Design and Analysis of an Adjustable and Configurable Bio-inspired Heuristic Scheduling Technique for Cloud Based Systems

by

Ali Al Buhussain

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Cloud computing environments mainly focus on the delivery of resources, platforms, and infrastructure as services to users over the Internet. More specifically, Cloud promises user access to a scalable amount of resources, making use of the elasticity on the provisioning of recourses by scaling them up and down depending on the demand. The cloud technology has gained popularity in recent years as the next big step in the IT industry. The number of users of Cloud services has been increasing steadily, so the need for efficient task scheduling is crucial for improving and maintaining performance. Moreover, those users have different SLAs that imposes different demands on the cloud system. In this particular case, a scheduler is responsible for assigning tasks to virtual machines in an effective and efficient matter to meet with the QoS promised to users. The scheduler needs to adapt to changes in the cloud environment along with defined demand requirements. Hence, an Adjustable and Configurable bio-inspired scheduling heuristic for cloud based systems (ACBH) is suggested. We also present an extensively comparative performance study on bio-inspired scheduling algorithms namely Ant Colony Optimization (ACO) and Honey Bee Optimization (HBO). Furthermore, a networking scheduling algorithm is also evaluated, which comprises Random Biased Sampling (RBS). The study of bio-inspired techniques concluded that all the bio-inspired algorithms follow the same flow that was later used in the development of (ACBH). The experimental results have shown that ACBH has a 90% better execution time that it closest rival which is ACO. ACBH has a better performance in terms of the fairness between execution time differences between tasks. HBO shows better scheduling when the objective consists mainly of costs. However, when there is multiple optimization objectives ACBH performs the best due to its configurability and adaptability.

Acknowledgements

I am very thankful to Allah for blessing me with the chance to pursue an important step in my life. Also, I am very grateful to my supervisor professor Azzedine Boukerche for his guidance, his advice, and encouragement during my Master's journey. I would like to extend my gratitude to Dr. Robson Eduardo De Grande for his advice and his kind approach throughout my research. For the PARADISE lab members, thank you all for attending all my presentations and advising me about my work. My Father and mother back home, words cannot begin to express how thankful I am to you both for teaching the importance of hard work and for trusting and believing in my abilities. Last but not least, to my lovely wife Sara, thank you for your patience throughout this journey, thank you for your constant help and encouragement. Finally, I would like to thank the Saudi Cultural bureau for funding my research and making my dream a reality.

Glossary

VM Virtual Machines

Cloudlet Cloud Tasks

QoS Quality of Service

SaaS Software as a Service

PaaS Platform as a Service

IaaS Infrastructure as a Service

PBSH Population Based Scheduling Heuristics

ACO Ant Colony Optimization

HBO Honey Bee Optimization

RBS Random Biased Sampling

NID Node In Degree

WIL Walk In Length

MOT Multi-Objective Theory

ACBH Adjustable and Configurable Bio-inspired Scheduling Heuristic

$\mathbf{W}_c$ ACBH Cost Factor Weight

$\mathbf{W}_{cp}$ ACBH Computation Factor Weight

$\mathbf{W}_{ld}$ ACBH Load Factor Weight

ACBH$_{ALL}$ ACBH with All Factors considered

ACBH$_{CC}$ ACBH with Cost and Computation Factors Considered

ACBH$_{CL}$ ACBH with Cost and Load Factors Considered

ACBH$_{COMP}$ ACBH with Only Computation Factor Considered

ACBH$_{LOAD}$ ACBH with Only Load Factor Considered

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces the purpose of my research and the contributions that this thesis are adding to the scheduling in the cloud computing field. A summery of the objective that my thesis are satisfying is provided. The outline of the thesis is shown at the end of this chapter.

1.1 Thesis statement and motivation

Cloud computing most important aspect is the elasticity of resources provided to users. This Elasticity can only be possible if the cloud has a scheduling system that can adapt to the changes imposed on demand for the cloud resources by the user. Cloud Computing is seen by many nowadays as the future of the IT industry. Cloud environments as any computing environment suffer from drawbacks that held it from reaching its maximum potential. Therefore, those drawbacks must be first identified and then resolved in accordance with the certain requirements provided by either users', providers, or both. One of those drawbacks is scheduling the use of resources (physical or virtual). Managing the cloud resources (i.e. scheduling) plays a crucial role in helping the cloud to effectively

and efficiently utilize those resources and hence, scheduling helps the cloud reach its maximum potential.

1.2 Thesis Objectives

The objectives of this thesis can be summarized as follows:

- To contextualize the scheduling of cloud resources. This aids in finding or suggesting new or modified algorithms to solve the cloud scheduling problem.
- To perform an extensive analysis on bio-inspired algorithms adapted in the literature to be able to compare them against each other. The study will aid in spotting drawbacks of those algorithms and hence aids in enhancing or inventing new ways to approach the scheduling problem of cloud resources.
- To develop an enhanced bio-inspired scheduler that is able to overcome the rigidity of the traditional bio-inspired techniques used in the related work.

1.3 Contributions

The main contribution of this thesis is the design and implementation of an adjustable and configurable bio-inspired Based Scheduling Algorithms (ACBH). ACBH is able to adapt to the QoS preferences of the user and schedule based on the current demands on the cloud virtual machines. Moreover, ACBH is elastic enough to work as a stand-alone scheduler or as a meta-heuristic. This makes ACBH a centralized scheduler or partially distributed if chosen to work as a meta-heuristic. Also, an extensive performance analysis of bio-inspired scheduling heuristics and suggesting future directions that can potentially be perused.

1.4 Thesis Outline

This thesis is structured as follows. Section 2 introduces the cloud environment, it's characteristics, and various service modules. Section 3 provides a brief description of the related works. Section 4 explains the suggested ACBH in details in terms of architecture and functionality. Section 5.1 provides details of the architecture and functioning of the HoneyBee Optimization algorithm. Section 5.2 thoroughly describes AntColony algorithm. Section 5.3 presents Random Biased Sampling algorithm. Section 6.1 introduces the experimental scenario and discusses the results obtained. Finally, Section 7.1 concludes the paper and provides directions for the future work.

Chapter 2

Cloud Computing

This chapter introduces the necessary characteristics of the cloud environment. Also, different service layers of the cloud are presented and discussed. These layers are relevant to the research topic as can be seen in Chapter 3. Furthermore, the importance of the scheduling in the cloud is presented.

Cloud computing provides an environment where the needs for hardware and software is minimized through flexible, agile provisioning. The Cloud enables the access to its services through the network or an on-line simple interface. NIST defined the cloud as a model for providing a set of ubiquitous, convenient, on-demand network access to a shared pool of configurable resources over the Internet [77]. Those resources should be provisioned and managed by the provider of the cloud services. The cloud computing environment is seen as the next step in the IT industry. As a result, Cloud computing has been given a lot of attention and effort by many large IT corporations, such as Amazon EC2, Microsoft Azure, and Google Cloud Platform. According to [77] any cloud environment has the following essential characteristics:

- On-demand self-service: The Cloud user must have control and ability to fetch the

resources needed (e.g. server time, storage) without any interaction with the cloud service provider.

- Broad network access: Users of the cloud services should be able to access the capabilities of the cloud through the Internet using any thin or thick client (smartphones, tablets, laptops, and workstations).

- Resource Pooling: The cloud service provider must provide the requested physical and virtual resources by the cloud users according to their demand. Those resources must be available at all times regardless of the location of the cloud user.

- Rapid Elasticity: The resources (physical and virtual) must be Elastically provisioned either manually or automatically to match the current demand of the cloud user. This capability must be available at all times.

- Measured Service: Cloud service provider must have a monitoring tool that is able to monitor, control, and report resources usage. This monitor provides transparency for both the cloud user and cloud service provider.

2.1 Cloud Types and Service Models

The cloud environment has three main types. The first type is the public cloud in which the access of its resources is open to everyone interested in subscribing to the services provided. For example, Amazon EC2, Microsoft Azure. Private clouds are similar to the public cloud but all of its capabilities are dedicated to a single organization. The hybrid as the name suggests it falls in between the public and the private cloud [82, 77, 55].

Cloud environment offers three main service models. The first service model offered is Software as a service (SaaS). The SaaS layer is the top most layer in which the cloud

providers use to provide software services to cloud users [82, 77, 55]. The users can access those services remotely. SaaS enables the users to access to software cheaper than buying the license and worrying about all the updates. Furthermore, remote hosting of those software means that the organizations using SaaS will also save on infrastructure (i.e. Hardware). The second service model is a platform as a service (PaaS). In PaaS as the name suggest provides cloud users with a platform to host their software. This model offers programming languages, libraries, and services and tools required to the cloud service consumers applications [82, 77, 55]. Finally, the infrastructure as a service (IaaS) is the last model offered in the cloud. In IaaS, the cloud provides complete infrastructure to deploy and run users applications. The cloud providers offer a range of resources such as datacenters, virtual machines, network, and storage [82, 77, 55]. Furthermore, the management of those resources in terms of scheduling, routing, and integration is offered by the cloud providers. All of the mentioned cloud service models as based on a pay-as-you-go model. In other words, users will pay for the resource they use.

2.2 Importance of Scheduling to Cloud Computing

Cloud computing services aim to provide resources for high availability. Those resources are scalable depending on the need of users. Due to its features, flexibility, and scale, a Cloud is a highly complex distributed environment, and all of its characteristics imposes tremendous challenges on allowing centralized governance. Therefore, there is an increasing need to identify distributed solutions that are able to govern the Cloud environment through local knowledge. A distributed governance system is expected to be self-organized and able to manage itself [90].

In the Cloud, the demands for resources change dynamically, and a Cloud provider is expected to be able to accommodate and react to these changes on the demand to

meet performance requirements, which involves time constants for real-time applications, money, which corresponds mostly to re-sizing resources according to dynamic application demands, and SLA agreement. Virtualization of physical resources provides the necessary dynamics to manage the resources in the cloud platform [62]. Therefore, virtualization plays a major role in the effective use of physical resources in the Cloud to meet the SLA agreement. Consequently, scheduling the assigned tasks to the virtual machines directly impacts on the performance of the Cloud and aids in balancing the distribution of load among the different physical servers. Thus, finding an effective scheduling scheme is very crucial.

Several scheduling algorithms have been proposed for allocating resources in distributed systems. However, the context of resource management systems for Cloud computing restricts the use of such algorithms to a narrower class of solutions. Bio-inspired algorithms, more specifically swarm or gang scheduling algorithms, have shown very suitable for Cloud environments as per results presented in some works. However, the analysis performed in previous have restricted themselves to limited scenarios.

Chapter 3

Related Work

This chapter presents an extensive survey of the existing cloud scheduling solutions that are implemented in each layer of the cloud environment. In this chapter, the existing solutions are categorized based on the layer of the cloud in which they operate. Moreover, a brief discussion of the the related work is added. A table summarizing all the techniques used in the literature is available as well.

3.1 Introduction to Scheduling in Cloud Environments

According to [117]. [58] there are three scheduling levels on the cloud environments. At the application layer (SaaS), virtualization layer (PaaS), and deployment layer (IaaS). The application layer scheduling is needed to map user tasks to the virtual machines. In the virtualization layer, virtual machines are deployed in the datacenters to achieve certain objectives such as load balancing and power saving. In terms of the deployment layer, a scheduler is needed to schedule services and data routing.

The cloud scheduling of task is an NP-Hard problem due to the size and the heterogeneity of the system. It is very important to find solutions to the scheduling problem in the cloud to meet the SLAs. In other words, for the cloud to be elastic, profitable, energy efficient, cost effective, available, and balanced in terms of the load; a good scheduler or group of schedulers must be used to achieve the cloud maximum potentials.

To formalize the scheduling problem in the cloud as illustrated in 3.2, we want to assign $task_i$ to the best fitted VM_j that is deployed to the best suited according to certain criteria PM_k. The scheduling in the cloud is a multi-objective problem in which we are trying to find the optimal mapping of Tasks T = T1,T2,,Tn to VMs (V = V1,V2,,Vm). Then VMs are deployed in Datacenters D = D1,D2,,Dk to achieve (Maximize or Minimize) a goal set we have G = G1,G2,,Gs. In other words, we are allocating resources to achieve a desired level of cost, energy level, time constraint, scalability, and so on.

The existing algorithms are mainly focusing on one section of the scheduling problem to find the best solution to that particular section. In other words, the developed schedulers are focusing on mainly solving the SaaS, PaaS, or IaaS scheduling problems separately. However, to have a more effective scheduler, we need to have interaction between the cloud layers to achieve better results.

3.1.1 Scheduling Objectives

Cloud scheduling objectives will differ from one layer to the next in the cloud and also will depend on the SLA between the user and provider. This variation in demands on the cloud require the scheduler to be adaptable. In order for the scheduler to be adaptable, it has to be self-managed, self-organized and distributed [91]. However, there exist common objectives to cloud schedulers in which those schedulers are developed to satisfy a selected few depending on the set of goals agreed upon in the SLA [117][58].

The scheduling objectives can be summarized as follow:

- Performance: Computing resources utilization measurements.
  - Response Time: The amount of time to finish scheduling tasks.
  - Throughput: A number of jobs finished per unit time.
  - Makespan: The total execution time of the tasks.
  - Reliability: trustworthiness of the algorithm.
  - Bandwidth: The amount of bandwidth used to process tasks.
  - Load balance: The amount of work that is assigned to the computing entity (Virtual Machine or Datacenter).
- Cost: The amount of money the will be consumed to execute the given tasks.
  - Cost of Operation (Provider): How much is that task in costing the cloud provider to execute it.
  - Price of use (user): How much does the cloud user will pay for that task to be executed.
- Eco: environment related measurements.
  - Energy consumption: The amount of energy consumed to execute users' tasks.
  - CO2 Emission: The amount of CO2 emissions from datacenters while executing uses' tasks.
- Others:
  - Security.

3.1.2 Resource Scheduling in the cloud

There have been some novice works to try and solve the scheduling problem in the cloud. Those initial work required the exhaustive search. Those works can be applied in a limited size environment, as they require a long time to make scheduling decision that will hinder the cloud capabilities in terms of availability and scalability. Others try to adopt the transitional (grid computing) scheduling schemes to the cloud but still those schedulers can achieve a single goal at a time.

At the beginning, there was some straightforward approach where developed like [91] suggested in amazon EC2 a rule-based scheduler was used. As the cloud grew and more resources were added, also did the rule set. This rule set was huge and taken long time to evaluate that hinders the cloud performance. Also, the rules have intersections in which the execution of one rule might cause another to trigger resulting in a cascade of rules. Thus, this solution is time consuming, centralized, not elastic to be implemented to a huge and heterogeneous cloud environment.Other approaches as seen in [43]. This work used developed an improved version of the Max-Min algorithm for task scheduling. The mechanism depended on not finding the maximum execution and assigned it to the least loaded computation node but also achieved lower make-span. In the work of [99] introduced a priority-based scheduling algorithm. The proposed scheduler evaluates a priority based on the cost of the execution of the task among other things. Then the tasks are divided into three groups of high, moderate, and low priority. The scheduler achieved task execution with lower cost. Furthermore, first come first serve (FIFO) is also was used to approach the scheduling problem and also used as a base test to compare purposed solutions as in [52].

Moreover, some other networking algorithms are also used to schedule the cloud tasks. Those heuristics were developed to challenge the scheduling problem in the cloud. Some

of these heuristics are driven from networking problems (e.g. Random Biased Sampling, Active Clustering). Those algorithms found some success in solving the scheduling problem but the scalability and randomness make them unpredictable. Also, the number of scheduling objective achieved is limited [91].

Based on new constraints, scheduling in clouds is faced as an NP-Hard problem [54, 115, 117, 58], requiring self-regulations for balancing load on entities. This self-organization is possible with the implementation of distributed algorithms that rely on local knowledge. In recent years, nature has influenced many works on seeking out solutions to the increasing scale and complexity of the Cloud system [6]. There are two general directions that are typically followed to solve this issue. The most powerful heuristics used to solve NP-Hard problems are population-based algorithms, or evolutionary optimization, such as fuzzy and neural controllers. Due to its complexity, the scheduling in the cloud is best tackled by the previously mentioned heuristics [93, 100], including genetic algorithms (GA), Particle Swarm optimization (PSO), Ant Colony optimization (ACO), and Honey Bee optimization (HBO).

There are major population heuristics approaches to schedule task in the cloud different layers [117, 58]. Genetic algorithms (GA) are a common approach that is being used recently as the survey illustrates later. 3.1 shows the general follow diagram of any GA. Moreover, Particle Swarm optimization (PSO) is also used to schedule task in the cloud. 3.1 explains the general step taken in PSO algorithms to make scheduling decisions. Ant Colony optimization (ACO) is heavily researched and is well defined. Hence, ACO is used as a scheduler in the cloud environments because it is easily extended to accommodate scheduling objectives as seen in the works bellow. Furthermore, Honey Bee optimization (HBO) is a viable scheduler that has been successfully used recently.

Based on the importance of the population based schedulers on the cloud environ-

ments, it is interesting to go over how these algorithms work, some objectives, and characteristics. In the next two sections, the algorithms will be explained and other schedulers used in the cloud will be mentioned. In addition, a table summing the objective, simulation used, test size, and other objectives of the schedulers.

3.2 Population Based Scheduling Heuristics (PBSH)

3.2.1 Overview

The PBSH follow the same step in their evolutionary process in order to make a decision. Those Algorithms have the following steps: I. Fitness evaluation. II. Candidate selection. III. Trial variation.

During the Fitness evaluation, the algorithm finds the best sources that it can exploit for benefits are usually defined in terms of the optimization objectives of the scheduling algorithms such as, cost and deadline. After that step is done, we will have a set of candidates solution that we can choose from to achieve the scheduling objective(s). Those candidates have different things to offer and some will help reach the scheduling objectives faster or cheaper than others. Thus, we have to choose the best out of the candidates to maximize the chances of reaching the scheduling objectives. Finally, trial variation is choosing when the scheduling algorithm stops and thus giving the best scheduling solution possible.

The GA, PSO, and ACO are non-deterministic in nature and require no prior guidance. Therefore, those schedulers represent a good for a complex and multi-objective environment like the cloud environment [117, 58]; however, in [117] there was no mention of one of the most studied and well-developed optimization technique that is HBO. The next section will have a brief introduction on the most common PBSH.

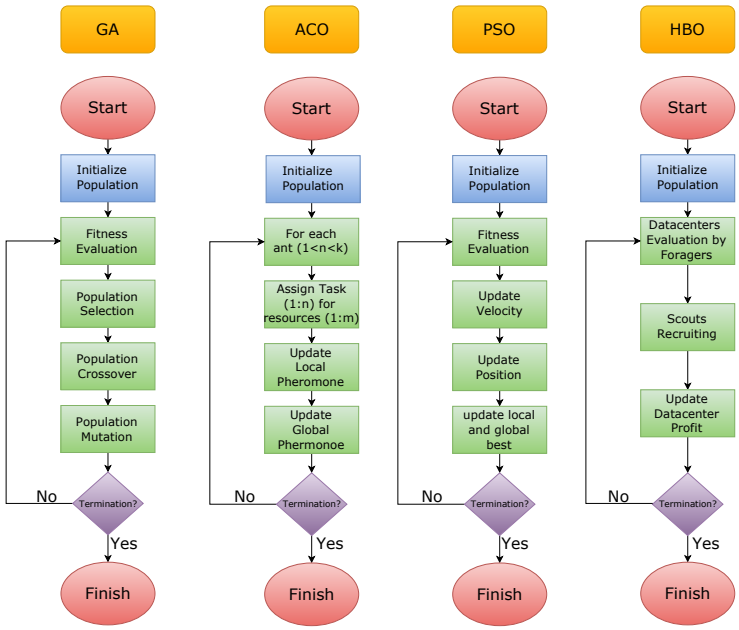

Figure 3.1: PBSH.

3.2.2 Most Common PBSH

(a) *Genetic Algorithm (GA):* In any genetic algorithms, any cloudlet (task) is considered to be a gene. Each task is coupled with a resource to form a chromosome. They construct a set of possible solutions to be chosen in the next steps of the GA. This step is done randomly to have an initial population of chromosomes. Then, GA starts the optimization process. An evaluation of the constructed chromosomes is done by evaluation their fitness. This evaluation depends on the objectives of the GA desired. Then, the chromosomes with the best fitness value are chosen to survive as they represent the best solutions thus far. After that, the set of best solutions is then crossed over and mutated to generate the new candidates for the next iteration. The GA evolutionary cycle will continue until a termination condition is reached (i.e. number of iterations set by the user).

(b) *Particle Swarm Optimization (PSO):* PSO works as GA when initializing the pop-

ulation as it creates a particle. It contains a pair made with a task and a value. This value represents the virtual machine id the particle will run on. The optimization in PSO happens when evaluating the velocity and position of the population pairs. The velocity and position updates guide the particle to fly towards the target. PSO terminates when reaching a certain number of iterations or the solution cannot be enhanced [68].

(c)*Ant Colony Optimization (ACO):* ACO initialization of the population means creating the desired number of ants then placing those ants randomly on the resources available. In this initial step, ants assign tasks to the resources they forage to randomly. Each ant will update the pheromone value by a constant value. Then each ant will choose the next resource for the task according to the pheromone and heuristic value. The local and global pheromone values are updated in every iteration. ACO stops when the maximum number of iterations that is specified by the user is reached or the ants completed their tours.

(d)*Honey Bee Optimization (HBO):* In HBO, the population is split into two types foragers and scouts. At the start, the foragers scour for resources that are most fitted to perform the task. Then the foragers will recruit scouts to use the most fitted source to execute the tasks.

3.3 Scheduling In Different Cloud Layers

The scheduling in the SaaS layer as seen in 3.2 means assign the task to the virtual machines to satisfy the scheduling objectives. Moreover, in the PaaS layer, scheduling means finding the best datacenter to deploy a virtual machine on. The deployment is done in a way to achieve the scheduling goals. Finally in the SaaS layer, the cloud service provider must have datacenters deployed in many geographical locations. Those datacenters must be managed effectively in terms of energy, load balancing, and cost.

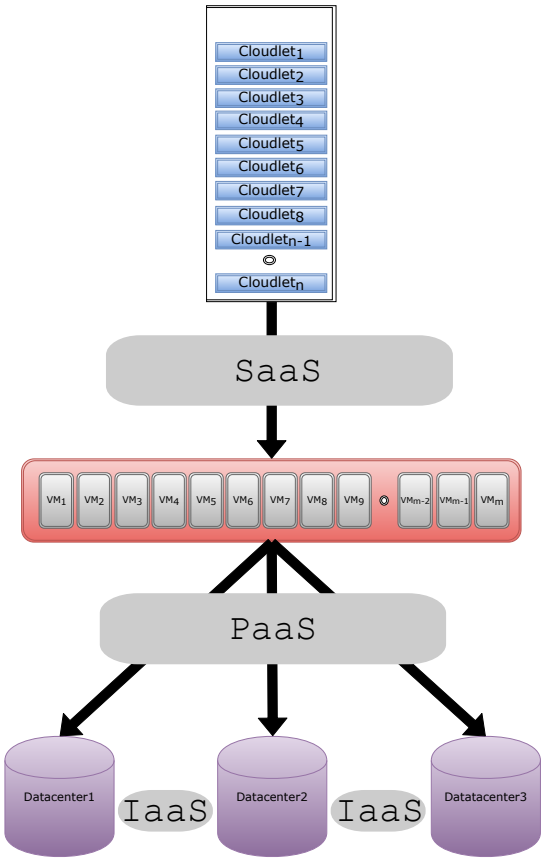

Figure 3.2: Cloud Service Models.

3.3.1 Scheduling in the SaaS Layer

According to [117, 58] the objectives of scheduling in this layer is mainly focused on performance (e.g. makespan) and cost of running the application. Cloud service providers also aim to schedule the resources efficiently. In other words, the schedulers not only satisfies the QoS for the user but also take into account maximizing the providers objectives such as saving carbon cost or energy consumed by running virtual machines on datacenters [36, 67]. In this layer, the scheduler is responsible for the assignment of tasks to virtual machines according to the scheduling objectives specified by the user and provider.

Scheduling in SaaS Using GA

The use of GA was suggested by [120] in 2009. The works suggest that GA was used and being successful in approaching the NP-Hard problem and thus cloud be used in scheduling the cloud resources. The GA used was simple as its encoding for chromosomes was straightforward. The main objective of this algorithm was to meet deadlines.

In [64] the encoding technique is similar to the work in [120] but the objectives considered are not only deadlines but also processing cost, throughput, and VMs utilization. The test performance was good; however, it was only compared to other simple schedulers such as round robin. [56] used similar encoding as in [120] combined with simulated annealing. The scheduler considered makespan, bandwidth, cost, distance, and reliability in making decisions. Some other works cluster several objectives such as the one in [76], which has been developed using a multi-objective scheduling algorithm. Reputation has also been used to introduce memory in the scheduling, such as work in [87]. Grouping based on memory consumption and computation requirements of tasks as in [120]. A GA scheduler that scans the entire job queue to make scheduling decision based was purposed in [53]. It aims to minimize the makespan of the tasks only. Thus, distributed, self-organized, and self-managed algorithms are needed to solve the cloud-scheduling problem. [69] combined the GA scheduler with Min-Min and Max-Min to enhance the initial population and hence a better scheduling solution can be found. Makespan was the focus of the previous algorithm. In [40] GA was scheduler goes through two optimization stages. At first, the deadline is used as the optimizer to find a feasible solution(s). After that, the set of feasible solutions is optimized further by using the cost.

There are other GA algorithms that are designed to optimize different objectives. Load balancing, energy consumption, utilization, and SLA are among those objectives. These extend the previous objectives that focus on QoS. Using load balancing as the

main optimization in GA as in [123], a multi-agent GA (MAGA) was purposed. The authors reduced the search space (number of tasks) by grouping tasks based on certain criteria. Also, a binary code is used to encode the chromosomes, not just an integer value.

Scheduling in SaaS Using ACO

The Ant Colony Optimization (ACO) uses the behavior of real ants in foraging for food to implement a solution for the cloud task scheduling. The ants leave the nest to search for food sources (VMs) in random. Then they evaluate the quality of the food source and carry it back to its nest. The ants leave a chemical trail on the ground. The strength of that chemical trail depends on the quality of the food source found [44]. Researchers used ACO to solve NP-hard problems such as traveling salesman problem, graph coloring problem, vehicle routing problem, and scheduling problem. In the context of Cloud computing, ACO is used to find the optimal way to schedule tasks to VMs [44, 75]. ACO differs from in the implementation of the transition rule. This rule has the heuristics information and pheromone update factors [44]. In [5] added to the transition rule another factor called execution matrix. It contains the expected execution time of task Ti on resource Rj. Both [75, 104] defined the ACO in terms of cloud entities. The heuristics were defined as the expected execution and transfer time for any given task. The pheromone update depends on the best current solution found in the previous iteration. Classification of tasks based on QoS parameters was used in the work of [124]. Tasks were grouped on the passes of bandwidth, completion time, system reliability, and cost. After that, ACO is used to schedule subgroups of the tasks on the cloud resources.

Others modified ACO to detect load imbalances between nodes as in works of [83]. The ants will start at the nodes that have many neighboring nodes. As the ants travel

they will detect the heavy loaded nodes and redistribute some of the load to the lightly loaded nodes. This process is two steps. The first step is to leave trail indicating how loaded is the node. The other step is to leave a trail to lightly loaded nodes. The work of [72] implemented ACO as in the [75, 104] but added a factor in the transition function to take into account the number of tasks assigned to a certain virtual machine to balance the load. In other words, this factor is added to ensure fairness when distributing tasks to virtual machines. ACO can also be merged with other algorithms such as PSO as in [111]. At first, ACO finds the best candidates (i.e. virtual machines) and then PSO performs crossover to avoid prematurity.

Scheduling in SaaS Using PSO

particle swarm optimization (PSO) is simple and fast as its only uses only two encodings. The first is the velocity and the second is the position. PSO is the fastest algorithm to converge when compared to GA and ACO [117]. The encoding in the work of [85] was similar to the GA in which that each task is associated with an integer holding the resource id. The PSO takes into account only the cost, both data transmission and computational, as optimization objective. This work is similar to [113] in terms of the optimization objective but they differ in form [85] as the PSO is discrete not continuous [117]. The optimization factors were then further extended in [39] to include the makespan, cost, reliability. The only problem with [39] is that there is no dependency between the optimization factors. The work done by [94] is encoding the scheduling by providing a rounded integer specifying the index of the resource assign to each task. The optimization objective was to reduce cost and meet deadlines. The index does not represent enough guidance or does not show the real features of the resources chosen [117]. Hence, [71] purposed to redefine what the index of resources should mean in PSO

by using a renumbering based on the price of the resource and the ability to reorder. Also [70] extended the [71] by using a include many other scheduling objectives.

Scheduling in SaaS Using HBO

Honey bee optimization algorithm, like the PSO, is simple. It is also fast to converge since its only has one encoding scheme. It chooses the most profitable source to utilize. The optimization can be extended to hold many scheduling objectives such as cost, computation time, load balancing, and so on [97]. In the work of [91], the authors used HBO to balance the loads in the physical servers. Others like [80] used cost as an optimization factor to derive the profit of the HBO. Although HBO is not researched as much as GA, ACO, and PSO, It represents a valid optimization solution to the cloud scheduling as HBO can be extended to accommodate many optimization objectives and is also fast to converge. HBO has been found to have the ability to solve a set of multi-objective optimization problems [86, 106]. Compared with the optimization algorithms GA, PSO, and ACO, the performance of HBO has shown a great competitiveness [116]. Other HBO like [57] built a fault tolerant a load-aware algorithm. In this HBO, the authors consider fault and load awareness to be essential for the cloud performance.

The table bellow summarizes the approaches used in the SaaS layers and shows that those approaches lack distribution and adaptability in comparison with ACBH.

Type	PBSH	Objectives	Distribution	Configurable
GA	Deadline-GA [64]	Deadline	NO	NO
	MGA [120]	Deadline Processing Cost Throughput VMs utilization	NO	NO
	Makespan-GA [53]	Makespan	NO	NO
	MinMax-GA [69]	Makespan	Partially Dis	NO
	Rep-GA [87]	Memory Makespan	NO	NO
	LocSearch-GA [56]	Makespan Bandwidth Cost reliability	Partially Dis	NO
	Load-GA [123]	Load Balance (CPU and Memory)	Partially Dis	NO
ACO	ACO [5]	Makespan	NO	NO
	ACO [75]	Makespan	NO	NO
	ACO [104]	Makespan Fairness	NO	NO
	Task-ACO [124]	Bandwidth Completion time Reliability Cost	Partially Dis	NO
	LoadBalance-ACO [83]	Load balance (VMs)	NO	NO
	PSO-ACO [111]	Maximize VMs utilization	Partially Dis	NO
PSO	MPSO[85]	Cost Bandwidth Makespan	NO	NO
	MPSO [113]	Cost Bandwidth Makespan	NO	NO
	MCR-PSO [39]	Makespan Cost Reliability	NO	NO
	RU-PSO[94]	Cost Deadline	NO	NO
	ReNum-PSO [70]	Cost	NO	NO
	ReNum-PSO [71]	Cost	Partially Dis	NO
HBO	Load-HBO[91]	Load Balance	NO	NO
	Profit-HBO [80]	Cost	NO	NO
	Fault-HBO [57]	Fault Load Balance	NO	NO
ACBH	Flexible-ACBH	Cost Bandwidth Makespan Fairness Memory Load	Partially Dis/No	Yes

3.3.2 Scheduling in the PaaS Layer

In this layer, the scheduler objective is to satisfy the or meet the requirement cloud provider, the user, or both. This layer is responsible for the virtualization management. In other words, assigning virtual machines to physical machines (i.e. datacenters) in a way to meet the specified requirements in the SLA between cloud user and provider

[117, 58]. The works in this area have similar scheduling objectives used to optimized the PBSH as in the SaaS Layer; however, instead of assigning tasks to virtual machines, we are going one step deeper and assigning virtual machines to datacenters as illustrated in 3.2. The vitalization enables the utilization or use of the physical resources over the Internet in an effective matter over the Internet [114].

The objectives of scheduling in this layer are similar to the objectives in the SaaS layer. However, the focus of schedulers in this layer is more towards load-balancing, maximizing the utilization of physical resources, and energy saving [117, 58]. Others also considered cost and makespan. The algorithms used in IaaS layer are GA, ACO, PSO, and HBO which is also similar to the previous layer with a change of perspective. The next subsections summarize the works related to this layer and their optimization goals.

Scheduling in the PaaS Layer Using GA

Most of the GA developed to have the same coding technique as in 3.1. The chromosomes represent the number of virtual machines available (live). Then, each gene is an integer. This integer is the id of the datacenter that the virtual machine is scheduled to run on. This encoding can be seen in the work of [121]. The optimization objectives included CPU usage, memory needed to by the virtual machine, and the bandwidth. This work used NSGA-II to solve the multi-objective problem; however, the test size is very small (10 VMs, 6 datacenters). Another GA is more focused on reducing the migration of virtual machines between datacenters along with having balanced datacenters such as the GA purposed by [63]. The encoding is enhanced to achieve the optimization objectives as by using a tree structure that changes as the number of virtual machines assign to datacenters change (e.g. migration, destroying). This encoding helps balance the system's load and reduce migrations.

Other GA algorithms are used to schedule the cloud tasks with the optimization objective focused on energy conservation. The purposed GA in [122] has a similar encoding representation as in 3.1. The GA schedule the virtual machines on the datacenters in such a way to maximize utilization. The authors represented the problem as an unbalanced assignment with the objective being maximizing the utilization of the datacenters. Hence, the GA purposed in [122] can use fewer datacenters to accomplish tasks and, therefore, saves energy. This GA only called initially and when the virtual machines number changes. However, in the work done by [61] work both offline (i.e. initially) and online to reconstruct the system state when the environment is not changing. The work done by [79] added predictor with the classical GA approach as the previously mentioned GAs. This predictor is added to make the GA run fast as the classical GA in known to be slow. The optimization objective is to maximize utilization in the datacenters to reduce energy consumption. The [79] used binary encoding. It represents the datacenter assign to that virtual machine along with the average request number of that datacenter is also represented in that encoding. Also, in the purposed hybrid GA as seen in [102]. The HGA optimization objectives are reducing the number of datacenters used and the amount of communication between virtual machines.

Scheduling in the PaaS Layer Using ACO

The ACO used to place virtual machines in datacenters has a similar flow as seen in 3.2. The ants will go on trips and assign the virtual machines to certain datacenters based on a specific optimization objectives. The ACO used by [73] is the same as the ACO is shown in 3.1 .The ACO has only one optimization objective, Finding the nearest underloaded datacenter. This datacenter is used to deploy new virtual machines or to migrate virtual machines to from heavy loaded datacenters.

Other ACO algorithm like the works of both [105, 51] in which the ants will place the virtual machines in their hosts based on multi-optimization objectives. Those objectives are to minimize both CPU and memory resources wastage. The ACO in [105, 51] maximizes the utilization of the cloud resources by efficiently utilizing available CPU and memory. The ACO of [49] represented the virtual machines placement problem on the least possible number of datacenters as the well known multi-dimensional bin packing (MDBP) problem. The optimization objective in the previous ACO of [49] was to reduce the number of datacenters used by placing as many virtual machines as each datacenter can handle to save energy. Some other ACO algorithms save energy but they consider different optimization objectives. For example, depended on the resources wastage and

3.3.3 Scheduling in the IaaS Layer

In this section, the PBSH are also used to solve problem arises in this layer. Most of the solution are based on GA and ACO algorithms. The importance of resources management in this layer is to manage the datacenters of the cloud providers in order to be able to deliver promised services to customers. Those datacenters can be located in one location, across country, or across continents to serve worldwide users. The issues encountered are placement of services, routing, and interoperability (Partner Federation). Service placement means deployment of the cloud service user tasks and their assigned virtual machines on the datacenters based on certain optimization objectives using GA or ACO. Routing means finding the desired datacenter to execute or run virtual machines based on provided optimization objectives of both cloud user and cloud provider.Interoperability means collaborating between different cloud provider (i.e. sharing datacenters) to provide services to users.

3.4 Discussion

The previous works done to find a solution to the scheduling problem in the cloud in any layer are nature driven. Those PBSH found success in solving specific issue only (i.e. the PBSH are tailored for specific scheduling optimizations). There exists a theory called multi-objective theory (MOT). MOT dose does not give a specific solution to the optimization problem but rather it finds the set of feasible solution to optimization objectives and constraints given (Perto set) [42].Some of the PBSH above uses MOT to reduce the search space as seen in [50, 65, 66, 107, 101, 60, 59, 78, 103].

However, the PBSH in the previous works is lacking simulations. Most of the testing on PBSH are done in math. Math is a good formalization of those algorithms but the simulation is necessary to really test PBSH under a cloud like setup. Moreover, simulations were done, however, few, are using a small test environment (i.e. few task, virtual machines, and datacenters). It is necessary to stress the PBSH under more realistic tests to see how they will perform in a real cloud environment.

Furthermore, all the previous works are missing Elasticity. In the cloud environments, the requirements of cloud users in terms of budget, and time changes all the time. In addition, the cloud service provider emphasis on the cloud infrastructure changes. Both changes must be reflected in the cloud scheduler in all the cloud service models to accommodate those changes and get the most out of the cloud. However, using the previously mentioned scheduling (PBSH) is limited as they are only good for what they are specifically designed for.

Moreover, the previous related work is not considering the interaction between cloud service models when it comes to scheduling. All the mentioned previous works consider only one cloud service model. This consideration is limiting cloud performance because the scheduling decisions are done in the SaaS layer, for example, has effects on the

placement of the assigned virtual machine in datacenters (PaaS). Developing a scheduler that is able to schedule through cloud service models can boost cloud performance.

Mobile cloud is an emerging computation platform that employs wireless networks technologies to bring cloud computing to mobile users such as cars and cellphones. The mobile cloud presents new challenges to meet its performance expectations[96]. Scheduling is one of the challenges of mobile cloud but there are other issues to be faced especially with networking due to the mobility of the nodes[96]. There are some works that has been done to accommodate to the new challenges of mobile cloud such as [110, 15, 41, 92] where the routing techniques are light, secure, and reliable. Other works such as [1, 11, 12, 24, 47, 23, 9, 20, 48, 119, 8]provided networking solution for mobile cloud which is an emerging computation platform that will use wireless networks to bring cloud computing to mobile users such as cars and cellphones. Some works are providing solutions to data transport and distribution across moving nodes (cars) as in [13, 118, 14, 31, 22], neighbor localization that are error aware will improve mobile cloud performance as seen in the works of [29, 33, 26, 25, 16, 30, 2, 84], and allocation of available nodes dynamically as demonstrated in [18]. The green computing is important in the cloud as it saves money and the environment and thus some research as been conducted to implement an energy aware cloud and mobile cloud as seen in [4, 21, 95, 10]. One main advantage of the mobile cloud is to bring live entertainment (i.e. streaming) to end user as in [19, 28, 27, 109] and safety message propagation between cars as in [17, 10, 108]. An important aspect of the cloud is to meet the service level agreements between the service users and providers. Therefore, not only the scheduling has to be QoS aware, the routing must be aware of the QoS too as seen in [8].

Chapter 4

Adjustable and Configurable Bio-inspired Heuristic Scheduling (ACBH)

One of the most important aspects of the cloud environments is to be able to scale and adapt to changes in demands by the different key players providing or using the cloud resources. The work suggested here takes a step forward into providing an adjustable scheduler that is able to accommodate and adapt to the change of requirements (QoS) in the cloud. The ACBH suggested by this thesis is detailed in this chapter. In this chapter, the idea behind the ACBH factors that represents the heart of the algorithm and it's vetoing system are explained. The significance of the ACBH is also mentioned.

It is extremely important to have a scheduler that can provide this kind of flexibility in the cloud because of the clouds high elasticity which differentiates it from grid computing. The cloud must be able to rapidly increase the amount of resources that users need and those resources must be utilized efficiently. Moreover, the cloud based system is service driven (i.e. the SLA agreements must be met because customers are paying for their

level of service). In other words, every customer has a different service level agreement that imposes a variety of demands on the cloud scheduler. For example, SLA1 will finish the tasks in 1 minute and costs 20$ and SLA2 will execute the same set of tasks in 1.5 minutes and will cost 10$. The customer has the choice to choose which service level best meets his/her requirements. Thus, it is crucial that the scheduler be easily adjustable and configurable to be able to insure that different SLAs are met [35, 112, 3, 34].

The notion of configurability is somewhat new to the cloud scheduling based systems as seen in section 3; however, it has been introduced in grid computing as seen in [32, 7, 46]. The authors of those works have combined existing techniques to reach a new scheme that is able to enhance the rigidity grid data distribution manager or to yield a more dynamic and efficient load balancing algorithm. The authors were able to enhance the quality of their algorithms by adding adaptability and dynamicity in a grid based system which is not as elastic as the cloud based systems. Thus, to enhance the performance in the cloud based scheduling the suggested scheduler is adding adjustability and configurability to be able to provide a better service quality to cloud customers.

As seen in the previous chapter in Fig 3.1, the PBSH schedulers are following a similar process in to make a scheduling decision as demonstrated in Fig 4.1 .Following that same way of decision making, a scheduler that is bio-inspired is developed that is also able to be adjusted and configured. The adjustability and reconfigurability of the implemented scheduler helps in enhancing the rigidity of the PBSHs. The idea behind ACBH is to create a scheduler that is able to change preferences or importance depending on the change in the requirements of the cloud service subscriber and the expectations of the cloud service provider as seen in Figure 4.1.

Similar to the PBSH heuristics, the ACBH has three main evolutionary steps to make the scheduling decision. The first step is called weight initialization in which

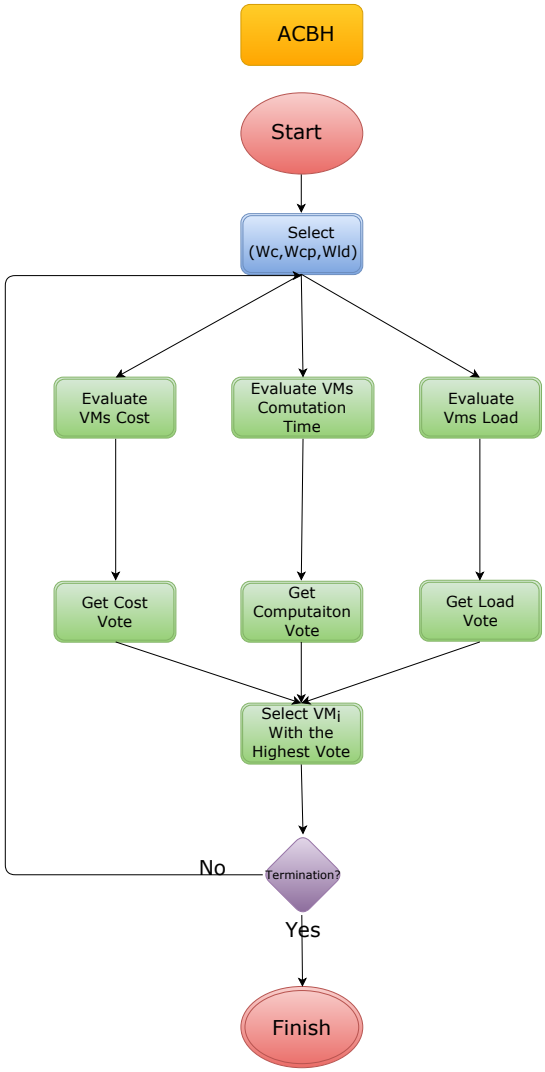

Figure 4.1: ACBH Architecture.

the algorithm chooses the current appropriate preference in terms of the optimization objectives. For example, the current preference could be 50% computation, 70% load, and 10% cost. The idea those weight is to be able to tweak the performance focus of the scheduling algorithm by only adjusting those three controlling weights. The second steps is called the candidate selection in which the ACBH evaluate the existing virtual machines using three deciding factors, they are explained thoroughly in later. After the evaluations are done, the values from each factor converted into relative values (i.e.

votes) in the voting system. Then those votes are to be adjusted using the weight chosen in step one of ACBH which will translate the objective of the scheduling to the voting system. At the end the adjusted votes are added to each virtual machine and the one with the maximum vote will be chosen to execute the given task.

The optimization objectives of ACBH are cost, computation. Moreover, the cloud service provider has a similar the computation and load. The meaning of those objectives are as follows:

- Cost: The cost of using VM_i to execute $Cloudlet_j$ in terms of RAM, MIPS, Storage, Bandwidth.
- Computation: The time VM_i will take to execute $Cloudlet_j$.
- Load: The number of Cloudlets assigned to VM_i.

The ACBH has works as a voting system in which each optimization objective is adjusted to the weight desired. Then the virtual machine with the most or the highest vote at time T_i will be assigned the current cloudlet. This property makes EBSH stand-alone scheduler because of the ability to choose the best virtual machine for the current task. Moreover, EBSH can be used as a fixable meta-heuristic since it can be extended to arrange the virtual machines in order of highest votes and then use any PBSH to assign the highest voted virtual machines to a set of tasks. The Elasticity of ACBH is to be able to work either as a heuristic and meta-heuristic enable it to be both centralized and partially distributed. The steps followed by EBSH are:

Algorithm 1 ACBH VM Assignment Algorithm

Require: $Cloudlet_{list}, VM_{list}, Datacenter_{list}, W_{cost}, W_{computation}, W_{load}$
1: $Size(m) \leftarrow getSize(CLOUDLET_{list})$
2: $Size(n) \leftarrow getSize(VM_{list})$
3: **for** $j = 1$ to m **do**
4: **for** $i = 1$ to n **do**
5: $CostVote \leftarrow getCostVote(VM_i, VM_i, Cloudlet_j)$ // as in 4.1
6: $AdjuCostVote \leftarrow adjustVote(W_c, Cost_Vote)$
7: $ComputationVote \leftarrow getComputationVote(VM_i, Cloudlet_j)$ // as in 4.5
8: $AdjuComputation_Vote \leftarrow adjustVote(W_{cp}, ComputationVote)$
9: $LoadVote \leftarrow getLoadVote(VM_i)$ // as in 4.6
10: $AdjuLoadVote \leftarrow adjustVote(W_{ld}, LoadVote)$
11: $SumAdjustedVotesVMi \leftarrow sumOfVotes(AdjuCostVote, AdjuComputationVote, AdjuLoadVote)$
12: **end for**
13: $VMToBeAssigned \leftarrow getMaximumVoteVM(SumAdjustedVotesVMi)$
14: $Assign(Cloudlet_j, VMToBeAssigned)$
15: **end for**

1. The estimation of the cost for executing $Cloudlet_j$ on VMi where i(1..n) is done. The cost is estimated based on the RAM, Storage, Bandwidth, and MIPS required by $Cloudlet_j$ from VMi.

2. The execution time of $Cloudlet_j$ on VMi where i (1..n) is estimated. This estimation is based on the worst case to ensure that we have the virtual machines with the least execution time.

3. The load on each virtual machine is calculated by the number of cloudlets assigned to it. this is also based on the worst case to ensure we get the least loaded virtual machine.

4. If stand-alone scheduler

   Convert all the optimization objectives from VM_i i (0..1).

   Add all the votes for every virtual machine and then find the one with the maximum vote. Finally, assign that virtual machine to the cloudlet to execute.

5. if meta-heuristic

   Convert all the optimization objectives from VM_i i (0..1).

Make a subset of virtual machines according to a threshold of the minimum number of votes. Then give the subset of virtual machines to any PBSH to assign tasks to that subset.

Table 4.1: ACBH Cost Factor Equations

Parameter	**Values**
T_{CL_j}	The cLength of the $Cloudlet_j$
$CPSVM_i$	The cost of storage used by Vm_i
DC_{CPS}	The Cost of storage of $Datacenter_i$
$size_{VM_i}$	The storage required by VM_i
$CPRVM_i$	The cost of RAM to execute $Cloudlet_j$ by VM_i
DC_{CPR}	Cost of RAM for executing $Cloudlet_j$ by VM_i
RAM_{VM_i}	The RAM required by VM_i
$CPBVM_i$	Cost of Bandwidth for executing $Cloudlet_j$ by VM_i
DC_{CPB}	$Datacnter_i$ cost per bandwidth.
Bw_{VM_i}	The needed bandwidth consumed by VM_i
$NOCVM_i$	The number of cloudlets assigned to VM_i
$MIPSVM_i$	The MIPS of VM_i

$$Cost_{Vote}^{i,j} = (CPSVM_i + CPRVM_i + CPBVM_i) \times (T_{CL_j}), i = 1...N, j = 1...M \tag{4.1}$$

Where,

$$CPSVM_i = DC_{CPS} \times size_{VM_i}, i = 1...N \tag{4.2}$$

$$CPRVM_i = DC_{CPR} \times RAM_{VM_i}, i = 1...N \tag{4.3}$$

$$CPBVM_i = DC_{CPB} \times Bw_{VM_i}, i = 1...N \tag{4.4}$$

The cost estimation functions are similar to the cost estimation followed by HBO.

$$Computation_{Vote}^{i,j} = \frac{(T_{CL_j} \times NOCVM_i)}{MIPSVM_i}, i = 1...N, j = 1...M \tag{4.5}$$

$$Load_{Vote}^{i,j} = NOCVM_i, i = 1...N, j = 1...M \tag{4.6}$$

All of the above equations are then used to get a vote for their optimization objective. This can be done by finding the minimum value calculated. Then this value will get a vote of 1. The others are divided by the minimum value given us the vote that is between (0 and 1). The importance of converting the optimization objectives to votes because the calculations will give different values. They are not close and adding them to gather affects the choice of virtual machines. Thus, converting them to votes (i.e. relative values) ensure fairness. Moreover, the voting system is necessary for the meta-heuristic mode.

Chapter 5

Population Based Scheduling Heuristics (PBSH)

This chapter of the thesis presents the chosen PBSH in details. The descriptions, flow charts, and pseudocodes of the PBSH are shown. Also, the details of the Random Biased Sampling algorithm is displayed.

5.1 Honey Bee Optimization (HBO)

The basic elements in HBO scheduling algorithm is a bee. Those bees find the most profitable source and exploit it. Foragers are also considered by shifting in quality or profit in the nectar sources [98]. HBO provides a self-managed and self-organized solution, by essence decentralized, to the scheduling problem [81]. Such characteristics fit it as a strong candidate for Cloud scheduling.

HBO scheduling algorithms are basically divided into two parts, as delimited in Algorithm 2. The first part consists of the foraging behavior of the bees as they look for food sources. The second part is the scouting where the bees start their search for the best

Algorithm 2 HBO VM Assignment Algorithm

Require: $Cloudlet_{list}, VM_{list}, Datacenter_{list}, fac_{LB}$
1: $Groups(q) \leftarrow divide(Cloudlet_{list})$
2: **for** $i = 1$ to q **do**
3: $length_i \leftarrow lenghtOfGroup_K(Groups_i)$
4: **end for**
5: **for** $k = 1$ to q **do**
6: $CloudLet_L \leftarrow max(Groups_k$
7: **while** $Groups_k \geq \{Groups_i | i = 1..q$ and $i \neq k\}$ **do**
8: **for** $s = 1$ to n **do**
9: $Datacenter_s \leftarrow select(Datacenter_{list})$ // as in Eq 5.1
10: **if** $fac_{LB} \leq VMsAssigned(Datacenter)$ **then**
11: $assign(Cloudlet_L, Datacenter_s(VM_{leastLoad}))$
12: **else**
13: $assign(Cloudlet_L, Datacenter_{i \neq s}(VM_{leastLoad}))$
14: **end if**
15: $decrement(length_k)$
16: **end for**
17: **end while**
18: **end for**

food source brought by the foragers. The HoneyBee procedure is described in essence according to the following elements:

1. The number of foraging VMs (n) is equal to the number of Datacenters (DCs) available. The choice of foraging VMs placement in DCs is random;

2. The DCs have its own characteristics (dch). Those characteristics (RAM, Storage, Bandwidth) help the foraging VMs to find the best food source for a cloudlet with a specific execution time;

3. The above-mentioned dch and the execution time are used to evaluate the fitness function. The outcome of the fitness value will then be the deciding factor by which we chose to run the cloudlet on which DC;

4. The DC with the highest fitness value, or the lowest cost as defined in Equation 5.1, receives a percentage of the tasks. The remaining subset of the tasks will be executed on the other DCs by repeating step 4 and excluding the chosen DCBest.

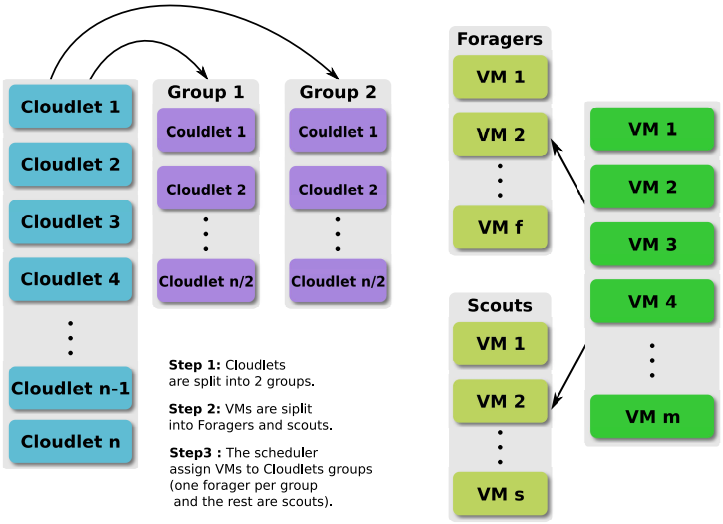

Figure 5.1: Honey Bee Structure.

$$DC_{Cost}^{i,j} = (Size_i + M_i + Bw_i) \times (T_{CL_j}), i = 1...N, j = 1...M \tag{5.1}$$

$$Size_i = dch_{CPS} \times size_{VM_i}, i = 1...N \tag{5.2}$$

$$M_i = dch_{CPR} \times RAM_{VM_i}, i = 1...N \tag{5.3}$$

$$BW_i = dch_{CPB} \times Bw_{VM_i}, i = 1...N \tag{5.4}$$

To simulate the honey bee for a Cloud environment, Figure 5.1 shows that Cloudlets are split into two groups forming a food source. Then, the VMs are split into foragers and scouts. Each forager is assigned to a cloudlet group where it is responsible for recruiting scouts VM to help execute the Tasks. Each scout chooses a task with certain probability

as delimited in Equation 5.1. This equation is formed by a sum of the size, quantity of memory, and bandwidth, respectively represented by Equations 5.2, 5.3, and 5.4 where the following parameters delimit them:

Parameter	**Values**
T_{CL_j}	The cLength of the $Cloudlet_j$
$Size_i$	The cost of storage used by Vm_i
dch_{CPS}	The Cost of storage of $Datacenter_i$
$size_{VM_i}$	The storage required by VM_i
M_i	The cost of RAM to execute $Cloudlet_j$ by VM_i
dch_{CPR}	Cost of RAM for executing $Cloudlet_j$ by VM_i
RAM_{VM_i}	The RAM required by VM_i
BW_i	Cost of Bandwidth for executing $Cloudlet_j$ by VM_i
dch_{CPB}	$Datacnter_i$ cost per bandwidth.
Bw_{VM_i}	The needed bandwidth consumed by VM_i

5.2 Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) uses the behavior of real ants in foraging for food to implement a solution for the cloud task scheduling. The ants leave the nest to search for food sources (VMs) in random. Then they evaluate the quality of the food source and carry it back to its nest. The ants leave a chemical trail on the ground. The strength of that chemical trail depends on the quality of the food source found [45]. Researchers used ACO to solve NP-hard problems such as traveling salesman problem, graph coloring problem, vehicle routing problem, and scheduling problem. In the context of Cloud computing, ACO is used to find the optimal way to schedule tasks to VMs [45, 74].

In ACO when ants search for food first, ants search randomly for food sources and once one is found, an ant leaves a chemical trail called pheromone leading to that food source. Then, other ants are attracted to that specific food source by following the

pheromone trail. This process continues until ants find the shortest path leading to a specific food source by accumulating huge amounts of pheromone on the shortest path leading to the food source [45]. The ACO can be applied to solve complex combination problems if the following elements are properly delimited for the algorithm:

- Problem statement: In this algorithm, ants find the optimal solution to the cloud-scheduling problem by moving from town to town (VM to VM) to choose the best VM to the Cloudlets. Ants move from At the start of the simulations, ants are placed randomly at different VMs with initial pheromone trail $\tau(0)$ as in Equation 5.5.

- Heuristic desirability η: the inverse of the expected execution time is used.

- Constraint satisfaction method: each ant is only allowed to visit a VM once to minimize scheduling time

- Pheromone-updating rule: each ant deposit a pheromone the depends on the quality of the solution. The pheromone is updated according to Equations 5.6-5.11.

- Probabilistic transition rule: the ant related to $tabu_k$ is updated by adding the visited VM in the initial step. Afterwards, Ant_k selects the next VM_j to execute $Cloudlet_i$ based on the probability defined in Equation 5.5.

$$\rho_{i,j}^{k} = \begin{cases} \left(\frac{[\tau_{i,j}(t)^{\alpha}] \times [\eta_{i,j}]^{\beta}}{\sum_s Allowed_k [\tau_{i,s}](t)^{\alpha} \times [\eta_{i,s}]^{\beta}} \right) & , j \epsilon Allowed_k \\ 0 & , \text{Otherwise} \end{cases} \tag{5.5}$$

- $\tau_{i,j}$(t) : is the pheromone concentration between task i and VM j.

- $allowed_k$: keeps track of the VMs that Ant_k can use at all times.

- $\eta_{i,j}$: is the heuristic value calculated in Eq(3) where $\eta_{i,j} = \frac{1}{d_{i,j}}$.

$$d_{i,j} = \left(\frac{TL\_Task_j}{Pe\_num_j \times Pe\_mips_j} + \frac{InFileSize}{VM\_bw_j} \right) \tag{5.6}$$

- α and β : to choose the relative weight of between the pheromone concentration and the heuristic value.

Equations 5.7, 5.8, 5.9, 5.10, and 5.11 are used to update the pheromone concentration. Initially each ant is placed on a VM randomly and pheromone value of $\tau(0)$ is given to all edges, as described in Algorithm 3.

$$\Delta\tau_{i,j}^k(t) = \begin{cases} \left(\frac{Q}{L_k(t)}\right) & ,i,j \epsilon T_k(t) \\ 0 & ,Otherwise \end{cases} \tag{5.7}$$

$$L_k(t) = argmax_{j \epsilon J} sum_{j \epsilon IJ}(d_{ij}) \tag{5.8}$$

$$\tau_{ij}^k(t) = (1-\rho)\tau_{ij}^k(t) + \Delta\tau_{ij}^k(t) \tag{5.9}$$

$$\Delta\tau_{ij}^k(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{5.10}$$

$$\tau_{ij}^k(t) = \tau_{ij}^k(t) + \left(\frac{Q}{L_k(t)}\right) if(i,j) \epsilon T_k(t) \tag{5.11}$$

$L_k(t)$ is the length of the current best tour done by the ants in the current iteration as in Eq(8). The local Pheromone value is updated by Eq(9) where ρ is the decay factor of the pheromone deposited before. Finally, the global pheromone value is updated by Eq(11).Multiple values were tested and the best were chosen and the are as follow:

ACO Parameter	Values
Ants	50
α	0.01
β	0.99
ρ	0.4
Q	100

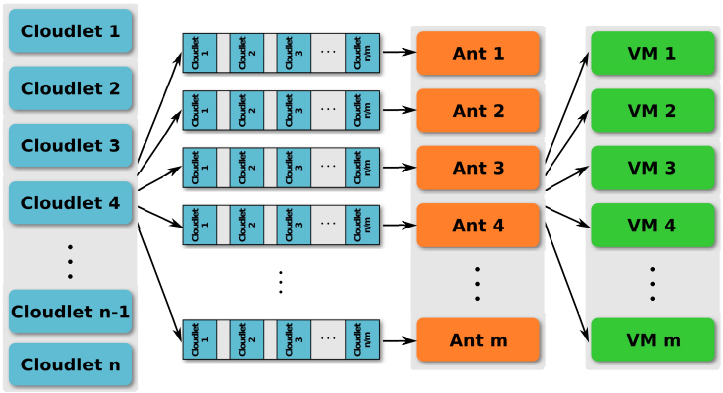

Figure 5.2: Ant Colony Architecture.

The diagram represented in Figure 5.2 illustrates in general terms the whole process of combining Cloudlets to VMs, which is described in Algorithm 3. The scheduler firstly creates the ants and then distribute the Cloudlets to each ant. Then, the ants evaluate the VMs based on the previously mentioned equations to choose the best VM. Finally, the tabu of each ant is updated with the chosen VM and the algorithm is repeated in the following iteration.

5.3 Random Biased Sampling (RBS)

The Random Biased Sampling (RBS) algorithm provides a way to construct a network of resources (VMs). Those networked resources are then divided into groups. The groups are given a degree. This degree on an average of all resource groups must be the same

Algorithm 3 AntColony VM Assignment Algorithm

Require: $\alpha, \beta, max_{iterations}, Cloudlet_{list}, VM_{list}$
1: **for** i in $Cloudlet_{list}$ **and** k in VM_{list} **do**
2: $pair_{Cloudleti,VM_k} \leftarrow \tau_{i,j}(0) = C$ // pheromone(C)
3: **end for**
4: $VM_k \leftarrow Ant_j \leftarrow randomPick(Ant_{pool})$
5: $Ant_j^{tabu} \leftarrow add(VM_k)$
6: **while** NOT done **do**
7: **for** $k = 1$ to m **do**
8: $VM_s \leftarrow select(Ant_k, VM_{list}, Cloudlet_{list})$ // as in Eq 5.5
9: $Ant_j^{tabu} \leftarrow add(VM_s)$
10: **end for**
11: **for** $k = 1$ to m **do**
12: $L_k \leftarrow calculate()$ // as in Eq 5.8
13: **end for**
14: $\tau_{i,j} \leftarrow update()$ // as in Eq 5.9
15: $pheromone_{global} \leftarrow update()$ // as in Eq 5.11
16: $increment(iterations)$
17: **end while**

achieving balanced scheduling. The created network of resources is self-organized and able to distribute the tasks based on only local knowledge. Thus, the Random Sampling algorithm represents a viable solution to the cloud task-scheduling problem [89].

RBS constructs a graph of resource each with a node in degree (NID) and a walk length threshold (υ). The tasks coming into the servers have an associated walk in length (ω) that is used to schedule those tasks to the appropriate resources. The NID value varies depending on the number of free resources on a given node. When a $task_i$ comes into the servers, $node_k$ will only executes a task if the execution test is fulfilled ($\omega \geq \upsilon$). If the previous condition is not satisfied, ω is incremented by one and sent to the other nodes and the execution test is applied again. Algorithm 4 represent the general procedure of RBS in a cloud environment. The following steps shortly describe the functioning of the algorithm:

- Step1: The VMs are split into n groups. Each group contains an equal number of VMs;
- Step2: Each Group is assigned walk length threshold (υ) and an NID. The NID is equal to the number of free VMs in that group;

Algorithm 4 Random Biased Sampling VM Assignment Algorithm

Require: $Cloudlet_{list} and VM_{list}$
1: **for** $k = 1$ to n **do**
2: $Groups(q) \leftarrow divide(VM_{list}, groupSize(number(r)))$
3: **end for**
4: **for** $k = 1$ to q **do**
5: $Group_k \leftarrow ascending(WIL)$
6: **end for**
7: **for all** $Cloudlet_i$ in $Cloudlet_{list}$ **do**
8: $Cloudlet_i \leftarrow random(WIL)$
9: **end for**
10: **for** $k = 1$ to m **and** i to q **do**
11: **if** $Cloudlet_k.WIL \geq Group_i.WIL$ **then**
12: $Group_i \leftarrow Cloudlet_k$
13: **else**
14: $increment(Clouldet_k.WIL, 1)$
15: $Group_{i+1} \leftarrow Cloudlet_k$
16: **end if**
17: **end for**

- Step3: Each Cloudlet is given a random ω;
- Step4: The execution test is performed to assign $Cloudlet_i$ to $group_j$;
- Step5: The NID of groupj is reduced by one;
- Step6: The assignment inside the VMs groups is done in a cyclic way;
- Step7: Repeat starting from step 3.

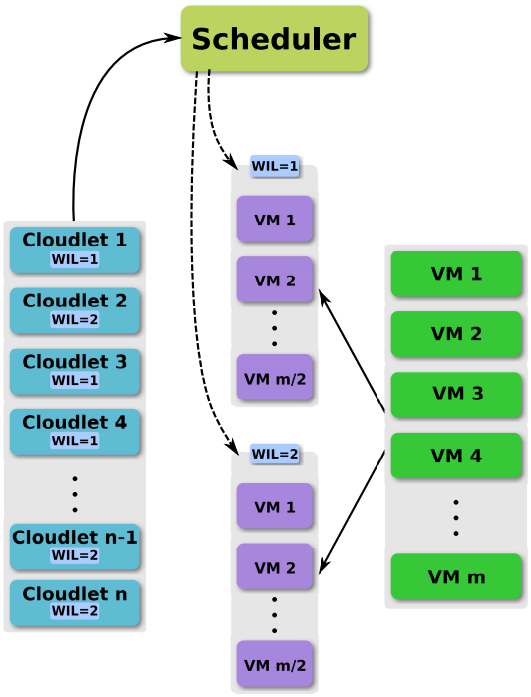

Figure 5.3: Random Biased Sampling Match-Making.

Figure 5.3 depicts the match-making of Cloudlets to VMS of the algorithm on a simple diagram. In the figure, the RBS scheduler divides the VMs available into two groups. Then, the created VM groups are assigned a walk in length (WIL) values (WIL = 1 .. n). Following this step, the Cloudlets are given a random WIL value between the available WIL values assigned to the VM groups. The RBS assigns a Cloudlet to a VM group if the Cloudlet WIL is equal or greater to the WIL of the VM group. If this condition is not applied, then RBS scheduler will increase the Cloudlet WIL in incremented by one and the algorithm is run again.

Chapter 6

Performance Evaluation

The performance evaluation chapter presents the results of the simulation. In the first part of this chapter, the homogeneous simulations scenario is introduced and it's simulation results and explanations are presented. In the second part of the experiments, the heterogeneous simulations are detailed and presented along with their results, which are discussed in details. Finally, the simulations of ACBH, as well as its variations, are also displayed.

6.1 Simulation Scenarios

Two scenarios have been used to analyze the bio-inspired schedulers fully and thoroughly. The first scenario comprises a homogeneous setup of physical resources that receives a homogeneous load, Cloudlets that impose the same amount of workload. The use of this scenario aims at testing the suggested algorithms against a base test, which is expected to be the optimum solution in this type of homogeneous setup. On the other hand, the second scenario represents a more realistic Cloud environment, in which heterogeneous resources and load are most likely to exist.

The homogeneous environment scenario was conducted to show that even in the worst case conditions in which no scheduler is needed, the bio-inspired algorithms are expected to converge to the optimal scheduling performance, and the only difference in this particular case resides on the time each scheduler takes to produce the assignment of load, which shows that the bio-inspired scheduling requires significantly more time than the base test.

Tables 6.1 and 6.2 show the experimental setup used in the homogeneous scenario respectively for the environment, VMs, and the load, Cloudlets. The experiment was conducted with the number of VMs ranging from 1000 to 100000 and 1000000 Cloudlets. There is a focus on computational performance while conducting this analysis, so negligible network delay is observed on the execution of the Cloudlets during the simulation; thus, the default network topology provided by CloudSim was used in running the experiments. Several different aspects have been considered in the analysis in order to determine effectiveness and efficiency of the algorithms. The first measurement observed in the experiments consists in the scheduling time of the selected scheduling algorithm. The second measurement criteria involve the execution time that the Cloudlets generated in order to complete their work. The third measurement considers the execution time imbalance which gives a sense of the balancing ratio of the load during its execution in the simulator.

Since all elements in this scenario present the same characteristics and capabilities, it represents the worst case for the scheduling algorithms, and the base test is necessarily the optimal solution for any time to scheduling setup due to equally assign Cloudlets to the virtual resources in a cyclic fashion.

The heterogeneous environment scenario was implemented to mimic real cloud environment where task and virtual machines are not similar. In other words, different

Table 6.1: VM Characteristics in the Homogeneous Setup

VM characteristics	**Values**
vmMips	1000
vmSize	5000
vmRam	512
vmBw	500
vmPesNumber	1

Where:

- vmMips: million instructions per second.
- vmSize: the size of the virtual machine in MB.
- vmRam: the RAM of the virtual machine.
- vmBw: the bandwidth of the virtual machine.
- vmPesNumber: the number of processing elements of the virtual machine.

Table 6.2: Cloudlet Characteristics in the Homogeneous Setup

Cloudlet characteristics	**Values**
cLength	250
cFileSize	300
cOutputSize	300
cPesNumber	1

Where:

- cLength: The required MIPS for the cloudlet.
- cFileSize: The required memory for the cloudlet.
- cOutputSize: The size of the output file of the cloudlet.
- cPesNumber: The required processing elements number for the cloudlet.

workloads were submitted to virtual machines that also have different capabilities. For the setup on this scenario, a different range of parameters was used, restricting to smaller dimensions; however, such restrictions did not impact on the results obtained. Thus, the

number of virtual machines was reduced to 50 and the number of Cloudlets was reduced to 5000. Tables 6.3, 6.5, and 6.4 list the characteristics of the VMs, Datacenters, and the tasks, respectively.

Table 6.3: VM Characteristics in the Heterogeneous Setup

Heterogeneous VM characteristics	**Values**
vmMips	500-4000
vmSize	5000
vmRam	512
vmBw	500
vmPesNumber	1

Table 6.4: Cloudlet Characteristics in the Heterogeneous Setup

Heterogeneous Cloudlet characteristics	**Values**
cLength	1000-20000
cFileSize	300
cOutputSize	300
cPesNumber	1

Table 6.5: Datacenter Characteristics in the Heterogeneous Setup

Datacenter characteristics	**Values**
CostPerMemeory	0.05-0.01
CostPerStorage	0.004-0.001
CostPerBandwidth	0.05-0.01
CostPerPrcessing	3

All the simulations below were executed for ten times and the results displayed in the figures are the averages of those runs. Moreover, the graphs have confidence intervals

shown as well. The simulation parameters (i.e. Cloudlet, VM, Datacenter characteristics) used are based on what has been extensively used in the literature. These parameters have to be distinct so that the results shows the differences in performance of the scheduling algorithms as suppose to using similar simulations parameters that will not show performance differences.

6.2 CloudSim

CloudSim emerged from the need to have a cloud simulation environment that would extend the traditional distributed systems simulators (Grid and Network) [37]. Moreover, testing on real cloud environment is costly and imposes several challenges, which considerably increase the complexity in conducting large-scale experiments; several factors drive such challenges, such as (i) variation in the clouds demand, supply pat tern, system size, and resources; (ii) the heterogeneous characteristics of user and QoS requirements in these dynamic environments; and (iii) variations in applications performance, dynamic, and scaling requirements. Moreover, a simulator is crucial in providing controlled scenarios in which results are reproducible over the most diverse combination of set up parameters [88].

Furthermore, aside from money and time when using commercial clouds, testing the scheduling and allocation of resources are challenging and not practical due to many factors. The first factor is variation in the clouds demand, supply pattern, system size, and resources. Secondly, the heterogeneousness of user and QoS requirements is due to the cloud being a dynamic environment. The third factor is the variations in applications performance, dynamic, and scaling requirements. Thus, the use of real cloud environments, such as Amazon EC2 and Microsoft Azure, cannot be used to test scheduling and allocations of resources (benchmarking) under varying conditions because of the rigidity

imposed by infrastructure. Thus, repeating test cases and getting reliable performance results is an extremely hard task because of the need to reconfigure benchmark across the cloud platform for multiple runs. Therefore, the need arises for a more controllable and developer-friendly cloud simulation environment to perform benchmarks and get reliable results. Hence, CloudSim was created to fulfill this need.

The need for a simulation tool that is able to provide control to the developer is very crucial in testing the hypothesis. A tool where it is easy to reproduce results will benefit researchers and IT companies greatly. Among the benefits of using a simulation based cloud environment are the controllability over the environment and experimenting with a verity of workloads and resources for developing and testing new scheduling and provisioning algorithms [88].

CloudSim: a new, generalized, and extensible simulation framework that allows seamless modeling, simulation, and experimentation of emerging Cloud computing infrastructures and application services. This simulation tool facilitate develops to configure the test environment with ease and build test scenarios. CloudSim provides test results and can be enhanced and added to by the developers. Furthermore, CloudSim saves time because of the ease of use and very flexible because a developer can build and test environment and scheduling algorithms for testing their applications in a heterogeneous cloud environment was developed to simulate real cloud environment and more specifically for testing the scheduling of tasks, Cloudlets, to virtual machines (VMs). Moreover, CloudSim is also capable of testing the virtual machines migration from one physical server to another to balance the loads on those servers.

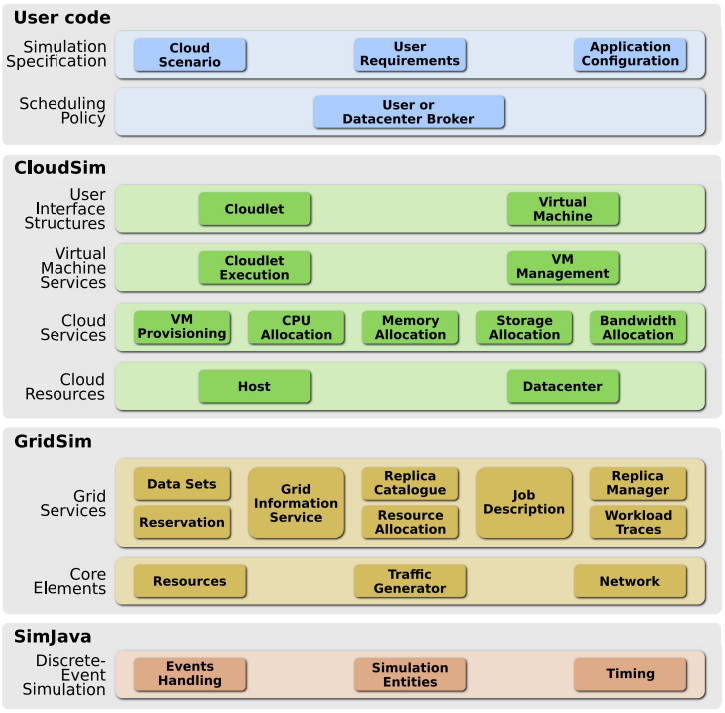

Figure 6.1: CloudSim Architecture [37]

6.2.1 CloudSim Architecture

The Figure 6.1 shows that CloudSim is a multi-layered Simulation environment. This environment was based on SimJava but then changed to be able to support advanced operations that are not supported by it. CloudSim provides the core functionalities, such as queuing and event processing. Also, the creation of the cloud core entities (services, host, data center, broker, and VMs) is handled by CloudSim [38].

The main classes that are needed to be extended are listed below. Those classes are essential to be overwritten in order to implement our own scheduling algorithms.

- DatacenterBroker: this class mediates between the user and services. In this class, the user adds the scheduling algorithms for assigning the Cloudlets to VMs. Thus, DatacenterBroker class must be extended by researchers and developers to simulate the scheduling algorithms.

- Cloudletscheduler: This class implements policies for scheduling or sharing of a VM by Cloudlets. There are two policies offered: time-shared and space-sharing. Developing new scheduling policies is possible by overriding the methods of this class.

- VmScheduler: This class implements policies for scheduling host processing cores to VMs (time-shared, space-sharing). Developing new scheduling policies is possible by overriding the methods of this class.

- VmAllocationPolicy: This class determines the provisioning policy for VMs utilization by hosts. It is responsible for the deployment of the VMs to appropriate host that meets the requirements (memory, storage, and availability) [38].

6.2.2 Base Test

In the experiment performed in this work, the default scheduler in CloudSim was used as the base test to compare with the studied algorithms. The algorithm of this Base Test is a simple scheduler that assigns Cloudlets to VMs in a cyclic manner. For instance, in an attempt to allocate a set of virtual machines (vm1, vm2) for a set of Cloudlets (c1,c2,...,cn), the Base Test assigns vm1 to c1, vm2 to c2, vm1 to c3 and so forth until all Cloudlets are assigned, showing an equally distributed load of the virtual resources. This scheduler represents the best solution in a homogeneous setup and, for this reason, is chosen to compare with the other solutions.

6.3 Performance Metrics

Three metrics were used to provide an overview of the performance of the algorithms on the experiments. Each of the metric is detailed in the following subsections.

(a)*Scheduling Time:* The scheduling time of the scheduling algorithms is obtained by retrieving the startup time in which the algorithm has been triggered and the final time when the algorithm returned with the solution of a specific combination setup. Due to the large scale of the setup environments used in our experiments, the unit of the scheduling time is in hour even though the precision of the retrieved metrics are on the scale of milliseconds. The times for the ACO, HoneyBee, and RBS are roughly the same. They are larger than the base test as they require more computation to schedule the Cloudlets to the VMs.

(b)*Execution Time:* The execution time corresponds to the maximum overall time that the tasks took to complete the execution. As described in Equation 6.1, this metric consists in the difference between earliest start execution time of a Cloudlet and finish time of the latest Cloudlet to end its execution. Please note that this metric unit corresponds to milliseconds of simulation wall clock time.

$$T_{exe} = T_{maxFinishTime} - T_{minStartTime} \tag{6.1}$$

Where:

- T_{exe} : simulation time of the Cloudlets.
- $T_{maxFinishTime}$: maximum finish time of the Cloudlets.
- $T_{minStartTime}$: minimum start time of the Cloudlets.

Through this simulation time metric, we can determine the effectiveness of the scheduling algorithms.

(c)*Time Imbalance:* The time imbalance provides an overview on the fairness of the Cloudlets execution on VMs. In summary, this metric represents the variation in the execution time of the tasks. Equation 6.2 details how time imbalance values are obtained based on the execution times of the Cloudlets.

$$T_{im} = \frac{[T_{max}] - [T_{min}]}{T_{avg}} \tag{6.2}$$

Where:

- T_{im} : time imbalance.
- T_{max} : maximum execution time of Cloudlets.
- T_{min} : minimum execution time of Cloudlets.
- T_{avg} : average execution time of Cloudlets.

(d)*Processing Cost:* The processing cost of the Cloudlets is defined in terms of the assigned virtual machine Vm_i uses of the resources provided by the datacenter in executes on. Those resources are defined as the following characteristics:

- Cost of Memory : the memory used by Vm_i in its datacenter;
- Cost of Bandwidth : the amount of bandwidth consumed by Vm_i;
- Cost of MIPS : the cost of computation (MIPS) to execute the task.

6.4 Experimental Results

The results obtained in the experiments are presented and discussed separately in two sections: homogeneous scenario and heterogeneous scenario.

6.4.1 Homogeneous Scenario

In Figure 6.2, the simulation shows that the performance in terms of Cloudlet simulation time is similar to the optimum scheduler (base test). As the number of virtual machines increases the simulation time of the Cloudlets decreases. The significance of this Figure

is that it shows that even in the worst case scenario, the algorithms behave closely to the Base test. ACO started with the worst performance but as the number of virtual machines grows it caught up to the optimal solution. The HBO began and ended the closest of the algorithm to the base test while the RBS started close to the best solution and ended the worst due to the randomness in assigning tasks a WIL (walk in length) value.

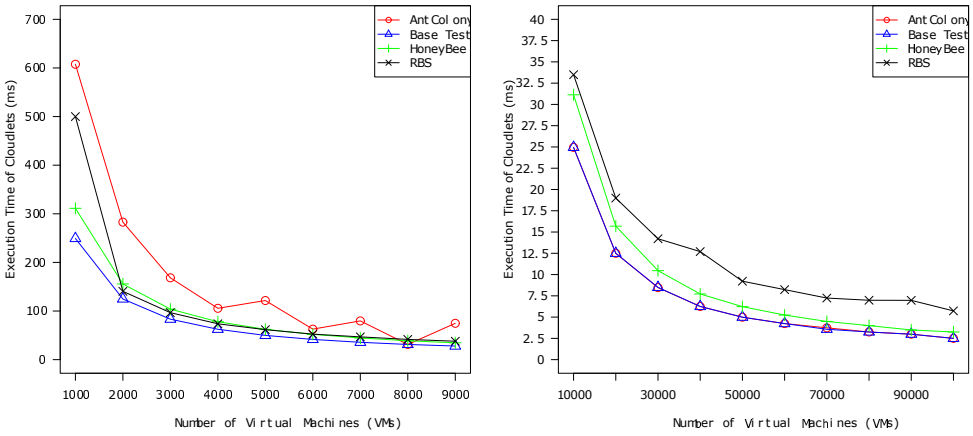

Figure 6.2: Simulation Time of the Homogeneous Scenarios.

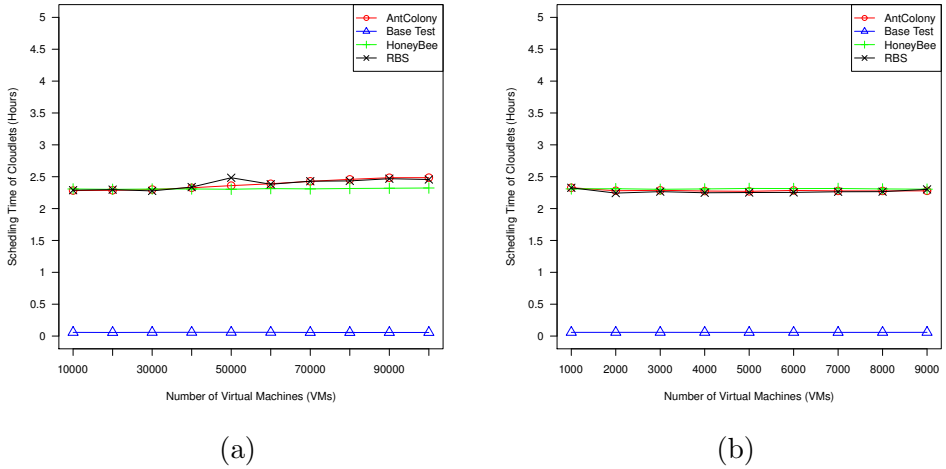

Figure 6.3: Scheduling Time in the Homogeneous Scenarios.

In Figure 6.3, the scheduling time for the base test is significantly less that the other algorithms. The reason behind the base test having the shortest scheduling time is that it does not have any computation to make the scheduling decision; it simply assigns the Cloudlet to the next virtual machine cyclically. The rest of the schedulers (ACO, HBO, And RBS) require computations to assign the Cloudlets to take a longer time to make a scheduling decision. Also, the size of the test makes the scheduling time difference significant (1000000 Cloudlets and up to 100000 VM to choose from).

6.4.2 Heterogeneous Scenario

In this scenario, the number of virtual machines is reduced significantly from the homogeneous scenario; the test used 50 virtual machines and up to 1000 Cloudlets. The scenario was used to evaluated the algorithms according to the same metrics described previously.

Table 6.6 demonstrates the configurations of the ACBH. This configuration is used in this part of the simulations. Many other setups of ACBH are explained and compared in Section 6.4.3.

Table 6.6: ACBH Factors

ACBH factors	**Weight %**
W_c	0
W_{cp}	100
W_{ld}	50

In Figure 6.4, the obtained execution times are shown for an increasing number of cloudlets and keeping the number of virtual machines the same. In this part of the simulation that tests the execution time of the algorithm, it can be clearly seen that

the ACBH has the best performance of all the algorithms. ACBH factors are set up to focus on computation (i.e. finish fast) and fairness is also considered as seen in 6.6. Thus, ACBH has the fastest execution time. ACO perform the better that the remaining algorithms, namely (HBO, RBS, and Base test). The factors of the ACO were chosen to give the best possible performance of the ACO in terms of the execution time of cloudlets. The ACBH scheduler has the edge on ACO in execution time because ACBH does not have a restriction of reusing virtual machines as the ACO when scheduling cloudlets. Although, fairness is considered in this set up of ACBH but the implementation of ACBH makes sure that each factor has a fair saying in the scheduling decision. However, the current setup of ACBH focuses more on the computation aspect.

Figure 6.5 shows the time each scheduler takes to finish assigning virtual machines to Cloudlets. The time was recorded in seconds. The Figure describes that the base test presented the shortest scheduling time because of its simplicity and no computation needed to make the scheduling choices. The second best scheduling times is the RBS due to its simple assignment decisions as it only checks for the WIL (walk in length). The third best scheduling is the HBO as it only identifies the least cost of the virtual machines to execute the Cloudlets. ACO and ACBH are taking the longest to provide a scheduling solution as they need to perform more complex, iterative computations and multiple searches to schedule the Cloudlets.

As shown in Figure 6.6, it measures the time degree imbalance to find the best load distribution of Cloudlets among virtual machines. In other words, it measures the fairness of the execution time between the cloudlets. ACBH in the current setup 6.6 of its factors has the fairness factor considered and its performance matches the Base Test that has the edge in load distribution due to its nature of scheduling as it gives each of the virtual machines an equal number of Cloudlets; hence, the execution times of those Cloudlets

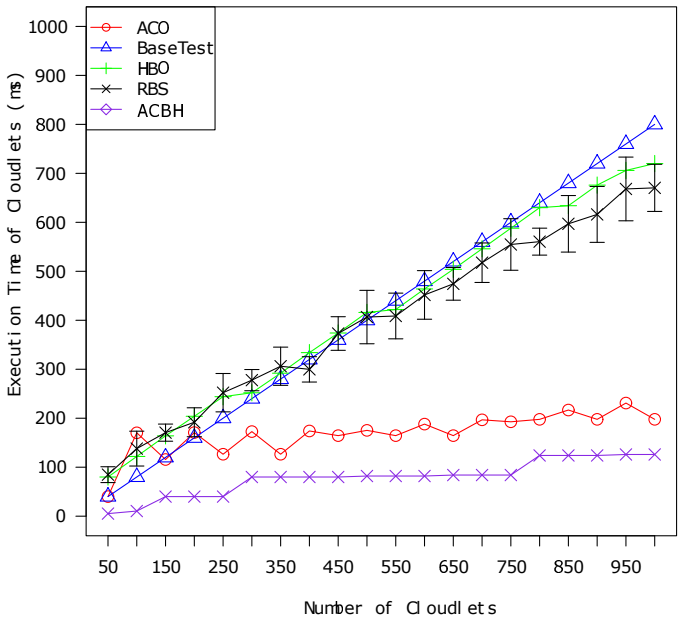

Figure 6.4: Execution Time of the Heterogeneous Scenarios.

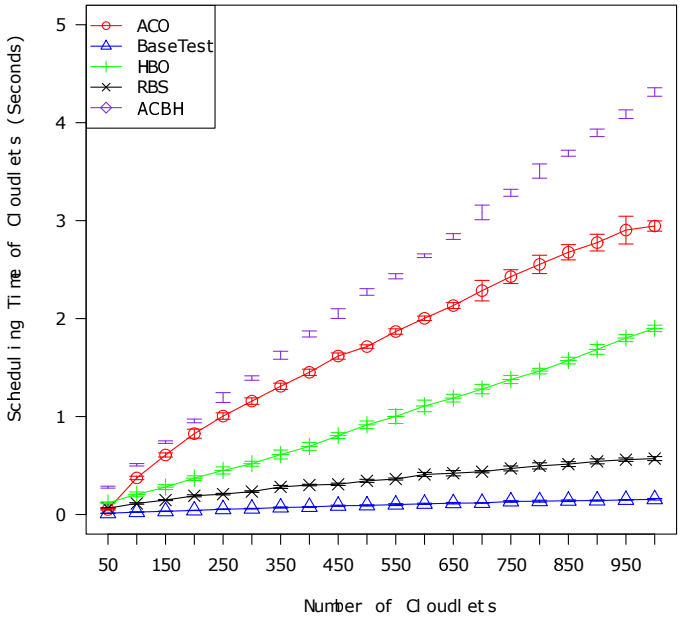

Figure 6.5: Scheduling Time for the Heterogeneous Scenarios.

are close to each other. However, in the case of the Base Test, it does not mean that the Cloudlets finish fast as proven in Figure 6.4. HBO and ACO have similar fairness performance. HBO has the load balancing factor it used to the allocating resources and

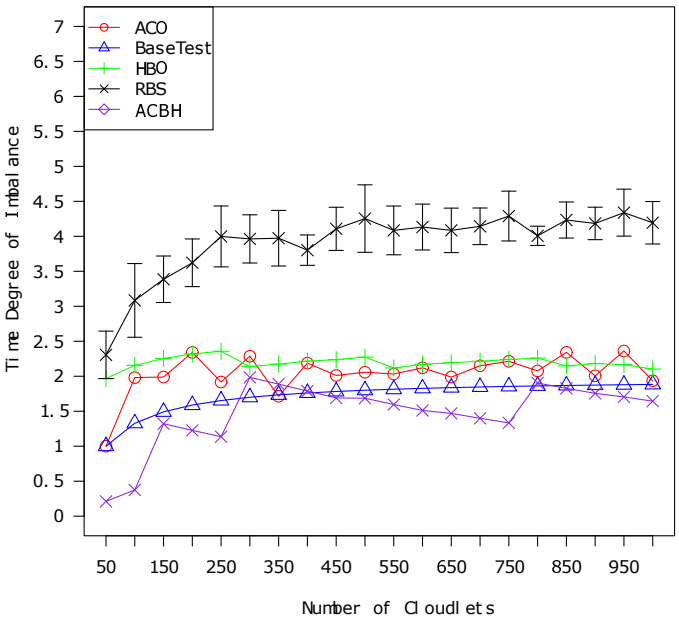

Figure 6.6: Degree of Time Imbalance in the Heterogeneous Scenarios.

thus it has a good performance when it comes to fairness. ACO has a restriction factor that prevents ants from choosing a virtual machine twice in a single ACO iteration. This restriction leads to fairness between cloudlets execution.

To calculate the cost of cloudlet processing, we took into account the bandwidth, memory, and MIPS needed. As depicted in Figure 6.7, HBO presents the best price value as it does consider the lowest processing cost when scheduling even though the load balancing factor shows an effect on the decision making but it is minimal. The Base Test and RBS have a good cost because they have no restrictions on choosing any virtual machine and they distribute the cloudlets. ACBH in the current setup 6.6 does not consider the vote from the cost factor and hence, it has a high cost. ACO has a high cost because it focuses on using the strongest virtual machines when it schedules cloudlets.

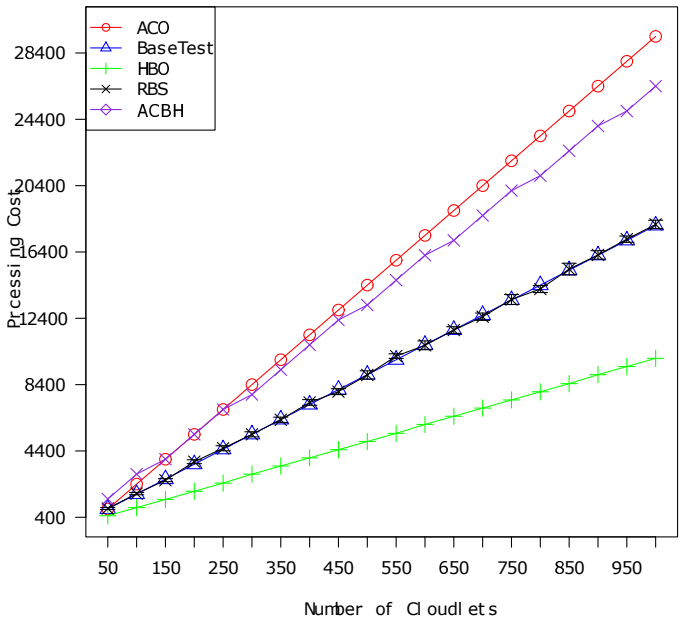

Figure 6.7: Processing Costs for Heterogeneous Scenarios.

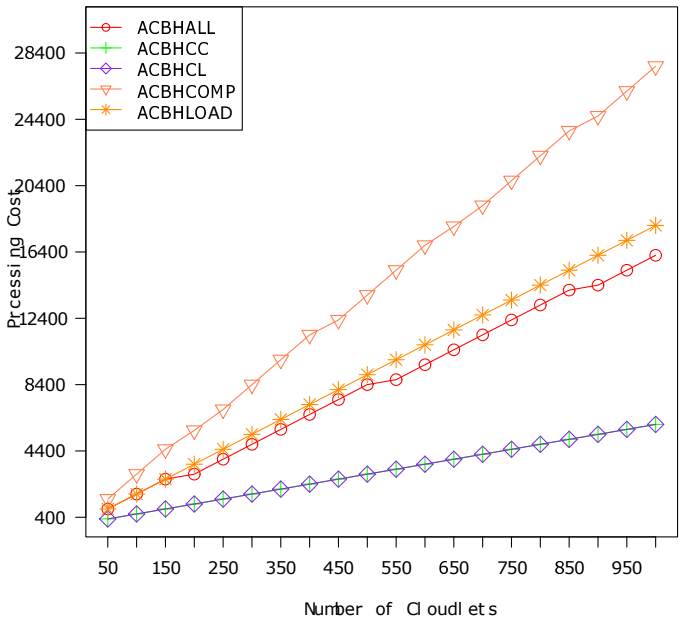

Figure 6.8: ACBH Variations Cost.

6.4.3 ACBH Variations

ACBH has the edge on the existing algorithms because of its ability to change and adapt to different optimization objectives that are decided by the cloud service provider and the cloud service subscribers. ACBH is able to change focus by a simple change to the weights of each factor. The following graphs represent ACBH ability to change by shifting focus based on the change of the controlling weights.

The cost of different variations of ACBH is shown in Figure 6.8. It shows that the ACBH ($ACBH_{COMP}$) that focus on only computation only had the most expensive solution. The previous setup is suitable for use when the requirement is to finish the tasks as fast as possible such as real-time application tasks. Also, Figure 6.8 demonstrates that the cost of $ACBH_{LOAD}$ focused and $ACBH_{ALL}$ with all the optimization objectives has a lower cost than the $ACBH_{COMP}$. The best variation in terms of cost that is even better than HBO is $ACBH_{CC}$ and $ACBH_{CL}$ that are both focused on cost as a primary optimization objective and coupled with either computation or load to ensure fairness.

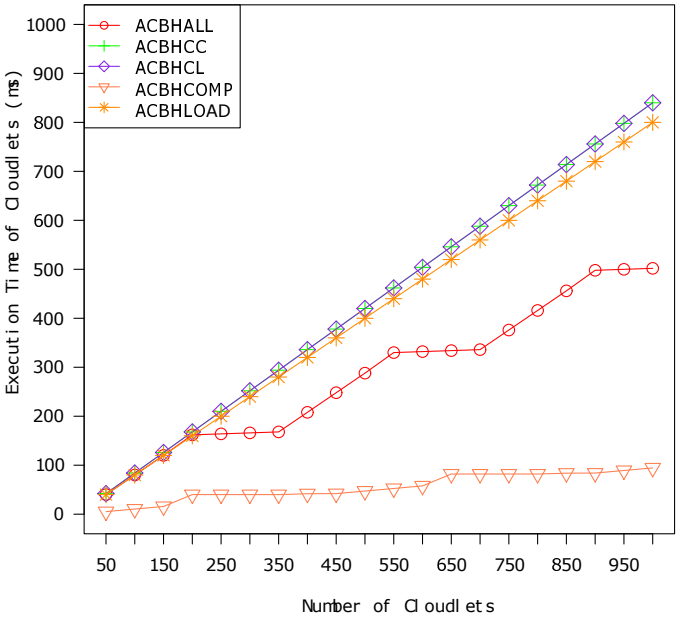

Figure 6.9: ACBH Variations Execution Time.

The ACBH variations are also evaluated for their execution time. Figure 6.9 shows the results of the simulation. $ACBH_{COMP}$ has the best (i.e. least Execution time) which is even better than all of the PBSH and the ACBH variation shown in Figure 6.4. The Second best of ACBH variation is $ACBH_{ALL}$ because the focus is on all the scheduling objectives. The last three variations of ACBH, namely $ACBH_{LOAD}$ $ACBH_{CL}$ $ACBH_{CC}$ have a similar execution time to HBO, RBS, and Base Test as seen in Figure 6.4

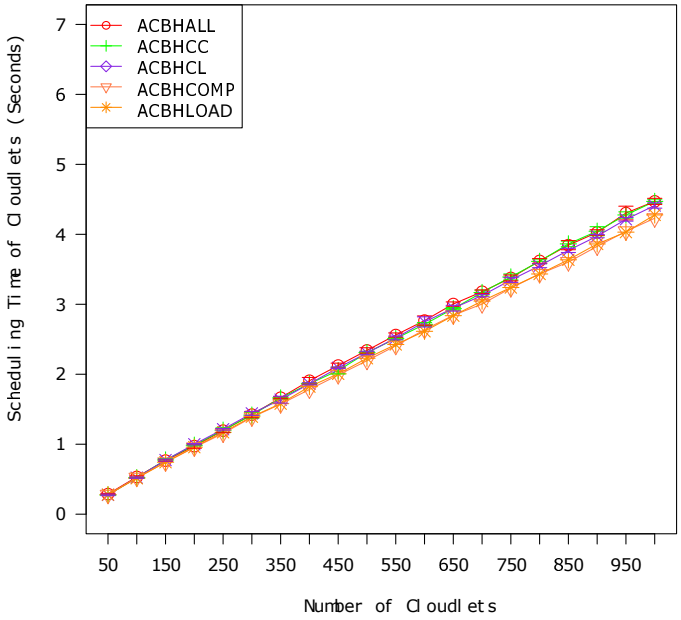

Figure 6.10: ACBH Variations Scheduling Time.

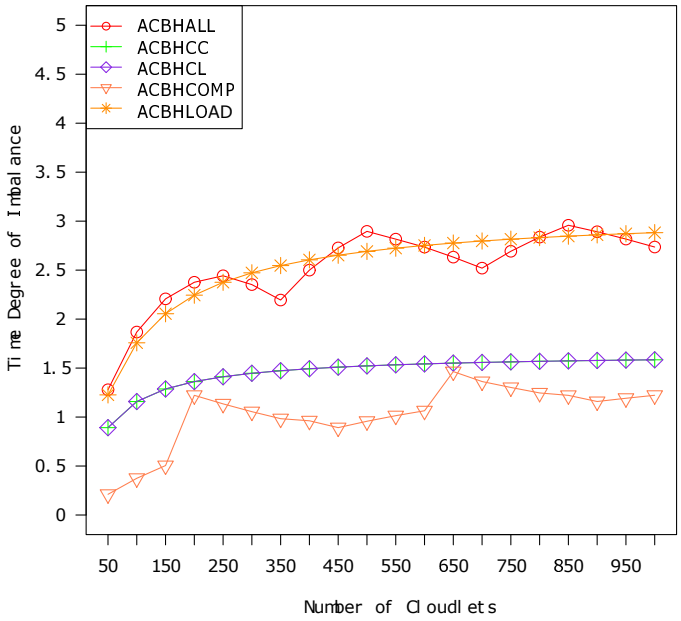

Figure 6.11: ACBH Variations Time Degree Imbalance.

All the ACBH variations have the same scheduling time as seen in Figure 6.10. For all the variations of ACBH, the algorithm has to evaluate all the deciding factors presented

in 6.6 to calculate the votes. Therefore, the variation of ACBH takes the same time to make a scheduling decision.

Moreover, ACBH variations are measures under time degree of imbalance 6.2. $ACBH_{COMP}$ has performs the best out of all the other variations because of the focus on finding the strongest virtual machines to perform the tasks which ensure that tasks have a similar execution time. $ACBH_{CL}$ $ACBH_{CC}$ have a good time degree of imbalance as seen in 6.11 that is close to the Base Test in Figure 6.6 because these two variations ensure that all the cloudlets with the least price get their fair share of tasks. $ACBH_{LOAD}$ has a slightly high time degree of imbalance because it only ensures that all virtual machines get tasks but those virtual machines have different capabilities and hence leading to the high time degree of imbalance. The previous statement is also applicable for $ACBH_{ALL}$ in terms of time degree of imbalance.

Chapter 7

Conclusion and Future Work

The last chapter of this thesis includes a conclusion of the research that summarizes the results of the simulations. Moreover, his chapter also presents directions for the future works on my thesis.

7.1 Conclusion

In this thesis, we conducted an extensive set of experimental analysis on bio-inspired scheduling algorithms, HBO and ACO, as well as Random Biased Sampling algorithm. The purpose of the analysis was to stress the algorithms at its extremes to determine the conditions in which they would perform best in a Cloud environment to come up with a superior scheduler that follow the same logic of the PBSH but is more flexible. Our experiments were divided into two different scenarios: homogeneous and heterogeneous. In the homogeneous scenario, the base test performed better than the rest of the schedulers since no advanced decision-making was necessary. In that scenario, the simulation environment was large and time-consuming.

In the second scenario, the heterogeneous, the schedulers were tested in the heteroge-

neous setup with different cloudlets demands and virtual machines capabilities. In this test, the ACBH showed the best Execution time and fairness as it's set to focus on those two factors. HBO had the minimal processing cost. However, since the cost factor in ACBH is not considered in this simulation ACBH has a high cost as does ACO. It terms of the time it takes each algorithm to make a scheduling decision for cloudlets, ACBH and ACO took the longest due to the relative complexity of both algorithms comparing to the others.

The third part of the simulation showed different variations of ACBH and demonstrated ACBH ability to adapt and change to meet different requirements. ACBH is able to alter its focus and solve different optimization objectives. Also, ACBH has the ability to pinpoint the best virtual machine not only give a set of viable solutions as in multi-objective theory (MOT). This set of solutions given by MTO is called the Perto set. This set will give the solutions that meet the constraints. However, this set will also include solutions that are meeting one constraints more that the others as demonstrated in the following works [50, 65, 66, 107, 101, 60, 59, 78, 103]. ACBH has the ability to provide a better set of solutions because it is possible to control the quality of the solutions.

From the simulations, it was noticed that finding the best weights (W_c, W_{cp},W_{ld}) seen in figure 4.1 is a challenging as it does require extensive experimental analysis to accommodate parameter values accordingly. Moreover, ACBH produces the same performance for some different (W_c, W_{cp},W_{ld}) setups as the case in Section 6.4.3, namely $ACBH_{CC}$ and $ACBH_{CL}$. Having similar performance with different setting adds to the flexibility of ACBH but also makes it harder to find the best setup to achieve the scheduling objectives. These issues will be further investigated in the near future to enhance the ACBH further.

7.2 Future Work

As future work, we will automate the factor weights selection of ACBH using machine learning which will enhance the performance of the scheduling algorithm further. Also, we will extend the ACBH to include the deployment decisions of virtual machines on datacenters. This attempt will help in solving the problem of load balancing of datacenters. Better load distribution of load will then assist in reducing migration moves of virtual machines between datacenters and hence save on the network bandwidth (less load on the local scheduler). Moreover, we plan to look at the power saving and CO2 emotions of datacenters. The attempt to build an environmentally friendly scheduler will improve the performance of the cloud platform and save cost for the cloud service providers. scheduling for fault tolerance is to be considered because it is crucial for any cloud platform. Fault tolerance is important to the cloud because it is high availability system. Interoperability across cloud platforms is a research area that is given attention lately. Therefore, a cross-platform scheduler is to be investigated. The scheduler then will be evaluated in cross-platform cloud environments. In the future, we will develop the scheduler to be fully distributed not just partially distributed. This is important to be achieved due to the size of the cloud system.

At the current stage, the ACBH supports task scheduling on virtual machines only and it has the option of being partially distributed. In the future, the scheduler will be extended to virtual machines deployment and to account for virtual machines movements between datacenters (i.e. migration moves). Also, the environment effects of datacenters will be looked at so that we can save on consumed energy. Furthermore, more efforts are to be put into investigating scheduling for fault tolerance.

Bibliography

[1] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *Intelligent Transportation Systems, IEEE Transactions on*, 12(3):717–735, 2011.

[2] O. Abumansoor and A. Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *IEEE Transactions on Vehicular Technology*, 61(1):275–285, Jan 2012.

[3] M. Alhamad, T. Dillon, and E. Chang. Conceptual sla framework for cloud computing. In *4th IEEE International Conference on Digital Ecosystems and Technologies*, pages 606–610, April 2010.

[4] Thanasis Antoniou, Ioannis Chatzigiannakis, George Mylonas, Sotiris Nikoletseas, and Azzedine Boukerche. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. In *Proceedings of the 37th annual symposium on Simulation*, page 43. IEEE Computer Society, 2004.

[5] Soumya Banerjee, Indrajit Mukherjee, and P Mahanti. Cloud computing initiative using modified ant colony framework. *World academy of science, engineering and*

technology, 56:221–224, 2009.

[6] Saurabh Bilgaiyan, Santwana Sagnika, and Madhabananda Das. An analysis of task scheduling in cloud computing using evolutionary and swarm-based algorithms. *International Journal of Computer Applications*, 89(2), 2014.

[7] A. Boukerche and R. E. D. Grande. Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems. In *Distributed Simulation and Real Time Applications, 2009. DS-RT '09. 13th IEEE/ACM International Symposium on*, pages 175–183, Oct 2009.

[8] Azzedine Boukerche, Regina B Araujo, and Leandro Villas. A wireless actor and sensor networks qos-aware routing protocol for the emergency preparedness class of applications. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 832–839. IEEE, 2006.

[9] Azzedine Boukerche and Luciano Bononi. Simulation and modeling of wireless, mobile, and ad hoc networks. *Mobile ad hoc networking*, pages 373–409, 2004.

[10] Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *Computer communications*, 29(4):477–489, 2006.

[11] Azzedine Boukerche and Amir Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM Computing Surveys (CSUR)*, 47(2):22, 2015.

[12] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Swimnet: a scalable parallel simulation testbed for wireless and mobile networks. *Wireless Networks*, 7(5):467–486, 2001.

[13] Azzedine Boukerche, Yan Du, Jing Feng, and Richard Pazzi. A reliable synchronous transport protocol for wireless image sensor networks. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 1083–1089. IEEE, 2008.

[14] Azzedine Boukerche and Caron Dzermajko. Performance evaluation of data distribution management strategies. *Concurrency and Computation: Practice and Experience*, 16(15):1545–1573, 2004.

[15] Azzedine Boukerche and Xin Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad hoc networks*, 5(8):1303–1316, 2007.

[16] Azzedine Boukerche and Xin Fei. A voronoi approach for coverage protocols in wireless sensor networks. In *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pages 5190–5194. IEEE, 2007.

[17] Azzedine Boukerche, Xin Fei, and Regina B Araujo. An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange. *Computer Communications*, 30(14):2708–2720, 2007.

[18] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A distributed algorithm for dynamic channel allocation. *Mobile Networks and Applications*, 7(2):115–126, 2002.

[19] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *Parallel and Distributed Systems, IEEE Transactions on*, 13(9):911–923, 2002.

[20] Azzedine Boukerche and Xu Li. An agent-based trust and reputation management scheme for wireless sensor networks. In *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, volume 3, pages 5–pp. IEEE, 2005.

[21] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks. *Mobile Networks and Applications*, 13(6):614–626, 2008.

[22] Azzedine Boukerche, Nathan J McGraw, Caron Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *Simulation Symposium, 2005. Proceedings. 38th Annual*, pages 259–266. IEEE, 2005.

[23] Azzedine Boukerche and Sotiris Nikoletseas. Protocols for data propagation in wireless sensor networks. In *Wireless Communications Systems and Networks*, pages 23–51. Springer, 2004.

[24] Azzedine Boukerche and Mirela Sechi M Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *Journal of Parallel and Distributed Computing*, 62(9):1476–1490, 2002.

[25] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Secure localization algorithms for wireless sensor networks. *Communications Magazine, IEEE*, 46(4):96–101, 2008.

[26] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo Freire Nakamura, and Antonio AF Loureiro. Dv-loc: a scalable localization protocol using voronoi diagrams for wireless sensor networks. *Wireless Communications, IEEE*, 16(2):50–55, 2009.

[27] Azzedine Boukerche and Richard Werner Nelem Pazzi. Remote rendering and streaming of progressive panoramas for mobile devices. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 691–694. ACM, 2006.

[28] Azzedine Boukerche, Richard WN Pazzi, and Jing Feng. An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks. *Computer Communications*, 31(11):2716–2725, 2008.

[29] Azzedine Boukerche, Cristiano Rezende, and Richard W Pazzi. Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6. IEEE, 2009.

[30] Azzedine Boukerche and Steve Rogers. Gps query optimization in mobile and wireless networks. In *Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on*, pages 198–203. IEEE, 2001.

[31] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *Journal of Parallel and Distributed Computing*, 62(3):366–392, 2002.

[32] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *Journal of Parallel and Distributed Computing*, 62(3):366–392, 2002.

[33] Azzedine Boukerche, Anis Zarrad, and Regina B Araujo. A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks. *Parallel and Distributed Systems, IEEE Transactions on*, 21(7):911–924, 2010.

[34] R. Buyya, S. K. Garg, and R. N. Calheiros. Sla-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 1–10, Dec 2011.

[35] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pages 5–13, Sept 2008.

[36] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[37] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[38] Rodrigo N Calheiros, Rajiv Ranjan, César AF De Rose, and Rajkumar Buyya. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*, 2009.

[39] Wei-Neng Chen and Jun Zhang. A set-based discrete pso for cloud workflow scheduling with user-defined qos constraints. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 773–778. IEEE, 2012.

[40] Zong-Gan Chen, Ke-Jing Du, Zhi-Hui Zhan, and Jun Zhang. Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 708–714. IEEE, 2015.

[41] Horacio Antonio Braga Fernandes De Oliveira, Azzedine Boukerche, Eduardo Freire Nakamura, and Antonio Alfredo Ferreira Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *Computers, IEEE Transactions on*, 58(5):677–691, 2009.

[42] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014.

[43] S. Devipriya and C. Ramesh. Improved max-min heuristic model for task scheduling in cloud. In *Proceedings of the International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, pages 883–888, Dec 2013.

[44] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2):243–278, 2005.

[45] Marco Dorigoa and Christian Blumb. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344:243–278, 2005.

[46] Elie El Ajaltouni, Azzedine Boukerche, and Ming Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*, pages 61–68. IEEE, 2008.

[47] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In

Proceedings of the 4th ACM international workshop on Mobility management and wireless access, pages 18–27. ACM, 2006.

[48] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 165–172. ACM, 2006.

[49] E. Feller, L. Rilling, and C. Morin. Energy-aware ant colony based workload placement in clouds. In *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*, pages 26–33, Sept 2011.

[50] Sören Frey, Florian Fittkau, and Wilhelm Hasselbring. Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 512–521, Piscataway, NJ, USA, 2013. IEEE Press.

[51] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Yang Hou, and Liang Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230–1242, 2013.

[52] Yujia Ge and Guiyi Wei. Ga-based task scheduler for the cloud computing systems. In *Proceedings of the International Conference on Web Information Systems and Mining (WISM)*, volume 2, pages 181–186, Oct 2010.

[53] Yujia Ge and Guiyi Wei. Ga-based task scheduler for the cloud computing systems. In *Proceedings of the Web Information Systems and Mining (WISM), 2010 International Conference on*, volume 2, pages 181–186, Oct 2010.

[54] Thiago AL Genez, Luiz F Bittencourt, and Edmundo RM Madeira. Workflow scheduling for saas/paas cloud providers considering two sla levels. In *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, pages 906–912. IEEE, 2012.

[55] Eugene Gorelik. *Cloud computing models*. PhD thesis, Massachusetts Institute of Technology, 2013.

[56] Gan Guo-ning, Huang Ting-lei, and Gao Shuai. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In *Proceedings of the 2010 International Conference on Intelligent Computing and Integrated Systems*, pages 60–63, 2010.

[57] Punit Gupta and Satya Prakash Ghrera. Load and fault aware honey bee scheduling algorithm for cloud infrastructure. In *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*, pages 135–143. Springer, 2015.

[58] M. Guzek, P. Bouvry, and E.-G. Talbi. A survey of evolutionary computation for resource management of processing in cloud computing [review article]. *Computational Intelligence Magazine, IEEE*, 10(2):53–67, May 2015.

[59] Mateusz Guzek, Johnatan E Pecero, Bernabé Dorronsoro, and Pascal Bouvry. Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems. *Applied Soft Computing*, 24:432–446, 2014.

[60] Mateusz Guzek, Johnatan E Pecero, Bernabé Dorronsoro, Pascal Bouvry, and Samee U Khan. A cellular genetic algorithm for scheduling applications and energy-

aware communication optimization. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 241–248. IEEE, 2010.

[61] Ligang He, Deqing Zou, Zhang Zhang, Hai Jin, Kai Yang, and S.A. Jarvis. Optimizing resource consumptions in clouds. In *Proceedings of the 12th IEEE/ACM International Conference on Grid Computing (GRID)*, pages 42–49, Sept 2011.

[62] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Proceedings of the Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 89–96, Dec 2010.

[63] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Proceedings of the Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 89–96, Dec 2010.

[64] Sung Ho Jang, Tae Young Kim, Jae Kwon Kim, and Jong Sik Lee. The study of genetic algorithm-based task scheduling for cloud computing. *International Journal of Control and Automation*, 5(4):157–162, 2012.

[65] Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. A pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2496–2503. IEEE, 2013.

[66] Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. A pareto-based metaheuristic for scheduling hpc applications on a geographically distributed cloud federation. *Cluster Computing*, 16(3):451–468, 2013.

[67] Dzmitry Kliazovich, Sisay T Arzo, Fabrizio Granelli, Pascal Bouvry, and Samee U Khan. e-stab: energy-efficient scheduling for cloud computing applications with traffic load balancing. In *Proceedings of the IEEE Green Computing and Communications (GreenCom)*, pages 7–13. IEEE, 2013.

[68] Raghavendra V Kulkarni and Ganesh Kumar Venayagamoorthy. Particle swarm optimization in wireless-sensor networks: A brief survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(2):262–267, 2011.

[69] Pardeep Kumar and Amandeep Verma. Independent task scheduling in cloud computing by improved genetic algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(5), 2012.

[70] Hai-Hao Li, Zong-Gan Chen, Zhi-Hui Zhan, Ke-Jing Du, and Jun Zhang. Renumber coevolutionary multiswarm particle swarm optimization for multi-objective workflow scheduling on cloud computing environment. In *Proceedings of the Companion Publication on Genetic and Evolutionary Computation Conference*, pages 1419–1420. ACM, 2015.

[71] Hai-Hao Li, Yu-Wen Fu, Zhi-Hui Zhan, and Jing-Jing Li. Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 870–876. IEEE, 2015.

[72] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and D. Wang. Cloud task scheduling based on load balancing ant colony optimization. In *Proceedings of the Chinagrid Conference (ChinaGrid), 2011 Sixth Annual*, pages 3–9, Aug 2011.

[73] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and D. Wang. Cloud task scheduling based on load balancing ant colony optimization. In *Chinagrid Conference (ChinaGrid), 2011 Sixth Annual*, pages 3–9, Aug 2011.

[74] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and D. Wang. Cloud task scheduling based on load balancing ant colony optimization. In *Proceedings of the Sixth Annual Chinagrid Conference (ChinaGrid)*, pages 3–9, Aug 2011.

[75] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and Dan Wang. Cloud task scheduling based on load balancing ant colony optimization. In *Proceedings of the Sixth Annual Chinagrid Conference (ChinaGrid)*, pages 3–9. IEEE, 2011.

[76] Jing Liu, Xing-Guo Luo, Xing-Ming Zhang, Fan Zhang, and Bai-Nan Li. Job scheduling model for cloud computing based on multi-objective genetic algorithm. *IJCSI International Journal of Computer Science Issues*, 10(1):134–139, 2013.

[77] Peter Mell and Tim Grance. The nist definition of cloud computing. Technical Report Special Publication 800-145, National Institute of Standards and Technology, U.S. Department of Commerce, 2011.

[78] Mohand Mezmaz, Nouredine Melab, Yacine Kessaci, Young Choon Lee, E-G Talbi, Albert Y Zomaya, and Daniel Tuyttens. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *Journal of Parallel and Distributed Computing*, 71(11):1497–1508, 2011.

[79] Haibo Mi, Huaimin Wang, Gang Yin, Yangfan Zhou, Dianxi Shi, and Lin Yuan. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 514–521, July 2010.

[80] Sunil Nakrani and Craig Tovey. On honey bees and dynamic server allocation in internet hosting centers. *Adaptive Behavior*, 12(3-4):223–240, 2004.

[81] Sunil Nakrani and Craig Tovey. On honey bees and dynamic server allocation in internet hosting centers. *Adaptive Behavior*, 12(3-4):223–240, 2004.

[82] Mohsin Nazir, Prashant Tiwari, Shakti Dhar Tiwari, and Raj Gaurav Mishra. Cloud computing: An overview. *Book Chapter of Cloud Computing: Reviews, Surveys, Tools, Techniques and Applications-An Open-Access eBook published by HCTL Open*, 2015.

[83] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K.P. Singh, N. Nitin, and R. Rastogi. Load balancing of nodes in cloud using ant colony optimization. In *Proceedings of the 14th International Conference on Computer Modelling and Simulation (UKSim)*, pages 3–8, March 2012.

[84] Horacio ABF Oliveira, Eduardo F Nakamura, Antonio AF Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 71–78. ACM, 2005.

[85] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 400–407. IEEE, 2010.

[86] DT Pham and Afshin Ghanbarzadeh. Multi-objective optimisation using the bees algorithm. In *Proceedings of the Innovative Production Machines and Systems Virtual Conference*, 2007.

[87] F. Pop, V. Cristea, N. Bessis, and S. Sotiriadis. Reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments. In *Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 772–776, March 2013.

[88] A. Quiroz, Hyunjoo Kim, M. Parashar, N. Gnanasambandam, and N. Sharma. Towards autonomic workload provisioning for enterprise grids and clouds. In *Proceedings of the 10th IEEE/ACM International Conference on Grid Computing*, pages 50–57, Oct 2009.

[89] O Abu Rahmeh, P Johnson, and A Taleb-Bendiab. A dynamic biased random sampling scheme for scalable and reliable grid networks. *INFOCOMP Journal of Computer Science*, 7(4):1–10, 2008.

[90] M. Randles, D. Lamb, and A. Taleb-Bendiab. A comparative study into distributed load balancing algorithms for cloud computing. In *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 551–556, April 2010.

[91] M. Randles, D. Lamb, and A. Taleb-Bendiab. A comparative study into distributed load balancing algorithms for cloud computing. In *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 551–556, April 2010.

[92] Yonglin Ren and Azzedine Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 2129–2133. IEEE, 2008.

[93] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. *IEEE Transactions on Industrial Informatics*, 9(1):132–141, 2013.

[94] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, 2(2):222–235, 2014.

[95] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *Vehicular Technology, IEEE Transactions on*, 60(2):656–663, 2011.

[96] Zohreh Sanaei, Saeid Abolfazli, Abdullah Gani, and Rajkumar Buyya. Heterogeneity in mobile cloud computing: taxonomy and open challenges. *Communications Surveys & Tutorials, IEEE*, 16(1):369–392, 2014.

[97] Thomas D Seeley, Scott Camazine, and James Sneyd. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28(4):277–290, 1991.

[98] Thomas D Seeley, Scott Camazine, and James Sneyd. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28(4):277–290, 1991.

[99] S. Selvarani and G.S. Sadhasivam. Improved cost-based algorithm for task scheduling in cloud computing. In *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–5, Dec 2010.

[100] Meie Shen, Zhi-Hui Zhan, Wei-Neng Chen, Yue-Jiao Gong, Jun Zhang, and Yun Li. Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks. *IEEE Transactions on Industrial Electronics*, 61(12):7141–7151, 2014.

[101] C. Szabo and T. Kroeger. Evolving multi-objective strategies for task allocation of scientific workflows on public clouds. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[102] Maolin Tang and Shenchen Pan. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Processing Letters*, 41(2):211–221, 2014.

[103] Fei Tao, Ying Feng, Lin Zhang, and TW Liao. Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Applied Soft Computing*, 19:264–279, 2014.

[104] Medhat Tawfeek, Ashraf El-Sisi, Arabi E Keshk, Fawzy Torkey, et al. Cloud task scheduling based on ant colony optimization. In *Proceedings of the 8th International Conference on Computer Engineering & Systems (ICCES)*, pages 64–69. IEEE, 2013.

[105] Medhat A Tawfeek, Ashraf B El-Sisi, Arabi E Keshk, and FA Torkey. Virtual machine placement based on ant colony optimization for minimizing resource wastage.

In *Advanced Machine Learning Technologies and Applications*, pages 153–164. Springer, 2014.

[106] Sisi Tian, Quan Liu, Wenjun Xu, and Junwei Yan. A discrete hybrid bees algorithm for service aggregation optimal selection in cloud manufacturing. In *Proceedings of the Intelligent Data Engineering and Automated Learning*, pages 110–117. Springer, 2013.

[107] Jinn-Tsong Tsai, Jia-Cen Fang, and Jyh-Horng Chou. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research*, 40(12):3045–3055, 2013.

[108] Leandro Villas, Azzedine Boukerche, Regina Borges De Araujo, and Antonio AF Loureiro. Highly dynamic routing protocol for data aggregation in sensor networks. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 496–502. IEEE, 2010.

[109] Leandro A Villas, Azzedine Boukerche, Horacio ABF De Oliveira, Regina B De Araujo, and Antonio AF Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, 12:69–85, 2014.

[110] Leandro A Villas, Azzedine Boukerche, Heitor S Ramos, Horacio ABF de Oliveira, Regina Borges de Araujo, and Antonio AF Loureiro. Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *Computers, IEEE Transactions on*, 62(4):676–689, 2013.

[111] Xiaotang Wen, Minghe Huang, and Jianhua Shi. Study on resources scheduling based on aco allgorithm and pso algorithm in cloud computing. In *Proceedings of the 11th International Symposium on Distributed Computing and Applications to Business, Engineering Science (DCABES)*, pages 219–222, Oct 2012.

[112] L. Wu, S. K. Garg, and R. Buyya. Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 195–204, May 2011.

[113] Zhangjun Wu, Zhiwei Ni, Lichuan Gu, and Xiao Liu. A revised discrete particle swarm optimization for cloud workflow scheduling. In *Proceedings of the International Conference on Computational Intelligence and Security (CIS)*, pages 184–188. IEEE, 2010.

[114] Fei Xu, Fangming Liu, Hai Jin, and Athanasios V Vasilakos. Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. *Proceedings of the IEEE*, 102(1):11–31, 2014.

[115] Jielong Xu, Jian Tang, Kevin Kwiat, Weiyi Zhang, and Guoliang Xue. Enhancing survivability in virtualized data centers: a service-aware approach. *IEEE Journal on Selected Areas in Communications*, 31(12):2610–2619, 2013.

[116] Baris Yuce, Michael S Packianather, Ernesto Mastrocinque, Duc Truong Pham, and Alfredo Lambiase. Honey bees inspired optimization method: the bees algorithm. *Insects*, 4(4):646–662, 2013.

[117] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.*, 47(4):63:1–63:33, July 2015.

[118] Zhenxia Zhang, Azzedine Boukerche, and Richard Pazzi. A novel multi-hop clustering scheme for vehicular ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*, pages 19–26. ACM, 2011.

[119] Zhenxia Zhang, Richard W Pazzi, and Azzedine Boukerche. A mobility management scheme for wireless mesh networks based on a hybrid routing protocol. *Computer Networks*, 54(4):558–572, 2010.

[120] Chenhong Zhao, Shanshan Zhang, Qingfeng Liu, Jian Xie, and Jicheng Hu. Independent tasks scheduling based on genetic algorithm in cloud computing. In *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, 2009.

[121] Jianfeng Zhao, Wenhua Zeng, Min Liu, Guangming Li, and Min Liu. Multi-objective optimization model of virtual resources scheduling under cloud computing and it's solution. In *Proceedings of the International Conference on Cloud and Service Computing (CSC)*, pages 185–190, Dec 2011.

[122] Hai Zhong, Kun Tao, and Xuejie Zhang. An approach to optimized resource scheduling algorithm for open-source cloud systems. In *Proceedings of the Fifth Annual ChinaGrid Conference (ChinaGrid)*, pages 124–129, July 2010.

[123] Kai Zhu, Huaguang Song, Lijing Liu, Jinzhu Gao, and Guojian Cheng. Hybrid genetic algorithm for cloud computing applications. In *Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC)*, pages 182–187. IEEE, 2011.

[124] Linan Zhu, Qingshui Li, and Lingna He. Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm. *IJCSI International Journal of Computer Science Issues*, 9(5):54, 2012.