MASTER THESIS

Task Scheduling Optimization in Cloud Computing Using Multi-Objective Evolutionary Algorithms With User-in-the-Loop

Author:
Ismat Marouf

Supervisor:
Dr. Abdel Salam Sayyad

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in Computing

at Birzeit University, Palestine

May 13, 2019

Approved by the thesis committee:

_____

Dr. Abdel Salam Sayyad, Birzeit University

_____

Dr. Radi Jarrar, Birzeit University

_____

Dr. Majdi Mafarja, Birzeit University

_____

Date approved:

_____

Declaration of Authorship

I, Ismat MAROUF, declares that this thesis titled, "Task Scheduling Optimization in Cloud Computing with User-in-the-Loop" and the work presented in it are my own. I confirm that:

- This work was completely done or mainly while in candidature for a masters degree at Birzeit University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

Cloud Computing platforms provide the supply of computing resources on the basis of demand. The optimization of a task workflow in cloud environments aims at reducing the overall execution time while consuming the least amount of cloud resources. Choosing a certain workflow from a number of competing schedule methods for Pareto efficient solutions can be a complex and confusing task for the cloud's end user. We used two types of workflow. The first workflow is a balanced structure as Montage, the second workflow is a CyberShake and the structure is not balanced. Those data types of workflows provided by CloudSim [15] tool. The User-in-the-Loop (UIL) aims at enabling the user to interact with and guide the algorithmic search. This study presents a multi-objective improvement with UIL approach to Cloud Workflow Optimization. In addition to the overall objective of minimizing the total execution time, we seek to minimize the runtime for each Virtual Machine (VM) and minimize the number of VMs utilized in a particular schedule. In comparison to the existing workflow planning algorithms, we demonstrate the advantage of this study approach, which allows the decision maker to save on cloud resources without detriment to the overall runtime. In this study, we utilize jMetal [23] for the wide range of metaheuristic algorithms that are implemented in it, and CloudSim [15] for its cloud simulation capability. The goal is to set the tasks workflow and their resources allocation while enabling the user to interact to focus the search around the user preferences.

The results have shown that the evolutionary algorithms are able to outperform existing algorithms. The user-in-the-loop process is able to guide the search in order to find the best optimal solution among alternative solutions available in the search space. The evolutionary algorithms are able to execute tasks in the cloud environment with a minimum number of machines, hence, the cost will be reduced too in cloud environments. The evolutionary algorithms are able to balance the executing process of tasks on machines too, which means the task will be distributed in the cloud with minimum costs of resource on those machines.

المستخلص

تخطيط جدولة المهام في الحوسبة السحابية باستخدام الخوارزميات التطورية متعددة الأهداف مع إدخال المستخدم في دورة التخطيط

اعداد : عصمت معروف

تهدف بيئة الحوسبة السحابية الى توفير موارد الحوسبة عبر الشبكة بناءا على طلب المستخدمين او الموردين . يهدف تحسين سير عمل المهام وتسلسلها في البيئة السحابية إلى انشاء خطة تهدف الى تقليل وقت تنفيذ المهام الكلية وتقليل استخدام الموارد المتاحة في البيئة السحابية. تهدف الدراسة الى تحقيق هذه الاهداف من خلال استخدام افضل الوسائل التكنلوجية المتوفرة والطرق الخوازرمية، كما ان الدراسة تنظر الى الدراسات الاخرى وما تم تحقيقه في هذا المجال من اجل الحصول على افضل مخرجات يتم التوصل اليها خلال عملية البحث. ان عملية دمج المستخدم خلال عملية تنفيذ المهام تهدف الى اشراك المستخدم في ايجاد افضل الموارد المتاحة بافضل سيرعمل من حيث تسلسل المهام، وتمكين المستخدم وقدرته على التفاعل لتوجيه مجالات بحث الخوارزميات في ايجاد افضل خطة سير عمل ومن ثم يتم تنفيذ هذه الخطة على البيئة السحابية. تقدم هذه الدراسة تطورا متعدد الأهداف ودمج المستخدم او متخذ القرار لتحقيق هذه الاهداف المقترحة. ان الهدف الأكثر شيوعيا هو تقليل وقت التنفيذ الإجمالي للمهام المتسلسلة ، بالاضافة الى تقليل وقت التشغيل لكل جهاز على الشبكة ، كما تهدف الدراسة الى تقليل عدد أجهزة المستخدمة لتنفيذ هذه المهام بالمقارنة مع خوارزميات قائمة ودراسات سابقة ومقارنة النهج الجديد المقترح بالقائم لاظهار ميزة منهج الدراسة ، والذي يسمح لصانع القرار بالحفاظ على موارد السحابة دون الإضرار بوقت التشغيل الإجمالي.

Acknowledgements

رسالة شكرا

نتقدم بجزيل الشكر والعرفان إلى جميع أعضاء الهيئة التدريسية بجامعة بيرزيت.
كما اتوجه بالشكر والتقدير إلى الدكتور القدير عبد السلام صياد المشرف على هذا المشروع
احتراما لجهوده ولما أبداه من ملاحظات قيمة واضافات لامعة.

كما ونتوجه بجزيل الشكر إلى لجنة المناقشة كل من الدكتور : راضي جرار ومجدي مفارجة
لمساعدتهم لنا في انهاء هذا البحث وملاحظتهم القيمة
خلال المراحل السابقة لهذا البحث.
وكل الشكر والاحترام إلى من ساهم في إنجاز هذا المشروع المتواضع.

اهداء

إلى كل من وهب نفسه في سبيل إيصال أمته إلى الرقي والعلياء
إلى كل معلم بذل ما في وسعه لإيصال العلم إلى تلاميذه
. الدكتور الأمين : عبد السلام صياد.
إلى حملة راية العلم في جامعة بيرزيت.
إلى كل من ساعدنا في تحقيق ما تمنيناه ونتمناه.
إلى الشموع التي ذابت لتنير دربنا.
إلى الذي وعدته أن أكون فكنت الذي مد ذراعيه جسرا فعبرت.
إلى رمز العطاء الدائم الذي ألهمني بالمسير.
أبي العزيز.
إلى من تجسدت السعادة في أحضانها وارتسمت الفرحة في عينيها.
إلى أحن وأغلى قلب إلى أسمى لحن عزفه قلبي قبل لساني.
إلى من ربتني وليداً، وسقتني من حنانها شهد المدام .
أمي الحبيبة.
إلى ينبوع الحياة الدافئ إلى الحياة وبدونها لا حياة.
زوجتي الحبيبة.

إليهم جميعا نقدم شكرنا و امتناننا ونعدهم أن نكون عند حسن ظنهم في متابعة رسالتهم التي
بدؤوها.

Contents

List of Figures

List of Tables

List of Abbreviations

UIL	**U**ser-**I**n-the-**L**oop
UOL	**U**ser-**O**ut-of-the-**L**oop
MOGA	**M**ultiobjective **G**enetic **A**lgorithms
NSGA	**N**ondominated **S**orting **G**enetic **A**lgorithms
EMO	**E**volutionary **A**lgorithm **O**ptimization
SBSE	**S**earch-Based **S**oftware **E**ngineering
GA	**G**enetic **A**lgorithm
EA	**E**volutionary **A**lgorithm
MOOP	**M**ulti-**O**bjective **O**ptimization **P**roblem
EPS	**E**xceedance **P**robability **S**core
IGD	**I**nverted **G**enerational **D**istance
PF	**P**areto **F**ront
PSO	**P**article **S**warm **O**ptimization
SA	**S**imulated **A**nnealing
DCS	**D**istributed **C**omputing **S**ystem
DAG	**D**irected **A**cyclic **G**raph
PGA	**P**arallel **G**enetic **A**lgorithms
VMs	**V**irtual **M**achines
FCFS	**F**irst **C**ome **F**irst **S**erve
ISBSE	**I**nteractive **S**earch-**B**ased **S**oftware **E**ngineering
WSA	**W**orkflow**S**im **A**lgorithms

Chapter 1

Introduction

Search-Based Software Engineering (SBSE) uses the Metaheuristic search techniques, such as genetic algorithms and simulated annealing. It seeks to find optimal solutions or a range of alternative solutions to software engineering problems [33]. Over the past decades, the way that applications deal with data, networks, and cloud has changed significantly. More complex tasks have surfaced, requiring more resources to accomplish. Thus, finding efficient task scheduling optimization and planning techniques has become a big challenge under these circumstances [50, 31].

1.1 Problem Statement

The cloud computing[52] environment has two optimization algorithm approaches; the first one works on the optimization of resources [49, 76] and the second one aims to optimize tasks in cloud computing [50, 57]. Both approaches are designed to optimize tasks in cloud computing environments. Otherwise, the resources allocation approach aims at minimizing the usage of the number of resources available at the cloud environments during operation.

The problem of cloud workflow optimization is traditionally defined as follows:

$$\begin{aligned}&\text{Given a set of required Tasks } [T_1, T_2, ..., T_m]\\&\text{and a set of available Virtual Machines } (VM_1, VM_2, ..., VM_n)\\&\text{assign an ordered subset of tasks to each virtual machine}\\&\text{each task should be assign to one virtual machine only}\\&\text{the total task execution time } Runtime_{total}\\&\text{is minimized.}\end{aligned} \tag{1.1}$$

Therefore, a set of tasks are to be assigned to the same resources in cloud environment; without balancing between resources and the total execution time for the workflow. On the other hand, the scheduling algorithms use the spec of tasks, thus, the resources in this approach will not be considered, because this approach aims at assigning these tasks to their resources with a minimum total execution time. Both approaches have consistent behavior and no diversity in their solutions. So if the workflow is repeated multiple times, very similar results will be

generated with little variability, and there are no enhancements in the solutions as the workflow is re-executed. The System model of task workflow scheduling in Cloud Computing is shown in Figure 1.1.

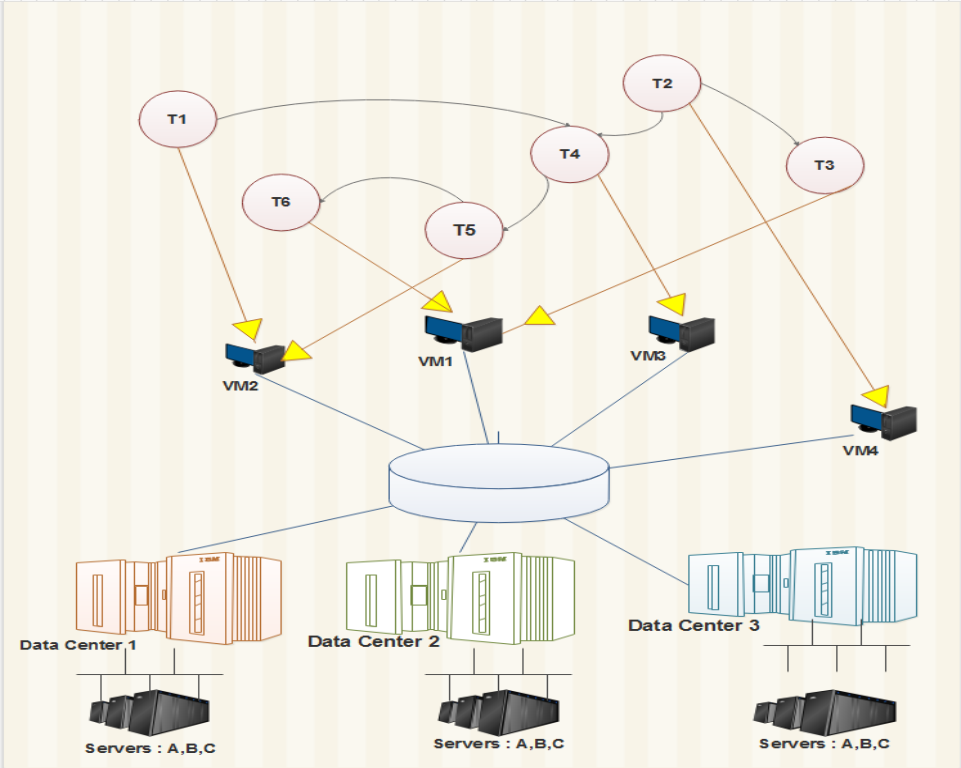


Figure 1.1: The System model of task workflow scheduling in Cloud Computing

1.2 Proposed Solution

Researches started introducing new approaches in SBSE that are called interactive approach [61, 34]. This approach means that the user is able to guide search process or make a reference point to be available in evaluating solutions. This study proposes a framework to execute a workflow of tasks in cloud computing with and without User-In-the-Loop (UIL). Meta-heuristic workflow is a new challenging task schedule, especially when these tasks are executed in a cloud computing environment that has multiple processes and a shared memory in different data centers.

The proposed framework works as follows: We need to distribute tasks to a set of virtual machines provided by Cloudsim and WorkflowSim. Then, calculate the total executing time for all tasks with the minimum resource available. We use jMetal to run multi-objective evolutionary algorithms (MOEAs) and find a set of Pareto-optimal workflows. The results from our experiment should be comparable with the traditional algorithms results in order to find the best optimal solution for the tasks scheduling problem.

The study has two phases, the first phase creates the scheduling work-flow based on metaheuristic algorithms to compare the results with the WorkflowSim results ([12]). In the second phase, we test the hypotheses and introduce the user-in-the-loop approach. The

study uses formatted Directed Acyclic Graph (DAG) files as a data set that was provided by WorkflowSim.

We used CloudSim ([15]) and jMetal ([23]) in our research in order to execute and simulate the problem to find near-optimal solutions for Task Scheduling in Cloud Computing using Heuristic Algorithms ([31]).

1.3 Impact of the proposed solution

The study presents a new approach with MOEAs and UIL. We conducted the experiments in two phases. The first phase aims at integrating WorkflowSim [12] and jMetal [23] tools to be able to execute the workflow in a Multi-Objective approach [70] and compare the results between algorithms. For the second phase we used an integrated Evolutionary Algorithms [6] with UIL in order to enhance solutions through guiding the search process of Evolutionary Algorithms. The tasks of workflows have many characteristics such as time, length, and size of input data files. The Virtual machines on cloud tools also have many characteristics such as CPU, RAM and data center characteristics. We distribute these tasks to a set of virtual machines available in a could computing environment and align them to minimize the resource allocation, thus decreasing the cost of resource usage, and allowing the user to select a region of interest to guide the algorithm search scope. The contributions of this study are:

1. Integrating jMetal and WorkflowSim tools.
2. The results of MOEAs are comparable with traditional algorithms.
3. The user is able to select a region of interest and interact with algorithm through a GUI.
4. The user is able to stop and restart the process to start the selection region of his interest through GUI.
5. The user is able to change the region selected to a new one during algorithm execution.
6. Optimizing the execution of workflow in cloud computing environment.
7. Optimizing the cost of processing tasks in cloud computing environment.
8. Optimizing the resource allocation in cloud computing environment.

1.4 Research Questions

This study seeks to answer the following research questions:

RQ1: How do MOEAs compare with traditional WorkflowSim algorithms with regards to minimizing the total task running time?

RQ2: How do MOEAs compare with traditional WorkflowSim algorithms with regards to minimizing the maximum task running time across virtual machines?

RQ3: How do MOEAs compare with traditional WorkflowSim algorithms with regards to minimizing the number of virtual machines required for a certain workflow?

RQ4: How does the User-In-the-Loop approach improve the ability of MOEAs in minimizing the three objectives?

1.5 Organization of this document

The rest of this study is organized as follows: Chapter 2 reviews background material on Cloud Computing, Traditional Optimization Algorithms, Genetic Algorithms, Crossover operation, Mutation operation, multi-objective Optimization, Interactive Search-Based Software Engineering and Quality Indicators, and multi-objective evolutionary algorithms (MOEAs). Chapter 3 introduces the related works in finding optimal solution for task scheduled problem and using reference points to guide the scope of searching algorithm. Chapter 4 introduces our framework and user-in-the-loop, in addition to existing technical tools overview, namely jMetal, WorkflowSim and CloudSim frameworks. Chapter 5 presents the research methodology, Experiment Setup and Procedures plan. Chapter 6 shows the results comparing the solutions of MOEAs with traditional algorithms, and the results when we involved user in the loop. This Chapter ends with discussing the results and the Limitation of the Study. Finally, the researcher will provide conclusions and Future Work.

Chapter 2

Background and Related Work

This chapter presents the main concepts and terminology in cloud computing and search-based software engineering. It also presents the most commonly used task optimization methods. Then, it introduces Multi-objective Evolutionary Algorithms (MOEAs) and the interactive optimization approach. Finally, we review the closely related literature in the area of cloud workflow optimization and interactive methods.

2.1 Cloud Computing

Cloud Computing is a distributed system that consists of a number of resources such as CPUs, memories, databases, networks and other hardware and software components [52]. Cloud platforms aim at optimizing the available resources allocated for all customers in demand with good quality. The cost of this allocation with good quality will be affected based on the resource allocation. Therefore, the Cloud system is useful for the applications that are time critical and depend on the user. However, the Cloud computing system has a number of core limitations. One of them is the limited number of available resources. The Cloud system provides a set of hosts such as, virtual machines (VMs) and computing servers that have a limited capacity. Furthermore, the Cloud system has two models, the public cloud and the private cloud. The two models have a different type of ownership and provide a different set of access rights. Thus, the users are able to control these resources if they own the software and resource allocations policies and techniques.

The main differences between grid computing[26] and cloud computing are: in the cloud computing the user is able to control it and has a price chargeable, that means the user owns it, obverse grid computing resource sharing the resources and doesn't have a price but it could have a quota. The cloud computing is a new technology and it is not widely used, but as perspective views technology it could be more flexible than the grid computing in managing and provisioning the resource [44].

2.2 Traditional Scheduling Algorithms

In a scheduling system, there are component process that compete for processing, where each component needs a certain execution time to process. Task Scheduling algorithm is a set of roles that aim at controlling the order of tasks that are being performed by machines.

The tasks could be inter-dependent or independent ones, which means the policies could be changed according to the type of tasks. The Scheduling Workflow tasks is not easy and there are some issues addressed as following: The process time and costs ordering tasks to execute is on domain; performance costs of these tasks when executed; The communication costs between machines and tasks; the resource availability in the domain; finally the bandwidth and capability of the network in order to execute tasks. There are various types of scheduling algorithms for distributed computing systems. Scheduling algorithms help in managing the CPU and memory availability as well as getting the maximum resource utilization [37]. The main advantage of scheduling algorithms is to achieve high performance computing and to get the optimum system throughput. Task scheduling is a process of allocating one or more time slots to one or more resources [37]. Cloud task scheduling is about assigning multiple tasks to different virtual machines. A cloud environment scheduling scheme is applied to minimize the completion time of a specific task or the makespan of a system. However, it is difficult to schedule a set of submitted tasks from different users on a shared set of computing resources. Thus, in this study we develop a scheduling algorithm that improves the process of assigning multiple tasks that have different parameters to multiple virtual machines by using the genetic algorithm (GA) in order to optimize the task planning process.

There are two approaches in task scheduling [28]. The first one is online or real time scheduling, which means the tasks will be executed immediately using the available resources without any delay, this kind of scheduling applied to dynamic scheduling which means we need to consider and utilize the resource available rather than execution running time. The second approach is offline task scheduling, which means this static scheduling tasks, and there is a time to program algorithms, also there is a time to estimate execution time. This approach enables us to prepare a provisioning plan for tasks to assign resources to them, then execute the tasks according to this provisioning plan.

The following scheduling algorithms are used for optimizing tasks [49, 76].

- First Come First Serve: tasks are executed based on a first come first serve basis. This is a non-preemptive scheduling algorithm [8]. This algorithm has low performance as each task has a waiting time in order to be executed, which could cause a high average waiting time.
- Minimum Completion Time Scheduling (MICT): it assign a set of tasks to a machine using the calculation of smallest execution time for all available tasks in workflow.
- Maximum Completion Time Scheduling (MXCT): it assigns a set of tasks to a machine using the calculation of longest execution time for all available tasks in workflow.
- Data Aware Scheduling: improves performance by accounting for the location of the task's data when executing it.
- Round Robin Scheduling: a preemptive process scheduling algorithm. Each process is given a fixed time to execute and then it is preempted to allow other processes to be executed.

The following scheduling algorithms work in heterogeneous cloud environment to optimize the workflows [68]:

- Heterogeneous Scheduling with Improved Task Priority (HSIP) [30]: The study proposed to calculating the standard deviation of communication cost between edges instead of the mean value in algorithm to determine the priority of tasks, then re-assign tasks to their resources if the resource is needed by another process.
- Predict Earliest Finish Time (PEFT) [4]: the study presents a new approach based on optimistic cost table (OCT). The study claims that this is the first algorithm to outperform HEFT for a workflow that has the same time complexity.
- Heterogeneous-Earliest-Finish-Time (HEFT) [69]: Is one of heuristic algorithms [42] and it is able to work with heterogeneous resources and schedule tasks based on execution time. It also considers the communication time between these tasks. The algorithm ranks the tasks according to the calculated values and assigns them to the resources available in the cloud environments starting with the minimum time. Thus, the HEFT has two processing steps: first the rank process in order to set priority for the task to be executed, and this process calculates the communication costs between edges and cost of tasks according to critical path then rank these tasks. The second process is the selection process, which aims at assigning tasks to it, which means each tasks will be known on which machine it has to be executed. The process calculates the earliest complete time for a given task. HEFT doesn't consider the network topology and bandwidth allocation on environment.

Algorithm 1 Pseudo-code Heterogeneous-Earliest-Finish-Time

$Calculates$ the Computation costs of tasks and
costs of edges with mean values.
$Rank(U) All\_tasks$ using traversing graph upward through
starting from the exit task.
$Sort\_tasks\_in_s cheduling\_list(SCHLIST)$ by non-increasing order of
Rank(U).
while $TasksNotSchedulein \leftarrow SCHLIST$ **do**
  $Selectjobi \leftarrow SCHLIST(TasksNotSchedule)$ get job ID for
scheduling
  for $eachmachinek$ **do**
    $ComputeEST(i,k) value$using insertion-based scheduling
policy
    $Assign_t hejob(i)\_To\_machine(j)$ that minimized EFT of job i.
  end for
end while

- Distributed Heterogeneous-Earliest-Finish-Time (DHEFT) [12]: is a new version of HEFT. The aim of this algorithm is to optimize the communication cost instead of the average time as calculated in HEFT. The HEFT planning assigns tasks according to the ranking process for that tasks as explained, which means we know the cost of each task before starting the algorithm. First, DHEFT finds the costs of executing tasks

with no priority. Then it create a list of tasks that has all tasks ready to be executed on machines. Then DHEFT checks if a task is ready and all parts that belongs to this task are ready to use, then, starts the estimated execution time for that task, if not, then assigns the task to this machine and go to the next task to execute it, Every time the algorithm checks if this task is ready to be executed with their parts or not, hence, there is no need to wait the resources, so, it will list machines with tasks with less computation time, and less waiting time.

- Robustness Heterogeneous-Earliest-Finish-Time (RHEFT) [68] is schedule algorithm developed for independent tasks, also, RHEFT is a new algorithms combined between two algorithms HEFT and DHEFT.

2.3 Genetic Algorithm

Genetic algorithm (GA) [19] is a meta-heuristic algorithm built to use natural selection, and is a type of evolutionary algorithms (EA). It generates a random population then evolves this population of individuals in stochastic iterations (generations). GA was developed to emulate the natural optimization principles occurring in living organisms. The Genetic algorithm uses a population to find and optimize the best nearest solution. The initialization of the population size depends on the problem type. Then, the algorithm creates the solutions randomly with their features as chromosomes. The solutions could be presented in binary (i.e., bits in array), or in other format types such as, string or integer.

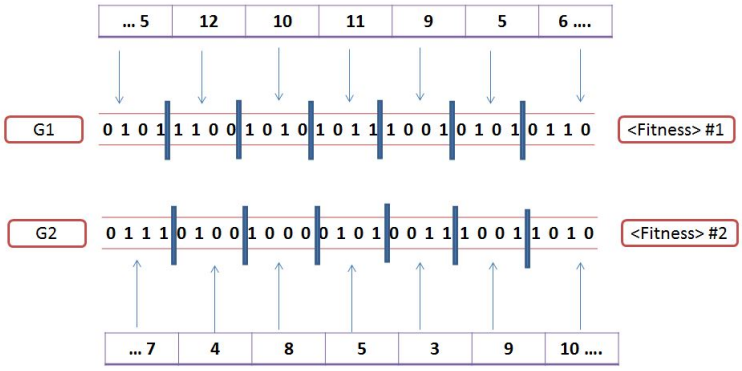


Figure 2.1: Genetic algorithms 4-bit chromosomes

Algorithm 2 Genetic algorithm pseudo-code

$StartTime \leftarrow LocalMachineStartTime$
$StopCond \leftarrow Cond_Time =$
$StopDone \leftarrow false$
$Popl \leftarrow CreatetheInitialofPopulation(PopulationSize \leftarrow PopulationSize)$
$Popl \leftarrow StartTheEvaluationofPopulation(Popl)$
$Popl \leftarrow AssignTheFitnessValueToEachMember(Popl)$
while $StopDone \neq True$ **do**
  $StartANewGeneration \leftarrow selectMembersOfPoplForCrossover(Popl)$
  $NewGeneration \leftarrow MutatePopulation(NewGeneration)$
  $Popl \leftarrow AssignTheFitnessValueToEachMember(Popl)$
  $EndTime \leftarrow LocalMachineTime$
  $Stop_Time \leftarrow EndTime - StartTime$
  if $Stop_Time >= StopCond$ **then**
    $StopDone \leftarrow True$
  end if
end while
$PrintFeasibleAfterEvalutaion \leftarrow ThefittestMemeberOfPopulation(Popl)$

2.3.1 Selection

The genetic algorithms (GA) are heuristic search algorithms influenced by the evolutionary ideas of natural selection and genetics. The GA algorithms generate solutions to the optimization problems using techniques inspired by natural evolution, such as the mutation, the selection, and the crossover. In general the algorithm starts with a randomly generated solutions set (individuals), each one of these individuals represents a feasible solution in the problem search space. The solution evolves over a number of generations based on some reproduction plan especially the crossover and the mutation. After each generation the individuals are evaluated based on some fitness functions. The individuals for the next generation are selected based on a selection policy and the fitness value, Also, the individuals that have higher fitness values will be given a higher probability in offspring process. The final step is reached when no improvement over a number of generations is observed, and at this step the final solution is considered to be the best solution. The GA allows to create sampling of search space, thus, the developer will be able to guide the search process using the previous sample space.

In fact the selection process evaluates the fitness of each individual and choose the best ones among all of them, while the crossover operation merges two individuals to provide new ones, finally the purpose of mutation allows moving each solution to one of its neighbors in order to maintain a good diversity during the process of optimization.

2.3.2 Crossover operation

The crossover operator merges the orderings of two parent elements. In a crossover operator GA uses two chromosomes in order to create the next generation of solutions. This means that the first two chromosomes are considered as the parents and the next set of created chromosomes are considered as the children, which means the chromosome will be a parent and

create from this chromosome his child's. Crossover is important for improving the population by exchanging the quality of high blocks. it inherits the value information to children from the parents, which means the sequences are combined from parents, thus, the information will pass to their children in one string. The common Crossover used are [6, 2].

- One-point crossover: It is a process to selects a random position point for parents, then the two parents will exchange all their bits with all information after that position.
- Two-point crossover: It is a process to select a random position two points for parents, then the two parents will exchange all their bits with all information after that position.
- k-point crossover:It is a process to select a random position k points for parents, then the parents will exchange all their bits with all information after that position.
- Uniform crossover:it exchange alleles with a given probability for each selected position points.

2.3.3 Mutation operator

Mutation is one of a genetic operator used to keep the genetic diversity between generations. Mutation modifies one or more gene values in the chromosome and changes its initial state (i.e., the mutation operators change some positions in an element). Thus, applying mutation more than one time will improve the GA solution. In mutation a permutation between two variables is performed, the two variables will be selected randomly according to probability defined, after that the two variables are swapped. The mutation probability should be kept low, as having high value of it could turn the problem into a primitive random search problem. Mutation operator is to allow moving from the current solution to the neighboring solutions. It is considered as a good step to find new solutions from two different variables. Because the initial population is randomized the mutation effects has the impact at the end of the process rather than in the beginning, flipping bits and the beginning will not have a great impact, while at the end it can change the things dramatically after the population converges. In our study we are using Binary Mutation that means each bit of offspring has a probability (mutation rate) to flip (0-1 or 1-0).

2.4 Multi-Objective Optimization

The optimization problem is finding acceptable optimal solutions for all objectives with their fitness functions. Multi-objective optimization problem maps a set of decision variables to a set of objective values, then, finding a vector from the decision space variables, which satisfies the objective functions and the constraints. Equation (2.1) is the formal equation for Multi-objective optimization problem (MOOP) [21, 1]:

$$\begin{aligned} &\underset{x}{\text{Minimize}}\ F(x) = [f_1(x), f_2(x), ..., f_M(x)] \\ &\text{subject to} \\ &G(x) = [g_1(x), g_2(x), ..., g_J(x)] \succeq 0, \\ &H(x) = [h_1(x), h_2(x), ..., h_K(x)] = 0, \\ &x_i^L \preceq x_i \preceq x_i^U, i = 1, ..., N. \end{aligned} \tag{2.1}$$

where

x = $(x_1, x_2, ..., x_N)^T$ and its a vector of the N decision variables.

M is the number of objectives f_i.

J inequality and K equality constraints.

x_i^L and x_i^U are respectively the lower and upper bound for each decision variables x_i.

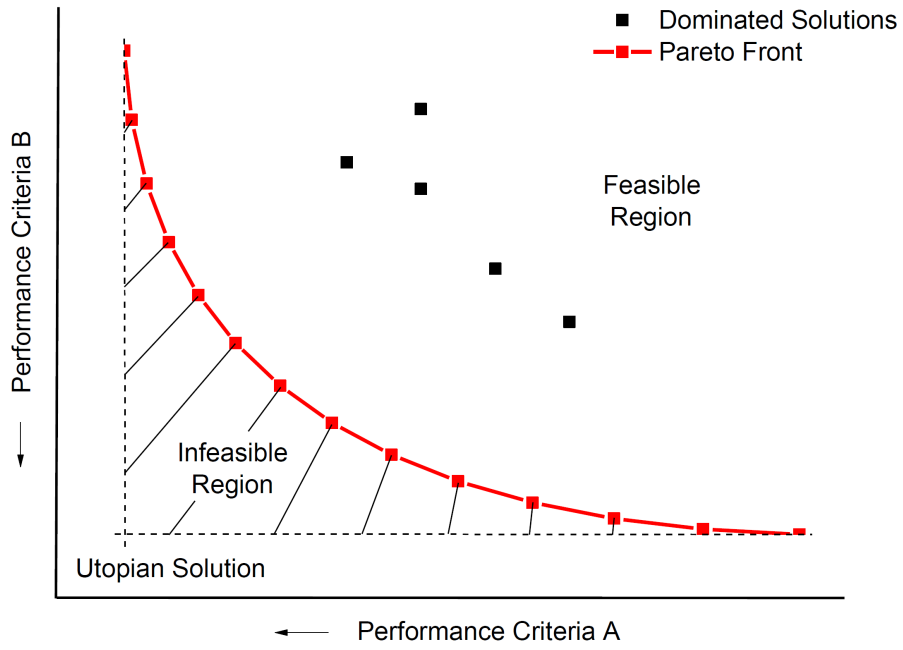


Figure 2.2: Multi-Objective Optimization [14]

The Pareto Front is the set of all non-dominated solutions. Non-dominated solutions represent those solutions that are the best according to at least one of the optimization objectives.

2.4.1 Search and Decision Making

There are two approaches to solving MOOP: The search process and the decision makers (DM) [35, 38]. The first approach starts to find optimal solutions, and these solutions will be feasible to end user, then, the user will be able to select the feasible solutions. The second approach creates a set of decision points, then the algorithm start using that points of decision as a reference through searching process to find optimal solutions. According to the optimization problem and how the decision process will be combined, the multi-objective optimization methods can be categorized as follows:

- The decision created before the search process started: which means the objectives of the MOOP are aggregated into a single object based on the reference point given by DM.

- Start search process before creating the decision: the algorithm starts the search process to find optimal sets of candidate solutions, then enabling the user to choose which solution is optimal.

- The decision is created during the search process: The DM is able to create the preferences while the search is running, by using the current state of optimization to guides the search and trade-off between objective.

2.4.2 Quality Indicators

Quality indicators are numeric measures of the coverage and diversity of solutions in the Pareto Front. The quality indicators below are used in our study for all algorithms used:

- Spread ([70]) indicator uses the members of Pareto Front set on the boundaries for its calculations. The linear scalarized algorithm obtains two (extreme) results, which are the results that have the highest distance value from each other (i.e., located far away from each other). The result is the best generalized spread value and maximized the distance metrics to measures how these elements far from non-dominated vectors.

- Inverted Generational Distance (IGD) ([54]) is used to calculate Euclidean distance between Pareto optimal and maximize the distance metrics to measure how far the elements in the set are from non-dominated vectors. In some cases we will find the value of IGD equal zero, which means all solutions created has been covered in search space.

- Epsilon quality indicators ([77]), it does not normalize the evaluation values. It applies the crowding distance metric among evaluation values. However, for the binary additive the epsilon indicator gives the minimum sum and exceeding probability score (EPS). Then, each vector in B can be added to every objective, thus, resulting approximation set is weakly dominated by A.

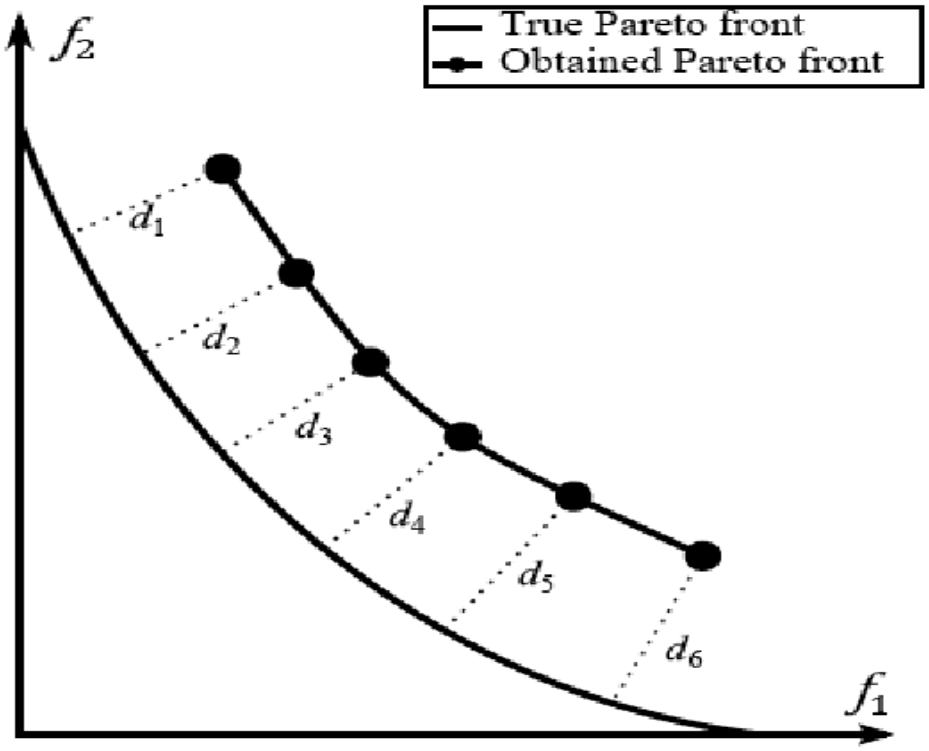


Figure 2.3: Epsilon Quality Indicator [64]

- Hyper-volume ([54]) indicator provide the hyper-volume between the estimated Pareto front (black) as a new solution and a reference point (red).

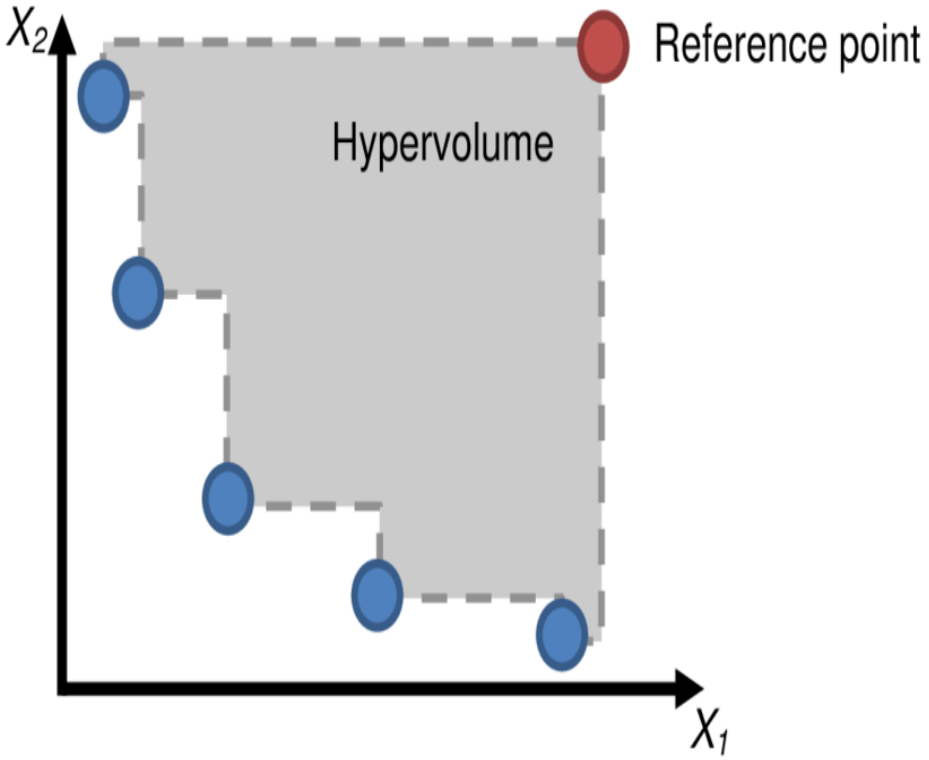


Figure 2.4: Hypervolume Quality Indicator [3]

2.5 Interactive Optimization

Interactive Search-Based Software Engineering (ISBSE) is a new approach that enables the user to interact with the search while the algorithm is running [48]. This approach helps in getting the optimal solution by controlling the crossover especially for GA and assess the experience by adding the user input to the search process. This approach is used in Software Testing [48, 46, 55, 18], where the users are allowed to interact with the system in order to guide the evolution of the search-based solutions. The results show that the users can affect the final outcome, however, there is some limitations as there is a need for expert users to be involved in the search process.

2.6 Exploration and Exploitation

The population based algorithms as NSGA-II, MOCell and IBEA aims at addressing the exploration and exploitation search space [17]. Exploration is a process that aims at visiting a new area of search space and find the best optimal points in that area. The Exploitation is a process aims at visiting the neighborhood of the visited area. The EA such as Genetic Algorithms is a search based algorithm. This means these algorithms looking to archive for a trade-off between Exploration and Exploitation. According to [17], the EAs are more effective and they have the best trade-off ration in Exploration and Exploitation processes. The good EAs could be developed with good understanding Exploration and Exploitation. The EAs aims at controlling and finding a balance between Exploration and Exploitation processes. The Exploration and Exploitation create a different approach and algorithm. The Exploration and Exploitation affects when the EAs start to find and use the best selection mechanism and crossover mutation in order to solve the problem, that means if we have high crossover and mutation rates it will gives high explore, but this is not a general role we need more experiments to find the ration between these parameters. Therefore, we tested our problem many times using different values for crossover and mutation to get the best trade-off between Exploration and Exploitation processes. The Exploration and Exploitation could be affected by population size and the representation of individuals, for example, if the population size is too large, the search space is explored more than with a smaller population size. The controlling of all parameters and setting in EAs will be considered to find the best ration between Exploration and Exploitation process, which means If the rates of each of crossover and mutation are too large, thus much of the space will be explored, but there is a high probability of missing good solutions and of failing to exploit existing solutions.

2.7 Multi-Objective Evolutionary Algorithms (MOEAs)

Many algorithms have been developed to solve multi-objective optimization problems. The most of MOEA approaches based on Pareto selection [16] that elite solutions to be used and improve the solutions to guide the search process [22]. The MOEAs create a set of non-dominated solutions called Pareto Fronts. The MOEAs aim to enhance those solutions,

and cover all possible solutions with among diversity between these points in the search space or scope. The main difference between multi-objective optimization algorithms is how the selection process, evaluating solutions, crossover, mutation and how these algorithms candidate non-dominated solutions to compare it in the next generation of the population. The Second population will create Pareto solutions and creates another generation and so on to find the optimal solutions among generation.

2.7.1 Multi-Objective Cellular genetic algorithm (MOCell)

Multi-Objective Cellular algorithm (MOCell) [56] is one of EA and it selecting two parents from defining neighborhood and the distance between them. MOCell creates non-dominated points in an external archive. The MOCell uses the archive to compare the non-dominated points with the new generated solution with existing solutions as the pseudo-code of algorithm below 3. MOCell ranking the solutions in archive area according to crowding distance to ordering these non-dominated solutions, then if the new solutions are worse than the existing then MOCell will replace it with the best one from the archive. The MOCell checking the size of external archive to make sure it will not be full, these concerns solved by removing the worse solutions from archive if they are worse than the optimal solutions.

Algorithm 3 Pseudocode of MOCell

```
Pareto front = Create Front()                    ▷ Creates an empty Pareto front
while StopCond ≠ True do
    for individual = 1to ← MOCell.popSize() do
        n_list ← GetNeighborhood(MOCell, position(individual));
        parents ← Selection(nlist);
        offspring ← Recombination(MOCell.Pc, parents);
        offspring ← Mutation(MOCell.Pm, offspring);
        Evaluate_Fitness(offspring);
        Insert(position(individual), offspring, MOCell, auxpop);
        Insert_Pareto_Front(individual);
    end for
    MOCell ← popauxpop;
    MOCell ← popFeedback(MOCell, ParetoFront);
end while
```

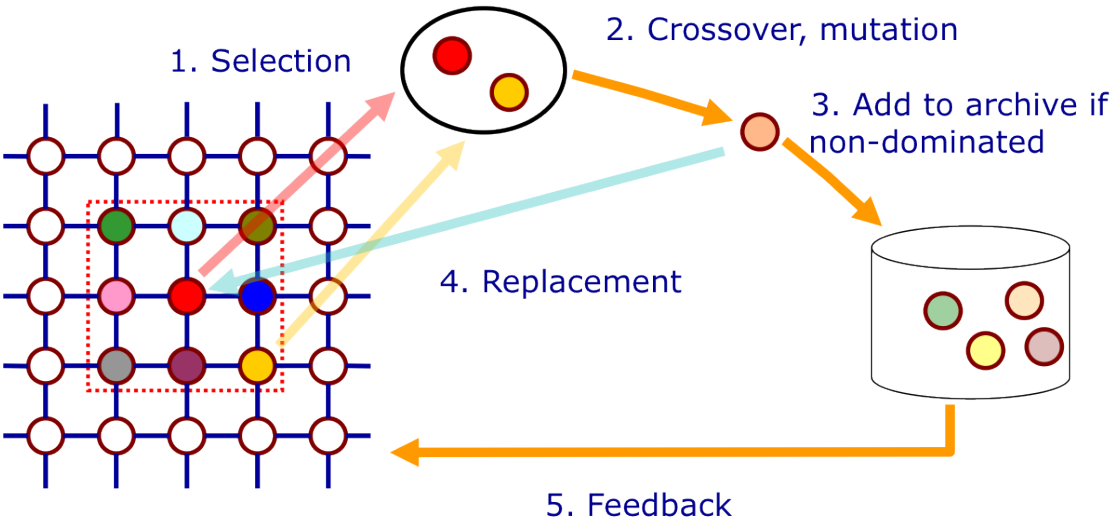


Figure 2.5: MOCell Process flow steps [23]

2.7.2 Indicator-Based Evolutionary Algorithm (IBEA)

IBEA [77] is one of the EAs comparing the individuals using a quality indicator to calculate the fitness values. So the fitness of individuals selected by the value of quality indicator calculated to that individual and the user able to be preferences, so, there is no need to create a diversity of populations and their fitness to be shared and used. The IBEA enabling to use various binary indicators and it works to eliminate the worst individual solutions from the population sets, then updating the fitness values of the remaining individuals.

Algorithm 4 IBEA Pseudocode

$Initialization$ Generates an initial population *P with N individuals*
while $StopDone \neq True$ **do**
  $Fitnessassignment$ Calculates the fitness values using the quality indicator
  $Enviromentalselection$ Until the size of *P does not exceed N,*
  removes the individual with the smallest fitness value,
  and recalculates the fitness value of the remaining individuals
  $Matingselection$ Performs binary tournament selection with replacement on *P,*
  in order to fill the temporary mating pool P
  $Variation$ Apply recombination and mutation operators to the mating pool *P,*
  and add the resulting offspring to*P*
  $EndTime \leftarrow LocalMachineTime$
  $Stop_Time \leftarrow EndTime - StartTime$
  if $Stop_Time >= StopCond$ **then**
    $StopDone \leftarrow True$
  end if
end while

2.7.3 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA [22] is an EA algorithm developed to improve the candidate solutions of population. The NSGA-II is a new version of NSGA. It was developed to support multi-objective problems to minimize the cost function. The population is sorted into a hierarchy of sub-populations based on the ordering of Pareto dominance. Within each order, the solutions are ranked according to the crowding distance metric.

Algorithm 5 Pseudocode for NSGA-II

$Initialization \leftarrow (PopulationSize, ProblemSize)$
$EvaluateAgainstObjectiveFunctions(Population)$
$FastNondominatedSort(Population)$
$Selected \leftarrow SelectParentsByRank(Population, PopulationSize)$
$Children \leftarrow CrossoverAndMutation(Selected, P\_mutation, P\_crossover)$
while $StopDone \neq True$ **do**
  $EvaluateAgainstObjectiveFunctions(Children)$
  $Union \leftarrow Merge(Population, Children)$
  $Fronts \leftarrow FastNondominatedSort(Union)$
  $EvaluateAgainstObjectiveFunctions(Children)$
  $CrowdingDistanceAssignment(Fronts); ParentsMerge(Parents, EachFront)$
  $SortByRankAndDistance(Fronts)$
  $Selected \leftarrow SelectParentsByRankAndDistance(Parents, PopulationSize)$
  $Population \leftarrow Children$
  $Children \leftarrow CrossoverAndMutation(Selected, P\_mutation, P\_crossover)$
  if $Stop_Time >= StopCond$ **then**
    $EndTime \leftarrow LocalMachineTime$
    $Stop_Time \leftarrow EndTime - StartTime$
  end if
  if $Stop_Time >= StopCond$ **then**
    $StopDone \leftarrow True$
  end if
end while

2.8 Related Work

This section discusses the related work in cloud workflow optimization, as well as related work in Search-Based Software Engineering with reference point and user-in-the-loop.

2.8.1 Cloud Workflow Optimization

Research in task scheduling whether independent or inter-dependent started many years ago [29, 10, 43, 62, 39]. During this period a lot of researchers started to find and solve the problem in both directions theoretical and technical experiments by providing some algorithms and modules for the scheduling tasks [25, 27, 9, 67, 59]. Grid computing was introduced as a new technology [11], which meant the same problem presented again but the environment is different, so the same problem should be solved using the grid technology with new approaches such as Meta-heuristics algorithms. [32, 74, 60, 11, 47, 58]. The researchers

started to find and rebuild their algorithms and modules to serve this technology and make their tools compatible with the new technology. Now cloud computing has been developed and it poses similar challenges. As a researcher, we have to solve the same problem with this technology and provide new approaches to solving it, using more flexible solutions [7], because the problem still active and needs more efforts and research to solve it as well in the dynamic environment.

Page and Naughton [57] investigated dynamic task scheduling in heterogeneous computing using GA. In their experiment some tasks were executed, while other tasks were listed in the queue to be presented for a processor in the future. This means that for each processor, the scheduler creates a list of tasks to be executed.

Jooyayeshendi and Akkasi [41] showed how to balance the load of multiple tasks on Distributed Computing Systems (DCS). They used the load-balancing algorithm framework to calculate node power for each system. They found a different rate of processing for each task because of the processors heterogeneity. The results showed that the algorithm run-time is increased linearly when the number of tasks increase. They also found that GA is better for load balancing compared to Simulated Annealing (SA).

Zhang et al. [76] introduced scheduling algorithm in grid computing. To solve it, they used particle Swarm Optimization (PSO) [24] and GA [19]. The study aimed at getting a schedule with minimum execution time and resources of tasks. Their results showed that the Genetic algorithm and the Simulated Annealing algorithms spent more time when the number of tasks increased. However, the problem was solved using both algorithms.

Zhan et al. [75] suggested mixing Particle Swarm Optimization (PSO) [24] and Simulated Annealing (SA) algorithms [19] to get a hybrid scheduling algorithm. They used CloudSim [15] to simulate the problem in the cloud computing environment. they used GA to get more convergence between solutions and help PSO to jump out of local optima and avoid sinking into the local optimal solution early. Their results showed that the Genetic algorithm and the simulated Annealing algorithms spent more time when the number of tasks increased. However, the new hybrid algorithm outperforms other algorithms.

Huang and Jie [36], executed workflow in cloud environment to minimize the cost of service in cloud computing environment under costs and time constraint using Genetic Algorithms. They used Workflow-sim to compare results obtained from GA with HEFT and MICT algorithms. The results showed that GA outperformed HEFT and MICT in optimizing scheduled tasks.

Yu et al. [73] introduced the same problem but in grid computing, and they proposed to solve it using GA after comparing the results with HEFT and greedy cost (GC). The GC approach seeking to minimize the execution cost for workflow while binding tasks to their resources available at the grid environment. They called these algorithms non-heuristic algorithms. To conduct the experiments, they used GridSim [73] seeking to minimize the cost and time of tasks for workflow with deadline constraint. The study used two workflow data types and non-balanced structure workflows. According to the results shown in the study, the GA outperformed the non-heuristic algorithms in complex workflow structures.

Jena and RK [40] used PSO algorithm to solve the resource optimization problem in cloud computing. The study used Cloud-Sim and they integrated PSO with EAs to get a good spread of solutions. The tasks were distributed to more than one Data center and each user was limited to assign their tasks to one Data center only. The study confirmed the ability of PSO to solve the task schedule problem in cloud computing in multi-objectives and the effectiveness of PSO to optimize the resources.

Srichandan et al. [66] introduced a new bacteria foraging algorithm (BFA). The BFA was integrated with GA in order to use a crossover and mutation operator and solve the problem in multi-objectives in cloud computing. The study supposed the resources are highly heterogeneous in the cloud environment. The study intended to minimize the execution time per machine and the commutation bandwidth cost among machines and Data centers. The study used the completion time task per request from the user which means the time per task was calculated from the total previous tasks to the current task. The study also defined the energy consumption model as a second objective. The researchers aimed at achieving load balancing, so they calculated the ideal ratio when starting the initial load distribution of tasks, but that was too risky because the machine could be too busy with other tasks when they calculated this ratio. According to the results shown in the study, the GA outperformed PSO. Notably, the study did not handle the dependency between tasks.

2.8.2 User-In-the-Loop Optimization

Deb and Kumar [20] presented a new approach to improve NSGA-II to get near-optimal solutions for a given problem. In this approach, the user selects one or more efficient solution sets. The selection of point from solutions set to be a reference for multiple solutions in multi-objective search for each population in each iteration. Then, the algorithm searches to find the best solutions. After that, the algorithm classifies all vectors as dominated and non-dominated. This is done after calculating crowding distance among vectors for other populations. That means when the user selects a point the search is redirected to search for all solutions closer to that point. Therefore, all Pareto fronts will be covered by this procedure. Thus, getting some points from the user can help other objectives and improve many objectives.

De Souza et al. [65] compared user performance in search algorithms for specific problems. They found that the users can be competitive in some cases. Thus, from their results we found that the user can help to guide the search algorithm, which evaluates the best optimal solution. Therefore, adding the user in the search process helps in decreasing the required time to evaluate and eliminate the solutions that the user does not want.

Yamany et al. [72], built on a previous study [63] about the configuration of software product lines. They presented a tool, which enables the user to select some options in order to find the best fitness for optimal solution. The selection is set as a reference when the algorithm restarts the search algorithm process for the optimal solutions. The user is included as part of the algorithm, and they used indicator-based evolutionary algorithm (IBEA) [77] since it proved to be the best method for this problem when compared to other algorithms.

The results of the study showed that the users can guide the search process and add value if we enable them to be part of the search algorithm.

Nebro et al. [55] proposed a new approach as extension to SMPSO. It allows the user to interact with non-dominated archiving points for PSO. This interaction will be considered during the algorithm execution and it could be changed by the decision marker. Thus, the reference is used to effectively focus the search on one or more regions of interest. The results showed that the new approach outperforms other traditional algorithms.

Longmei et al. [46]) produced a new paradigm modeling to prove that the preference point model can be used to allow focusing search on the interesting part of the Pareto front.

Ramirez et al. [61] referred to "human-in-the-loop" [18] approaches and the way of interaction to provide the user with current state, which enables the user to make his decision and choose realistic solutions. The study presented the interactive mechanisms used to handle many objectives and balance the trade-offs among these objectives. The human guided search is one of these interactive mechanisms. The study presented the classification of user interactive and search technique and showed the importance of SBSE and user interactive approaches with search process and how many publication published in this filed. According to that classification we use a mix of two approaches: Interactive re-optimization and Preference-based interactivity.

Li et al. [45], introduced a new approach "Target region-based" to solve the problem of Satellite Earth Observations. The study used Evolutionary Algorithm (EA) with users providing a target region. The results showed the user is able to affect the search scope. Also, the results gave an advantage of using NSGA-II, which is the fastest algorithm among MOEAs in the experiment.

2.8.3 Summary

This literature review shows that the task scheduling problem has been a common problem for many years, especially in cloud computing. Although evolutionary algorithms were used to tackle task scheduling, the previous studies didn't present the differences between work flow types and the could computing environment or dynamic machines. Furthermore, no previous study utilized the User-In-the-Loop approach to help guide the evolutionary search.

In our study, we compare multiple multi-objective evolutionary algorithms (MOEAs) and MOEAs with traditional algorithms. Then we enhance the search process by adding a user in the search process, and getting input from him/her to find the best optimal solutions. Then, we validate the results by comparing them to optimization results of existing algorithms.

The next chapter will present our methodology and the new Cloud Task Scheduling framework with user-in-the-loop, also, we will review existing tools has been integrated to our framework in order to conduct our experiment.

Chapter 3

Proposed Cloud Task Scheduling Framework

In this chapter, we present the framework and how it was built to allow multi-objective task scheduling for cloud computing, then, we review the existing tools which we utilized to build our proposed framework.

3.1 Proposed Cloud Task Scheduling Framework

In this study, we integrate existing tools and add some classes to these tools to run our Framework. The following are the changes added to the existing tools:

- MainClass: it has most of the settings needed to conduct our experiment.
- jMetal: It has most of the meta-heuristic algorithms and setting needed to conduct our experiment.
- WorkflowSim : We created new algorithms (metaheuristic) extended to BaseSchedulingAlgorithm.

  Figure 3.1 presents the integration process flow of workflow scheduling algorithm tasks. The results expected from this process find the best solution for the tasks scheduling problem in cloud computing.

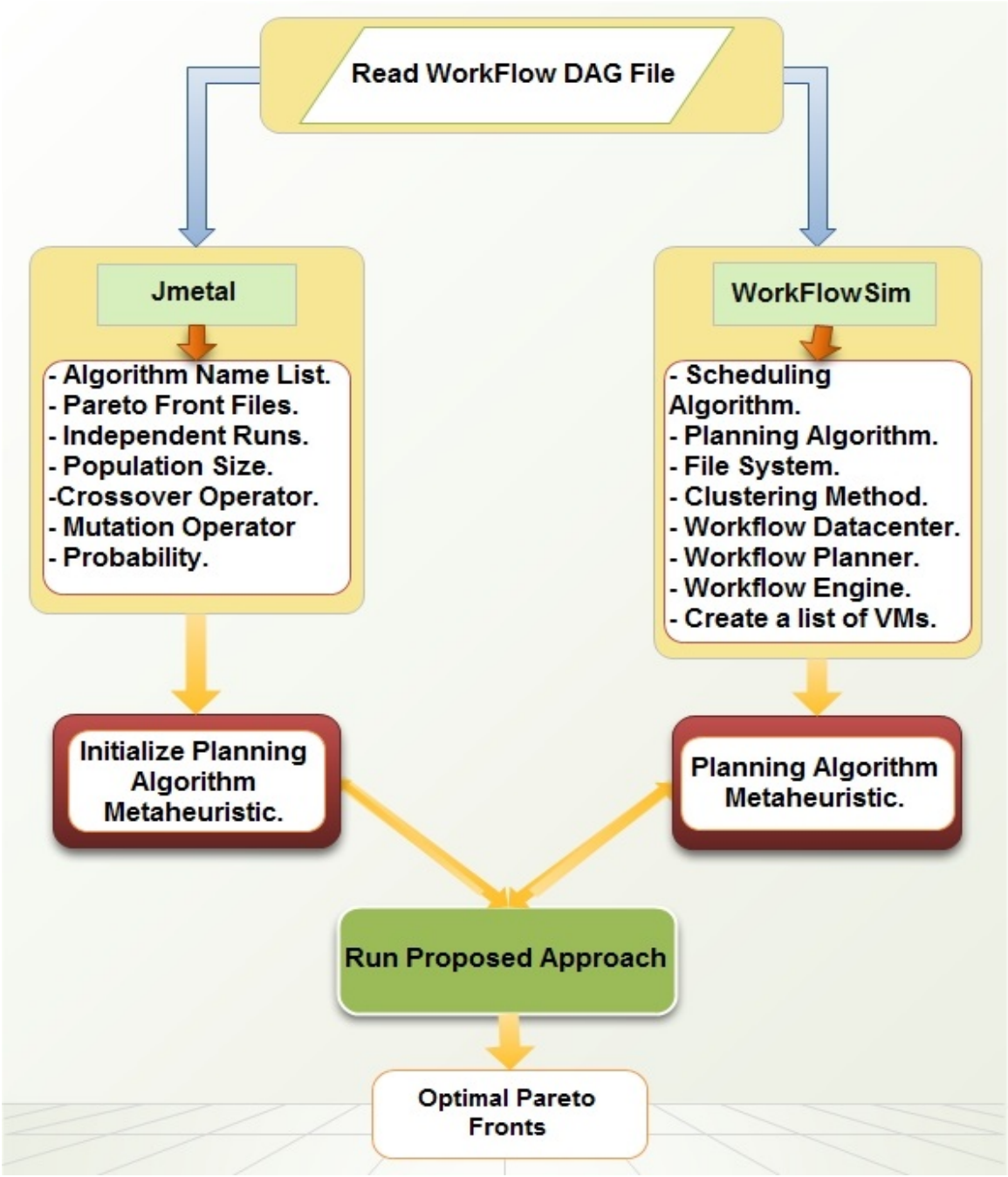


Figure 3.1: Framework Block Diagram

Figure 3.2 show the flow chart of the integration process. The flow chart presents how the components interact together and how we integrate our work with existing tools too.

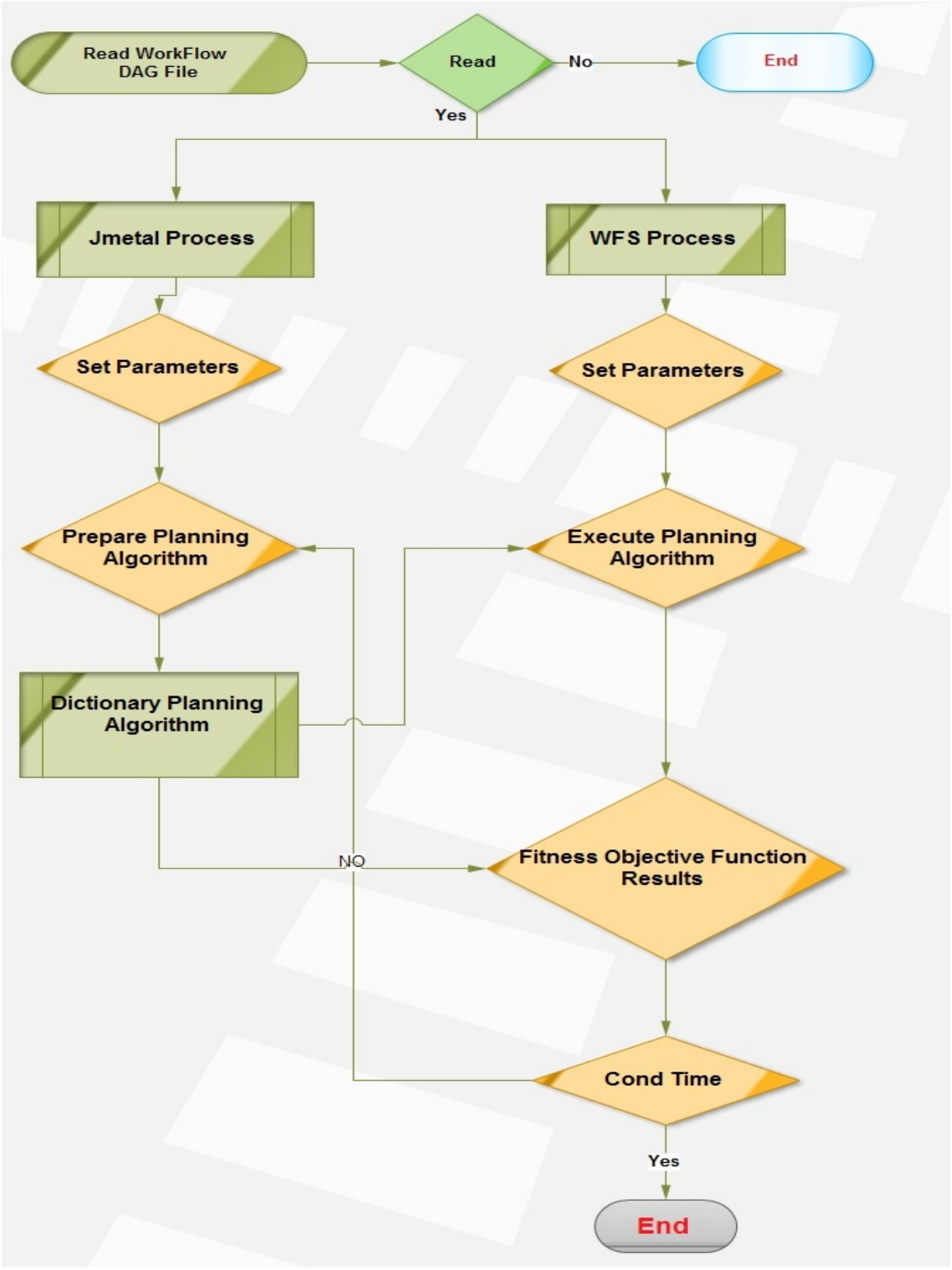


Figure 3.2: Flow Chart Process of Framework

3.2 Integrating User-In-the-Loop (UIL)

User-In-the-Loop (UIL) approach in task scheduling is one of the important contributions of this study. We need to distribute tasks to a set of virtual machines (VMs) that exist in a cloud environment using jMetal tools and previous integration between tools. After that, we will introduce the user as the main factor to effect the algorithm search process during running in order to minimize the objectives problem and find the optimal solutions.

In this part of the study, we will develop a GUI for the end user to enable them to interact with the integrated tools. The main goal of GUI is: to give the user an indication about the current state of running algorithms but interrupting them; the user will be able to change the scope of search if a region of interest is selected and generate selection file to existing running algorithm; also, the user can select many times on the same population even if the running process isn't interrupted.

Figure 3.3 presents the components and how these components interact together while integrated process is running. The Figure shows how the jMetal algorithms know if there is an interaction by the user. Otherwise, the running algorithm will continue even if the selection is done, because the data file of the selection is not created by the user and the process still pauses. This approach considers the interruption time, which means the selection time is not limited to the user.

Algorithm 6 Pseudo-code UIL

$StartTime \leftarrow LocalMachineStartTime$
$StopCond \leftarrow Cond_Time =$
$StopDone \leftarrow false$
$CheckFileUILExisit \leftarrow false$
$UIL \leftarrow false$
$Popl \leftarrow CreatetheInitialofPopulation(PopulationSize \leftarrow PopulationSize)$
$Popl \leftarrow StartTheEvaluationofPopulation(Popl)$
$Popl \leftarrow AssignTheFitnessValueToEachMember(Popl)$
while $StopDone \neq True$ **do**
  $StartANewGeneration \leftarrow selectMembersOfPoplForCrossover(Popl)$
  $NewGeneration \leftarrow MutatePopulation(NewGeneration)$
  $Popl \leftarrow AssignTheFitnessValueToEachMember(Popl)$
  $EndTime \leftarrow LocalMachineTime$
  $Stop_Time \leftarrow EndTime - StartTime$
  if $Stop_Time >= StopCond$ **then**
    $StopDone \leftarrow True$
  end if
  if $CheckFileStopExisit$ **then**
    $StopDone \leftarrow True$
  end if
  if $CheckFileUILExisit$ **then**
    $TouchFileSelection.TXT \leftarrow PrintThefittestMemeberOfPopulation >$
    $UIL \leftarrow True$
  end if
end while
$PrintFeasibleAfterEvalutaion \leftarrow ThefittestMemeberOfPopulation(Popl)$

Algorithm 7 Pseudo-code UIL Assign The Fitness Value To Each Member

$UIL \leftarrow false$
if $UserSelectionPointsexists$ **then**
$ReadSelectionPointFromUSer \leftarrow UserSelectionPoints.txt$
$RemoveFile \leftarrow UserSelectionPoints.txt$
•MinOpjective1•MaxOpjective1
•MinOpjective2•MaxOpjective2
•MinOpjective3•MaxOpjective3
•UIL = True
end if
if UIL **then**
for $i = 0to \leftarrow sizeofThefittestMemeberOfPopulation$ **do**
if $Fitness1 > MaxOpjective1||Fitness2 > MaxOpjective2||Fitness3 > MaxOpjective3$ **then**
fitness1[i] =888888 ▷ Dummy value to exclude the objective 1.
fitness2[i] =8888 ▷ Dummy value to exclude the objective 2.
fitness3[i] =88 ▷ Dummy value to exclude the objective 3.
sign real value
end if
end for
end if

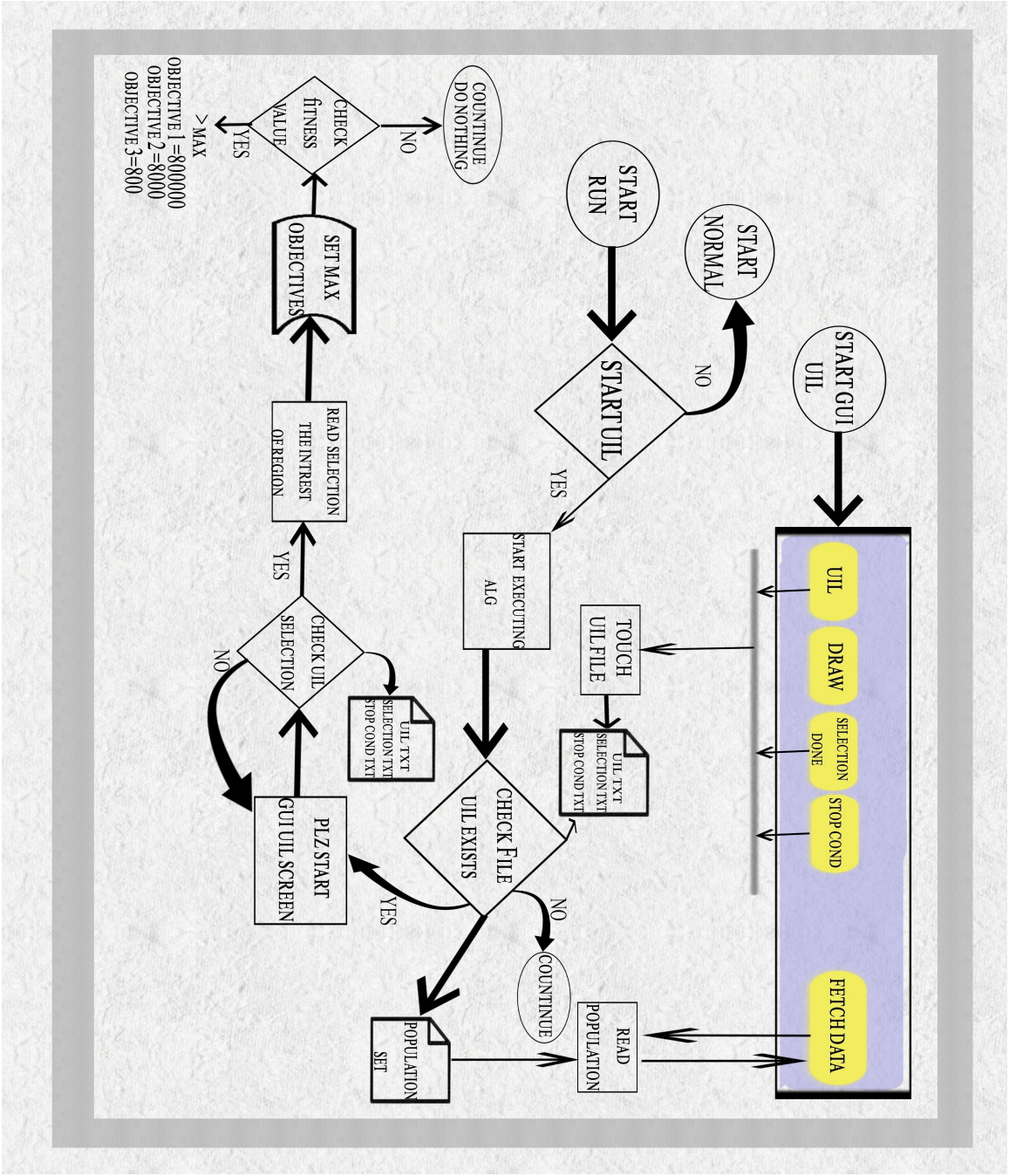


Figure 3.3: Flow Chart of Framework with UIL

3.3 Existing Tools

3.3.1 jMetal

jMetal [23] is a framework designed for meta-heuristic Algorithms for optimizing problems in multi-objectives. The framework has its own object oriented with their classes built for multi-objective problems (MOPs), and sharing their objects with complements to execute the MOPs and comparing them with different techniques and algorithms. JMetal provides many EAs to evaluate the fitness value of the feasible solutions in large search spaces. jMetal is built on base object oriented classes (SolutionSet, Solution, Variable, etc.) and their operations are intuitive, making it easy to understand and use. It is also easy to extend this package by adding new features, classes, functions in order to be used with the existing algorithms. The UML diagram below is a simplified version of the jMetal classes:

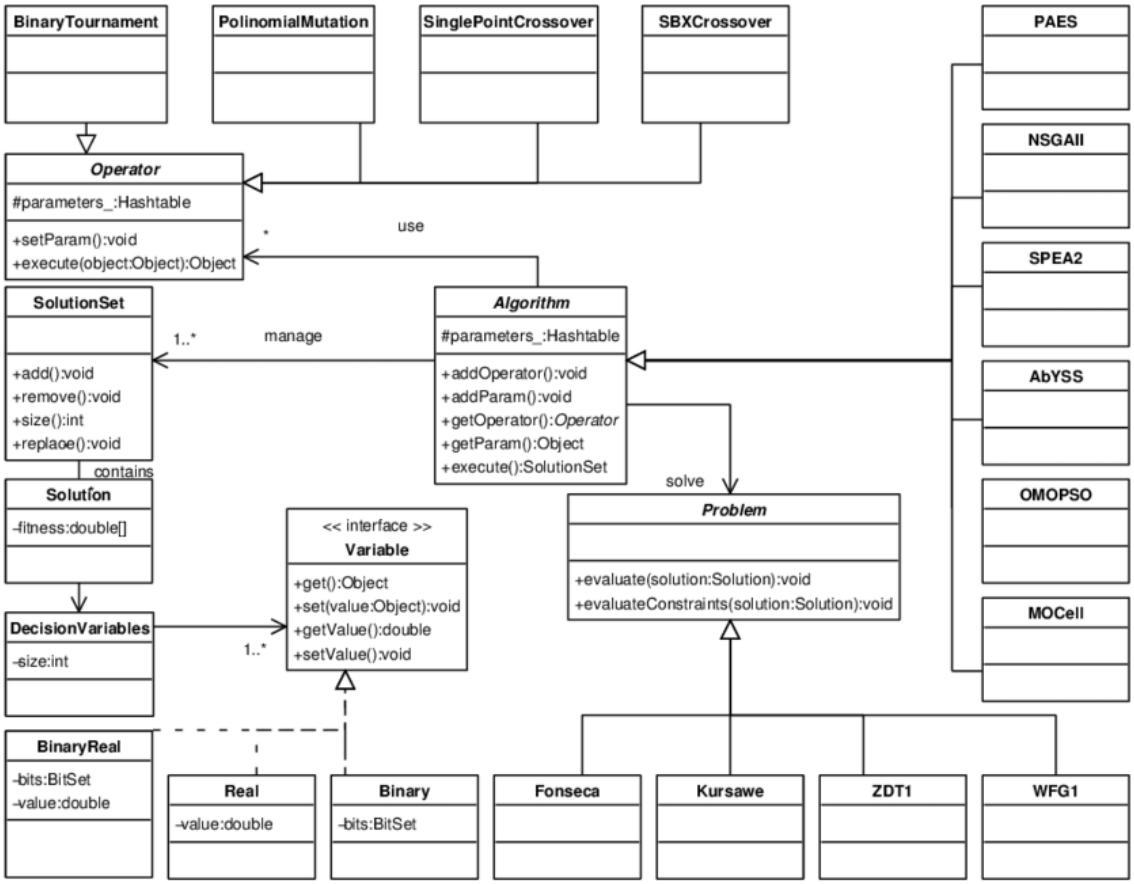


Figure 3.4: UML Class Diagram of jMetal [23]

3.3.2 CloudSim

CloudSim [15] is a framework designed for customers cloud and software's products in order to test their environments and applications before deploying them in real environments. That means this simulation is used to test environments by controlling variables. The cloud computing with all tools and methods aims to make sure the technology used is secure, available, sustainable, scale-able without any errors or issues. When we repeat the test and methodology used in a specific time, all of these goals aim at serving the evaluation of algorithms and applications in order to reduce the testing costs in the cloud computing technology.

CloudSim can estimate the resources that will be used and how the tasks can be distributed. It can tune the performance for the network. Using this Package can help us to calculate the cost of scale ability for data centers, and compute the infrastructure services and their applications. This means that CloudSim supports modeling and simulation such as the large scale cloud computing data centers simulation of virtualized server hosts, with customize policies for provisioning host resources to virtual machines, application containers, computational resources, data center network topologies and message-passing applications, federated clouds, dynamic insertion of simulation elements, stop, and resume of simulation, support for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines.

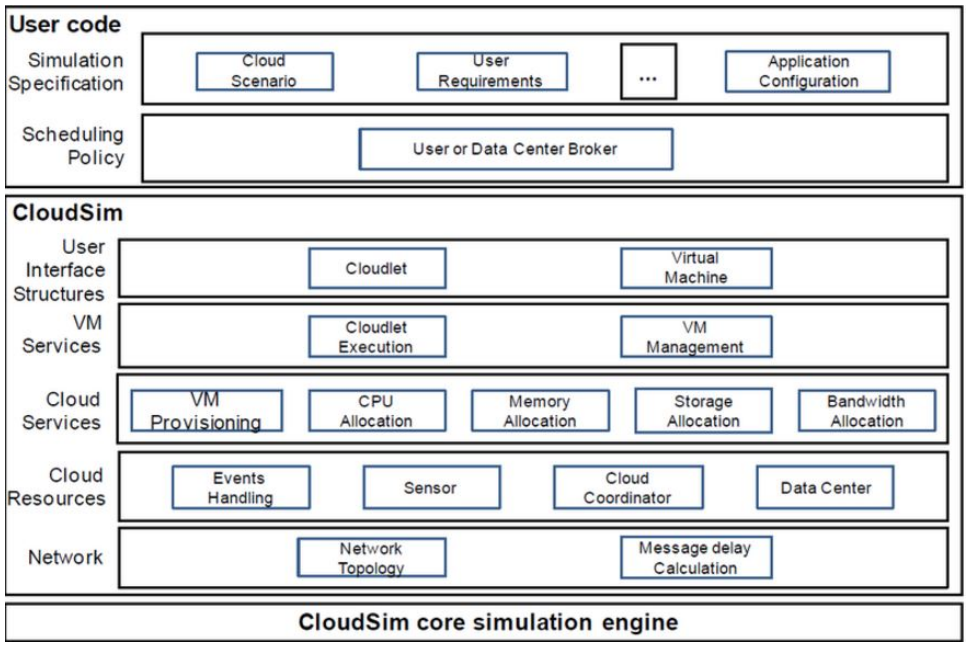


Figure 3.5: CloudSim Framework [15]

3.3.3 WorkflowSim

WorkflowSim [12] is a toolkit based on CloudSim [15]. WorkflowSim was built in order to support the workflow simulation level. It is combined with Directed acyclic graph (DAG) to support both dynamic and static workflow schedulers. The DAG file could be a balanced or unbalanced structure [5] as Figure 3.6. Therefore, WorkflowSim has some algorithms that support their modules functions and tools as follows:

Workflow Schedulers: which consists of many algorithms such as first come first serve scheduling algorithm, data aware scheduling algorithm, minimum completion scheduling algorithm, maximum completion time scheduling algorithm, round robin scheduling algorithm and static scheduling algorithm.

Workflow Planner: which is able to bind the tasks to their resource if available or not, that means this components is a global version for optimization algorithms instead of local Workflow Schedulers in WorkflowSim.

Task Clustering Algorithms: which is used for run-time based algorithms and data-oriented algorithms.

Workflow for fault tolerant clustering: which has algorithms with parameters used directly to learn from the traces of real executions.

WorkflowSim: has a lot of components created to solve many problems. These components are used in to find solutions for a specific problem not all. However, some of components such as

Workflow mapper: are used to read the data sets text and DAG files, and Workflow Engine is used to manage and create tasks, also, the Clustering engine is created to deal with dependency and the parents of each task in order to guarantee that the incoming task can be executed. Failure Generator is a component that is used to explore the task when it fails and after the last component is collected a failure record is returned back to the clustering Engine to adjust the scheduling strategies dynamically. This study takes a look at Workflow Scheduler, this component used for matching a task to a worker node and assign tasks to multiple virtual machines based on the user selection of each node. Based on this component the logarithms above were built, therefore, we will use the genetic algorithms to solve the same problems in different way.

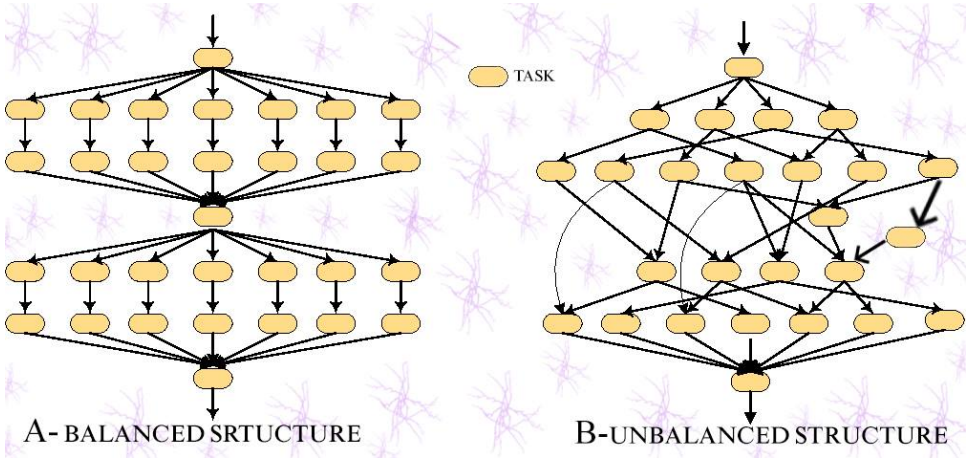


Figure 3.6: Directed Acyclic Graph Structure [71]

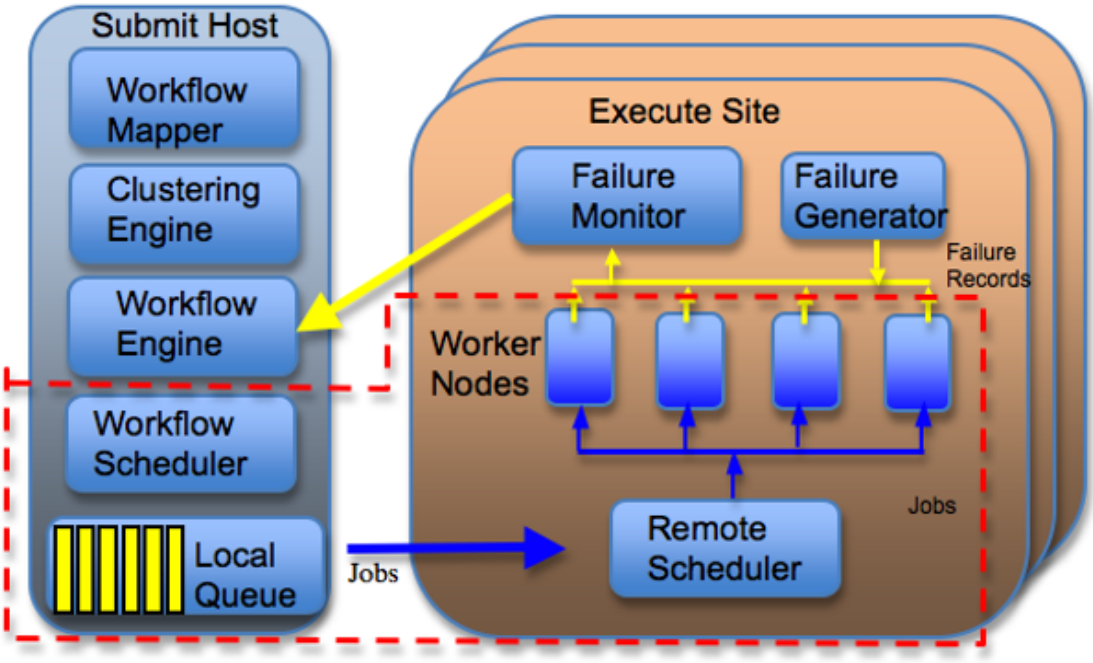


Figure 3.7: WorkflowSim Framework [12].

The Proposed Cloud Task Scheduling Framework shows our work and how we will implement this proposal to conduct our experiments and get significant results. The next chapter shows the setup environments and setting the existing tools and our framework and data set used in this study.

Chapter 4

Experimental Setup

We design the experiment to study the impact of MOEAs on selecting the best optimal solution for task scheduling. Thus, we selected hundreds of tasks with their characteristics, and sets of resources for VMware with their characteristics provided by CloudSim. Then, we implemented the problem using Java framework. After that, we investigated the MOEA solutions and calculated the impact of this selection.

4.1 Creating Tasks and Virtual Machines

To conduct this study we define the VMwares, the tasks and the problem that has to be presented using GA with quality indicator as follows:

- The CloudSim release version is 3.0, jMetal 5.0 and WorkflowSim 1.1.0.
- We created tasks and each task had attributes as follows: task ID; task size; task length; file type for task (input or output) and how many CPUs required for executing this task (Task CPU). The attributes of tasks will be read as into CloudSim as shown in Figure 4.1:

```
// cloudlet parameters
long length = 1000;
long fileSize = 3000;
long outputSize = 3000;
int pesNumber = 1;
UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet[] cloudlet = new Cloudlet[cloudlets];

for (int i = 0; i < cloudlets; i++) {
    cloudlet[i] = new Cloudlet(i, length+=800, pesNumber, fileSize+=1000,
    outputSize+=500, utilizationModel, utilizationModel,
            utilizationModel);
    // setting the owner of these Cloudlets
    cloudlet[i].setUserId(userId);
    list.add(cloudlet[i]);
}

return list;
```

Figure 4.1: Reading Tasks and Attributes for Simulation

- We created virtual machines (VMs), each having attributes as follows: VM ID; machine name, million instructions per second (MIPS); image VM size (size); vm memory (RAM); the network bandwidth (BW); number of CPUs (pesNumber). The attributes of virtual machines will be created as as shown in Figure 4.2:

```
// VM Parameters
long size = 10000; // image size (MB)
int ram = 512; // vm memory (MB)
int mips = 1000;
long bw = 1000;
int pesNumber = 1; // number of cpus
String vmm = "Xen"; // VMM name
// create VMs
Vm[] vm = new Vm[vms];

for (int i = 0; i < vms; i++) {
    vm[i] = new Vm(i, userId, mips, pesNumber, ram+=56, bw, size+=10, vmm,
    new CloudletSchedulerTimeShared());
    list.add(vm[i]);
}
return list;
```

Figure 4.2: Creating VMware on the Data Center

- We used nonidentical VMs too, using the equations below:

$$\begin{aligned} mips[i] &= mips[i] + 50. \\ ram[i] &= ram[i] + 256. \\ bw[i] &= bw[i] + 10. \end{aligned} \tag{4.1}$$

- We created VMs on data center. Each data center has attributes as follows: Ram, Storage, bandwidth (BW), million instructions per second (MIPS) and the host id.
- The results of simulation testing tasks and VMs are shown in Figure 4.3.

TaskID	STATUS	DatacenterID	VMID	Time	Start_Time	Finish_Time
0	SUCCESS	2	0	7.2	0.1	7.3
1	SUCCESS	2	1	7.8	0.1	7.9
2	SUCCESS	2	2	10.2	0.1	10.3
4	SUCCESS	2	1	12.6	0.1	12.7
3	SUCCESS	2	0	14.4	0.1	14.5
5	SUCCESS	2	2	15	0.1	15.1
7	SUCCESS	2	1	15.11	0.1	15.21
8	SUCCESS	2	2	17.4	0.1	17.5
6	SUCCESS	2	0	19.2	0.1	19.3
9	SUCCESS	2	0	21.6	0.1	21.7

Figure 4.3: Cloud Simulation Test

4.2 Solution Representation

We represented the problem for the GA, and the solution type is binary. The number of bits created based on Equation 4.2 and demonstrated in Figure 4.4. Each four bits represent one VMware, and the value of these bits is the VMware ID.

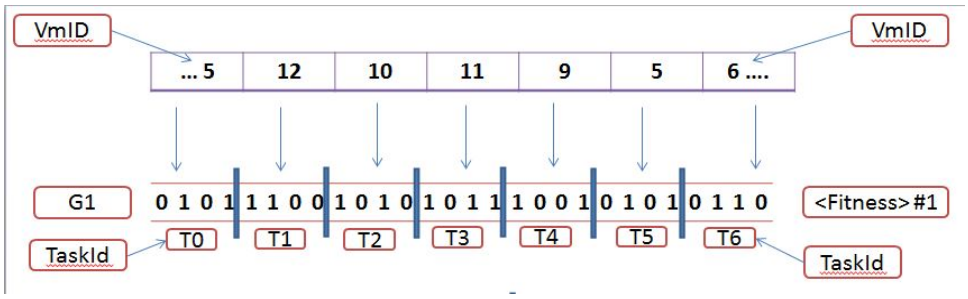


Figure 4.4: Resource Allocation for Problem as GA in a binary set

$$NumberOfbits = (\log NumberOfVMs) * (NumberOfTasks) \tag{4.2}$$

4.3 Stopping Condition

When Metaheuristic Algorithms are compared, different options exist for the stopping condition. Most studies assign a maximum number of fitness evaluations, and the algorithms are stopped once that number is reached. Some studies choose to stop the search once the solutions stop improving by much. We choose a maximum time for stopping the algorithms, for tow reasons: First, different algorithms take different amounts of time to reach maximum evaluations, and thus the comparison would be unfair. Second, real time is an important factor for the end user (i.e. the decision maker) as they wait for the algorithm to provide them with useful options that they can use.

We created a new variable called stop condition, to be used in jMetal tool. A time condition will stop running algorithms according to the value assigned by the user.

4.4 Optimization Objectives

The objectives we aim to optimize simultaneously using the MOEAs are as follows:

- The first objective is to minimize the total execution time across all VMs. It is calculated using Equation 4.3.

$$\sum_{Vm=0}^{NumberOfVMs} \sum_{Task=0}^{NumberOfTasks} FinishTime - StartTime \tag{4.3}$$

- The second objective is to minimize the maximum execution time across all VMs. The reason is that we would like to balance the task load across VMs. This objective is calculated using Equation 4.4. .

$$\max_{VMs} \sum_{Task=0}^{NumberOfTasks} FinishTime - StartTime \tag{4.4}$$

- The third objective is to minimize the number of virtual machines needed to execute the task load. This represents the economic concern, which is absent from the traditional workflow scheduling approaches.

4.5 Algorithm Settings

In this study, we compare 3 MOEAs, namely NSGA-II, MOCell and IBEA and two traditional WorkflowSim Algorithms (HEFT and DHEFT). We selected HEFT and DHEFT algorithms because they aim at optimizing workflows in heterogeneous cloud environments.

The parameter settings we used for these algorithms are shown in Table 4.1.

Table 4.1: Workflow Optimization Algorithm Settings

Multi-Objective Algorithms	• Stop Condition = 1 minute.
	• Mutation = BitFlipMutation [13].
	• Mutation Probability = 1.0 /Number of bits.
	• Selection = Binary Tournament [53].
	• Crossover = SinglePointCrossover.
	• Crossover Probability = 0.70
WorkflowSim Algorithms	• Planning Algorithm = No.
	• Local Replica Catalog File System = Yes.
	2.3 overheads Parameters = No.
	2.4 Clustering = No.

- The three quality indicators were selected as follows: HV, Spread and IGD.

4.6 DataSet of Study

To conduct our study we use three workflow types: Montage, CyberShake [51], and Heterogeneous Earliest Finish Time (HEFT) dataset [69] in the first phase. In the second phase (UIL) we only use Montage and CyberShake.

- Montage: is an application used to collect and create multiple images of the sky in order to create a one large-scale of big images. It was used by (NASA/IPAC).
- CyberShake: it is a probability model application used for predicting the earthquake hazard for a certain location or region.

The type structure workflows Montage and CyberShake are structure and unbalanced structure respectively [5]. Based on these data sets types we start running our algorithms to get results.

Chapter 5

Results

In our experiments we used the datasets for workflow with 100 tasks. The experiment has 4 major parts, in which the workflow tasks are assigned to 4, 8, 12 and 16 machines, respectively. Within each part, we repeat the experiments for identical VMs versus non-identical VMs.

Each algorithm is repeated for 30 independents runs, in order to get statistically significant results.

We executed a long run for an optimizing phase without any errors to get the best Pareto front for all Algorithms in WorkflowSim as a reference point to the MOEAs.

5.1 Comparing MOEAs with Traditional Algorithms

This section shows the results of running process when the user is out of the loop. We aim at finding the best algorithms and best optimal solutions to execute the workflows. The results will be compared to the exiting WorkflowSim approaches.

5.1.1 Solving for 4 Maximum VMs

Tables 5.1 and 5.2 show the median of the objective values for all five algorithms. The algorithms are ordered according to the best performance in the first objective. The results show that the three MOEAs (MOCell, IBEA, and NSGA-II) perform better than the WorkflowSim algorithms on the first two objectives, while no algorithm was able to fit all tasks on less than 4 VMs from Cybershak and Montage workflow. For the third objective the MOEAs outperform HEFT and DHEFT. In one case, MOCell is able to fit all of the HEFT study workflow on one virtual machine.

Table 5.1: 4 Identical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	MOCell	19776.25	1201.5	4
	NSGA-II	20229.39	1037	4
	IBEA	20508.92	1091	4
	HEFT	24732.47	1295	4
	DHEFT	25139.86	1789	4
HEFT_study	IBEA	381.61	64	2
	NSGA-II	381.73	65	2
	MOCell	381.84	68.5	2
	HEFT	538.81	94	3
	DHEFT	550.94	98	3
Montage	NSGA-II	3682.91	325	4
	MOCell	3700	334	4
	IBEA	3739.64	365	4
	HEFT	6324.16	552	4
	DHEFT	6575.98	668	4

TXT: Total Execution Time (Millisecond).
VMTXT: Total Execution Time for Virtual Machine (Millisecond).
Vm_Used: Number of Virtual Machines Used.

Table 5.2: 4 NonIdentical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	IBEA	19175.3	933	4
	NSGA-II	19391.67	1053	4
	MOCell	19451.81	968.5	4
	DHEFT	23512.61	1650	4
	HEFT	29923.46	1852.5	4
HEFT_study	MOCell	363.17	117	1
	NSGA-II	363.33	62.5	2
	IBEA	375.74	55	3
	HEFT	471.56	79.5	3
	DHEFT	482.21	76.5	3
Montage	NSGA-II	3414.72	265	4
	MOCell	3502.65	305	4
	IBEA	3586.1	339	4
	DHEFT	4657.12	499	4
	HEFT	5565.81	505.5	4

5.1.2 Solving for 8 Maximum VMs

Tables 5.3 and 5.4 show the median results of three objective values for all the five algorithms. The three MOEAs (MOCell, IBEA, and NSGA-II) perform better than the WorkflowSim algorithms on the two objectives, also MOCell and NSGA-II are able to fit all tasks on less than 8 VMs whether identical or non-identical.

Table 5.3: 8 Identical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	IBEA	20492.86	828.5	7
	NSGA-II	21054.37	663	7
	MOCell	21329.53	727	7
	HEFT	24949.64	711	8
	DHEFT	25576.87	1375.5	8
HEFT_study	IBEA	381.88	67	2
	NSGA-II	382.22	64	2
	MOCell	429.16	60.5	3
	HEFT	488.73	71	5
	DHEFT	496.77	68	5
Montage	NSGA-II	4151.3	237	7
	MOCell	4258.28	245	8
	IBEA	4520.13	237	8
	DHEFT	5663.07	393	6
	HEFT	6257.87	305	8

Table 5.4: 8 NonIdentical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	MOCell	19865.94	659	7
	NSGA-II	19972.08	884.5	7
	IBEA	21544.23	718	8
	DHEFT	23544.97	1427	8
	HEFT	26781.49	784.5	8
HEFT_study	NSGA-II	363.66	60	2
	IBEA	363.71	59	2
	MOCell	366.04	49	3
	DHEFT	443.16	61	4
	HEFT	460.96	68.5	4.5
Montage	MOCell	3904.81	190.5	7.5
	IBEA	4149.58	224	8
	NSGA-II	4241.06	252	7.5
	DHEFT	4861.69	317.5	7
	HEFT	5611.3	271.5	8

5.1.3 Solving for 12 Maximum VMs

Tables 5.5 and 5.6 show the median results of three objective values for all the five algorithms. The three MOEAs (MOCell, IBEA, and NSGA-II) perform better than the WorkflowSim algorithms on the first two objectives. Also, IBEA and NSGA-II are able to fit all tasks on less than 12 VMs. On the other hand, the MOEAs still perform better than all in three objectives if these machines were identical or non-identical.

Table 5.5: 12 Identical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	NSGA-II	20890.77	682	10
	IBEA	20997.87	642	10
	MOCell	21525.05	626	10
	DHEFT	23919.16	1148.5	10
	HEFT	25872.48	559	12
HEFT_study	IBEA	381.89	64	2
	NSGA-II	382.52	64	2
	MOCell	395.66	61	2.5
	HEFT	456.85	68	5
	DHEFT	461.36	61	6
Montage	NSGA-II	4082.79	204.5	9.5
	MOCell	4161.11	190	10
	IBEA	4199.94	195	10
	DHEFT	5526.05	310.5	9
	HEFT	5732.66	214	11

Table 5.6: 12 NonIdentical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	MOCell	20504.71	645	10
	IBEA	20580.38	659	9
	NSGA-II	20680.59	594	10
	DHEFT	22093.07	1045.5	9.5
	HEFT	24751.09	527.5	12
HEFT_study	IBEA	363.54	60.5	2
	NSGA-II	363.87	60	2
	MOCell	364.26	59	2
	DHEFT	428.42	55.5	5
	HEFT	435.82	66	5
Montage	NSGA-II	3920.37	211.5	8.5
	Ibea	3927.58	188	10
	MOCell	3965.77	207	9
	DHEFT	4829.67	262	9
	HEFT	5696.61	208.5	11

5.1.4 Solving for 16 Maximum VMs

Tables 5.7 and 5.8 show the median results of three objective values for all five algorithms. Two MOEAs (MOCell and IBEA) perform better than the WorkflowSim algorithms on the first objective, and MOEAs are able to fit all tasks on third objective and execute workflow on less than 16 VMs. On the other hand, the DHEFT does not perform better than MOEAs algorithms even though it outperforms in the third objective. The cause root of this outperform, all tasks is that it is able to execute in less than 16 machines, but if we increase the running time to be more than one minutes for MOEAs algorithms, then the MOEAs will outperform workflowsim algorithms in this objective. Increasing the time needed because we need more crossover and mutation and solutions for a long chromosome.

Table 5.7: 16 Identical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	NSGA-II	21711.13	488	13
	MOCell	22251.09	511	13
	IBEA	22539.92	513.5	13
	DHEFT	22550.03	926.5	10
	HEFT	24988.37	449	16
HEFT_study	MOCell	381.9	65	2
	NSGA-II	383.39	58	2.5
	IBEA	383.52	60.5	2.5
	HEFT	428.41	66	5
	DHEFT	434.16	60	5
Montage	IBEA	4353.78	155	13
	MOCell	4429.54	155	13
	NSGA-II	4441.52	157.5	13
	DHEFT	4940.73	251	11.5
	HEFT	5435.33	164	14

Table 5.8: 16 NonIdentical Virtual Machines

Workflow	AlgName	TXT	VMTXT	Vm_Used
CyberShake	IBEA	20921.98	557	13
	MOCell	21089.8	548.5	13
	NSGA-II	21126.57	497.5	13
	DHEFT	22426.16	1082	10.5
	HEFT	23231.99	418	16
HEFT_study	MOCell	363.73	60.5	2
	NSGA-II	368.68	47	3
	IBEA	371.43	45	3
	DHEFT	413.62	55	5
	HEFT	417.27	64	5
Montage	MOCell	4244.13	167	13
	IBEA	4389.7	170.5	13
	DHEFT	4438.24	188	11
	NSGA-II	4531.21	172	12
	HEFT	5253.99	157	15

5.1.5 Pareto Front plots

The common decision makers are interested in finding the best optimal solution among alternative solutions for different objectives. As researchers we are interested in showing the results of our approach to decision makers to fit their interest. The figures below show scatter plots for all optimal solutions we get for three objectives. We make 2D plots with 2 objectives only to demonstrate how the objectives trade-off against each other.

Figure 5.1: CyberShake Workflow Total Execution Time vs Total Virtual Machine Execution Time.

Figure 5.1 shows the results of Pareto fronts for a balanced structure workflow. Balanced workflow structure means the tasks that could be executed in parallel machines. Thus, if we increase the number of virtual identical machines the total execution time for workflow on those machines will decrease. For non-identical machines the MOAEs tend to allocate tasks to the highest spec machines and minimizing the total execution time. The distribution of Pareto Fronts shows that MOAEs have the best results, and all solutions are close together for both objectives. Thus, giving us a consistency in the solutions for the same region instead of scattered solutions for WorkflowSim algorithms.

Figure 5.2: HEFT$studyWorkFlowTotalExecutionTimevsTotalVirtualMachineExecutionTime$.

Figure 5.3: Montage WorkFlow Total Execution Time vs Total Virtual Machine Execution Time.

Figures 5.2 and 5.3) show the results of Pareto fronts for HEFT study and Montage workflows. The distribution of Pareto Fronts shows that MOAEs (NSGA-II, IBEA, and MOCell) produce minimal objective values, while HEFT and DHEFT produce higher values.

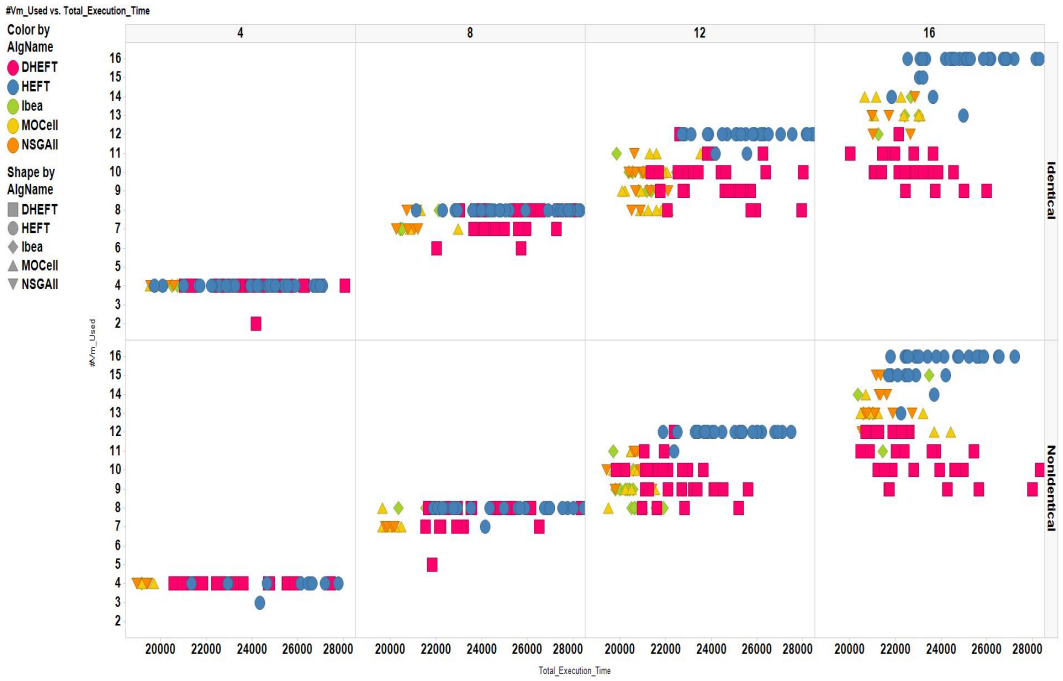


Figure 5.4: CyberShake Workflow Virtual Machines Used vs Total Execution Time.

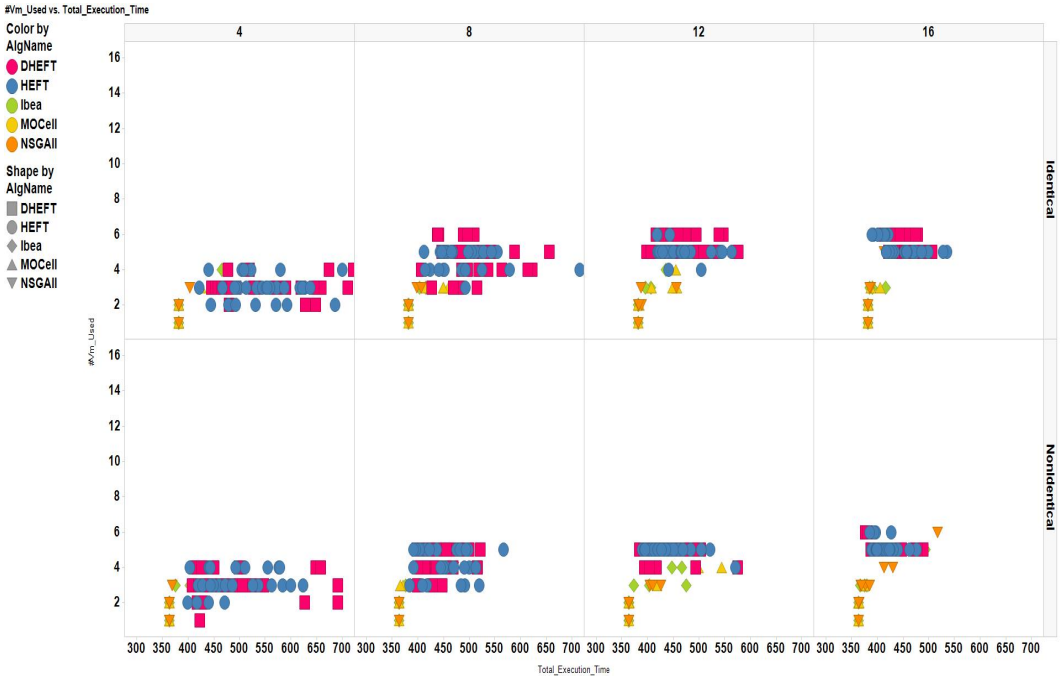


Figure 5.5: HEFT_study Workflow Virtual Machines Used vs Total Execution Time.

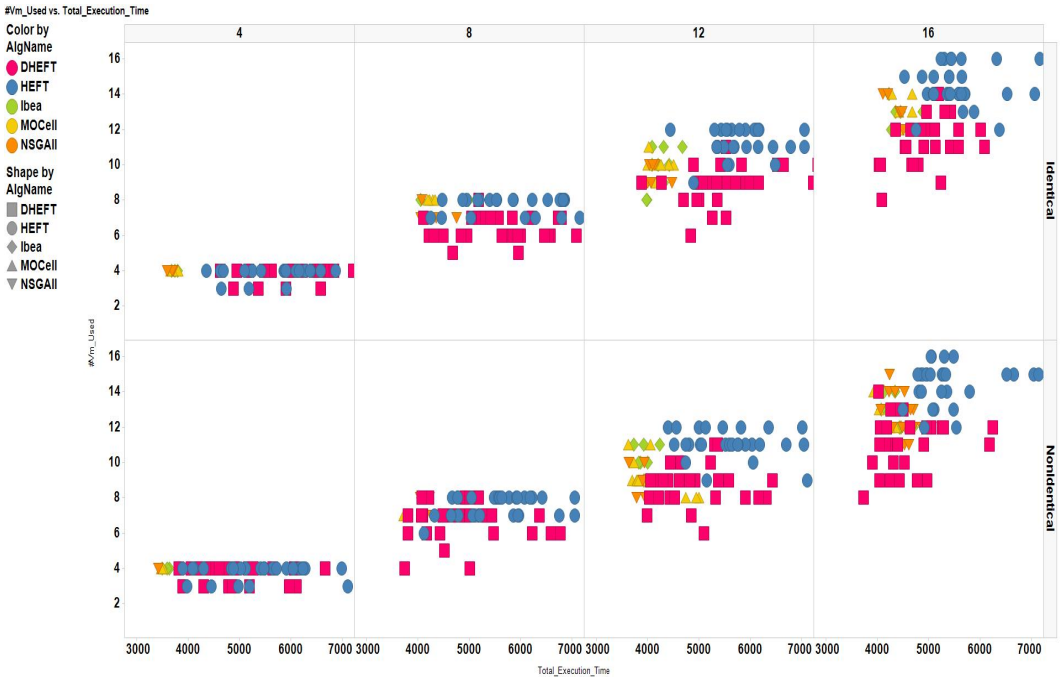


Figure 5.6: Montage Workflow Virtual Machines Used vs Total Execution Time.

The results of Figures 5.6, 5.5, and 5.4 show the consistency in the solutions provided by MOAEs for the same number of machines used and gives the best results for the first objective, whether the workflow type is a balanced or non-balanced structure.

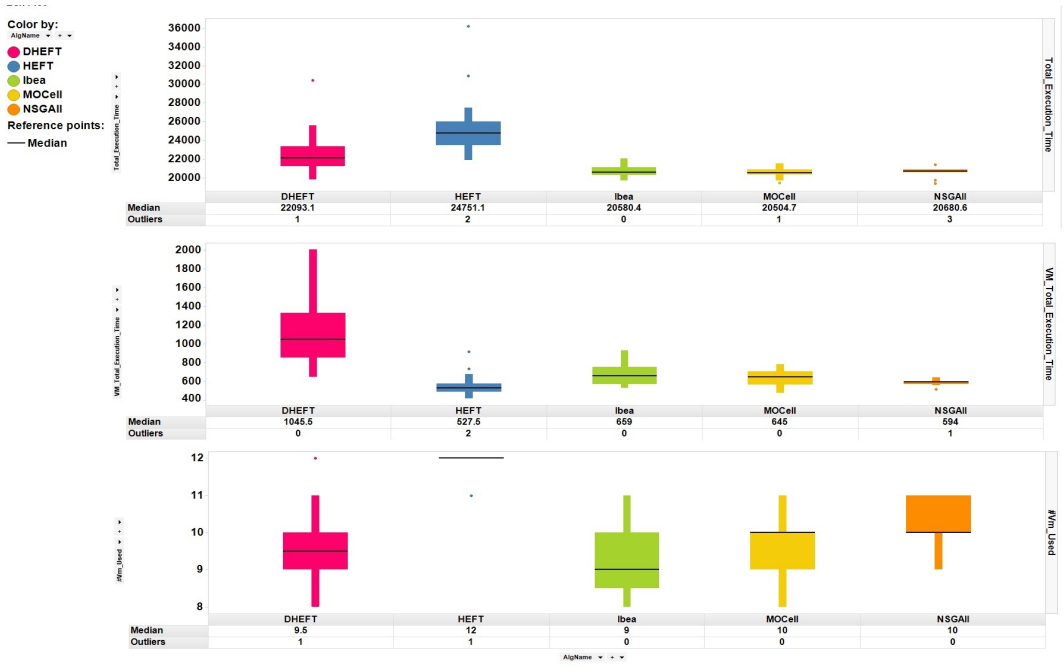


Figure 5.7: Non-Identical 12 Virtual Machines Box Plot for CyberShake Workflow.

The Box Plots in Figure 5.7 show the median of algorithms indicated by the line, and the outlier values for three objectives. The median of IBEA has better results for all objectives. There is also no outlier values. That means IBEA gives consistent results for our objectives. DHEFT was able to work with the same machine but it wasn't able to optimize the load balancing between machines (i.e. the second objective). MOCell results were close but not better than IBEA.

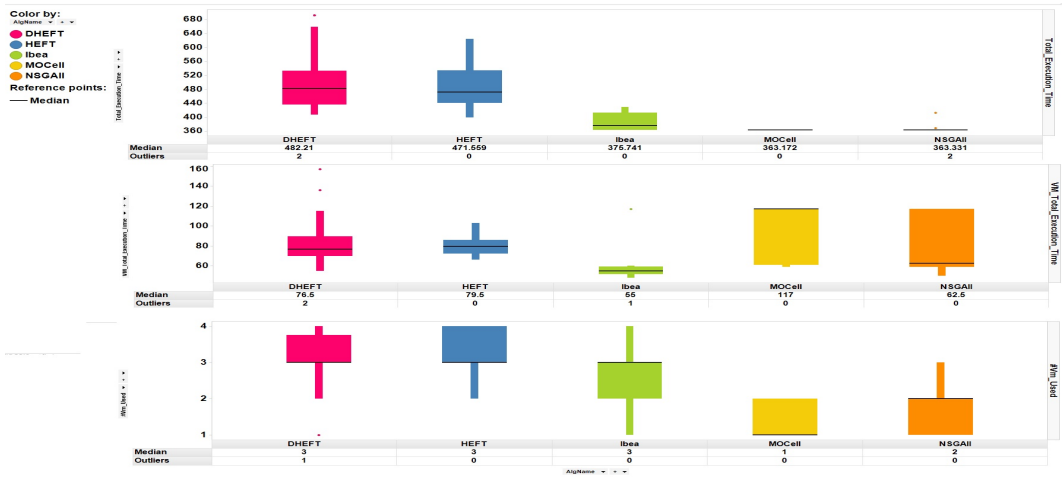


Figure 5.8: Non-Identical 4 Virtual Machines Box Plot for HEFT-study Workflow.

The Box Plot in Figure 5.8 presents small workflow on four non-identical machines. MOCell is able to handle the workflow in one machine given minimum total execution time, thus, it still has the best results among them in two objectives for high workload in a cloud computing environment.

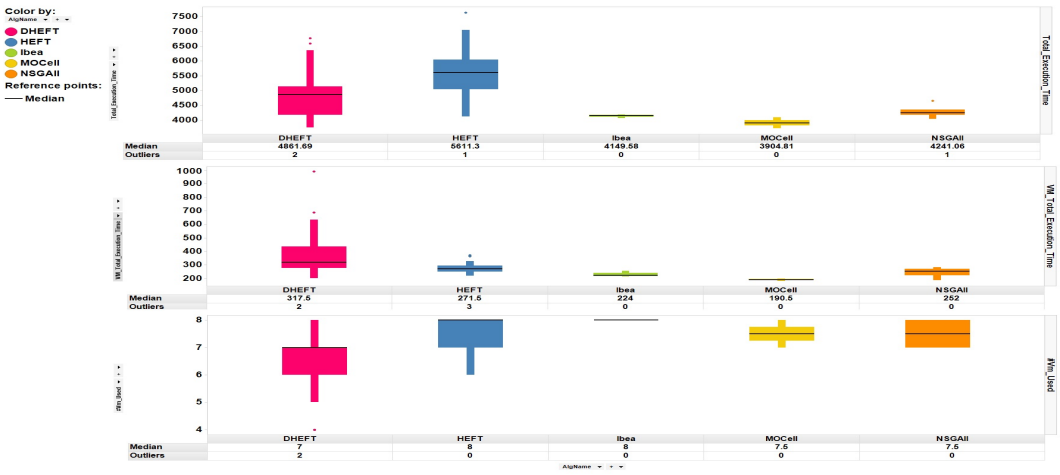


Figure 5.9: Non-Identical 8 Virtual Machines Box Plot for Montage Workflow.

The median of distribution values in Box Plot 5.9, clearly detects how the values are close together, also, the marginal values for MOEAs are not too big like DHEFT or HEFT.

5.2 User-In-the-Loop

This section details the results of running process when the user is involved in the framework, i.e. User-In-the-Loop (UIL) approach. The following are changes in our experiment to execute and implement UIL in order to get the solutions of our problem: the data sets used are CyberShake and Montage, these data sets have 100 tasks executed on cloud environment that has 8 and 12 machines. The stop condition is set to two minuets. The user is able to stop running the process before that time, and 5 minutes if the user is out of process. Each algorithm is run 10 independent times. We selected NSGA-II algorithm as a representative of meta-heuristic algorithms, since it is the most popular MOEA, and it produced comparable results with MOCell and IBEA.

The tables below show the median of three objective results when the time condition set to five minutes and the user is out of process, and two minutes when the user is in the loop.

Table 5.9: Results for User Out of the Loop

Workflow	TotalVm	TXT	VMTXT	Vm_Used
CyberShake	8	20322.24	681.00	8
	12	20971.72	643.50	11
Montage	8	3893.99	199.00	8
	12	4176.54	192.00	11

Table 5.10: Results for User In the Loop

Workflow	TotalVm	TXT	VMTXT	Vm_Used
CyberShake	8	20035.66	668.00	8
	12	21693.33	671.00	11
Montage	8	3887.62	200.50	8
	12	4373.36	202.00	11

The tables below show the median of three objective results for all non dominated points when the time condition set to five minutes and the user is out of process, and two minutes when the user in the loop.

Table 5.11: Non Dominated Points when The User is Out of Process

Workflow	TotalVm	TXT	VMTXT	Vm_Used
CyberShake	8	19809.29	731.00	8
	12	20392.63	587.00	11
Montage	8	3764.43	191.50	7
	12	3992.69	180.00	11

Table 5.12: Non Dominated Points when The User is the Process

Workflow	TotalVm	TXT	VMTXT	Vm_Used
CyberShake	8	19779.94	592.00	8
	12	20337.15	606.50	10.5
Montage	8	3715.08	188.00	8
	12	3943.89	178.00	11

The results of the second phase show that when the user is in and out of the search process in the algorithm. The results of table 5.9 represent the median of Pareto fronts calculated for the three objectives, where the stop condition time was set to five minutes and the user out of process. Then we generate the non-dominated points 5.11 in order to calculate the median of these points. On the other hand, we repeated the same procedure when the user-in-the-loop, but we set the stop condition time to two minutes as shown in the tables 5.10 and 5.12. We used two minutes as the stop condition time because the user-in-the-loop was able to guide the search process for the purpose of attaining the most favorable and efficient solution, thereby

decreasing the stop condition time from five minutes to two minutes. The user-in-the-loop results in two minutes outperforms the user-out-of-the-loop process for both data sets. Thus, the results of the user-in-the-loop in two minutes is competitive to the user-out-of-the-loop process in five minutes.

We also set the stop condition times equal to each other for both user-in-the-loop and user-out-of-the-loop. The stop condition time was set to a total of five minutes. As shown in tables 5.13 and 5.14, the median of the three objective points validate that the user-in-the-loop guided the search process to outperform the user-out-of-the-loop when they both were set to an equal stop condition time of five minutes.

Table 5.13: The User Involves in the framework processes five minutes

Workflow	TotalVm	TXT	VMTXT	Vm_Used
CyberShake	8	19827.62	614	8
	12	20749.19	622	11
Montage	8	3747.98	186.00	8
	12	4094.69	180	11

Table 5.14: Non Dominated Points when The User is the Process

Workflow	TotalVm	TXT	VMTXT	Vm_Used
CyberShake	8	19707.4	609	8
	12	20167	588	10
Montage	8	3706.58	169	8
	12	3941.83	170.50	10.5

The figures below show the Pareto fronts of three objective results for all and non dominated points when the time condition set to two minutes and the user in the loop against five minutes when the user out of procures.

Figure 5.10 present eight machines and non identical with CyberShake data set, and Figure 5.11 shows the same for Montage workflows:

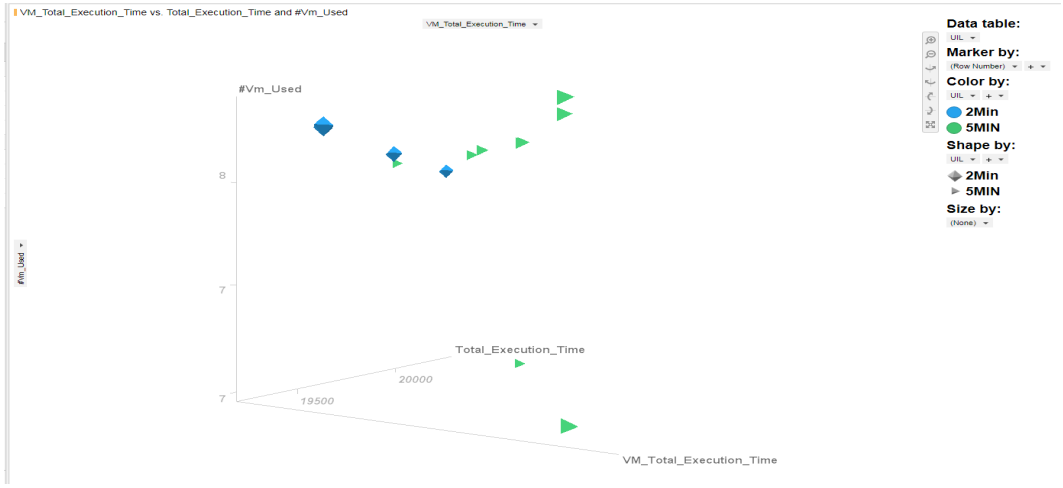


Figure 5.10: CyberShake workflow Non-Identical 8 Virtual Machines With and without UIL

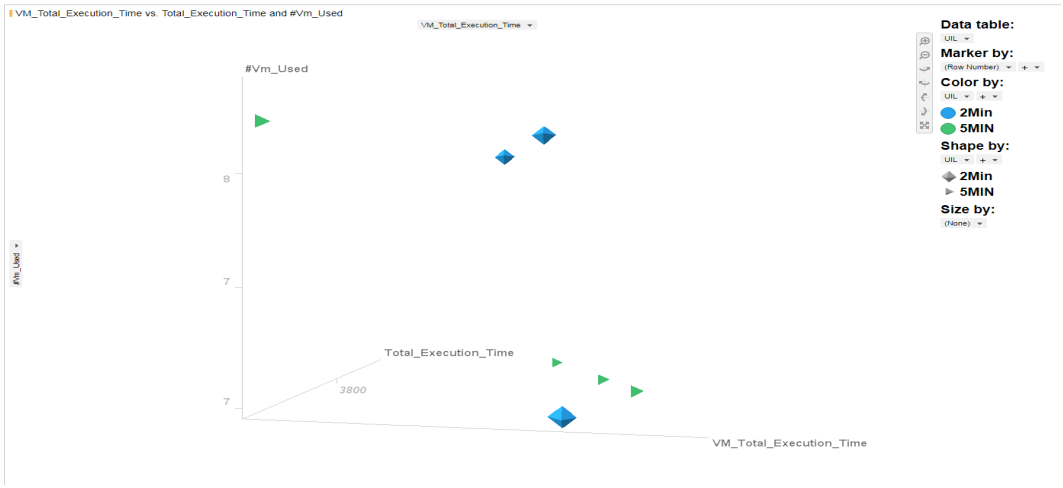


Figure 5.11: Montage workflow Non-Identical 8 Virtual Machines With and without UIL

The figures below present twelve machines and non identical with CyberShake data sets and Montage workflows:

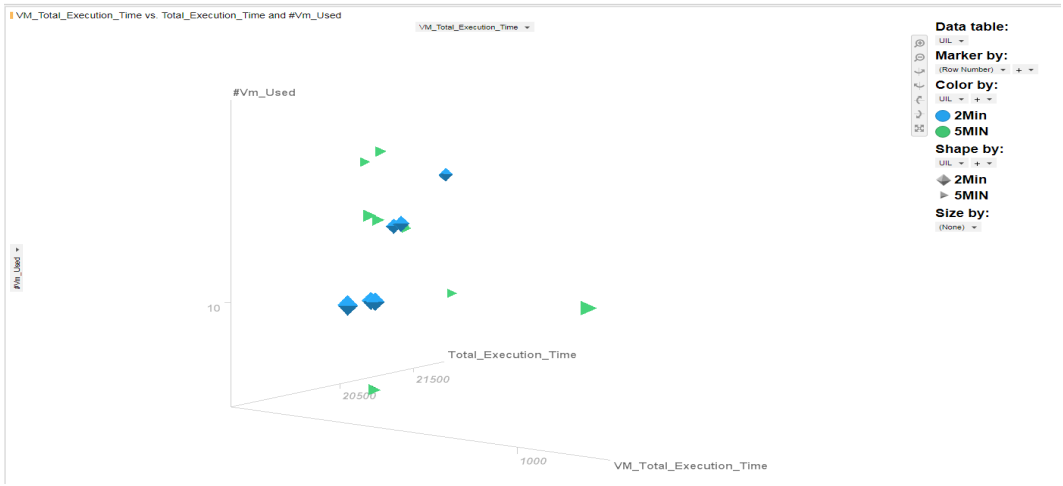


Figure 5.12: CyberShake Workflow and Non-Identical 12 Virtual Machines With and without UIL

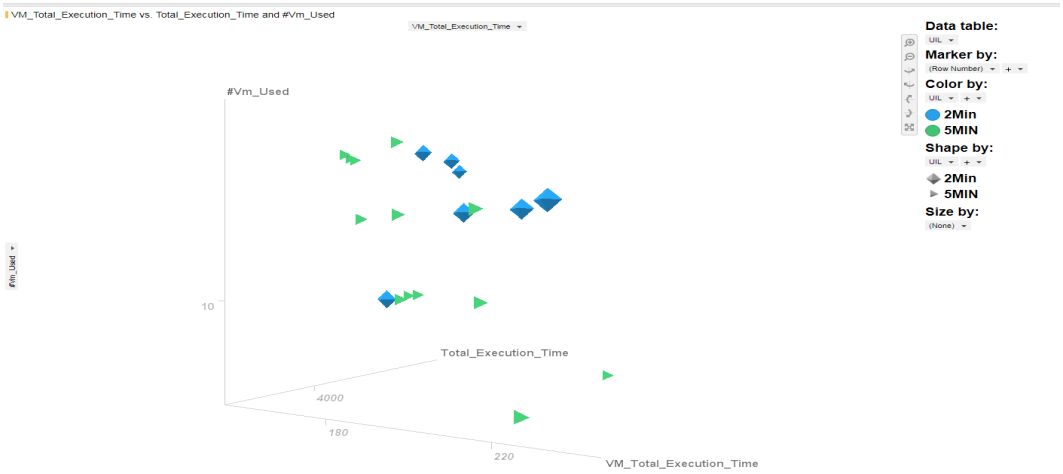


Figure 5.13: Montage workflow Non-Identical 12 Virtual Machines With and without UIL

In the previous tables, we discussed the results of user-in-the-loop in five minutes absolutely will outperform the results when the user out of the process. The Figures 5.10,5.11,5.12 and 5.13 presents the three objectives when the user-in-the-loop and the user out-of-process. The solutions with the user-in-the-loop in two minutes are closed to be optimal as a user out of the loop in five minutes for both workflow data types even the virtual machines are non-identical.

In Appendix A, we demonstrate the graphical interface that allows the user to interact with the search process.

5.3 Discussion

This section represents the behavior of results for the figures and tables above. We will also discuss the results of the box plots above and why the three MOEAs (MOCell, IBEA, and NSGA-II) outperform traditional algorithms. We will discuss why they are better than the WorkflowSim algorithms in the three objectives for all Workflow data types, for instance, whether these machines created are identical or non-identical.

The purpose of the second objective is to prevent the overload onto one machine, and achieve a balanced loading between machines that are available at cloud environment. We increased the number of tasks assigned on to the virtual machines which is less than the number of total virtual machines. Hence, the total time for the second objective will be increased with the minimum total execution time per one machine. The best evaluation of results is a comparison between the algorithms when there is an increased workload on the virtual machine resources. The results of Table 5.1 show that all machines were used for both workflows CyberShake and Montage because the number of tasks were too large, and because of this we needed to use all four machines. However, in reality we needed more than four machines for this scenario.

The robust MOEAs start looking to minimize the first objective and this can be done by optimizing the distribution tasks to machines available in the environment where all machine resources are identical or nonidentical. Thus, in the unbalanced structure the MOEAs looks

to execute all dependent tasks on the same machine, if we increased the stop condition time. As we see, the difference between MOEAs and WorkflowSim is too large in the results of the first objective. The results of Table 5.2 show MOEAs seeking to find the highest specification of machines available in order to assign multiple tasks to this machine, for example, the number of tasks assigned to D machine should be more than the tasks assigned to machine C or B, and also more than the same machine in the initial solution. Clearly, for workflow HEFT_study, the MOEAs outperform the algorithms in WorkflowSim in three objectives based upon the results in the two tables, but we noticed that the HEFT outperforms DHEFT, if the virtual machines are identical. These results were not expected because the DHEFT algorithm should have better results and outperform HEFT.

In order to optimize schedule tasks and test our experiment for the third objective, we needed to increase the number of virtual machines from four machines to eight machines and test them. By increasing this number, the total execution time per machine will decrease because the number of total tasks will also decrease because we distributed the tasks to more machines. We also expect to minimize the number of machines per workflow, thus, reducing the cost of usage of resources available in the cloud environment. The results of Tables 5.3 and 5.4, clearly show that we are able to minimize the number of virtual machines for HEFT_study workflow. In addition, we were able to minimize the number of machines. Therefore, MOEAs gives better results and is more efficient than traditional algorithms. The results of Table 5.3 show that the DHEFT algorithm outperforms MOEAs in the third objective if there is an unbalanced workflow, but it failed in the two remaining objectives. Hence, the DHEFT algorithm is of no use because of its limited efficiency with the remaining two objectives. The results of Tables 5.5, 5.6, 5.7 and 5.8 show the MOEAs still outperform traditional algorithms. The results in tables 5.6 and 5.7 show DHEFT exceeding the MOEAs for the first objective in certain cases, but that does not mean that DHEFT outperforms MOEAs; because MOEAs outperform DHEFT in the two remaining objectives.

The chromosome becomes very long when the number of virtual machines is set to sixteen machines and the number of tasks is set to one hundred. In order to outperform DHEFT we need to increase the stop condition time to be more than one minute for larger tasks and more machines, then we got better results than DHEFT for all objectives.

According to the results of tables and figures, we can conclude that the performance of the MOEAs consistently outperforms than traditional algorithms in the three objectives in cloud computing environments. These results are also observed when the machines created are identical or non-identical, and the workflow type structures are balanced or non-balanced.

The results in this study are highlighted as follows:

(a) Using Meta-heuristic or evolutionary algorithms is a viable method for scheduling tasks of workflow on cloud computing platforms. The traditional methods and approaches try to solve the scheduled tasks of the workflow using one or two objectives at most. Using MOEAs, we are able to solve the task scheduling problem for three objectives, including the economic concern of saving on utilized VMs. We arrive at an optimal schedule within a reasonable time. Using MOEAs for task scheduling in cloud computing reduces the overall execution time while consuming the least cloud

resources which allow the decision maker to save on cloud resources without detriment to the overall task run-time.

(b) The length of the chromosome can impact the results of evolutionary algorithms. When we have big chromosomes the algorithms need more time to get the better results because there is a lot of crossover and mutation to generate new solutions. However, more time is needed to execute big workflows using the existing tools. Thus, the MOEAs still generate better solutions for the same run-time cost.

(c) Impact of the user-in-the-loop on results: The user has experience about workflow tasks and the environment he has, even if this environment is not a cloud computing environment. So, we aim at utilizing this experience to reduce the algorithmic search time. This give us better solutions when the User is in the loop. This happen because we exclude uninteresting solutions and allow the algorithms to evolve interesting solutions only.

5.4 Limitations of the Study

We got our results from analyzing the data based on settings, parameters, and procedures that are followed throughout the study. There is a possibility that if the parameters change the results might change for all algorithms. We used the same properties for all CloudSim entities (such as data-center, hosts, Storage, and number of (users) and the same parameters settings for all meta-heuristic algorithms (such as selection, mutation and crossover type and probability). We cannot identify the expected behavior of these algorithms if their properties change, and how the changing of these properties could affect our objectives.

The Meta-heuristic start from initial population set randomly, that means this can affect our results too. to mitigate this effect, we repeated each algorithm 30 independent times for each dataset workflow, and that enables us to accept the results in spite of the randomness inherent in the algorithms.

The study used jMetal and WorkflowSim simulation in cloud computing. Our results may not represent the results in real life environment. The simulation works with specific tasks assigned to specific Virtual Machines. So, in real life environment we need to consider other variables such as other tasks running on the same machine, in addition to network and cloud computing structure and topology.

Chapter 6

Conclusions and Future Work

Optimizing the efficiency of task scheduling in cloud computing continues to be an essential job for system administrators. In this study we presented the importance of solving the task workflow problem and the research directions in that area. Our approach was to redefine the problem as a multi-objective optimization problem with three objectives and the use of multi-objective evolutionary algorithms (MOEAs) to present the user with Pareto efficient solutions to choose from. The results show the robustness of MOEAs compared to traditional algorithms in WorkflowSim for cloud computing environments. Specifically, MOCell and NSGA-II produced the best results in the studied experiments according to Pareto front quality metrics.

Furthermore, we introduced an approach for involving users in the optimization process that allows the decision maker to focus on a preference area within the objective space and focus the computation power on that area. The results of user-in-the-loop show that the MOEA has the ability to generate solutions close to the optimal Pareto Front and to distribute solutions uniformly through the non-dominated solution set.

As future work, workflow scheduling optimization algorithms in cloud computing still need more effort and time to reduce iteration time from 1 minute to 30 seconds or less. Also, we are interested in covering more Workflow types and larger data sets, and experimenting with different parameter settings to arrive at the best parameter tuning for our problem.

Appendix A

Appendix

A.1 UIL Test Case

This section presents a demo for our experiments in details. The UIL means the user is able to start the framework. the user has the ability to choose between two running modes the user in or out of the search process in the framework, based on this running mode we will start executing the framework.

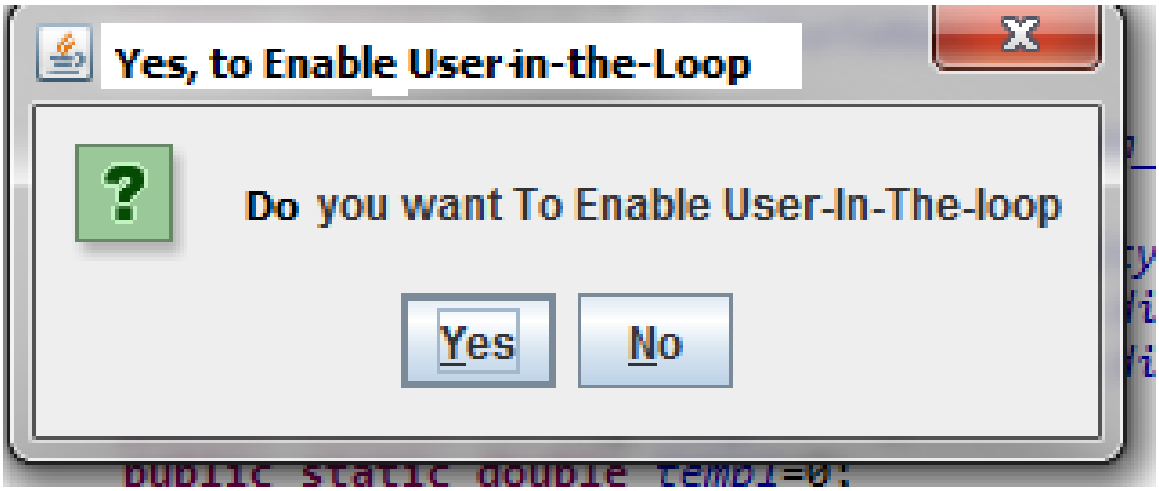


Figure A.1: UIL Running Mode.

The user will be able to start UIL or stop the running process through GUI if the running mode user is in the search process is running. The image below appears if the user presses on "Start UIL" button through GUI the results are: a user able to cancel this operation or interrupt it; the population will be created to the user and available to create the visualization of this population.

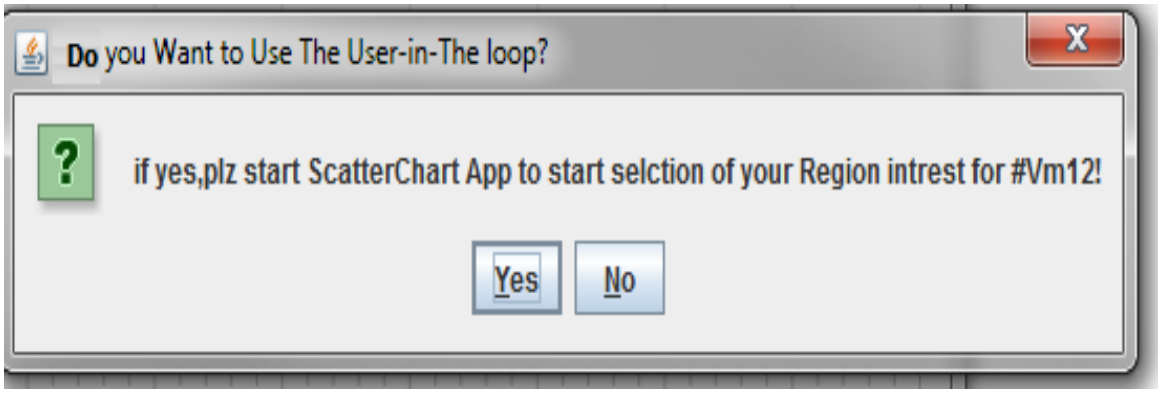


Figure A.2: Start UIL Process

The user has to select the region of his interest according to the decision of the running mode at the beginning of the process, So, this selection should be mandatory to end user and no way to exist without his selection. The frameworks will calculate the selection time and converts this time to actual running time.

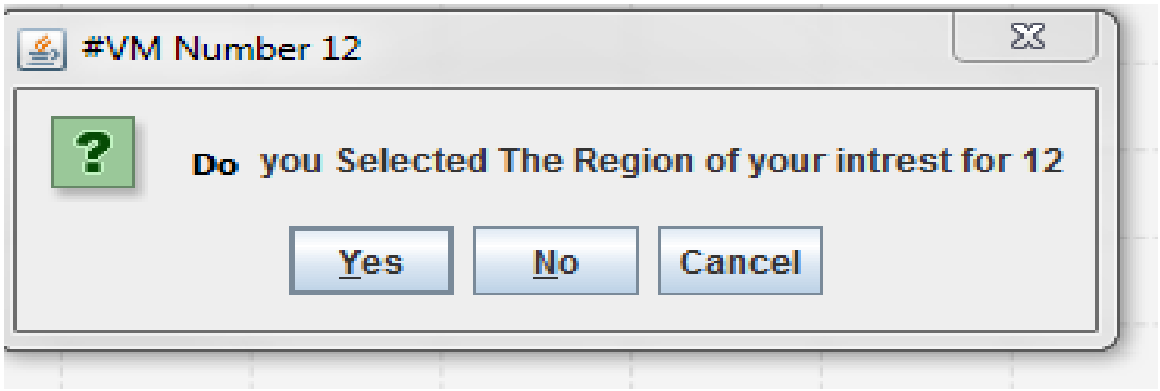


Figure A.3: Check the Input of Selection from User

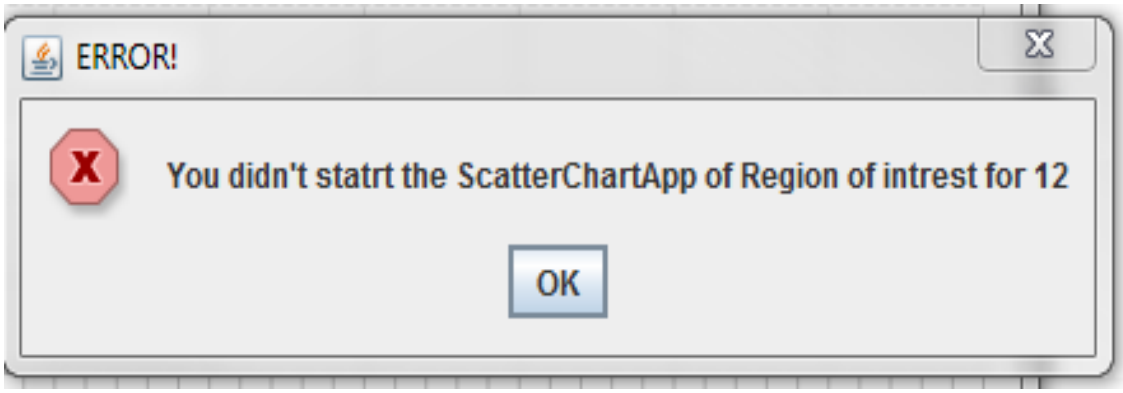


Figure A.4: User doesn't select the region

The button start data fetcher is used to read the population of solutions at the interruption point and prepares the data to be ready for drawing through "Draw Chart" button, also, the

next chart button enables the user to check and draw the objectives. The user can drag and drop a rectangle to select all the region of his interest.

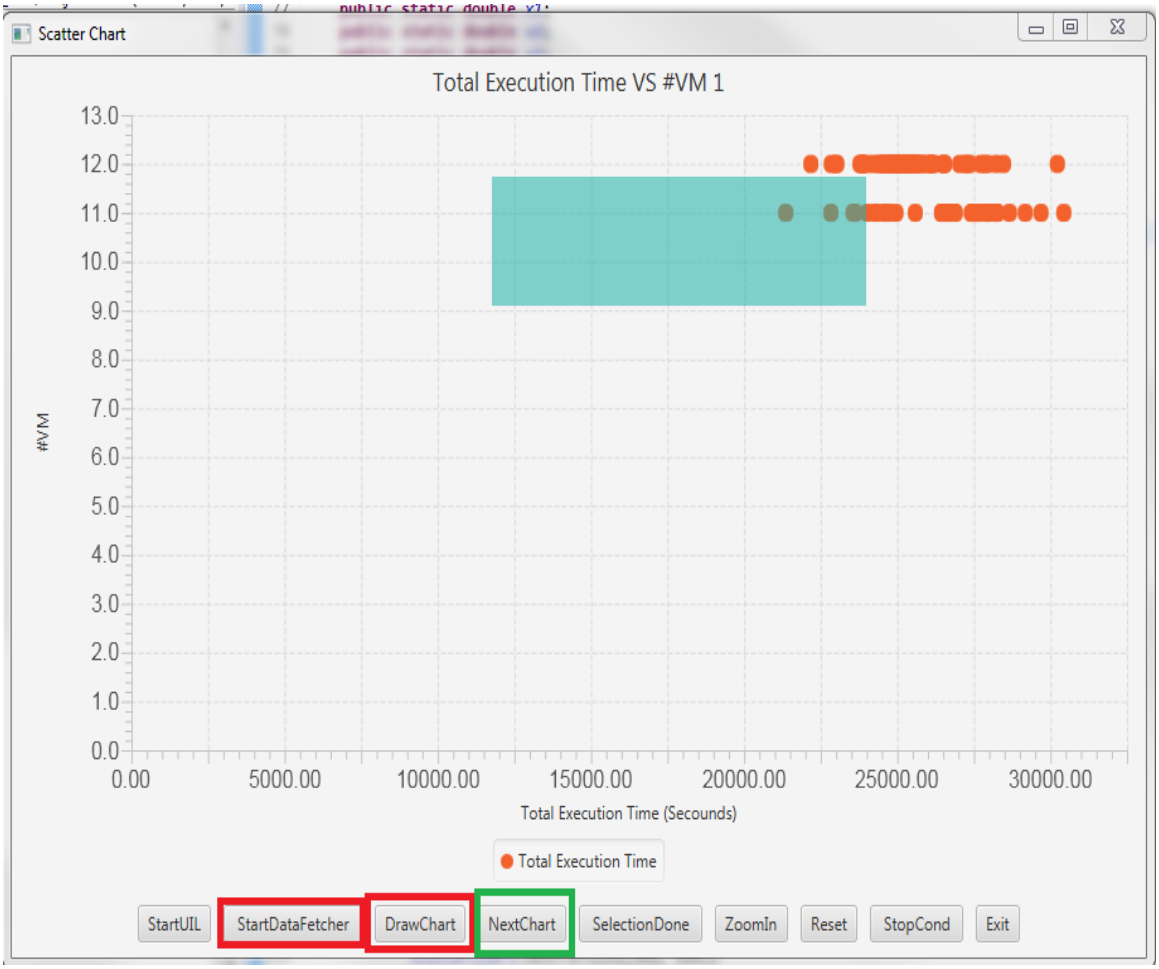


Figure A.5: Read and Draw the Population set

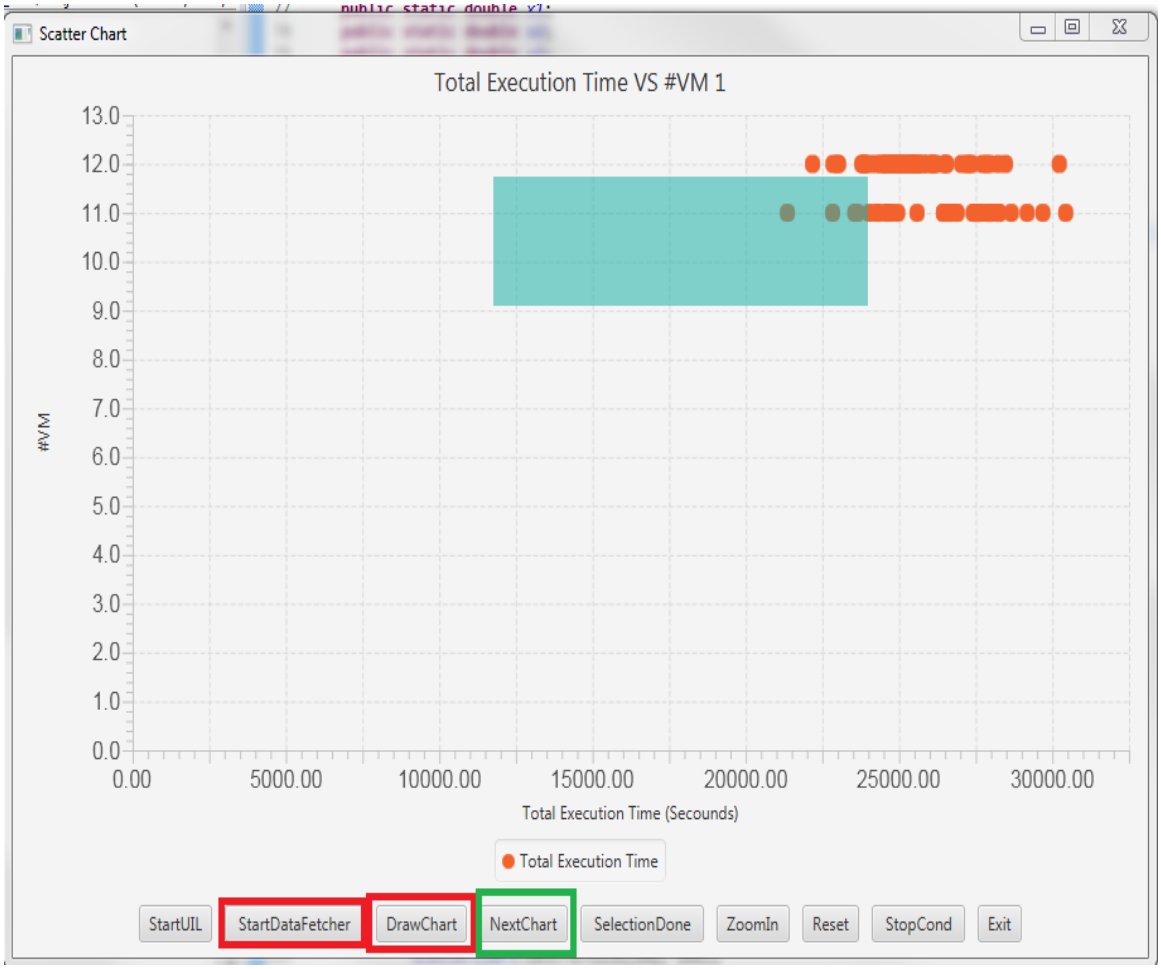
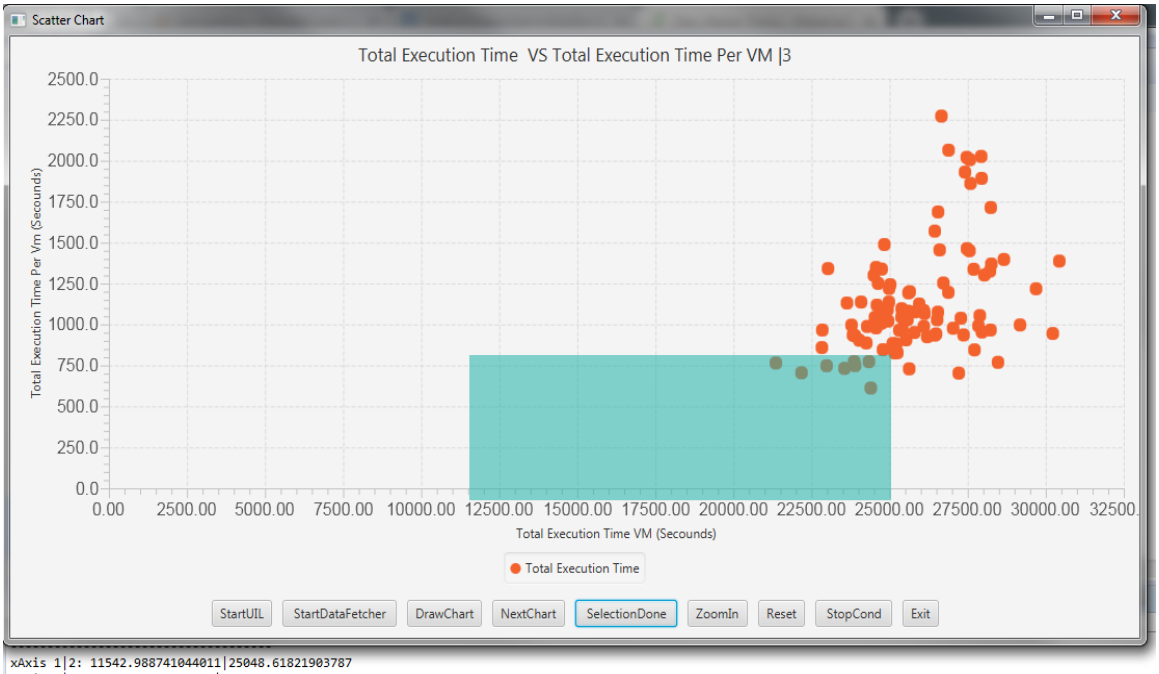


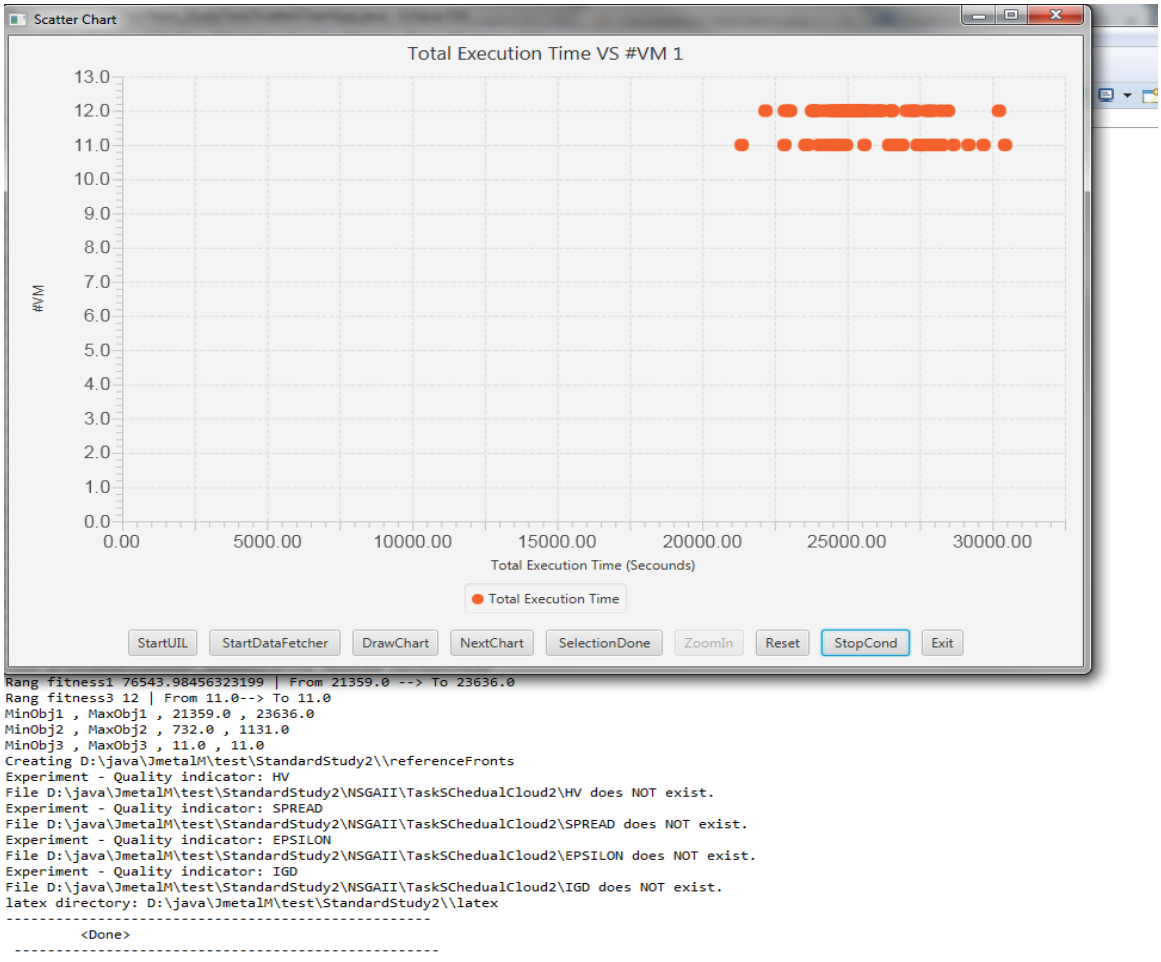
Figure A.6: Next chart from Population set

The button "Done" is used to create the region of interest of the user, and get all points from the population in that region. When this is done the algorithm will check the input file and read the values from a user selection to use them. the "Stop Cond" button is used to stop the execution process of algorithms.




```
xAxis 1|2: 11542.988741044011|25048.61821903787
yAxis 1|2: 989.3364928909953|195.49763033175356
Coord Selected#_0_23557_11_732
Coord Selected#_1_22179_12_705
Coord Selected#_2_21359_11_765
Coord Selected#_3_24398_12_612
Coord Selected#_4_22984_12_748
Coord Selected#_5_22833_11_859
Coord Selected#_6_24338_11_773
Coord Selected#_9_23864_12_773
Coord Selected#_10_23895_12_749
Coord Selected#_11_22845_12_966
Coord Selected#_12_23828_12_935
Coord Selected#_13_24256_11_887
Coord Selected#_14_24029_11_904
Coord Selected#_16_23889_12_932
Coord Selected#_18_24788_12_846
Coord Selected#_23_24568_11_979
Coord Selected#_24_24284_12_988
```

Figure A.7: Creates the Region of Interest from Population set




```
Rang fitness1 76543.98456323199 | From 21359.0 --> To 23636.0
Rang fitness3 12 | From 11.0--> To 11.0
MinObj1 , MaxObj1 , 21359.0 , 23636.0
MinObj2 , MaxObj2 , 732.0 , 1131.0
MinObj3 , MaxObj3 , 11.0 , 11.0
Creating D:\java\JmetalM\test\StandardStudy2\\referenceFronts
Experiment - Quality indicator: HV
File D:\java\JmetalM\test\StandardStudy2\NSGAII\TaskSChedualCloud2\HV does NOT exist.
Experiment - Quality indicator: SPREAD
File D:\java\JmetalM\test\StandardStudy2\NSGAII\TaskSChedualCloud2\SPREAD does NOT exist.
Experiment - Quality indicator: EPSILON
File D:\java\JmetalM\test\StandardStudy2\NSGAII\TaskSChedualCloud2\EPSILON does NOT exist.
Experiment - Quality indicator: IGD
File D:\java\JmetalM\test\StandardStudy2\NSGAII\TaskSChedualCloud2\IGD does NOT exist.
latex directory: D:\java\JmetalM\test\StandardStudy2\\latex
-----------------------------------------------------
        <Done>
 -----------------------------------------------------
```

Figure A.8: UIL Stop Condition

The UIL test case in this study presents how the user will be able to control and create the region of interest through GUI. According to our design and workflow to solve the problem in cloud computing, we expect to get significant results compared to WFS algorithms, and significant results when UIL and these results will outperform the results when the user is out of the process.

Bibliography

[1] Ajith Abraham and Lakhmi Jain. “Evolutionary multiobjective optimization”. In: *Evolutionary Multiobjective Optimization*. Springer, 2005, pp. 1–6.

[2] Ram Bhushan Agrawal, K Deb, and RB Agrawal. “Simulated binary crossover for continuous search space”. In: *Complex systems* 9.2 (1995), pp. 115–148.

[3] Victor Alves Ribeiro. “Multi-Objective Model Selection for Unmanned Aerial Vehicles Automatic Target Recognition Systems”. PhD thesis. May 2017. DOI: 10.13140/RG.2.2.35499.85283.

[4] Hamid Arabnejad and Jorge G Barbosa. “List scheduling algorithm for heterogeneous systems by an optimistic cost table”. In: *IEEE Transactions on Parallel and Distributed Systems* 25.3 (2014), pp. 682–694.

[5] Yalda Aryan and Arash Ghorbannia Delavar. “A bi-objective workflow application scheduling in cloud computing systems”. In: *Int J Integr Technol Educ* 3 (2014), pp. 51–62.

[6] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*. Vol. 1. CRC press, 2000.

[7] Anju Bala and Inderveer Chana. “A survey of various workflow scheduling algorithms in cloud environment”. In: *2nd National Conference on Information and Communication Technology (NCICT)*. sn. 2011, pp. 26–30.

[8] Sanjoy K Baruah. “The non-preemptive scheduling of periodic tasks upon multiprocessors”. In: *Real-Time Systems* 32.1-2 (2006), pp. 9–20.

[9] Christian Bierwirth and Dirk C Mattfeld. “Production scheduling and rescheduling with genetic algorithms”. In: *Evolutionary computation* 7.1 (1999), pp. 1–17.

[10] Jacek Blazewicz, Jan Karel Lenstra, and AHG Rinnooy Kan. “Scheduling subject to resource constraints: classification and complexity”. In: *Discrete Applied Mathematics* 5.1 (1983), pp. 11–24.

[11] Junwei Cao et al. “Gridflow: Workflow management for grid computing”. In: *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*. IEEE. 2003, pp. 198–205.

[12] Weiwei Chen and Ewa Deelman. “Workflowsim: A toolkit for simulating scientific workflows in distributed environments”. In: *E-science (e-science), 2012 IEEE 8th International Conference on*. IEEE. 2012, pp. 1–8.

[13] Francisco Chicano et al. “Fitness Probability Distribution of Bit-flip Mutation”. In: *Evol. Comput.* 23.2 (June 2015), pp. 217–248. ISSN: 1063-6560. DOI: 10.1162/EVCO_a_00130. URL: http://dx.doi.org/10.1162/EVCO_a_00130.

[14] Karim Chichakly. *Multiobjective Optimization*. [Online]. Available: blog.iseesystems.com/modeling-tips/multiobjective-optimization/. 2018.

[15] CloudSim. *CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services*. [Online]. Available: http://www.cloudbus.org/cloudsim/. 2016.

[16] David Corne, Joshua Knowles, and Martin Oates. “The Pareto Envelop-based Selection Algorithm for Multi-Objective Optimization”. In: vol. 1917. Sept. 2000, pp. 839–848. DOI: 10.1007/3-540-45356-3_82.

[17] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. “Exploration and exploitation in evolutionary algorithms: A survey”. In: *ACM Computing Surveys (CSUR)* 45.3 (2013), p. 35.

[18] Altino Dantas et al. “Interactive software release planning with preferences base”. In: *International Symposium on Search Based Software Engineering*. Springer. 2015, pp. 341–346.

[19] Lawrence Davis. “Genetic algorithms and simulated annealing”. In: (1987).

[20] Kalyanmoy Deb and Abhishek Kumar. “Interactive evolutionary multi-objective optimization and decision-making using reference direction method”. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM. 2007, pp. 781–788.

[21] Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. “Multi-objective optimization”. In: *Decision Sciences: Theory and Practice*. CRC Press, 2016, pp. 145–184.

[22] Kalyanmoy Deb et al. “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II”. In: *International Conference on Parallel Problem Solving From Nature*. Springer. 2000, pp. 849–858.

[23] Juan J. Durillo and Antonio J. Nebro. “jMetal: A Java framework for multi-objective optimization”. In: *Advances in Engineering Software* 42 (2011), pp. 760–771. ISSN: 0965-9978. DOI: DOI:10.1016/j.advengsoft.2011.05.014. URL: http://www.sciencedirect.com/science/article/pii/S0965997811001219.

[24] Russ C Eberhart, James Kennedy, et al. “A new optimizer using particle swarm theory”. In: *Proceedings of the sixth international symposium on micro machine and human science*. Vol. 1. New York, NY. 1995, pp. 39–43.

[25] Hesham El-Rewini and Ted G. Lewis. “Scheduling parallel program tasks onto arbitrary target machines”. In: *Journal of parallel and Distributed Computing* 9.2 (1990), pp. 138–153.

[26] Ian Foster et al. “Cloud computing and grid computing 360-degree compared”. In: *Grid Computing Environments Workshop, 2008. GCE’08*. Ieee. 2008, pp. 1–10.

[27] D.C. Gabriner et al. *System and method for genetic algorithm scheduling systems*. US Patent 5,848,403. 1998. URL: https://www.google.com/patents/US5848403.

[28] Alex Gantman et al. "Scheduling real-time tasks in distributed systems: A survey". In: (1998).

[29] Michael R Garey and Ronald L. Graham. "Bounds for multiprocessor scheduling with resource constraints". In: *SIAM Journal on Computing* 4.2 (1975), pp. 187–200.

[30] Hui Liu He Guo Guan Wang Yuxin Wang. "HSIP: A Novel Task Scheduling Algorithm for Heterogeneous Computing". In: *Scientific Programming* 2016.3676149 (2016), p. 11. URL: http://dx.doi.org/10.1155/2016/3676149.

[31] Lizheng Guo et al. "Task scheduling optimization in cloud computing based on heuristic algorithm". In: *Journal of Networks* 7.3 (2012), pp. 547–553.

[32] Irfan Habib et al. "Adapting scientific workflow structures using multi-objective optimization strategies". In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 8.1 (2013), p. 4.

[33] Mark Harman and Bryan F Jones. "Search-based software engineering". In: *Information and software Technology* 43.14 (2001), pp. 833–839.

[34] Mark Harman and Afshin Mansouri. "Search based software engineering: Introduction to the special issue of the IEEE transactions on software engineering". In: *IEEE transactions on Software Engineering* 6 (2010), pp. 737–741.

[35] Je rey Horn, Nicholas Nafpliotis, and David E Goldberg. "A niched Pareto genetic algorithm for multiobjective optimization". In: *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*. Vol. 1. Citeseer. 1994, pp. 82–87.

[36] Jie Huang. "The Workflow Task Scheduling Algorithm Based on the GA Model in the Cloud Computing Environment." In: *JSW* 9.4 (2014), pp. 873–880.

[37] Pham Phuoc Hung et al. "Task scheduling for optimizing recovery time in cloud computing". In: *Computing, Management and Telecommunications (ComManTel), 2014 International Conference on*. IEEE. 2014, pp. 188–193.

[38] C-L Hwang and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Vol. 164. Springer Science & Business Media, 2012.

[39] Oscar H Ibarra and Chul E Kim. "Heuristic algorithms for scheduling independent tasks on nonidentical processors". In: *Journal of the ACM (JACM)* 24.2 (1977), pp. 280–289.

[40] RK Jena. "Multi objective task scheduling in cloud environment using nested PSO framework". In: *Procedia Computer Science* 57 (2015), pp. 1219–1227.

[41] Azita Jooyayeshendi and Abbas Akkasi. "Genetic Algorithm for Task Scheduling in Heterogeneous Distributed Computing System". In: ().

[42] Natallia Kokash. “An introduction to heuristic algorithms”. In: (Dec. 2018).

[43] C Mani Krishna and Kang G. Shin. “On scheduling tasks with a quick recovery from failure”. In: *IEEE Transactions on Computers* 5 (1986), pp. 448–455.

[44] Neal Leavitt. “Is cloud computing really ready for prime time”. In: *Growth* 27.5 (2009), pp. 15–20.

[45] Longmei Li et al. “Multiobjective evolutionary algorithms based on target region preferences”. In: *Swarm and Evolutionary Computation* 40 (2018), pp. 196–215.

[46] Longmei Li et al. “Preference incorporation to solve multi-objective mission planning of agile earth observation satellites”. In: *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE. 2017, pp. 1366–1373.

[47] Tianchi Ma and Rajkumar Buyya. “Critical-path and priority based algorithms for scheduling workflows with parameter sweep tasks on global grids”. In: *Computer Architecture and High Performance Computing, 2005. SBAC-PAD 2005. 17th International Symposium on*. IEEE. 2005, pp. 251–258.

[48] Bogdan Marculescu et al. “Tester Interactivity makes a Difference in Search-Based Software Testing: A Controlled Experiment”. In: *CoRR* abs/1512.04812 (2015). URL: http://arxiv.org/abs/1512.04812.

[49] Teena Mathew, K Chandra Sekaran, and John Jose. “Study and analysis of various task scheduling algorithms in the cloud computing environment”. In: *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*. IEEE. 2014, pp. 658–664.

[50] Hitoshi Matsumoto and Yutaka Ezaki. “Dynamic resource management in cloud environment”. In: *Fujitsu Sci. Tech. J* 47.3 (2011), pp. 270–276.

[51] Gaurang Mehta, Gideon Juve, and W Chen. “Workflow Generator”. In: *confluence. pegasus. isi. edu* (2009). URL: https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator(Đt’ĐřÑĆĐřĐ¿ĐśÑĂĐřÑL’ĐμĐ¡ĐÿÑŕ:11.06.2016).

[52] Peter Mell and Tim Grance. “The NIST definition of cloud computing”. In: (2011).

[53] Brad L Miller, David E Goldberg, et al. “Genetic algorithms, tournament selection, and the effects of noise”. In: *Complex systems* 9.3 (1995), pp. 193–212.

[54] Antonio J Nebro et al. “AbYSS: Adapting scatter search to multiobjective optimization”. In: *IEEE Transactions on Evolutionary Computation* 12.4 (2008), pp. 439–457.

[55] Antonio J Nebro et al. “Extending the Speed-Constrained Multi-objective PSO (SMPSO) with Reference Point Based Preference Articulation”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2018, pp. 298–310.

[56] Antonio J Nebro et al. “Mocell: A cellular genetic algorithm for multiobjective optimization”. In: *International Journal of Intelligent Systems* 24.7 (2009), pp. 726–746.

[57] Andrew J Page and Thomas J Naughton. “Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing”. In: *19th IEEE international parallel and distributed processing symposium*. IEEE. 2005, 189a–189a.

[58] Radu Prodan and Thomas Fahringer. “Dynamic scheduling of scientific workflow applications on the grid: a case study”. In: *Proceedings of the 2005 ACM symposium on Applied computing*. ACM. 2005, pp. 687–694.

[59] Marjan Kuchaki Rafsanjani and Amid Khatibi Bardsiri. “A new heuristic approach for scheduling independent tasks on heterogeneous computing systems”. In: *International Journal of Machine Learning and Computing* 2.4 (2012), p. 371.

[60] Mustafizur Rahman, Srikumar Venugopal, and Rajkumar Buyya. “A dynamic critical path algorithm for scheduling scientific workflow applications on global grids”. In: *e-Science and Grid Computing, IEEE International Conference on*. IEEE. 2007, pp. 35–42.

[61] Aurora Ramirez, Jose Raul Romero, and Christopher Simons. “A systematic review of interaction in search-based software engineering”. In: *IEEE Transactions on Software Engineering* (2018).

[62] Sartaj K Sahni. “Algorithms for scheduling independent tasks”. In: *Journal of the ACM (JACM)* 23.1 (1976), pp. 116–127.

[63] Abdel Salam Sayyad, Tim Menzies, and Hany Ammar. “On the value of user preferences in search-based software engineering: a case study in software product lines”. In: *2013 35th International Conference on Software Engineering (ICSE)*. IEEE. 2013, pp. 492–501.

[64] L.K. Sheng et al. “Multi-Objective particle swarm optimization algorithms – A leader selection overview”. In: 15 (Aug. 2014), pp. 6–19. DOI: 10.5013/IJSSST.a.15.04.02.

[65] Jerffeson Teixeira de Souza et al. “The human competitiveness of search based software engineering”. In: *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*. IEEE. 2010, pp. 143–152.

[66] Sobhanayak Srichandan, Turuk Ashok Kumar, and Sahoo Bibhudatta. “Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm”. In: *Future Computing and Informatics Journal* 3.2 (2018), pp. 210–230.

[67] Hiroyuki Tarumi. *Workflow system for rearrangement of a workflow according to the progress of a work and its workflow management method*. US Patent 6,115,640. 2000.

[68] Jyoti Thaman and Manpreet Singh. “Green cloud environment by using robust planning algorithm”. In: *Egyptian Informatics Journal* 18.3 (2017), pp. 205–214.

[69] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. “Performance-effective and low-complexity task scheduling for heterogeneous computing”. In: *IEEE transactions on parallel and distributed systems* 13.3 (2002), pp. 260–274.

[70] David A Van Veldhuizen and Gary B Lamont. *Multiobjective evolutionary algorithm research: A history and analysis*. Tech. rep. Citeseer, 1998.

[71] Amandeep Verma and Sakshi Kaushal. "Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud". In: *International Journal of Grid and Utility Computing* 5.2 (2014), pp. 96–106.

[72] El Yamany et al. "OPTI-SELECT: an interactive tool for user-in-the-loop feature selection in software product lines". In: *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools-Volume 2*. ACM. 2014, pp. 126–129.

[73] Jia Yu and Rajkumar Buyya. "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms". In: *Scientific Programming* 14.3-4 (2006), pp. 217–230.

[74] Jia Yu, Rajkumar Buyya, and Kotagiri Ramamohanarao. "Workflow scheduling algorithms for grid computing". In: *Metaheuristics for scheduling in distributed computing environments* (2008), pp. 173–214.

[75] Shaobin Zhan and Hongying Huo. "Improved PSO-based task scheduling algorithm in cloud computing". In: *Journal of Information & Computational Science* 9.13 (2012), pp. 3821–3829.

[76] Lei Zhang et al. "A task scheduling algorithm based on PSO for grid computing". In: *International Journal of Computational Intelligence Research* 4.1 (2008), pp. 37–43.

[77] Eckart Zitzler and Simon Künzli. "Indicator-based selection in multiobjective search". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2004, pp. 832–842.