\ \right) \qquad (3\text{-}3)$$

S.t.

$$\sum_{i=1}^{n} w_i x_{ij} \le K y_j, \forall j \qquad x_{ij}, y_j = 0,1\ \ \forall i, j$$

$$\sum_{j=1}^{n} x_{ij} \le b_j, \forall j \qquad x_{ij} = 0,1\ \ \forall i$$

Where.

$$x_i = \begin{cases} 1 & job\ i\ is\ used \\ 0 & job\ i\ is not\ used \end{cases} \qquad y_j = \begin{cases} 1 & resource\ j\ is used \\ 0 & resource\ j\ is not used \end{cases}$$

The aim is to find the minimum number of virtual machines y_j to minimize the objective function. The values of L and C and T (load balancing, cost and response time) are calculated based on the number of virtual resources y_j. The variable of x_{ij} demonstrates that the i^{th} job is processing in j^{th} virtual machine, and if its value is equal to 0, it means that there is no enough resource in j^{th} virtual machine and if its value is equal to 1, it means that there is enough resource to allocate the j^{th} virtual machine. Every job has the capacity of w_i. The first limitation shows that total capacity of all jobs can be placed at the maximum K available resources. The second limitation shows the maximum capacity of each virtual resource. b_j is the capacity of each virtual resource.

4.2.4. Searching process for optimal solution

Grey wolves often search based on to the position of their leaders (alpha, beta, and delta). They diverge from each other to search for prey and converge to attack prey [85]. Rotational movements of wolves can be modeled in multi-dimensional spaces. Thus, the most of multi-

objective problems such as resource allocation in distributed systems can be modeled. The pseudo code of the proposed algorithm is presented in table 4-1:

Table 4-1: Pseudo code of the proposed algorithm

```
Initialize the grey wolf population Xi=(i=1,2,...,n)
Initialize a,b and c
Calculate the fitness of each search agent
X1=the best Search gent
X2=the second best Search Agent
X3=the third best Search agent
While t<Max number of iterations)
   For each search agent
     Update the position of the current search agent by equation
     End for
Calculate the fitness of all search agents
Update X1,X2,X3
t=t+1
If not improve solution
  Begin
    sol_wolf=Solution_grey_wolf
         Initialize sol_wolf for initialize_solution for TLBO
        Sol_TLBO=Do TLBO with Initialize Population with sol_wolf
        Intialize the grey wolf population Xi= Sol_TLBO, Initialize a,b and c
        Calculate the fitness of each search agent
X1=the best Search agent, X2=the second best Search agent, X3=the third
best Search agent
  end
end while
return X1
```

To see how proposed method is theoretically able to search in solution space, the algorithm steps can be summarized as follows:

- In problem search space, the optimal solution is difficult to find. Therefore, based on mathematically simulation of hunting behavior in gray wolves, we consider three top values (Alpha, Beta, and Delta) and save them as the best solutions obtained so far. In the next step, other search agents (omegas wolves) are obliged to update their positions (value) according to the position of the best search agents.
- The main advantage of this algorithm is that if there was no improvement in gray wolf algorithm, according to the teaching-learning process, we try to find a better solution. If the problem is stuck in local optimum, teaching-learning process can introduce the new area of space based on training phase, which may improve the solution. Because of the

accuracy of gray wolf algorithm in the local behavior (defect of the grey wolf algorithm), after each iteration, the position of wolves is updated. These positions can be improved by the teaching-learning algorithm, and then gray wolf algorithm is repeated again. This process increases the accuracy of gray wolf algorithm. Figure 4-2 illustrates flow diagram of proposed algorithm.

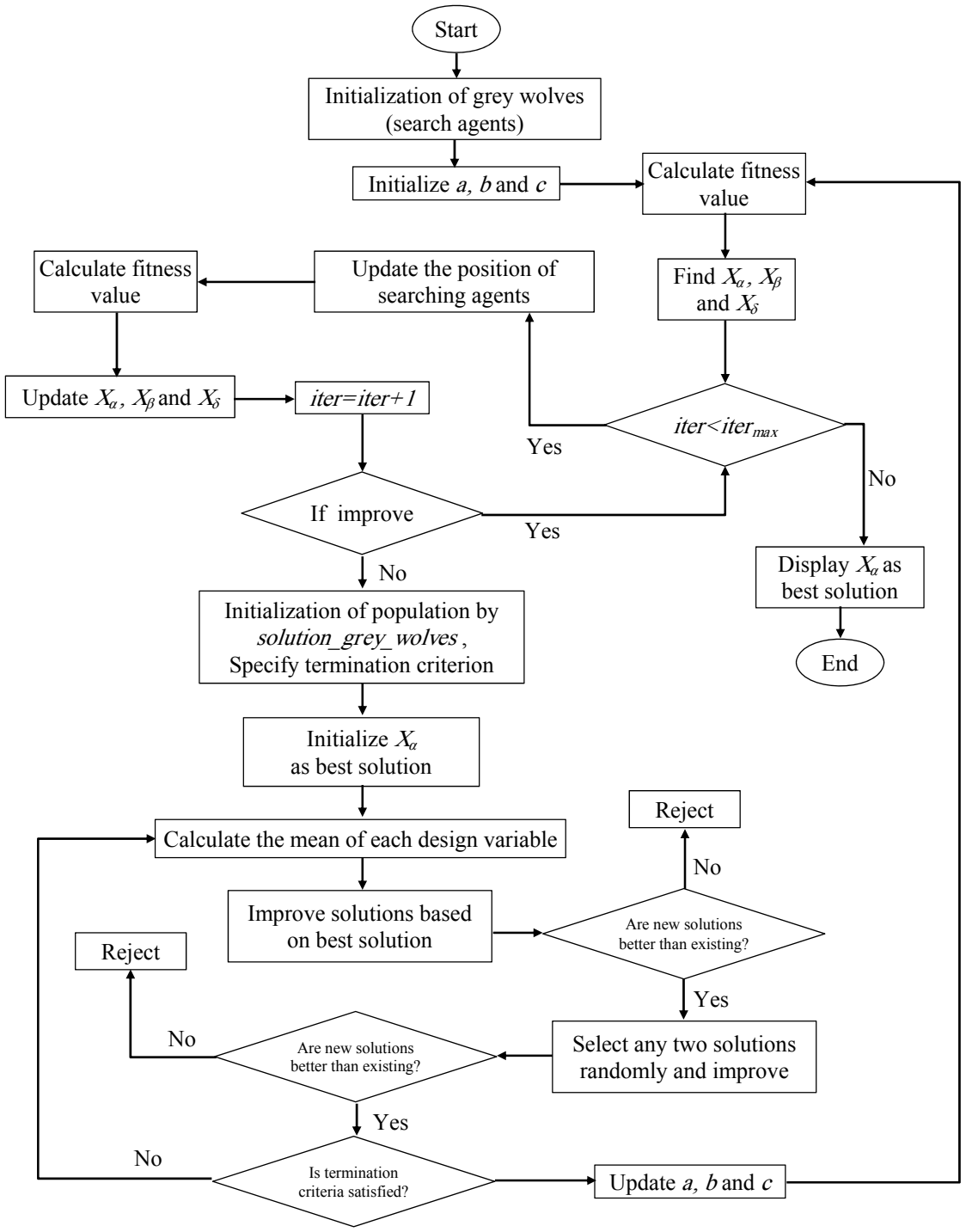


Figure 4-2: Flow diagram of proposed algorithm

In the initial state, a series of random numbers with uniform or normal distribution are considered as the initial population, and a basic solution is considered for the problem with initialization of coefficients variation *a, b, c*.

Each wolf is considered as a solution. In other words, each wolf is considered as a solution to the problem. Three best solutions (Alpha, Beta, and Delta) are selected as optimal solutions. These selections are selected on the basis of the objective function and fitness function in grey wolf optimization. Then, the program enters the main loop and update the position of other wolves (other solutions) with the capability of exploitation and exploration in grey wolf optimizer.

This means that, according to the first three solutions (α, β, and δ), the algorithm considers more value for the probability of reaching better solutions. Correspondingly, the algorithm values the wolves of beta, delta and gamma classes and a new position of wolves' community and their classification can be obtained. Now, the fitness function is calculated again for the wolves and three new solutions (wolves) are selected. If a better solution is found in the new classification than old solutions, the algorithm continues the same process to find the optimal solution. If there is no any improvement in solutions, the best solution between the wolves is considered as the initial solution (initial population) for teaching-learning-base algorithm.

Therefore, the problem continues by the teaching-learning-base algorithm and its solution is considered as the initial population to start the gray wolf algorithm again and the gray wolf algorithm starts again. It should be noted that in the gray wolf algorithm, each wolf represents a solution in the solution space, which the best solution will be chosen in any stage according to the position of other wolves.

Generally speaking, the best solution space of the gray wolf is considered as the initial solution for teaching-learning-base algorithm. After that, the output is implemented as the initial solution space for the next iteration of the gray wolf algorithm.

4.2.5. Complexity of the algorithm

Time and space are two important resources which an algorithm needs to solve a problem[99]. The time complexity of an algorithm implicates the number of steps and total time required by the algorithm during computational process. The most common metric for calculating time complexity is *Big-O* notation [100]. The most common estimation method in order to the calculation of time complexity is counting the number of primary execution functions by the algorithm. Since each algorithm is developed based on different functions and may vary with types of its own input data, therefore, worse case is usually considered for an algorithm in order to estimate of time complexity. Hence, this is the maximum time taken to solve a problem of size *n*.

4.2.6. Big-O notation of the proposed algorithm

An algorithm has a complexity *f (n) = O(g(n))* if there exist positive constants *n0* and c such that $\forall n > n0, f(n) \leq c \cdot g(n)$. In this case, the function *f (n)* is upper bounded by the function *g(n)*. The *Big-O* notation can be used to compute the time or the space complexity of an algorithm.

In order to compute complexity of the proposed algorithm we assume:

Let $S = \{s_1, s_2, s_3, ...s_m\}$ be the set of items and $S_i = \{s_1, s_2, s_3, ...s_i\}$

Let L_i be the set of sizes of all $S' \subseteq S_i$ which are not larger than t (t is capacity of each Bin)

The largest subset of S (of size at most t) is the largest number in L_m

We compute L_i from L_{i-1}: $L_i = L_{i-1} \cup (L_{i-1} + s_i)$

Where $(x + s_i) \in (L_{i+1} + s_i)$ iff $x \in L_{i-1}$ and $x + s_i \leq t$

The algorithm:

- Let $L_0 = \{0\}$ → $O(1)$ time
- For $i = 1...m$:
  - Compute $(L_{i-1} + s_i)$ from L_{i-1} → $O(|L_{i-1}|)$ time
  - Compute $L_i = L_{i-1} \cup (L_{i-1} + s_i)$ → $O(|L_i|)$ time
- Output the largest number in L_m → $O(|L_m|)$ time

Each L_i is of length $|L_i| \leq t$

The overall time complexity is therefore $O(mt)$

If $m \leq n$ the time complexity would be $O(nt)$

Theorem:

We say that an algorithm is polynomial time

If it runs in polynomial when all the numbers are integer $\leq n^c$ (c is constant)

Therefore the algorithm run time is: $O(nt) = O(n^{c+1})$

4.2.7. Worse-Case and Best-Case

Combinatorial problems are deceptive because they are defined easily but often very difficult to solve. For instance, there is no any algorithm to find the optimal solution for Travelling Salesman Problem (TSP) [101]. Similarly, there is no any algorithm to guarantee satisfactory of a given Constraint Satisfaction Problem (CSP) [102] instance in polynomial time. Accordingly, this phenomenon has been encountered on a wide variety of problems and led to the development of new theories, in particular, to the theory of *NP*-completeness. The main aim is the classification of the problems based on that how difficult they are to solve. The class of *NP*-complete problems is solved by a non-deterministic Turing machine in a polynomial time. *NP*-complete problems are considered as inherently intractable based on computational viewpoint. Therefore, it is obvious that in the worse-case, any algorithm that tries to solve an *NP*-complete problem requires polynomial time. Thus, based on defined polynomial objective function and calculated time complexity our proposed algorithm in worse-case needs polynomial run time $O(n^c)$, where n is the number of workloads and c is the number of virtual machines. In the best case the algorithm needs a linear run time $\Omega(n)$ where the number of virtual machines are equal to 1.

4.3. Running example of proposed algorithm

As explained earlier, the proposed algorithm is a kind of approximate algorithm which works with a solution space during the searching process and reaching the optimal solution. The solution space may contain a lot of solution and in each iteration, the algorithm tries to

search optimal and convergence other solution to optimal solution. Since the nature of this algorithm is population-base, therefore using this algorithm in order to see the functionality of algorithm in small scales in hard. However, we applied proposed algorithm on a uniformly distribution with 15 values between [0-1] as an example in the bellow and received surprising results. In order to explain example, we tried to present another example with the greedy method which is used in most of the resource allocation methods, in order to clarify the difference of proposed method in optimization. Figure 4-3 presents a running example of proposed algorithm.

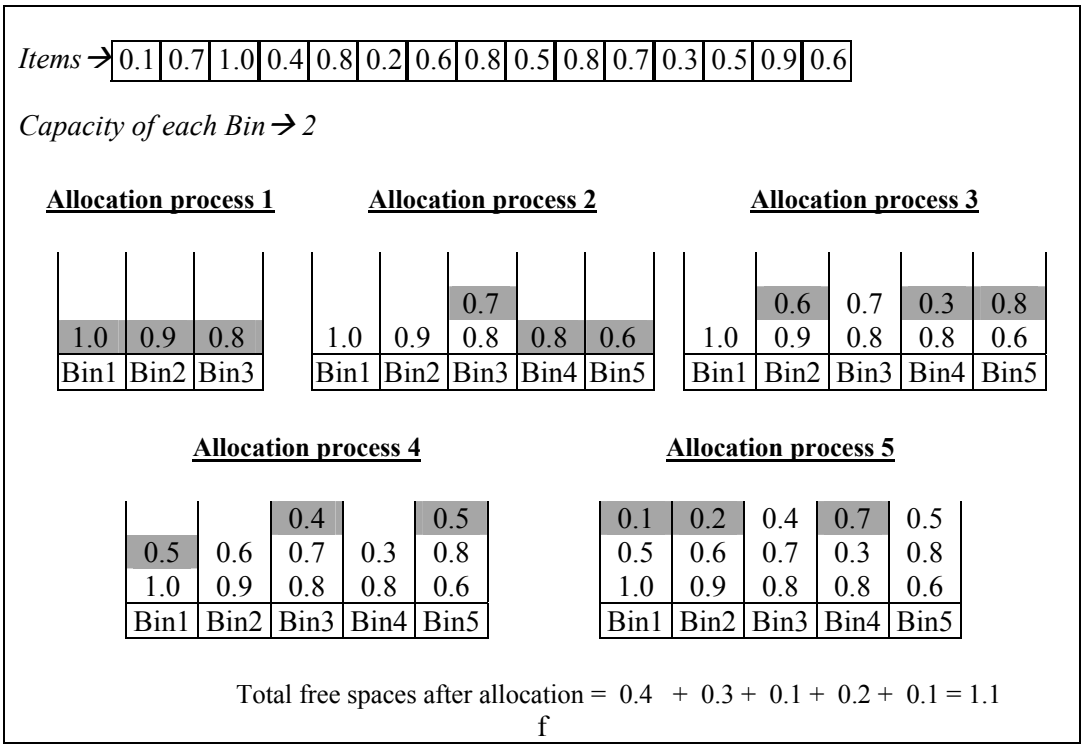


Figure 4-3: An example of the proposed method process in bin packing problem space

In this example, we have a uniform distribution as input, which is shown as the items in the example. Assume these items considered as workloads in the cloud system. Also, assume we have some virtual machines in the cloud system and capacity of each virtual machine is equal to 2 (It is noteworthy that, because we want to solve this example as a bin packing problem, therefore each bin represents a Virtual machine).

The workloads applied on algorithm and algorithm has to find the optimal resource for each workload. The algorithm aim is that allocates the least number of virtual machines to

workload. With decreasing number of machines, the cost of energy consumption also decreased. And also waste spaces in each virtual machine to be decreased, which means increasing load balancing. The process of allocation for each iteration in example is as follows:

Step 1: All possible solutions are calculated for each job.

Step 2: Three best solutions are selected as best solutions (alpha, beta, and delta).

Step 3: Other solutions are obliged to get convergence to best solutions. If in the convergence process, there was any improvement, then best three new solutions are replaced old best solutions.

Step 4: if there was not any improvement in convergence process, the best solutions are optimal solutions and solution are global optimum (this process will be done by TLBO algorithm).

Step 5: The resource is assigned.

This process will be done in each iteration, therefore we will have high accuracy to find optimal solution and avoid local optimum in our algorithm. This example has 5 allocation process and proposed algorithm allocate the least number of bins (Virtual machines) to workloads (Items).

The Bin packing problem instance is considered with bin capacity equal to 2 and 15 items. The best resource (Bins) is selected for each requested job (each item). In each allocation process, optimal resource (the optimal resource is selected based on the best solution in solution space of algorithm- which is Alpha) is selected for requested jobs. In this executed example, three allocations are considered in each allocation process.

In each allocation process, all possible solution for each item (load) is calculated. It means in each allocation process we have a solution space. Consequently, the optimal solution which is the optimal resource for a load is gained. The optimal resource is calculated for every single load after avoiding all local optimums.

As we mentioned earlier, In order to compare the processes of proposed algorithm, we represent the second example in figure 4-4. We have used the first-fit greedy algorithm in the second example and applied the same items as workloads. In a short comparison, the difference is clear between examples. The number of used bins (virtual machines) is 5 bins in proposed algorithm and 6 bins in the second example. The total amount of waste space in proposed algorithm is 1.1, while it is 3.1 in the second example. Load balancing is visible in the first running example.

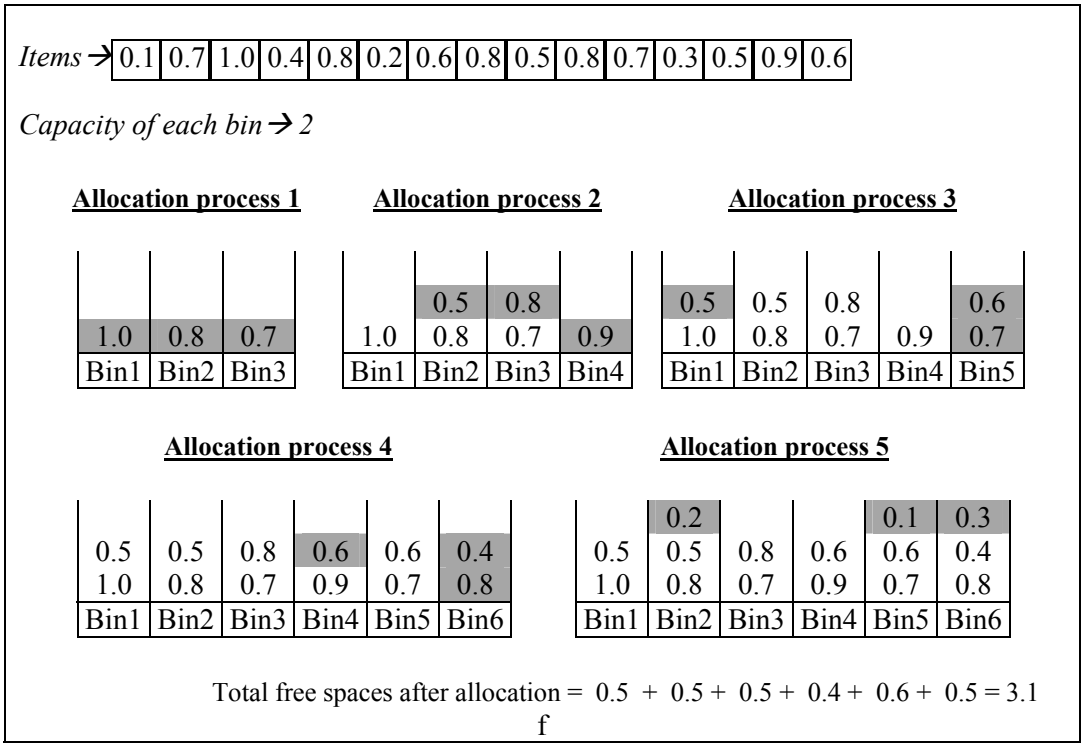


Figure 4-4: An example of first fit algorithm process in bin packing problem space

Chapter 5

Computational and Simulation Results

5.1. Implementation details

Since it is not enough just to propose a new algorithm, a number of different computational tests are performed in order to verify the functionality of algorithm to ensure that the proposed algorithm is efficient. This section contains a computational comparative study between proposed algorithm and several explained previous algorithms (Grey Wolf Optimizer, Particle Swarm Optimization and Biogeography-Based Optimization (BBO) [103]). The proposed algorithm was programmed by the author in the Matlab (V2016a) in Windows7 (64-bit, Professional edition, version 6.1- Build 7601: service pack 1) mode. The computational experiments were performed on a desktop computer with an Intel(R) core(TM) i5-2410M 2.30GHz and installed memory(RAM) 4.00 GB.

Table 5-1: Pseudo code of PSO algorithm (left) and BBO algorithm (right)

For each particle *Initialize particle* *END* *Do* *For each particle* *Calculate fitness value* *If the fitness value is better than the best fitness value (pBest) in history* *set current value as the new pBest* *End* *Choose the particle with the best fitness value of all the particles as the gBest* *For each particle* *Calculate particle velocity according equation (a)* *Update particle position according equation (b)* *End*	*Initialize a population of N candidate solutions{* x_k *}* *While not(termination criterion)* *For each , set emigration probability* μ_k *fitness of* x_k *, with* $\mu_k \in [0,1]$ *For each* x_k *, set immigration probability* $\lambda_k = 1 - \mu_k$ *For each individual* $z_k \ (k = 1,...,N)$ *For each independent variable index* $s \in [1.n]$ *Use* λ_k *to probabilistically decide whether to* *immigrate to* z_k *If immigrating then* *Use* $\{\mu_i\}$ *to probabilistically select the* *emigrating individual* x_j *;* $z_k(s) \leftarrow x_j(s)$ *End if* *Next independent variable index:* $s \leftarrow s+1$ *Probabilistically mutate* z_k *Next individual:* $k \leftarrow k+1$ *;* $\{x_k\} \leftarrow \{z_k\}$ *Next generation*

To implement other mentioned algorithms, the basic pseudo code was downloaded from their official website and then programmed in the Matlab (V2016a) by author (see Appendix B). These algorithms also performed on the same machine which proposed algorithm performed. Table 5-1 shows pseudo code of PSO and BBO algorithms. Although there was no any difficulty during implementation processes, programming of optimization algorithms need high accuracy and require taking much time.

5.2. Computational experiments

In order to compare different optimization algorithms, their behavior have to be investigated when faced with solving different mathematical functions. Therefore, to compare the proposed algorithm against other algorithms, at first, the proposed algorithm is benchmarked on 15 benchmark function [104]. Table 5-2 shows the list of these functions.

Table 5-2: Used benchmark functions

Function	Function	Domain	Optimal	mode
Sphere	$F(\mathrm{x})=\sum_{i=1}^{D} x_i^2$	*-50≤ xi ≤ 50*	$f(x*)=0$	*Unimodal*
Chung Reynolds	$F(\mathrm{x})=(\sum_{i=1}^{D} x_i^2)^2$	$-100 \le xi \le 100$	$f(x*)=0$	*Unimodal*
Schwefel 2/22	$f(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	*-10 ≤ xi ≤ 10*	$f(x*)=0$	*Unimodal*
Schwefel 2/21	$f(x)=\max_i\{\lvert x_i\rvert, 1\le i\le n\}$	*-100 ≤ xi ≤100*	$f(x*)=0$	*Unimodal*
Cube	$f(x)=100(x_2-x_1^3)^2+(1+x_1)^2$	$-10 \le xi \le 10$	$f(x*)=0$	*Unimodal*
Dixon & Price	$f(x)=(x_1-1)^2+\sum_{i=2}^{D} i(2x_i^2-x_{i-1})^2$	$-10 \le xi \le 10$	$f(x*)=0$	*Unimodal*
Griewank	$f(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod\cos(\frac{x_i}{\sqrt{i}})+1$	$-100 \le xi \le 100$	$f(x*)=0$	*Multimodal*
Rosenbrock	$f(x)=\sum_{i=1}^{D}(100(x_i^2-x_{i+1})^2+(1-x_i)^2)$	$-20 \le xi \le 20$	$f(x*)=0$	*Multimodal*
Ackley	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+\ell$	$-32 \le xi \le 32$	$f(x*)=0$	*Multimodal*
Rastrigin	$f(x)=\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$-10 \le xi \le 10$	$f(x*)=0$	*Multimodal*
Brown	$f(x)=\sum_{i=1}^{D-1}(x_i^2)^{(x_{i+1}^2+1)}+(x_{i+1}^2)^{(x_i^2+1)}$	$-4 \le xi \le 4$	$f(x*)=0$	*Multimodal*

Test functions or benchmark functions are important to validate and compare the performance of optimization algorithms [105]. Consequently, to verify achieved results, the results are compared with three above optimization algorithms.

Each algorithm has been investigated in different conditions, i.e. changing the number of iteration (200, 1000, 1500), holding the iteration size constant 30, changing the iteration size (20, 30, 50), and holding the number of iteration constant 1000. It is noteworthy that the algorithms have been run 20 times for each of the abovementioned conditions on a benchmark function, and the final result has been obtained from the average of 20 times of running so that the rate of error decreases. Benchmark functions used in table 5-2 are divided into two groups: Unimodal and Multimodal.

All of these test functions are presented here in order to examine the performance of global optimization methods. The behavior of these test functions to reach optimal result varies to cover most difficulties faced in the area of continuous global optimization. It is important to note that unimodal functions are appropriate for benchmarking exploitation and multimodal functions have many local optima with the number of increase exponentially along with different dimensions. This makes them suitable for benchmarking the exploration ability of an algorithm [85].

5.2.1. Results

Although, the results are different in number of iteration and population size, the computational results showed that concerning unimodal functions like sphere and Chang Reynolds , which are simple functions with no local optima. If we have many or few iterations, large or small population size, hybrid algorithm outperforms other algorithms. This rule also applies to Schwefel 2/21 function because not only it is a simple function, but it also does not have any local optima. As results show in tables 5-3, 5-4, 5-5, 5-6, and 5-7, hybrid algorithm outperformed in comparison with all other algorithms in unimodal and multimodal functions.

Table 5-3: Result of benchmark functions in number of iteration 200 and population size of 30

Function	*Hybrid method*	*GWO*	*PSO*	*BBO*
sphere	*0*	*0*	*0.064355*	*0.045546*
Chung Reynolds	*0*	*0*	*0*	*0.063455*
Schwefel 2/22	*0*	*0.049545*	*0.035231*	*0.024245*
Schwefel 2/21	*0.005634*	*0.015366*	*0.104434*	*0.223567*
Cube	*0*	*0.094653*	*0.073244*	*0.083556*
Dixon & Price	*0.064556*	*0.034444*	*0.042324*	*0.075743*
Griewank	*0.034567*	*0.043433*	*0.047651*	*0.124456*
Rosenbrock	*0.022345*	*0.022655*	*0.107431*	*0.144677*
Ackley	*0.014238*	*0.072762*	*0.052764*	*0.034357*
Rastrigin	*0.013251*	*0.094749*	*0.074321*	*0.073534*
Brown	*0.025231*	*0.030769*	*0.060328*	*0.053567*

Table 5-4: Result of benchmark functions in number of iteration 1000 and population size of 30

Function	*Hybrid method*	*GWO*	*PSO*	*BBO*
sphere	*0*	*0*	*0*	*0*
Chung Reynolds	*0*	*0*	*0*	*2.78E-04*
Schwefel 2/22	*0*	*0*	*6.34E-03*	*3.31E-04*
Schwefel 2/21	*0*	*0*	*0*	*0*
Cube	*0.052115*	*0.115553*	*8.37E-03*	*6.22E-04*
Dixon & Price	*0*	*0.025688*	*5.45E-02*	*7.82E-02*
Griewank	*0*	*0.014521*	*0.012366*	*0.001343*
Rosenbrock	*0*	*0.013483*	*0.010828*	*0.013231*
Ackley	*0.003451*	*0.081342*	*0.017007*	*0.005432*
Rastrigin	*0.004783*	*0.022348*	*0.010613*	*0.024389*
Brown	*0.004532*	*0.038901*	*0.007164*	*0.043265*

Table 5-5: Result of benchmark functions in number of iteration 1500 and population size of 30

Function	*Hybrid method*	*GWO*	*PSO*	*BBO*
sphere	*0*	*0*	*0*	*0*
Chung Reynolds	*0*	*0*	*0*	*0*
Schwefel 2/22	*0*	*2.54E-07*	*0*	*0*
Schwefel 2/21	*0*	*0*	*2.65E-07*	*0.005696*
Cube	*0*	*10.55E-06*	*4.64E-08*	*8.57E-06*
Dixon & Price	*0*	*6.35E-04*	*7.34E-04*	*0.007046*
Griewank	*0*	*0*	*0.035662*	*0.070721*
Rosenbrock	*0.002345*	*0.004236*	*0.000823*	*0.006457*
Ackley	*0.003211*	*0.005673*	*0.007012*	*0.006542*
Rastrigin	*0.002144*	*0.006578*	*0.070687*	*0.005465*
Brown	*0.002754*	*0.004265*	*0.006423*	*0.003333*

Table 5-6: Result of benchmark functions in number of iteration 1000 and population size of 20

Function	*Hybrid method*	*GWO*	*PSO*	*BBO*
sphere	*0*	*0*	*0*	*0*
Chung Reynolds	*0*	*0*	*0*	*0*
Schwefel 2/22	*0*	*0*	*0*	*0*
Schwefel 2/21	*0*	*0*	*6.84E-03*	*3.43E-04*
Cube	*0*	*7.26E-05*	*5.34E-04*	*7.24E-03*
Dixon & Price	*0*	*7.73E-03*	*0.06578*	*5.63E-03*
Griewank	*0*	*0*	*0.053578*	*0.002556*
Rosenbrock	*0*	*0.025335*	*0.012645*	*0.024674*
Ackley	*0.003435*	*0.056322*	*0.012446*	*0.025467*
Rastrigin	*0.007432*	*0.012467*	*0.013234*	*0.036456*
Brown	*0.004562*	*0.025367*	*0.002355*	*0.045366*

Table 5-7: Result of benchmark functions in number of iteration 1000 and population size of 50

Function	*Hybrid method*	*GWO*	*PSO*	*BBO*
sphere	*0*	*0*	*0*	*0*
Chung Reynolds	*0*	*0*	*0*	*0*
Schwefel 2/22	*0*	*0*	*0*	*0*
Schwefel 2/21	*0*	*0*	*6.42E-04*	*0*
Cube	*0*	*8.25E-03*	*0*	*5.43E-05*
Dixon & Price	*0*	*0*	*5.34E-03*	*6.43E-04*
Griewank	*0*	*0.035654*	*0.025435*	*0.003656*
Rosenbrock	*0*	*0.054353*	*0.025435*	*0.026545*
Ackley	*0.008767*	*0.076544*	*0.014679*	*0.003765*
Rastrigin	*0.006576*	*0.037654*	*0.017654*	*0.015434*
Brown	*0.003757*	*0.027645*	*0.004868*	*0.025436*

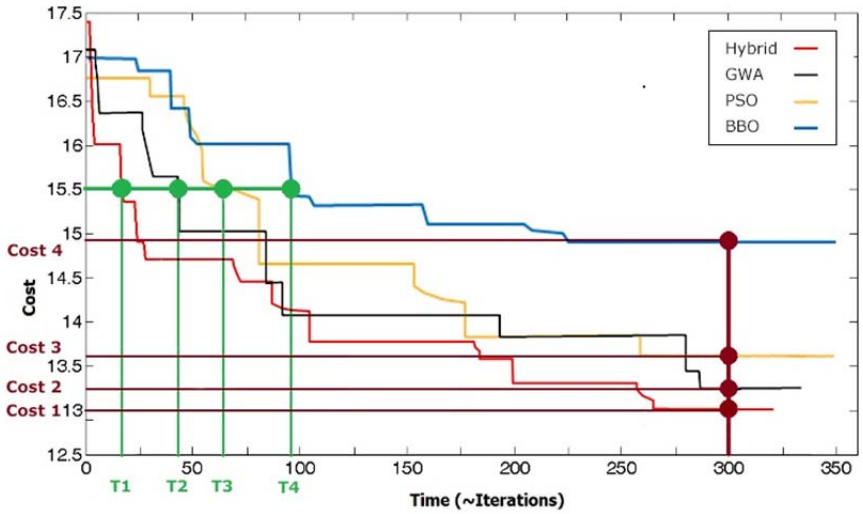


Figure 5-1: Comparison of CPU time to reach optimum result in different algorithms

According to the figure 5-1, results show comparing of the CPU time to reach the optimum in 4 algorithms, hybrid algorithm reaches the optimum, sooner than other algorithms in all life conditions.

Generally speaking, hybrid algorithm is the best algorithm to solve a problem for the simple functions that do not have any local optima. Regarding Schwefel 2/22 function, hybrid algorithm delivered better results than other algorithms in almost all cases. This function is a bit more complex than Sphere function, for it contains both the sum and the product of variables. In addition, this function contains local optima. Therefore, hybrid algorithm could be used to solve the complex problems that contain local optima. Rosenbrock function is a rather complex function which does not have local optima or products of the variables. When we face the restriction of iterations, hybrid algorithm still yields better results. Yet, when there is no iterations restriction, Bees algorithm works better. In conclusion, in order to obtain better results from Bees algorithm for rather complex functions that do not contain local optima, more iterations and larger iteration are required. In Restring and Ackley functions that there are many local optima like Schwefel 2/22 which have local optima, hybrid algorithm outperforms than others in changing the number of generation and iteration size as well. In Griewank algorithm, which is a rather complex function, hybrid algorithm still delivers the best results on the condition that the iteration size and rate of iterations are small or big, low or high because this function does not contain local optima.

The overall computational result shows the GWA is usually trapped in local optima. So, it is not suitable for the problems that contain local optima. Due to its high rate of convergence, this algorithm is an appropriate option for solving other problems. Unlike GWA, hybrid algorithm will not be trapped in local optima, and it seems that taking advantage of TLBO was appropriate to revise GWO algorithm. In general, for very complex problems the results show that proposed hybrid algorithm is suitable and will help us achieve the optimized result especially in resource allocation in cloud environments where there are many local optima.

5.3. Introducing assessment Index

Load balancing is the process of allocation of the total load to cloud resources in order to improve energy efficiency by reducing virtual resources and eliminate a condition in which the numbers of nodes are heavily loaded, while others are idle. The impact of load balancing based on reducing the number of resources and load balancing is called load balancing index. The amount of efficiency of each resource $Ri_{efficiency}$ is equivalent to the percent of resource that has been used compared to the total resource.

Formally, the coefficient of variation of resource efficiency is called the lack of load balancing. This variable indicates the amount of deviation extent from productivity. According to statistical indicators, if this variable is zero, all resources absolutely will be used. This variable is equal to the division of standard deviation productivity σ in resources on mean of the number of resources. If the variable is close to zero, the load balancing will be better done. The standard deviation in system is calculated as:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Ri_{efficiency} - mean\, R_i)^2} \qquad (5\text{-}1)$$

Therefore, load balancing factor *(Flb)* and load imbalance factor *(NFlb)* are introduced using following equations:

$$NFlb = \frac{\sigma}{mean(R_i)} \quad i = 1...n \qquad (5\text{-}2)$$

$$Flb = \frac{C - \sigma}{n} \qquad (5\text{-}3)$$

Where C is the capacity of each resource or virtual machine.

5.4. Simulation Results

To assess the performance of proposed algorithm, we used two different probability distributions, uniform and normal distribution, which are considered as distribution of workload in cloud computing. Uniform distribution workloads represent the uniform amount of small, medium and large size of loads while normal distribution workloads show the

symmetrical fashion in the cloud environment. In all of the simulations, the following parameters were used for our proposed algorithm:

a) MaxIt = 500; % Maximum Number of Iterations
b) nPop = 250; %for each test must be updated Population Size
c) Dim = number of variables (e.g. a=0.0354, b=38.3055, c=1243.531)

MaxIt is the number of iterations for the algorithm. Whatever it increases, the time needed to implement increases, and in some cases, accuracy increases, but increasing iteration always doesn't guarantee improvement of the solution. *nPop* is proportion of the number of package in bin packaging problem which is the same processes that can be processed by cloud resources. This amount according to the bin packing data set will be equal to 250, 500, 1000 and etc. Coefficients *a, b, c* have been selected according to the resource [49]. Dim is the number of packets in the bin packaging dataset.

5.4.1. Uniform distribution workload

According to the assessment index, load imbalance and the number of resources allocated are compared with each other. In this case, bin packing problem is considered as a model of resource allocation in the cloud computing. The packages on the bin packing are considered as processes requests that are processed in the cloud virtual servers. We want to allocate the packages or processes with minimum number of boxes or servers. Fewer boxes cause the resources to be used in a better manner, load balancing can be done better, and fewer servers will be effective on costs and energy consumption.

To produce random numbers, uniform distributed in the interval [0, 1] is considered, which in case of need for greater interval, it has become a larger scale. Experiments have been conducted on ready data sets called Falkenauer [106]. This data set is presented in eight categories, which the sets one to four are uniform, and they are displayed with the letter *u*. The next number after *u* indicates the number of packets. These sets have been distributed between 20 and 100 and the capacity of boxes is 150. The second series of sets five to eight are displayed with the letter *t*. The next number after *t* indicates the number of packets. This

set is called triplet, the packets are distributed between 25 and 50, and the size of the boxes is 100. There are 60 packages in the dataset binpack5 with the capacity of 100 boxes, as follows:

36.6000	26.8000	36.6000	43.0000	26.3000	30.7000	41.4000
28.7000	29.9000	49.5000	25.1000	25.4000	47.4000	25.2000
27.4000	37000	26.9000	36.1000	47.3000	25.2000	27.5000
47.2000	25.9000	26.9000	44.4000	25.8000	29.8000	43.9000
27.3000	28.8000	44.5000	27.2000	28.3000	41.9000	26.1000
32.0000	36.3000	27.1000	36.6000	35.5000	27.3000	37.2000
46.6000	26.2000	27.2000	35.7000	29.2000	35.1000	39.5000
25.5000	35.0000	35.0000	30.3000	34.7000	45.0000	25.2000
29.8000	41.0000	27.5000	31.5000			

The optimal solution to solve this problem is the allocation to the following form. The optimal solution of the allocation is 20 boxes. Our proposed method has achieved an approximate solution of 23. The solution has acted better than the GW and TLBO algorithms alone, it is shown in Table 5-8.

Table 5-8: Allocation with the proposed algorithm in data set binpack5

Dataset	***Gray wolf***	***TLBO***	***BBO***	***PSO***	***Proposed method***	***Optimal result***
First part of binpck5 dataset	24	26	25	24	23	20

Table 5-9: Allocation in boxes dataset binpack5

Bin1	49.5,47.4,	Bin13	35,35,29.9,
Bin2	47.3,47.2,	Bin14	34.7,32,31.5,
Bin3	46.6,45,	Bin15	30.7,30.3,29.8,
Bin4	44.5,44.4,	Bin16	29.8,28.8,28.7,
Bin5	43.9,43,	Bin17	28.3,27.5,27.4,
Bin6	41.9,41.4,	Bin18	27.3,27.3,27.2,
Bin7	41,39.5,	Bin19	27.2,26.9,26.9,
Bin8	37.2,37,25.5,	Bin20	26.8,26.2,26.1,
Bin9	36.6,36.6,26.3,	Bin21	25.9,25.8,25.4,
Bin10	36.6,36.3,27.1,	Bin22	25.2,25.2,25.2,
Bin11	36.1,35.7,27.5,	Bin23	25.1,
Bin12	35.5,35.1,29.2,		

Table 5-9 shows the allocation of all packages of binpak5 dataset to 23 boxs. Each box is shown as a bin which numbers in front of that shows the packages weights (e.g. weights of 49.5, 47.4 have placed in the first box). Other boxes have allocated to different packages in

the same manner. The percentage of unallocated free space in data set Binpack5 is 0.1304 or 13.04% and load balancing (the space allocated) is equivalent to 0.8696.

The best possible mode is the division of the sum of the packages' values to the space of 100, which 20 is obtained as the optimal solution. Since the problem is a kind discrete type, there is the possibility of load imbalance in the optimal solution. In the following, the comparison of these methods is continued by different experiment datasets. Table 5-10 shows the results.

This table shows that with increasing data, the accuracy of the proposed method reduces, but in total, in solving the resource allocation problem with bin packaging problem, our method has higher accuracy than other methods. In the second experiment, according to data dispersion and greater capacity of the box compared with the packages, it will be more difficult; however, the proposed method seems desirable and outperforms other methods. Figures 5-2 and 5-3 illustrate the results.

Table 5-10: Differences between the proposed method and other methods in terms of solution of packages allocated

Exp.	***Dataset***	***No. of Jobs***	***Resource capacity***	***TLBO***	***GWO***	***Hybrid***	***Optimal solution***
First	*Binpack1-U250_00*	*250*	*150*	*100*	*100*	*100*	*99*
First	*Binpack1-U250_01*	*250*	*150*	*102*	*101*	*101*	*100*
First	*Binpack2-U250_00*	*250*	*150*	*104*	*100*	*100*	*99*
First	*Binpack2-U250_01*	*250*	*150*	*104*	*101*	*101*	*100*
First	*Binpack3-U500_00*	*500*	*150*	*206*	*203*	*201*	*198*
First	*Binpack3-U500_01*	*500*	*150*	*208*	*207*	*204*	*201*
First	*Binpack4-U1000_00*	*1000*	*150*	*413*	*407*	*403*	*399*
First	*Binpack4-u1000_01*	*1000*	*150*	*423*	*419*	*411*	*406*
Second	*Binpack5-T60_00*	*60*	*100*	*23*	*23*	*23*	*20*
Second	*Binpack5-T60_01*	*60*	*100*	*23*	*23*	*23*	*20*
Second	*Binpack6-T120_00*	*120*	*100*	*46*	*45*	*45*	*40*
Second	*Binpack6-T120_01*	*120*	*100*	*47*	*45*	*45*	*40*
Second	*Binpack7-T249_00*	*249*	*100*	*99*	*96*	*94*	*83*
Second	*Binpack7-T249_01*	*249*	*100*	*103*	*98*	*95*	*83*
Second	*Binpack8-T501_00*	*501*	*100*	*203*	*198*	*190*	*167*
Second	*Binpack8-T501_01*	*501*	*100*	*207*	*201*	*191*	*167*

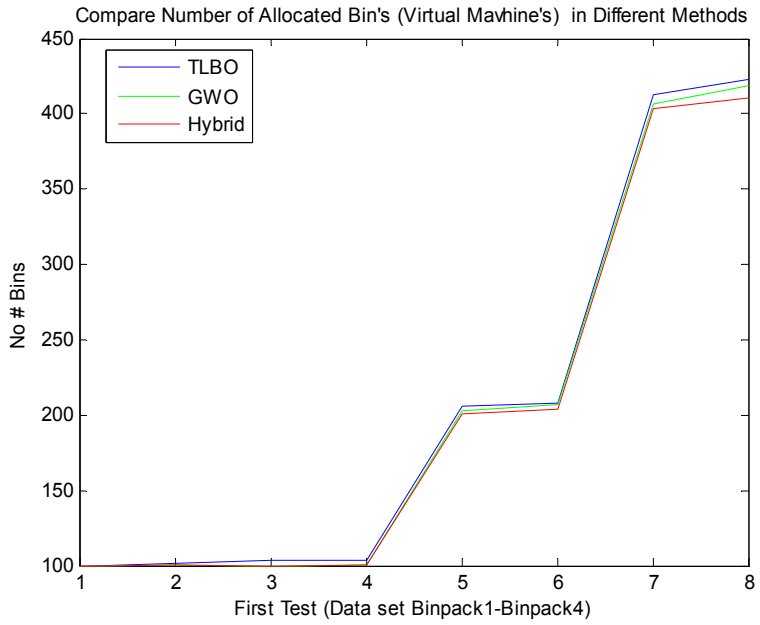


Figure 5-2: Comparison of hybrid method with other methods in the first experiment to achieve optimal solution

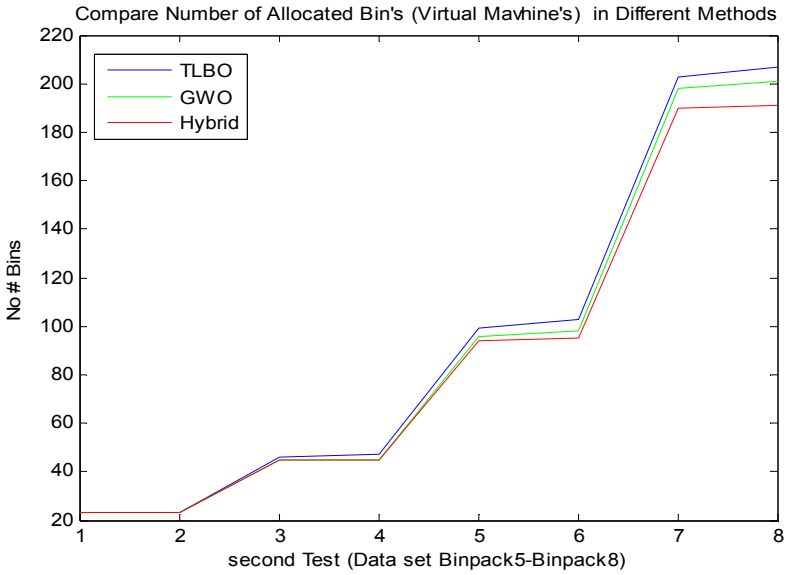


Figure 5-3: Comparison of hybrid method with other methods in the second experiment to achieve optimal solution

According to recent articles [107-108], CGA-CGT and HI-BP methods have provided the best solution for bin packing problem. We compared the proposed method with these two methods in next experiment. In order to implement these methods, we also used Matlab 2016 and performed the implemented program on the same machine which is used for all of the

simulations as explained earlier (see Appendix).The proposed hybrid method shows outperforms than these two methods. Table 4-3 shows it.

Table 5-11: Comparison of the proposed method with HI_BP and CGA-CGT

Exp.	*Dataset*	*No. of Job*	*Resource capacity*	*HI_BP*	*CGA-CGT*	*Hybrid*	*Optimal solution*
First	*Binpack1-U250_00*	*250*	*150*	*100*	*100*	*100*	*99*
First	*Binpack1-U250_01*	*250*	*150*	*101*	*101*	*101*	*100*
First	*Binpack2-U250_00*	*250*	*150*	*100*	*100*	*100*	*99*
First	*Binpack2-U250_01*	*250*	*150*	*101*	*101*	*101*	*100*
First	*Binpack3-U500_00*	*500*	*150*	*204*	*201*	*201*	*198*
First	*Binpack3-U500_01*	*500*	*150*	*204*	*204*	*204*	*201*
First	*Binpack4-U1000_00*	*1000*	*150*	*404*	*404*	*403*	*399*
First	*Binpack4-u1000_01*	*1000*	*150*	*414*	*413*	*411*	*406*
Second	*Binpack5-T60_00*	*60*	*100*	*23*	*23*	*23*	*20*
Second	*Binpack5-T60_01*	*60*	*100*	*23*	*23*	*23*	*20*
Second	*Binpack6-T120_00*	*120*	*100*	*45*	*45*	*45*	*40*
Second	*Binpack6-T120_01*	*120*	*100*	*47*	*45*	*45*	*40*
Second	*Binpack7-T249_00*	*249*	*100*	*96*	*94*	*94*	*83*
Second	*Binpack7-T249_01*	*249*	*100*	*101*	*97*	*95*	*83*
Second	*Binpack8-T501_00*	*501*	*100*	*202*	*194*	*190*	*167*
Second	*Binpack8-T501_01*	*501*	*100*	*204*	*199*	*191*	*167*

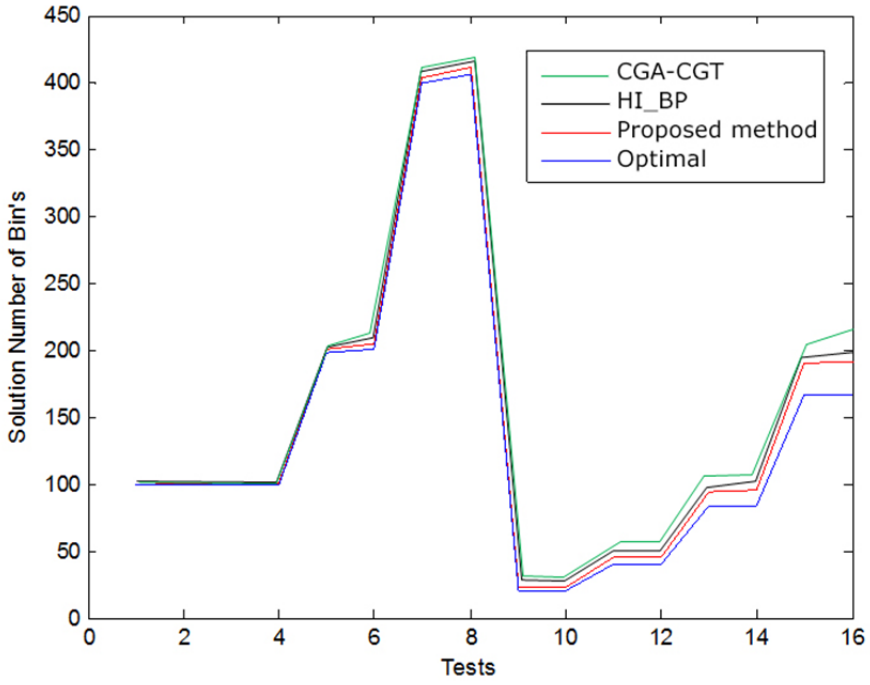


Figure 5-4: Difference between the proposed method and CGA-CGT, HI-BP, and optimal solution

Figure 5-4 shows the difference between the approximate solutions with the optimal solution which the difference is within acceptable limits. Table 5-12 shows the calculating

load balancing in different tests. As the results show, with the increasing data, the proposed method has superior performance than the two other methods.

Nflb index shows the lack of load balancing, which increasing its values demonstrate better load balancing. Increasing load imbalance shows that the maximum resource capacity is used. It means that the amount of empty capacity of the resources is low, so with increasing data, load balancing is performed better in the proposed algorithm. Table 5-12 shows load imbalance between the methods in the first and fourth experimental set.

Table 5-12: Comparison of the load imbalance between the proposed algorithm and other algorithms

Exp.	***DataSet***	***No. of Job***	***Resource capacity***	***Nflb-TLBO***	***Nflb-GWO***	***Nflb-Proposal method***
First	*Binpack1-U250_00*	*250*	*150*	*0.0145*	*0.0145*	*0.0145*
First	*Binpack1-U250_01*	*250*	*150*	*0.0193*	*0.0195*	*0.0195*
First	*Binpack2-U250_00*	*250*	*150*	*0.0138*	*0.0145*	*0.0145*
First	*Binpack2-U250_01*	*250*	*150*	*0.0178*	*0.0195*	*0.0195*
First	*Binpack3-U500_00*	*500*	*150*	*0.0159*	*0.0163*	*0.0170*
First	*Binpack3-U500_01*	*500*	*150*	*0.0143*	*0.0147*	*0.0155*
First	*Binpack4-U1000_00*	*1000*	*150*	*0.0097*	*0.0103*	*0.0113*
First	*Binpack4-u1000_01*	*1000*	*150*	*0.0119*	*0.0127*	*0.140*
Second	*Binpack5-T60_00*	*60*	*100*	*0.1304*	*0.1304*	*0.1304*
Second	*Binpack5-T60_01*	*60*	*100*	*0.1304*	*0.1304*	*0.1304*
Second	*Binpack6-T120_00*	*120*	*100*	*0.1023*	*0.1111*	*0.1111*
Second	*Binpack6-T120_01*	*120*	*100*	*0.1013*	*0.1111*	*0.1111*
Second	*Binpack7-T249_00*	*249*	*100*	*0.1010*	*0.1090*	*0.1170*
Second	*Binpack7-T249_01*	*249*	*100*	*0.1203*	*0.1226*	*0.1263*
Second	*Binpack8-T501_00*	*501*	*100*	*0.1107*	*0.1139*	*0.1211*
Second	*Binpack8-T501_01*	*501*	*100*	*0.1219*	*0.1231*	*0.1257*

Index of load imbalance acts in a reverse manner compared with load balancing index. With increasing NFLB, the performance of load balancing will be increased. Due to the incorrect understanding structure of training in the experiments, TLBO method doesn't have a good load balancing with increasing data. The proposed method shows a good performance with increasing data in load balancing. Load balancing can be calculated by the lack of load balancing. Load balancing can show appropriate allocation. The second experiment in Table

5-12 confirms the performance of the proposed method in the fifth to eighth data set.The related charts have shown in figure 5-5 and 5-6.

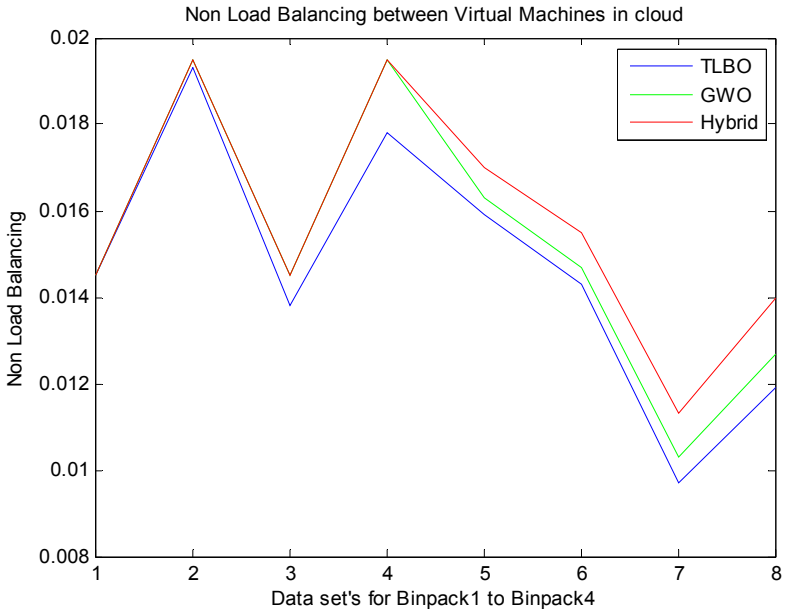


Figure 5-5: Load imbalance in the first test set

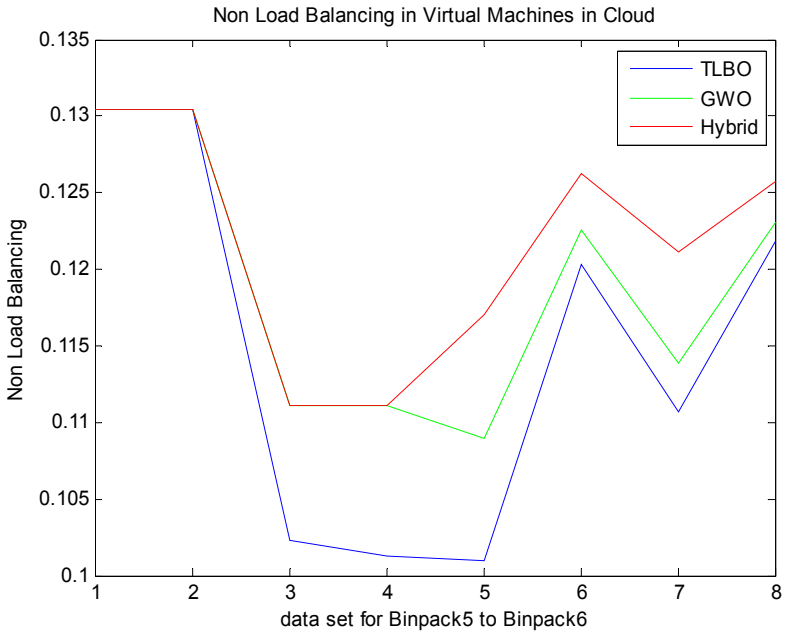


Figure 5-6: Load imbalance in the second test set

Variable of relative percentage changes in comparison with the best answer can demonstrate the accuracy of the algorithm. Therefore, in order to evaluate robustness of the proposed algorithm in next experiment we calculate *RPD (Robust Parameter Design)*

parameter which is a normalized change percentage than the best answer where $f_{heuristic}$ is meta-heuristic value and $f_{optimal}$ is the optimal value.

$$RPD = 100 \times \frac{f_{heuristic} - f_{optimal}}{f_{optimal}} \quad (5\text{-}4)$$

This criterion shows the performance and robustness of the algorithm with the increase of the data. Table 5-13 demonstrates comparison of RPD between proposed algorithm and optimal solution.

Table 5-13: Comparison of RPD between proposed method and optimal solution

Exp.	*Dataset*	*No. of Job*	*Resource capacity*	*Hybrid*	*Optimal solution*	*RPD*
First	*Binpack1-U250_00*	*250*	*150*	*100*	*99*	*1/99*
First	*Binpack1-U250_01*	*250*	*150*	*101*	*100*	*1/100*
First	*Binpack2-U250_00*	*250*	*150*	*100*	*99*	*1/99*
First	*Binpack2-U250_01*	*250*	*150*	*101*	*100*	*1/100*
First	*Binpack3-U500_00*	*500*	*150*	*201*	*198*	*3/198*
First	*Binpack3-U500_01*	*500*	*150*	*204*	*201*	*3/201*
First	*Binpack4-U1000_00*	*1000*	*150*	*403*	*399*	*4/399*
First	*Binpack4-u1000_01*	*1000*	*150*	*411*	*406*	*5/406*
Second	*Binpack5-T60_00*	*60*	*100*	*23*	*20*	*3/20*
Second	*Binpack5-T60_01*	*60*	*100*	*23*	*20*	*3/20*
Second	*Binpack6-T120_00*	*120*	*100*	*45*	*40*	*5/40*
Second	*Binpack6-T120_01*	*120*	*100*	*45*	*40*	*5/40*
Second	*Binpack7-T249_00*	*249*	*100*	*94*	*83*	*11/83*
Second	*Binpack7-T249_01*	*249*	*100*	*95*	*83*	*12/83*
Second	*Binpack8-T501_00*	*501*	*100*	*190*	*167*	*23/167*
Second	*Binpack8-T501_01*	*501*	*100*	*191*	*167*	*22/167*

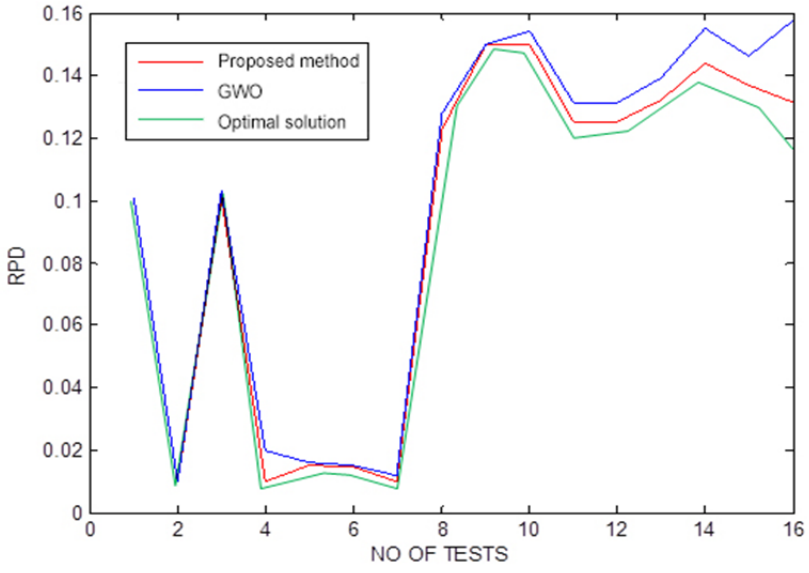


Figure 5-7: Comparison of RPD between GWO, optimal solution, and proposed algorithm

The results indicate that the proposed method has superior performance with increasing data. The average of relative percentage changes is increased in an appropriate way in experimental data and has a better flow in comparison with other algorithms. Figure 5-7 indicates improvements of the RPD parameter in comparison with other results. The results show, with increasing data, the performance is observed to be stable in the proposed method.

5.4.2. Normal distribution workload

In order to evaluate proposed algorithm using normal distribution workload, we used CloudSim simulator in two different environments. CloudSim has the capability to simulate for modelling cloud computing in the homogeneous and heterogeneous environment.

Table 5-14: CloudSim setting for homogeneous and heterogeneous cloud environments

Entities	Parameters	Values for homogeneous Env.	Values for heterogeneous Env.
User	No. of Users	20	20
CloudLet	No. of CloudLets	120-800	120-800
Host	No. of host	5	5
	Ram	4096MB	25GB
	Storage	500GB	2TB
Virtual Machines	Network Bandwidth	5GB	10GB
	No. of VMs	12	20
	MIPS		
	RAM	2048MB	128MB to 15GM
	Bandwidth		128MB to 15GM
	VMM	Xen	Xen
	Operation system	Linux	Linux
	No. of CPUs	2	2
	No. of Datacenters	2	2

CloudSim toolkit is an open source framework that enables developers to model and different layers of cloud computing infrastructure and application services. Table 5-14 shows presents cloudlets, cloud users, virtual machines, and host and data center properties for two

different homogeneous and heterogeneous cloud environments respectively. The efficiency of the algorithm under uniform distribution through Falkenauer datasets was investigated.

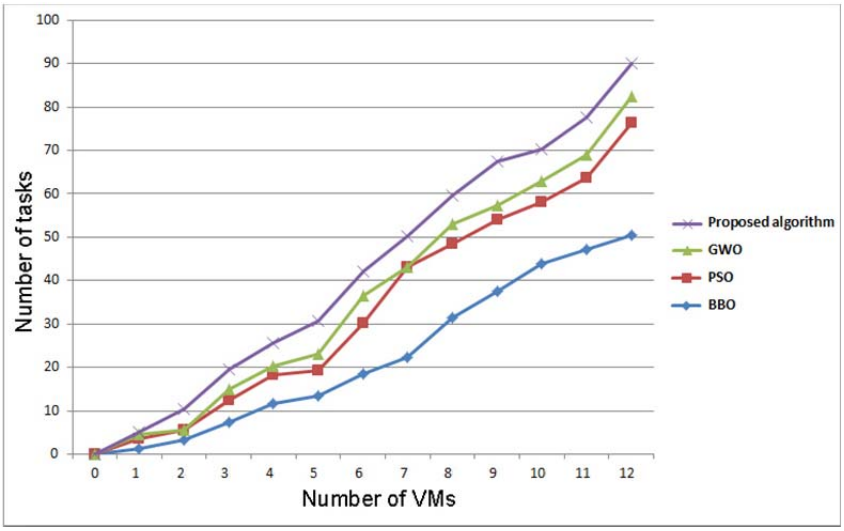


Figure 5-8: Performance evaluation between different algorithms in homogeneous environment

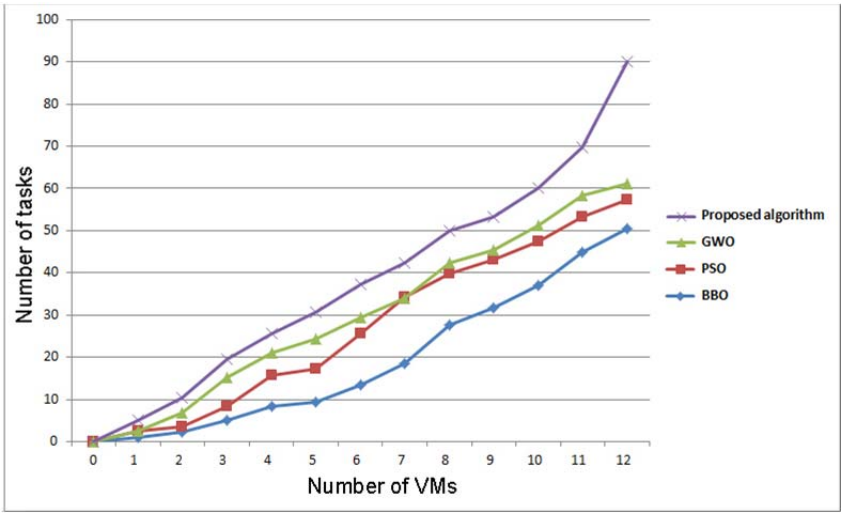


Figure 5-9: Performance evaluation between different algorithms in heterogeneous environment

Figure 5-8, 5-9 show the performance between proposed algorithm, GWO, PSO, and BBO with using normal distribution workloads in homogeneous and heterogeneous environments in the cloud system respectively. The x-axis and y-axis are the numbers of virtual machines and number of loads respectively. The results clearly show that proposed algorithm has better performance for using maximum capacity of virtual machines. The figures show that proposed algorithm maximizes the utilization of resources. As it is clear, with increasing the

loads, the proposed algorithm uses the least number of virtual machines. It means the simulated cloud system using the proposed method can balance loads across virtual machines.

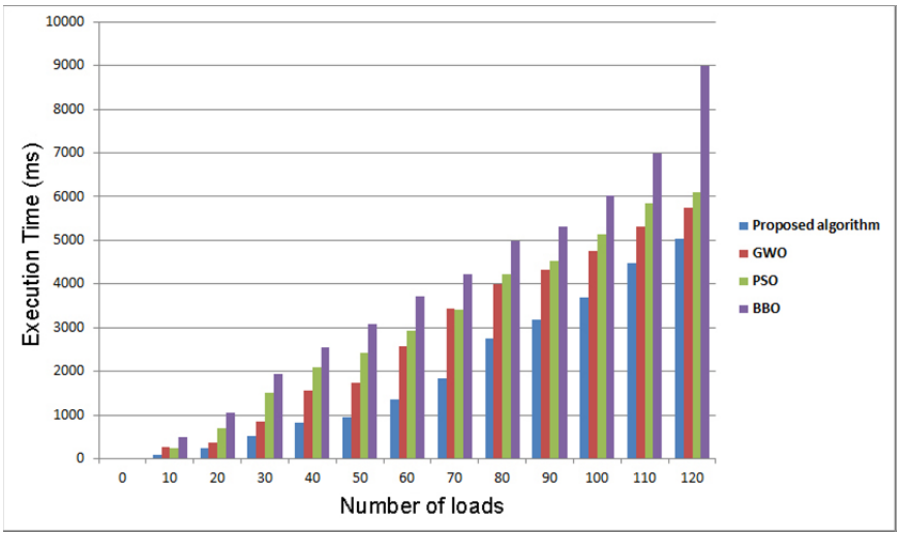


Figure 5-10: Comparison of makespan between different algorithms in homogeneous environment

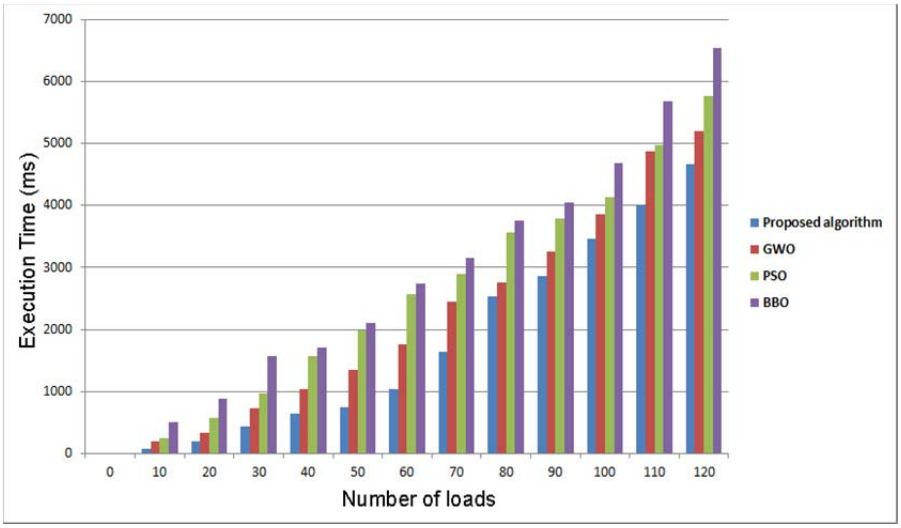


Figure 5-11: Comparison of makespan between different algorithms in heterogeneous environment

Figure 5-10 and 5-11 illustrates the comparison of makespan between proposed algorithm, GWO, PSO, and BBO with using normal distribution workloads in homogeneous and heterogeneous environments in the cloud system respectively. The x-axis and y-axis are the numbers of cloudlets and execution time respectively. The results clearly depict that proposed

algorithm outperforms in terms of time reduction. Reduction of the makespan has a direct effect on response time. The results indicate the algorithm has superior performance in both homogeneous and heterogeneous environments in comparison with other methods.

The comparison of throughput between proposed algorithm, GWO, PSO, and BBO is shown in figures 5-12 and 5-13. This comparison is based on normal distribution workloads in both the heterogeneous and homogeneous environments in the cloud system. The measurement of this factor is calculated based on the number of the performed tasks and time. The results indicate that the number of performed tasks by proposed algorithm is impressive in comparison with other algorithms. The overall results reflect superiority of the proposed algorithm in comparison with other algorithms.

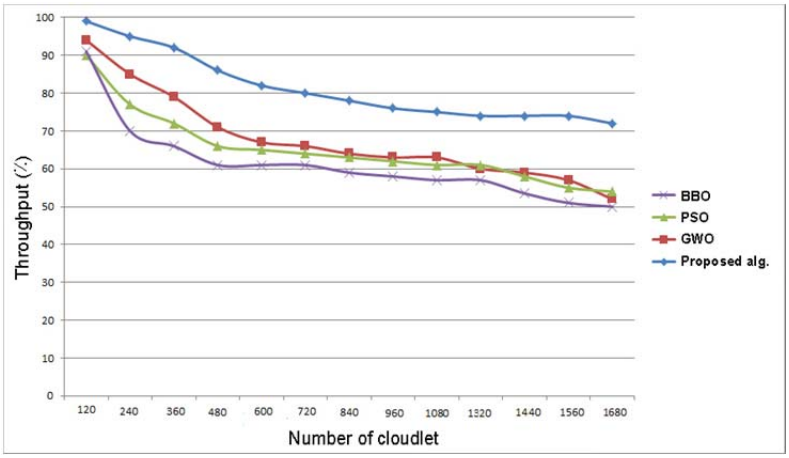


Figure 5-12: Comparison of throughput between different algorithms in homogeneous environment

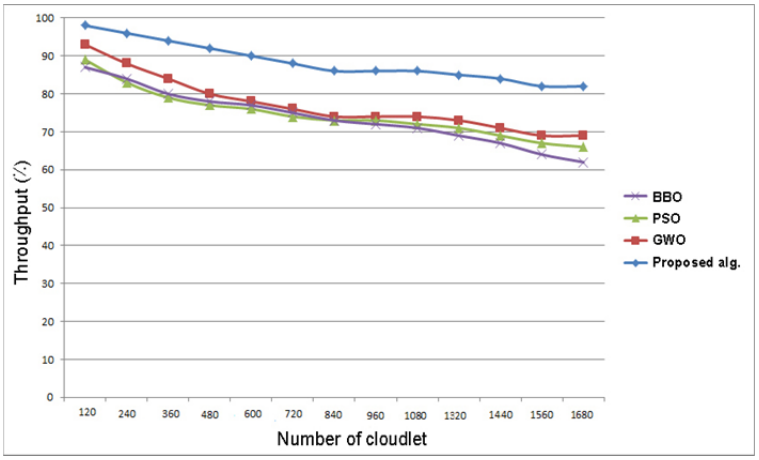


Figure 5-13: Comparison of throughput between different algorithms in heterogeneous environment

Chapter 6

Conclusions and Future Directions

6.1. Conclusions and Discussion

The high workloads across virtual machines are one of the main challenges of the cloud computing. Therefore, appropriate resource allocation and load balancing techniques and methods are becoming increasingly vital for cloud environments. The requested tasks by a client have to wait for sending to the virtual machines where appropriate resources are available. These allocations are independent of the executive priority of the tasks. However, cloud client may offer larger value for its requests in order to raise his/her task priority and eventually may succeed in taking control over the resources needed. Current resource allocation methods such as FIFO and Round-Robin which is used in the clouds do greedy and unfair allocation regardless of priority between users' jobs.

This thesis designs a new method for dynamic resource allocation problem in cloud computing. This method is developed for the bin-packing problem, where the packages were introduced as workloads and each Bin introduced as a virtual machine in cloud computing environment. The ultimate goal of the problem solving was that the allocation of packages to Bins is done so that using the minimum number of Bins.

A new combinatory meta-heuristic algorithm using gray wolf optimizer and teaching-learning-base optimizer was introduced in this regard. The research issues and the major contributions which have been made in this thesis are summarized as follows:

- In chapter 2, the concept of cloud computing and theoretical foundations in virtualized environments are introduced. This chapter further investigates the problem of resource allocation and explains that the load balancing leads to reduce traffic load in the cloud network, and as a result, energy consumption costs will be decreased.

- Chapter 3 presents a comprehensive survey of the state-of-the-arts on resource allocation, load balancing and scheduling techniques, algorithms, and methods in cloud environments. In this chapter, we provide an overview of the related works of techniques related to resource allocation at different dimensions and levels such as objectives, optimization methods, simulation and implementation tools, and the executable environment. The main goal to organize the state-of-the-arts has been to

reach a deep and clear comprehension of the problem, to identify the key factors, issues, challenges in relation to existing related works.

- Chapter 4 proposes a bin packing based approach for dynamic resource allocation and load balancing using two relatively new multi-objective optimization techniques. By an efficient allocation process, the number of rented virtual machines is reduced, and it causes decreasing number of cloud physical servers, and consequently, cost reduction in cloud infrastructures. Also, it has direct effects on reducing energy consumption. This process leads to maximize utilization of resources in virtual machines. Therefore, the amount of waste spaces is decreased across virtual machines and helps to increase load balancing in the cloud environment. The proposed algorithm is an approximation algorithm.

- In chapter 5, in order to evaluate the performance of our method, the proposed algorithm is benchmarked on eleven test functions and a comparative study is conducted to verify the results with other existing algorithms. Also, the proposed algorithm is simulated in two different tools (Matlab and CloudSim) and the experimental results are presented. The evaluation results indicate that the proposed method in high workloads for resource allocation in Cloud Scheduler has better performance than other existing methods. The complexity of the algorithm is polynomial. The results of the load balancing experiments in two homogeneous and heterogeneous environment show that by applying uniform and normal workload distributions, the proposed method outperforms in comparison with other existing techniques.

6.2. Future Research Directions

Our main focus in this study was on suggesting a novel hybrid meta-heuristic algorithm to improve the performance of resource allocation process in cloud computing environments. Some issues related to resource allocation in cloud computing that needs further research, have not been addressed in this thesis. This practical study can be considered as the starting

point of the variety of researchers. The potential future directions of this research include the following:

- Semi-automated admission control mechanisms for resource allocation in the cloud to decide which user requests to be accepted. This important issue is related to reservation of the cloud resource in advance. The main problem of traditional resource allocation techniques in the cloud is that requested resource by user's application may be not available in time and the related requests will be refused by the system. Advanced reservation request technique is used to guarantee the availability of required resources at the specified time. The aim is to integrate a semi-automated admission control mechanism with our algorithm to improve the optimality of our method.

- Load forecasting techniques are extremely important in order to reduce energy consumption and predict overall workloads in cloud computing. Therefore, as future work, we can improve the stability and performance of our algorithm with forecasting algorithms.

- One of the important usages of our proposed algorithm is to be integrated with OpenStack(see Appendix A). Therefore, implementation of proposed algorithm in order to provide an efficient resource allocation and scheduling policies to this cloud environment can be an important future work.

- Most of the state-of-the-arts on resource allocation in cloud computing focus on workflows with independent tasks. Extension of the proposed algorithm for workflows with dependent jobs can be done in future.

- Trying to extend proposed algorithm to different distributed environment. For example, simulating the algorithm for resource allocation problem in Apache Hadoop and evaluation of the algorithm for this environment.

- The most of the research focus to present new resource allocation methods and techniques while there is no any resource allocation framework that is practical for

different cloud deployment models. Therefore, developing a resource allocation framework is needed.

- The definition and calculation of energy and reducing network traffic through dynamic resource allocation, load balancing, and cloud scheduling using proposed algorithm can be another area for future research.

- More accuracy in the calculation of choices by the teaching-learning-base algorithm can be another area of future research. Neural networks can be used in order to predict. Neural network doesn't guarantee the convergence, but the combination of our proposed method with neural network techniques can guarantee convergence and increase the speed.

- Implementation of the proposed method with the real workloads in a real cloud environment can provide more accurate solutions for the future research and introduce more research areas for different types of clouds.

References

[1] Subashini, S. and Kavitha, V.: A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, *34*(1), pp.1-11, 2011.

[2] Chen, S.L., Chen, Y.Y. and Kuo, S.H.: A novel load balancing architecture and algorithm for cloud services. *Computers & Electrical Engineering*, *58*, pp.154-160, 2017.

[3] Nema, L., Sharma, A. and Jain, S.: Load Balancing Algorithms in Cloud Computing: An Extensive Survey. *International Journal of Engineering Science*, pp. *63-74, 2016.*

[4] Murata, Y., Egawa, R., Higashida, M. and Kobayashi, H.: A history-based job scheduling mechanism for the vector computing cloud. In *Applications and the Internet (SAINT), 2010 10th IPSJ International Symposium,* pp. 125-128, IEEE, 2010.

[5] Wang, W., Zeng, G., Tang, D. and Yao, J.: Cloud-DLS: Dynamic trusted scheduling for Cloud computing. *Expert Systems with Applications*, *39*(3), pp.2321-2329, 2012.

[6] Pawar, C.S. and Wagh, R.B.: A review of resource allocation policies in cloud computing. *World Journal of Science and Technology*, *2*(3), pp.165-167, 2012.

[7] Alakeel, A.M.: A guide to dynamic load balancing in distributed computer systems. *International Journal of Computer Science and Information Security*, *10*(6), pp.153-160, 2010.

[8] Ghribi, C., 2014. Energy efficient resource allocation in cloud computing environments, *Doctoral dissertation, Institut National des Télécommunications,* 2014.

[9] Wu, L., Garg, S.K. and Buyya, R.: SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments. *Journal of Computer and System Sciences*, *78*(5), pp.1280-1299, 2012.

[10] Li, X. and Du, J.: Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing. *IET Information Security*, *7*(1), pp.39-50, 2013.

[11] Liu, K., Jin, H., Chen, J., Liu, X., Yuan, D. and Yang, Y.: A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform. *The International Journal of High Performance Computing Applications*, *24*(4), pp.445-456, 2010.

[12] Alkhanak, E.N., Lee, S.P. and Khan, S.U.R.: Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities. *Future Generation Computer Systems*, *50*, pp.3-21, 2015.

[13] Chiu, C.F., Hsu, S.J., Jan, S.R. and Chen, J.A.: Task scheduling based on load approximation in cloud computing environment. In *Future Information Technology*, pp. 803-808, Springer, 2014.

[14] Yang, X., Liu, D., Ma, H. and Xu, Y.: Online approximate solution of HJI equation for unknown constrained-input nonlinear continuous-time systems. *Information Sciences*, *328*, pp.435-454, 2016.

[15] Singh, S. and Chana, I.: A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of Grid Computing*, *14*(2), pp.217-264, 2016.

[16] Toosi, A.N., Calheiros, R.N. and Buyya, R.: Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)*, *47*(1), p.7-19, 2014.

[17] Rong, C., Nguyen, S.T. and Jaatun, M.G.: Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering*, *39*(1), pp.47-54, 2013.

[18] Boutaba, R., Zhang, Q. and Zhani, M.F.: Virtual machine migration in cloud computing environments: Benefits, challenges, and approaches. *Communication Infrastructures for Cloud Computing*, pp.383-408, 2013.

[19] Mousavi, S.M. and Gábor, F.: A novel algorithm for Load Balancing using HBA and ACO in Cloud Computing environment. *International Journal of Computer Science and Information Security*, *14*(6), p.48, 2016.

[20] Singh, P., Dutta, M. and Aggarwal, N.: A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowledge and Information Systems*, *52*(1), pp.1-51, 2017.

[21] Pooranian, Z., Shojafar, M., Abawajy, J.H. and Abraham, A.: An efficient meta-heuristic algorithm for grid computing. *J. Comb. Optim.*, *30*(3), pp.413-434, 2015.

[22] Gandomi, A.H., Yang, X.S. and Alavi, A.H.: Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with computers*, pp.1-19, 2013.

[23] BoussaïD, I., Lepagnot, J. and Siarry, P.: A survey on optimization metaheuristics. *Information Sciences*, *237*, pp.82-117, 2013.

[24] Khetan, A., Bhushan, V. and Gupta, S.C.: A novel survey on load balancing in cloud computing. *Proc. International Journal Of Engineering Research & Technology (IJERT) ISSN*, pp.2278-0181, 2013.

[25] Singh, S. and Chana, I.: A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of Grid Computing*, *14*(2), pp.217-264, 2016.

[26] Randles, M., Lamb, D. and Taleb-Bendiab, A.: April. A comparative study into distributed load balancing algorithms for cloud computing. In *Advanced Information Networking and Applications, 2010 IEEE 24th International Conference,* pp. 551-556, 2010.

[27] Khan, N., Shah, A. and Nusratullah, K.: Adoption of Virtualization in Cloud Computing: A Foundation Step towards Green Computing. *International Journal of Green Computing (IJGC)*, *6*(1), pp.40-47, 2015.

[28] Erl, T., Puttini, R. and Mahmood, Z.: Cloud computing: concepts, technology & architecture. *Pearson Education*, Book, 2013.

[29] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L. and Leaf, D.: NIST Cloud Computing Reference Architecture. *National Institute of Standards and Technology of the US Department of Commerce*, Special Publication 500-292, 2014.

[30] Mousavi. SM.,Fazekas. G.: Increasing QoS in SaaS for low Internet speed connections in cloud. *The 9th International Conference on Applied Informatics, Eger, Hungary,* pp. 195-200, 2014.

[31] Kalloniatis, C., Mouratidis, H. and Islam, S.: Evaluating cloud deployment scenarios based on security and privacy requirements. *Requirements Engineering*, *18*(4), pp.299-319, 2013.

[32] Kavis, M.J.: Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS). *John Wiley & Sons*, Book, 2014.

[33] Chun, S.H. and Choi, B.S.: Service models and pricing schemes for cloud computing. *Cluster Computing*, *17*(2), pp.529-535, 2014.

[34] Lin, J.W., Chen, C.H. and Lin, C.Y.: Integrating QoS awareness with virtualization in cloud computing systems for delay-sensitive applications. *Future Generation Computer Systems*, *37*, pp.478-487, 2014.

[35] García-Valls, M., Cucinotta, T. and Lu, C.: Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture*, *60*(9), pp.726-740, 2014.

[36] Lubomski, P., Kalinowski, A. and Krawczyk, H.: June. Multi-level virtualization and its impact on system performance in cloud computing. In *International Conference on Computer Networks*, pp. 247-259, Springer International Publishing, 2016.

[37] Durairaj, M. and Kannan, P.: A study on virtualization techniques and challenges in cloud computing. *International Journal of Scientific and Technology Research*, *3*(11), pp.147-51, 2014.

[38] Blenk, A., Basta, A., Reisslein, M. and Kellerer, W.: Survey on network virtualization hypervisors for software defined networking. *IEEE Communications Surveys & Tutorials*, *18*(1), pp.655-685, 2016.

[39] Xiao, Z., Song, W. and Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE transactions on parallel and distributed systems*, *24*(6), pp.1107-1117, 2013.

[40] Um, T.W., Lee, H., Ryu, W. and Choi, J.K.: Dynamic resource allocation and scheduling for cloud-based virtual content delivery networks. *ETRI Journal*, *36*(2), pp.197-205, 2014.

[41] Anuradha, V.P. and Sumathi, D.: February. A survey on resource allocation strategies in cloud computing. In *Information Communication and Embedded Systems (ICICES), 2014 International Conference,* pp. 1-7, IEEE, 2014.

[42] Shyamala, K. and Rani, T.S.: An analysis on efficient resource allocation mechanisms in cloud computing. *Indian Journal of Science and Technology*, *8*(9), pp.814-821, 2015.

[43] Domanal, S.G. and Reddy, G.R.M.: January. Optimal load balancing in cloud computing by efficient utilization of virtual machines. In *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference,* pp. 1-4, IEEE, 2014.

[44] Shahzad, F.: State-of-the-art survey on cloud computing security Challenges, approaches and solutions. *Procedia Computer Science*, *37*, pp.357-362, 2014.

[45] Shamsi, J., Khojaye, M.A. and Qasmi, M.A.: Data-intensive cloud computing: requirements, expectations, challenges, and solutions. *Journal of grid computing*, *11*(2), pp.281-310, 2013.

[46] Zhan, Z.H., Liu, X.F., Gong, Y.J., Zhang, J., Chung, H.S.H. and Li, Y.: Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Computing Surveys (CSUR)*, *47*(4), p.63, 2015.

[47] Dave, A., Patel, B. and Bhatt, G.: October. Load balancing in cloud computing using optimization techniques: A study. In *Communication and Electronics Systems (ICCES), International Conference,* pp. 1-6, IEEE, 2016.

[48] Ardagna, D., Casale, G., Ciavotta, M., Pérez, J.F. and Wang, W.: Quality-of-service in cloud computing: modeling techniques and their applications. *Journal of Internet Services and Applications*, *5*(1), p.11, 2014.

[49] Haas, C., Caton, S., Chard, K. and Weinhardt, C.: January. Co-operative infrastructures: An economic model for providing infrastructures for social cloud computing. In *System Sciences (HICSS), 2013 46th Hawaii International Conference,* pp. 729-738, IEEE, 2013.

[50] Mohamaddiah, M.H., Abdullah, A., Subramaniam, S. and Hussin, M.: A survey on resource allocation and monitoring in cloud computing. *International Journal of Machine Learning and Computing*, *4*(1), p.31, 2014.

[51] Gouda, K.C., Radhika, T.V. and Akshatha, M.: Priority based resource allocation model for cloud computing. *International Journal of Science, Engineering and Technology Research*, *2*(1), pp.200-215, 2013.

[52] Zhang, L., Li, Z. and Wu, C.: April. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *INFOCOM, 2014 Proceedings IEEE,* pp. 433-441, IEEE, 2014.

[53] Al-Ayyoub, M., Jararweh, Y., Daraghmeh, M. and Althebyan, Q.: Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure. *Cluster Computing*, *18*(2), pp.919-932, 2015.

[54] Han, F.F., Peng, J.J., Zhang, W., Li, Q., Li, J.D., Jiang, Q.L. and Yuan, Q.: Virtual resource monitoring in cloud computing. *Journal of Shanghai University (English Edition)*, *15*(5), pp.381-385, 2011.

[55] Ramezani, F., Lu, J. and Hussain, F.K.: Task-based system load balancing in cloud computing using particle swarm optimization. *International journal of parallel programming*, *42*(5), p.739-750, 2014.

[56] LD, D.B. and Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing*, *13*(5), pp.2292-2303, 2013.

[57] Chaczko, Z., Mahadevan, V., Aslanzadeh, S. and Mcdermid, C.: September. Availability and load balancing in cloud computing. In *International Conference on Computer and Software Modeling, Singapore,* Vol(14), 2011.

[58] Milani, A.S. and Navimipour, N.J.: Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications*, *71*, pp.86-98, 2016.

[59] Mohanty, S., Patra, P.K. and Mohapatra, S.: Dynamic Task Assignment with Load Balancing in Cloud Platform. In *Emerging Research Surrounding Power Consumption and Performance Issues in Utility Computing*, pp. 363-385, 2016.

[60] Ergu, D., Kou, G., Peng, Y., Shi, Y. and Shi, Y.: The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. *The Journal of Supercomputing*, pp.1-14, 2013.

[61] Masdari, M., ValiKardan, S., Shahi, Z. and Azar, S.I.: Towards workflow scheduling in cloud computing: a comprehensive analysis. *Journal of Network and Computer Applications*, *66*, pp.64-82, 2016.

[62] Salot, P.: A survey of various scheduling algorithm in cloud computing environment. *International Journal of Research in Engineering and Technology*, *2*(2), pp.131-135, 2013.

[63] Dreshti P.H., Altaf B.M.: Improve Performance by Task Scheduling Beneficial to Both User and Cloud Provider in Cloud Computing. *International Journal of Computer Science and Mobile Computing*, 3(4), pp.1283-1288, 2014.

[64] Wu, L.: A revised discrete particle swarm optimization for cloud workflow scheduling, *Journal of Information and Communication Technologies, 18(3)*, pp. 1-5, 2012.

[65] Chen, C.L., Huang, S.Y., Tzeng, Y.R. and Chen, C.L.: A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem. *Soft Computing*, *18*(11), pp.2271-2282, 2014.

[66] Izakian, H., Ladani, B.T., Abraham, A. and Snasel, V.: A discrete particle swarm optimization approach for grid job scheduling. *International Journal of Innovative Computing, Information and Control*, *6*(9), pp.1-15, 2010.

[67] Wang, F.S. and Chen, L.H.: Particle swarm optimization (PSO). *Encyclopedia of Systems Biology*, pp.1649-1650, 2013.

[68] Engelbrecht, A.: July. Particle swarm optimization. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation* (pp. 381-406). ACM, 2014.

[69] Du, K.L. and Swamy, M.N.S.: Particle swarm optimization. In *Search and Optimization by Metaheuristics*, pp. 153-173, Springer International Publishing, 2016.

[70] Banerjee, S.,Mukherjee, I., and Mahanti., P.: Cloud computing initiative using modified ant colony framework, World academy of science, engineering and technology, pp. 200–203, 2009.

[71] Ünal, M., Ak, A., Topuz, V. and Erdal, H.: Ant Colony Optimization (ACO). In *Optimization of PID Controllers Using Ant Colony and Genetic Algorithms*, pp. 31-35, Springer, 2013.

[72] Gao, Y., Guan, H., Qi, Z., Hou, Y. and Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, *79*(8), pp.1230-1242, 2013.

[73] Ludwig, S.A. and Moallem, A.: Swarm intelligence approaches for grid load balancing. *Journal of Grid Computing*, *9*(3), pp.279-301, 2011.

[74] Yuce, B., Packianather, M.S., Mastrocinque, E., Pham, D.T. and Lambiase, A.: Honey bees inspired optimization method: the bees algorithm. *Insects*, *4*(4), pp.646-662, 2013.

[75] Zhao, S., Lu, X. and Li, X.: Quality of service-based particle swarm optimization scheduling in cloud computing. In *Proceedings of the 4th International Conference on Computer Engineering and Networks,* pp. 235-242, Springer, 2015.

[76] Karthick, A.V., Ramaraj, E. and Subramanian, R.G.: February. An efficient multi queue job scheduling for cloud computing. In *Computing and Communication Technologies (WCCCT), 2014 World Congress,* pp. 164-166, 2014.

[77] Abdullah, M. and Othman, M.: Simulated annealing approach to cost-based multi-quality of service job scheduling in cloud computing enviroment. *American Journal of Applied Sciences*, *11*(6), p.872-885, 2014.

[78] Xu, G., Pang, J. and Fu, X.: A load balancing model based on cloud partitioning for the public cloud. *Tsinghua Science and Technology*, *18*(1), pp.34-39, 2013.

[79] Deb, K.: Multi-objective optimization. In *Search methodologies,* Book, pp. 403-449, Springer US, 2014.

[80] Xue, S., Liu, F. and Xu, X.: An Improved Algorithm Based on NSGA-II for Cloud PDTs Scheduling. *Journal of Software*, *9*(2), pp.443-450, 2014.

[81] Bensmaine, A., Dahane, M. and Benyoucef, L.: A non-dominated sorting genetic algorithm based approach for optimal machines selection in reconfigurable manufacturing environment. *Computers & Industrial Engineering*, *66*(3), pp.519-524, 2013.

[82] Ruiz, R., Maroto, C. and Alcaraz, J.: Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, *34*(5), pp.461-476, 2006.

[83] Salimi, R., Motameni, H. and Omranpour, H.: Task scheduling using NSGA II with fuzzy adaptive operators for computational grids. *Journal of Parallel and Distributed Computing*, *74*(5), pp.2333-2350, 2014.

[84] Cheng, B.: Hierarchical cloud service workflow scheduling optimization schema using heuristic generic algorithm. *Przeglad Elektrotechniczny*, *88*(2), pp.92-95, 2012.

[85] Mirjalili, S., Mirjalili, S.M. and Lewis, A. Grey wolf optimizer. *Advances in Engineering Software*, *69*, pp.46-61, 2014.

[86] Mirjalili, S., Saremi, S., Mirjalili, S.M. and Coelho, L.D.S.: Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, *47*, pp.106-119, 2016.

[87] Guha, D., Roy, P.K. and Banerjee, S.: Load frequency control of large scale power system using quasi-oppositional grey wolf optimization algorithm. *Engineering Science and Technology, an International Journal*, *19*(4), pp.1693-1713, 2016.

[88] Zhang, S. and Zhou, Y.: Grey wolf optimizer based on Powell local optimization method for clustering analysis. *Discrete Dynamics in Nature and Society*, 14(3), pp. 47-61, 2015.

[89] Saremi, S., Mirjalili, S.Z. and Mirjalili, S.M.: Evolutionary population dynamics and grey wolf optimizer. *Neural Computing and Applications*, *26*(5), pp.1257-1263, 2015.

[90] Rao, R.V., Savsani, V.J. and Vakharia, D.P.: Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, *43*(3), pp.303-315, 2011.

[91] Gomathi, B., and Karthikeyan, K.: Task Scheduling algorithm Based on hybrid particle swarm optimization in cloud computing environment, *Journal of Theoretical and Applied Information Technology*, pp. 33-38, 2013.

[92] Pandey, S., Wu, L., Guru, S.M. and Buyya, R.: April. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference,* pp. 400-407, IEEE, 2010.

[93] Madni, S.H.H., Latiff, M.S.A. and Coulibaly, Y.: Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities. *Journal of Network and Computer Applications*, *68*, pp.173-200, 2016.

[94] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A. and Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *41*(1), pp.23-50, 2011.

[95] Xia, B. and Tan, Z.: Tighter bounds of the First Fit algorithm for the bin-packing problem. *Discrete Applied Mathematics*, *158*(15), pp.1668-1675, 2010.

[96] Coffman Jr, E.G., Csirik, J., Galambos, G., Martello, S. and Vigo, D.: Bin packing approximation algorithms: survey and classification. In *Handbook of Combinatorial Optimization*, pp. 455-531, Springer New York 2013.

[97] Liu, Z., Qu, W., Liu, W., Li, Z. and Xu, Y.: Resource preprocessing and optimal task scheduling in cloud computing environments. *Concurrency and Computation: Practice and Experience*, *27*(13), pp.3461-3482, 2015.

[98] Tripathy, L. and Patra, R.R.: Scheduling in cloud computing. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, *4*(5), pp.21-37, 2014.

[99] Neri, F.: An Introduction to Computational Complexity. In *Linear Algebra for Computational Sciences and Engineering* (pp. 349-362). Springer, 2016.

[100] Tomita, E., Tanaka, A. and Takahashi, H.: The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, *36*(1), pp.28-42, 2006.

[101] Hoffman, K.L., Padberg, M. and Rinaldi, G.: Traveling salesman problem. In *Encyclopedia of operations research and management science*, pp. 1573-1578, Springer US, 2013.

[102] Barto, L. and Kozik, M.: Robustly solvable constraint satisfaction problems. *SIAM Journal on Computing*, *45*(4), pp.1646-1669, 2016.

[103] Simon, D.: Biogeography-based optimization. *IEEE transactions on evolutionary computation*, *12*(6), pp.702-713, 2008.

[104] Jamil, M. and Yang, X.S.: A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, *4*(2), pp.150-194, 2013.

[105] Kyrö, P.: Benchmarking as an action research process. *Benchmarking: An International Journal*, *11*(1), pp.52-73, 2004.

[106] Falkenauer, E.: Genetic algorithms and grouping problems. *John Wiley & Sons, Inc*, 1998.

[107] Quiroz-Castellanos, M., Cruz-Reyes, L., Torres-Jimenez, J., Gómez, C., Huacuja, H.J.F. and Alvim, A.C.: A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research*, *55*, pp.52-64, 2015.

[108] Alvim, A.C., Ribeiro, C.C., Glover, F. and Aloise, D.J.: A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, *10*(2), pp.205-229, 2004.

Publications

List of Related publications

1. **Mousavi. SM**., Mosavi. A., Varkonyi-Koczy, A.R., Fazekas., G.: Dynamic resource allocation in Cloud Computing. *Journal Acta Polytechnica Hungarica,* 14(3), pp. 80-101, 2017.
2. **Mousavi. SM**., Fazekas.G.: A Novel Algorithm for Load Balancing using HBA and ACO in Cloud Computing Environment. *International Journal of Computer Science and Information Security,* 14(6), pp.48-52, 2016.
3. **Mousavi. SM**.,Fazekas. G.: Increasing QoS in SaaS for low Internet speed connections in cloud. *The 9th International Conference on Applied Informatics, Eger, Hungary,* pp. 195-200, 2014.
4. **Mousavi. SM**. Fazekas.G.: Dynamic resource allocation in cloud computing: A survey and taxonomy*", Journal of Engineering and Applied Sciences,* Accepted (Mar 2017), in press.
5. **Mousavi. SM**. et al: A load balancing algorithm for resource allocation in cloud computing. *Advances in Intelligent Systems and Computing, Recent Global Research and Education: Technological Challenges,* Springer International Publishing 2017, Accepted, in press.
6. **Mousavi. SM**., Fazekas. G.: Comparison of efficiency of meta-heuristic algorithms to solve optimization problems in cloud resource allocation. *Journal of computers*, (Submitted).

Further publications

7. **Mousavi. SM.**: A new architecture for Iran's communication Infrastructures for national Cloud Computing. *International conference in Electrical and Computer Engineering, Tehran, Iran, Aug 2015,* http://cbconf.ir/en/.

8. **Mousavi. SM.**, Fazekas G.: Cloud Computing Auditing Roadmap and Process. *Journal of Engineering and Applied Sciences,* Accepted (March 2017), in press.

9. Farzaneh. Y., **Mousavi. SM**., Mousavi. A., Azodinia. M.: Improving Client Access License for Apache Hadoop Application. *Journal of computers,* (Submitted).

Appendix A

OpenStack

OpenStack Nova is the OpenStack compute project. It is a compute controller that pools computing resources like CPU, memory, etc... Nova provides API's to control on-demand scheduling of compute instances like virtual machines on multiple virtualization technologies, bare metal, or container technologies. Nova uses images to launch instances or VMs. In this chapter, we provide an explanation of the steps to create an instance with Nova in order to implement the proposed algorithm on OpenStack Nova as future work.

1. Create a simple credential file:

   vi creds
   # Paste the following :
   export OS_TENANT_NAME = admin
   export OS_USERNAME = admin
   export OS_PASSWORD = admin_pass
   export OS_AUTH_URL =" http ://192.168.100.11:5000/ v2 .0/"

2. Upload the cirros cloud image:

   source creds
   glance image - create --name "cirros -0.3.2 - x86_64 " --is - public true \
   --container - format bare --disk - format qcow2 \
   --location http :// cdn . download . cirros - cloud .net /0.3.2/ cirros -0.3.2 - x86_64 - disk . img

3. List Images:

   glance image – list

4. Create an external network:

   source creds
   # Create the external network :

neutron net - create ext - net -- shared -- router : external = True
Create the subnet for the external network :
neutron subnet - create ext - net --name ext - subnet \
--allocation - pool start =192.168.100.101 , end =192.168.100.200 \
--disable - dhcp -- gateway 192.168.100.1 192.168.100.0/24

5. Create an internal (tenant) network:

source creds
Create the internal network :
neutron net - create int - net
Create the subnet for the internal network :
neutron subnet - create int - net --name int - subnet \
--dns - nameserver 8.8.8.8 -- gateway 172.16.1.1 172.16.1.0/24

6. Create a router on the internal network and attach it to the external network:

source creds
Create the router :
neutron router - create router1
Attach the router to the internal subnet :
neutron router - interface -add router1 int - subnet
Attach the router to the external network by setting it as the gateway :
neutron router - gateway - set router1 ext – net

7. Generate a key pair:

ssh – keygen

8. Add the public key:

source creds
nova keypair - add --pub - key ~/. ssh / id_rsa .pub key1

9. Verify the public key is added:

nova keypair – list

10. Add rules to the default security group to access your instance remotely:

Permit ICMP (ping):
nova secgroup -add - rule default icmp -1 -1 0.0.0.0/0
Permit secure shell (SSH) access :
nova secgroup -add - rule default tcp 22 22 0.0.0.0/0

11. Launch your instance:

```
NET_ID =$( neutron net - list | awk '/ int -net / { print $2 }')
nova boot -- flavor m1. tiny --image cirros -0.3.2 - x86_64 --nic net -id= $NET_ID \
--security - group default --key - name key1 instance1
```

12. Note: To choose the instance parameters these commands could be used:

```
nova flavor - list : -- flavor m1. tiny
nova image - list : --image cirros -0.3.2 - x86_64
neutron net - list : --nic net -id= $NET_ID
nova secgroup - list : --security - group default
nova keypair - list : --key - name key1
```

13. Check the status of your instance:

```
nova list
```

Appendix B

All implemented algorithms and used dataset in this thesis are downloadable using the following link:

https://drive.google.com/file/d/0B_YXoYhgc4BqZ2tJY1Jkc2MwZlU/view?usp=sharing